anchore

Container Security for US Government Information Systems

info@anchore.com

www.anchore.com

Contents

Introduction–
Container Security

Timely updating and upgrading of software, remains number one advisory in the NSA Top 10 Mitigation Strategies for Cybersecurity:

Apply all available software updates, automate the process to the extent possible, and use an update service provided directly from the vendor. Automation is necessary because threat actors study patches and create exploits, often soon after a patch is released. These "N-day" exploits can be as damaging as a zero-day. Vendor updates must also be authentic; updates are typically signed and delivered over protected links to assure the integrity of the content. Without rapid and thorough patch application, threat actors can operate inside a defender's patch cycle.

Efficient patching was once the easy hit for security and operations teams. However, the emergence of containerization at the core of the modern continuous integration and continuous development (CI/CD), has introduced an additional, opaque layer of complexity for platform managers and security teams monitoring the software cycle.

If this complexity is not managed and monitored effectively, it can provide a significant new attack surface, where threat actors can introduce critical vulnerabilities into otherwise secure government platforms.

Federal organizations need to be aware that effective container security involves far more than the ability to scan the contents of a container. To work efficiently, security must be integrated into the CI/CD workflow and it must operate in synchronization with the speed and processes of modern containerized development.

Purpose

The purpose of this document is to provide information security best practices for Federal organizations who have chosen to deploy containers at scale. The paper offers specific guidance and recommendations for these organizations on implementing an effective DevSecOps process.

Federal authorities have published a notable document covering basic aspects of container security: NIST 800-190 Application Container Security Guide. However, much of the additional, relevant industry knowledge is then spread across various peripheral documents such as, NIST 800-53 Rev.4 :Security and Privacy Controls for Federal Information Systems and Organizations , NIST 800-171:Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations, U.S. Department of Defense Cloud Computing Requirements Guide and various FedRAMP publications.

This paper seeks to consolidate some of the relevant information from these various guidelines. It also presents some timeless information security best practices as they pertain to container security. The aim is to provide DevOps and SecOps teams, and their management, with a set of simple solutions and methodologies to maintain secure control of their containers throughout the deployment lifecycle.

Our goal is to provide a simple approach for implementing and maintaining control of container security at scale within Federal information systems.

Audience

The intended audience for this paper is for security engineers, DevOps engineers, security administrators, information system security officers (ISSO), information system security managers (ISSM) and Chief Information Security Officers (CISO). The document is relevant for anyone who is interfacing with containers within their information system and authorization boundaries within the U.S. Department of Defense (DoD) and other U.S. Federal organizations.

The content assumes the audience has a solidified understanding of cloud architecture, DevSecOps, containerization technologies, and a moderate understanding of a CI/CD workflow in a Federal workspace.

The Container Advantage

Introduction

Containers are a powerful tool that is highly complementary to existing VM or Cloud instances. They are gaining wide adoption within the development community, and now enjoy strong support from a wide array of major vendors.

By passing around immutable images, containers offer strong development advantages, both in developer experience and potential automation.

Although they bring unique security challenges, these are manageable using the correct tooling and by adopting a strong container-focused security model. With the correct approach, containers can not only increase developer velocity, but also substantially strengthen security within all Federal organizations.

Lifespan and Mutability

Containers bring considerable advantages that extend far beyond older virtual machine (VM) infrastructure. In certain ways, containers can be seen as the next evolution of VM's. And increasingly, existing VM infrastructure and practices are being repurposed to accommodate container platforms.

Virtual Machines are essentially server hardware encapsulated within hypervisor software. This allows a single large server to be presented as many smaller ones. It enables a more effective use of resources through the more efficient sharing of CPU and RAM. VM's are also fully isolated from each other, which adds to security as long as the Hypervisor software is secure.

However, once they are launched from a 'Golden Image', VM's are mutable, and can be altered in much the same way as the servers they replace. Tools such as Ansible, Puppet or Chef are used in an attempt to manage changes and control configuration drift. But where problems arise, operators will often change configuration manually as part of any fix.

A container should be viewed as a wrapper for a single process, rather than a general machine

Because 'state' within VM's is long-lived and mutable, they can be challenging to monitor and manage, and almost impossible to keep in lockstep with each other over time. This is perhaps the most significant difference between VM's and Containers.

Containers are often thought of as lightweight virtualization, they are short-lived to the point of being disposable and core aspects of their immutability can be set in stone.

A container should be viewed as a wrapper for a single process, rather than a general machine. They offer a very isolated way to run a process. They are run on top of a shared operating system kernel, but this is within their own highly isolated user environment. This increased granularity makes containers even more efficient than virtual machines for sharing resources, with orchestration platforms such as Kubernetes offering the ability to manage many thousands of containers efficiently.

Containers and the processes running within them, are constantly started and stopped and may only exist for a fraction of a second. Any changes made to the container during its lifetime, do not survive restarts. And because the images are both immutable and containers short-lived, this makes it relatively trivial to avoid configuration drift.

System Sharing

One key advantage that containers have over VM's is that they are easy to package and share as a complete deployable, covering everything from developed code through to operating system configuration.

Container images are generally developed using a simple and declarative configuration language (although, existing tools such as Puppet or Ansible can also be readily used to build images). Once created, the images are simple to distribute, either as compressed files or, more commonly, using a 'registry'. The registry provides a central image store on either the public or private network where developers can then pull down an image to the local machine with a simple command. Shared images are then ready to use, either by running them directly, or using them as a compositional piece in a new image.

This flexibility creates a massive developer advantage, allowing them to create, share and improve upon an entire system.

This flexibility creates a massive developer advantage, allowing them to create, share and improve upon an entire system. Rather than having to intuit how to set up the sometimes-complex set of packages and configuration to run a system, a developer can simply run the image.

It also allows developers to share their variants of these systems easily. Organizations may determine a particular version of a product, such as Postgres, that must be used when developing internal solutions. Containers make it trivial to prescribe and provide these bespoke configurations to developers for use in their development cycle.

This collaborative approach offers another crucial advantage for developers, letting them begin developing immediately against a production-like environment. It also allows them to discover issues far earlier in the development cycle. Bugs that would typically have only been caught when code was promoted into a managed environment, can now be found and fixed much earlier on the developers' laptop.

The Core of Modern Development

Developers can now easily package and share, not just isolated containers, but full environments. This has led to an explosion of container use. Increasingly, containers are being offered as the first-class installation experience, with more 'traditional' models of packing becoming less and less prevalent.

This enthusiasm for containerization has spread to even the largest vendors. Companies such as Red Hat and Microsoft are offering software in containers and engaging and contributing to the broader ecosystem. Increasingly, developer tooling such as IDE's can integrate with containers, leveraging their ease of use and collaborative properties to ease the development lifecycle.

Kubernetes is now firmly entrenched as the container orchestration tool of choice, with every significant cloud provider now offering a managed version. Developers are increasingly turning to containers to both develop against known and trusted development environments, but also to massively increase development velocity. Where a developer may once have had to spend half a day installing a Postgres database to develop against, they can now install it in seconds using a container available within the public registry. In the past, developers may have had to maintain many different versions of a language and framework on their laptops, and use workarounds to make sure each project used the correct one, now they can do so easily by isolating each one within a container.

This enthusiasm for containerization has spread to even the largest vendors

Containers have been transformative to how developers work, and Anchore is built to maintain and optimise security in a world where containers will increasingly become the de facto way to package and distribute software.

Essential Container Security

New Threats...

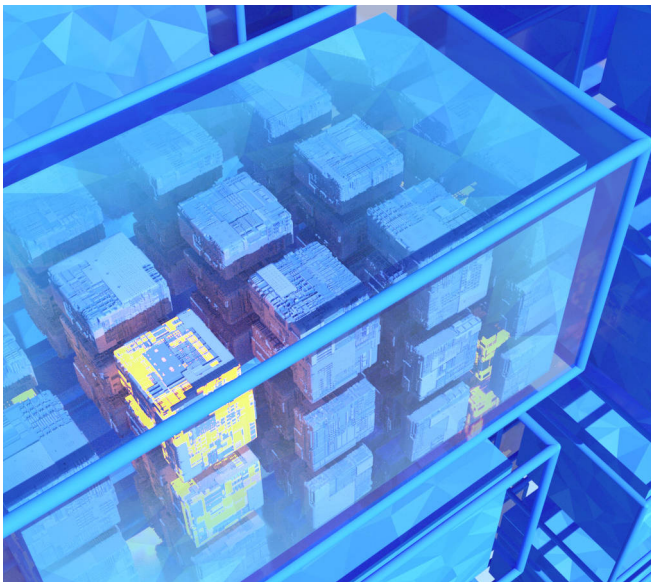
While containers can dramatically increase the speed of development and foster greater collaboration, they have created their own, new set of security challenges.

Container images are commonly constructed from other parent images. Developers will rarely create an image from scratch. Instead, most will turn to existing parent images such as Ubuntu, or the slimline Alpine images.

However, these images are sparse, usually composed of a shell and some form of package management software. For developers using a language such as Python or Ruby, these parent images still require additional work to be usable. As a result, they may turn to an image that makes use of the upstream Ubuntu images but also installs the language of their choice.

If our developer then wishes to make use of a framework such as Rails they may still have some work to do. In this case, the developer may turn to a parent image that has already packaged Rails. In this scenario, the developer is using a parent image that inherits from at least two, and possibly more parent images itself.

With every fresh indirection at the parent image level, attackers have a further opportunity to implement malicious software. Moreover, this threat would not be immediately apparent to the developer, as it would be hidden behind the convenience that parent images provide.



Therefore, if any container is left unchecked, it becomes a potential vehicle for any aggressor to move potential malicious software within the security perimeter. Air Gapping would not necessarily prevent this; since images can be passed as compressed bundles and so, could be passed in without requiring access to a public network.

In this way, containers compound the already well-understood threat of supply chain attacks.

Recent attacks, such as the Magecart attack, have relied on bad actors smuggling malicious software into projects via a seemingly innocuous library that has been consumed by an upstream developer. Tools exist that can scan for these threats. However, many were developed prior to mass container adoption, do not work well with containers and are unable to inspect them effectively.

This gap in tooling creates a dual-threat where either the underlying operating system or the application code in a container can be vulnerable to supply chain attack. In addition, the simple configuration language used to construct containers can lead to bad practices creeping in.

There are many well-acknowledged techniques to secure containers, such as ensuring the Root user is not used to run the process, or that ports are not exposed without good reason. However, with most tools, these are not policed at run time. Again, the composable nature of containers means that parent images that fail to use these good practices, can pass vulnerabilities downstream, even if developers inside an organization would typically implement best practice themselves.

Containers running processes with heightened privileges increase the possibility of an aggressor being able to launch attacks on the underlying container host, and this ability must be constrained. However, detecting inappropriate access permissions within a container image is challenging without specialist tooling.

...and New Advantages

It is evident that containers bring new security challenges; however, they can also bring profound security advantages, especially in high-security situations.

By their very nature, container images are immutable; once created, they can be analyzed, approved and given a unique hash to ensure provenance. This unique feature of containers gives them a massive advantage over VM images.

VM images may start in the same place, with an image that is identifiable and certified, but once in use it's mutability renders this moot. VM's must be kept under constant scrutiny, to track identify and remediate changes. To be effective, this scanning and remediation must be rapid, frequent and reliable.

> Containers offer an improved security profile during development, giving developers a working replica of the environment in which their code will execute

Containers do not have these needs. If for some reason a container is considered suspect, it can be killed, and a new, identical container will take its place. Using orchestration systems, this advantage can be extended further. By ensuring a container's useful life is measured in minutes before it is killed, the attack surface of the container is massively reduced. It ensures that the useful time an attacker has to establish a beachhead, explore the system and enact countermeasures is virtually non-existent. Every time the container is restarted, it is in the certified state that is known to the development, operations and security teams.

Containers also offer an improved security profile during development, giving developers a working replica of the environment in which their code will execute. Rather than working on assumed knowledge of the upstream environments, they will be able to reason and work with a replica and identify issues early. This approach massively reduces inadvertent issues introduced at the development stage and makes it easier for developers, platform and security engineers to collaborate effectively.

Finally, due to the composable nature of containers, it is relatively trivial for organizations to create and prescribe approved parent images. These parent images can be created from an organizational base layer, allowing DoD and Federal organizations to have fine-grained control of exactly what is deployed, and how it is configured, within their platform environment.

Managers can not only provide a set of secure parent images, they can black-list bad images or even limit any ability for developers to pull public images. In this way, platform managers and security operatives can be assured that all images in use are configured in accordance with best practice.

Core Guidance for Container Security Implementation

1) Follow the 30/60/90 rule to keep images secure

Anchore recommends following the 30/60/90 rule to satisfy the guidance outlined in the DoD Cloud Computing Security Requirements Guide. This rule sets out the number of days it should take for security issues to be fixed, once they have been discovered: critical vulnerabilities should be fixed in 30 days, high vulnerabilities in 60 days and moderate vulnerabilities within 90.

In support of this, it is also strongly recommended to make use of a tool that allows security teams to frequently update and validate vulnerability databases with new security data. This is needed to satisfy Security Controls RA-5(2), as well as being good practice to ensure security data is timely and relevant.

By following the 30/60/90 rule and ensuring that vulnerability databases and feeds are updated promptly, SecOps teams are empowered to respond and remediate new security challenges quickly and efficiently.

2) Make use of tools that support container image white/black listing

Federal Organizations should leverage container security tools that can enforce white and blacklisting of container images. Maintaining white and blacklists are common methods of securing networks and software dependencies, however, they are less common in a containerized environment.

This capability is crucial, as containers can potentially be used as a method to deploy blacklisted software into secure areas. Containers can obfuscate the software bill of materials (BOM) from existing scanning tools. It is crucial that the tools used can examine the contents of a container and can enforce white and blacklist safeguards.

Anchore advises that container image blacklisting should take place at the CI/CD stage to allow for rapid feedback. By shifting the security feedback to the developers, they receive immediate feedback on issues. This technique allows for faster remediation, as blacklisted container images, or the software contained within them, are flagged to the developer immediately.

3) Deploy a container security tool that can maintain strong configuration management over container images and software within those images

The software delivery and security operations teams should maintain an accurate inventory of all software deployed and used on any federal information system. This inventory gives both teams accurate situational awareness of their systems and enables more accurate decision making.

To maintain a sound security posture in line with NSA guidelines, security teams should then reference this inventory to actively remove any unwanted, unneeded, or unexpected software from the information system.

The NSA further explains, "Active enterprise management ensures systems can adapt to dynamic threat environments while scaling and streamlining administrative operations."

In line with relevant controls (CM-8(3)a), Anchore advises that Federal organizations leverage a container-native security tool that can systematically deconstruct and inspect container images for all known software packages and display it to information security personnel in an organized and timely manner.

4) Use a container scanning tool that can run on all Impact levels ranging from IL-2 through IL-6

A large number of Federal Organizations must leverage tools that keep any vulnerability data regarding the information system within their authorization boundaries. However, many scanning tools require an agent that connects to the vendor's external cloud environment. This is designated as interconnectivity between DoD/Federal systems and the tool vendor, and would rule out the use of any agent/cloud-based tool within an IL-6 classified environment.

Where organisations still choose to implement an agent-based container security tool, they are then responsible for ensuring that the security vendor maintains an up-to-date accreditation for their cloud environment. The environment must also have the relevant RMF/FedRAMP security controls that can be inherited by the Federal information system during the ATO process. In addition, any DoD or Federal agency should ensure the agent-based tool is capable of running in both classified/unclassified environments.

5) Leverage Container Native Tools to Support Continuous ATO

Traditional vulnerability scanners such as Nessus, Qualys, TrendMicro (amongst many others) are used to provide valuable artefacts to third-party auditors. These are used to support validation efforts before an ATO decision is made. Although vitally important, such tools may miss security issues inherent in a containerized environment.

As such, all Federal information systems should make use of container-native security tools that can produce reporting assets. These reports, focused on the container threat surface, are equally as important as other vulnerability reports, and are vital in forming an end-to-end view of security.

Reports need to be timely, digestible and offer a level of detail that can satisfy the needs of both technical non-technical stakeholders. Ideally, reports should be able to interoperate with other tools in use with security operations and cyber threat intelligence teams in Federal markets. These reports serve as essential artefacts to be reviewed in the Continuous ATO model.

6) Engage auditors that understand DevOps lifecycles

DevSecOps is the modus operandi of software development for the DoD for the foreseeable future and is seen in the broader civilian sector as being a key technique for secure software development.

Implementing a Continuous ATO process requires the DoD and other Federal agencies to deploy security control assessors (SCA) or Third Party Assessment Organization (3PAO). Where this process is being applied to container based development, it is essential that the auditor has a concrete understanding of the DevSecOps model.

Crucially auditors need a solid understanding of containerization security threats, trends, and risks as they pertain towards both RMF and FedRAMP ATO's. This fundamental understanding of container security best practices and relevant threat models, is crucial before undertaking any audit activities against a system that makes extensive use of containerization. Without this understanding, auditing may miss fundamental security issues that are specific to containers.

7) Shift Security Left

Shifting left' is a term used to describe tools and practices that improve and encourage more rapid feedback into the early stages of development. Feedback can be varying different types of operational and functional insight and information. However, the objective is always to hand bugs and fixes back to developers as part of a smooth, ongoing, continuous development process.

Unit testing is a familiar example of shifting left, by delivering early, user-experience feedback on functionality. This ensures that most problems are caught early, during the development stage, where it is quicker and simpler to remedy them.

By shifting security left, the handling of each vulnerability becomes an integral part of the CI/CD pipeline. This prevents a mass of vulnerabilities appearing as a single irritating blockage before systems can be admitted into production. More frequent vulnerability scanning during development ensures bugs and other issues can be dealt with quickly and easily, as they arise, and security becomes a part of the development process.

With the primary focus of CI/CD environments on fast, efficient development and innovation, security has to work efficiently as part of this process. Anchore advises that DoD and Federal security teams should use tools that can deliver rapid feedback into development. Security tools must integrate with the common CI/CD and container orchestration tools, and should promote early-stage interaction with developers.

8) Express security policy as code

Where possible, tools should be selected that enable security policy to be defined as code. This brings the ability for Security operations teams to establish best practices that can then be usefully automated and pushed to tools, either across the network or in more secure environments, via an air gap.

Expressing security policy as code also enables systems to be managed using existing software development life cycle (SDLC) techniques, allowing policies to be versioned, and for versions to be compared for configuration drift or unexpected changes. In essence, it will enable the policies themselves to be subjected to the same level as rigour as the code they are applied against.

Where the onus of implementing new security policies has been shifted left onto developers, it can be important not to tighten container security policies too far in one single step. Versioning also enables any agencies to improve and tighten security policy over time.

This iterative approach towards improving security, stops over-intrusive security policy from stalling development in the CI/CD pipeline, and prevents the emergence of any culture clash between developers and security operations. Security teams can begin with a policy base that delivers on minimum compliance standards and develop this over time towards evolving best practice.

Why does Container Security Matter for DoD and Federal environments?

In the past, the DoD has built applications on standardized virtual machines in accordance with Security Technical Implementation Guides (STIGs) and applicable hardening guides. To validate this, Security Control Assessment (SCA) teams have utilized STIG validation tools to ensure that proper STIG's are implemented for both the operating system and any additional software running on top of the base OS.

Within the DevSecOps model, it is vital to expand software management, application security, and software patching into container security. Container images must be inspected for vulnerabilities, suspicious software packages, outdated packages. Without this, it remains possible for vulnerable applications that are composed as container images to be placed into Federal production environments.

Security teams must first know what vulnerable software is running on the system and which images are affected by vulnerabilities. They must then ascertain whether there is a patch/fix available, or an updated version of the affected image. After they gain this insight, they can begin taking the necessary steps to introduce vulnerability remediation in a timely manner that complies with the following FedRAMP and DoD standard below:

Anchore believes this standard should be enforced for all Federal organizations deploying containers in order to provide a stabilized security posture for Federal information systems. The DoD Cloud Computing SRG explains, "For both FedRAMP and FedRAMP+ requirements, high and critical risk findings must be mitigated within 30 days. Moderate findings must be mitigated within 90 days," (Corresponding Security Controls: CA-5, CA-7).

It cannot be overstated that simply scanning for vulnerabilities does not represent a sufficient strategy for successful risk mitigation and container security within a DoD environment. To ensure timely remediation, scanning must be integrated into the CI/CD workflows: insights from scanning should be made available to developers as early as possible; any required action must become a smooth and seamless part of the ongoing development cycle; and security teams need the ability to monitor and report on progress.

A large part of success in the execution phase of container security lies in developing an effective DevSecOps culture within the organisation. However, security tooling designed around a DevOps workflow can be vital to foster and facilitate this change.

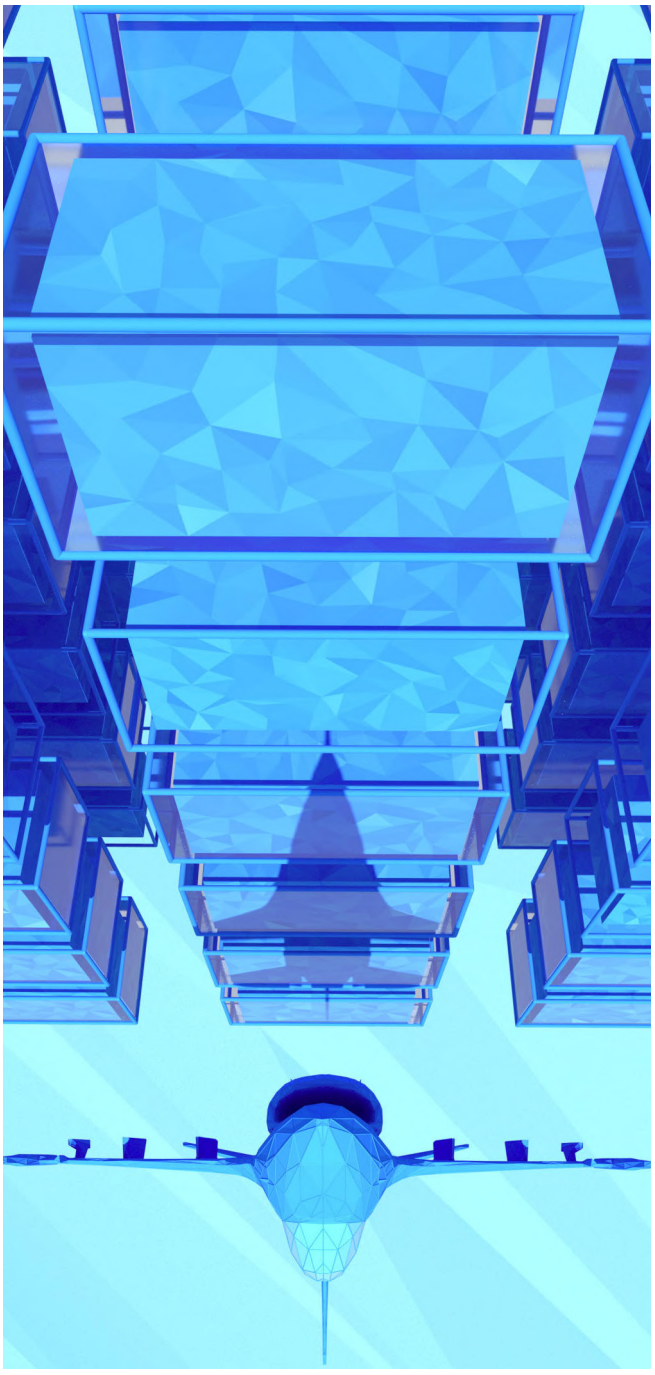
Classified Environments and Containerization

Containerization in classified environments is nothing new. And yet, while container security in classified environments has strategic importance to DoD customers, there has very little literature of successful implementation strategies.

Container security can be implemented on a classified system in a controlled manner through a typical DMZ set up in the environment. Any vulnerability feed can be imported to the DMZ and updated periodically according to the organization's standards and program requirements.

Security engineers must validate that the container security tool installation has up-to-date vulnerability data. The tool itself should never be connected directly to vulnerability sources via the internet. However, up-to-date vulnerability information can be pulled via a proxy server that is connected to the internet and that sits outside the DMZ.

Additionally, engineers must have a separate test environment from production that allows them to build images and then scan those images for relevant vulnerabilities and compliance issues. This allows software engineers and security teams alike to validate container images before introducing them into a production environment at the classified level.



How do container security tools help the U.S. Government with Continuous Monitoring and Continuous ATO?

CSP's and Federal information systems should already be providing vulnerability scanning artifacts for cloud and on-prem environments running traditional virtual machines (VM). These are relied on by multiple security control assessors and third party auditors.

The DoD Cloud Computing SRG explains, "Understanding existing vulnerabilities and risks within the enterprise is a key component in performing effective Cyberspace Defense analysis. The vulnerability reports and POA&Ms developed by the CSPs as part of continuous monitoring requirements supporting both FedRAMP and FedRAMP+ requirements will be made available to DISA's cloud services support team and subsequently to the organizations performing MCD and BCD Actions for their collective use in providing Cyberspace Defense" (CA-2).

Security control assessment teams should scan, collect, and analyze vulnerability scan data as part of best practice. This should be used to validate Federal information systems and CSP's are complying with relevant RMF/FedRAMP security controls, including RA-5; Vulnerability Scanning.

RA-5 states " Scans for vulnerabilities in the information system and hosted applications…". If the Federal agency is implementing DevOps and containerization, it is therefore essential for security control assessment teams to be validating vulnerability scans for any containers that will host applications living on U.S. government information systems.

Federal agencies should leverage container security tools that can support continuous monitoring and continuous ATO efforts. Agencies and supporting CSP's should be leveraging a container native tool that is possible of generating reports that can integrate with other vulnerability detection systems, augment cyber threat intelligence data, and provide the artifacts necessary to support risk assessment teams in the Continuous ATO model.

Preferably, these tools should then be integrated into their larger security suite, where vulnerability and log data can be centralized. This directly supports the continuous monitoring capabilities that should already be put in place for NIST 800-137 compliance.

Additionally, Anchore recommends providing monthly reviews of container vulnerability and compliance scans in-line with the guidance found in CSP Continuous Monitoring Strategy Guide produced by FedRAMP (RA-5a/5d,SI-2c/SI-2(2),CA-7g).

Why is Policy Important?

Employing a hardening methodology is fundamentally important due to the inherent nature of information systems not being secure out-of-the-box

Policy is important when maintaining a strong container security solution. It allows any container security tool to align with the various regulatory security requirements stipulated by each Federal agency. A security tool should be capable of incorporating policy elements, from both CIS benchmarks and the various DISA STIGs.

Although many STIG's have inherent security configurations that pertain to the kernel level, anything at the non-kernel level can be configured for the container images. As a result, organizations can leverage the policies to validate that they are meeting security engineering requirements and various compliance baselines ranging from the DoD to FedRAMP.

Anchore encourages organizations to create hardened base images that have met the select baselines of the Federal organization for developers to use.

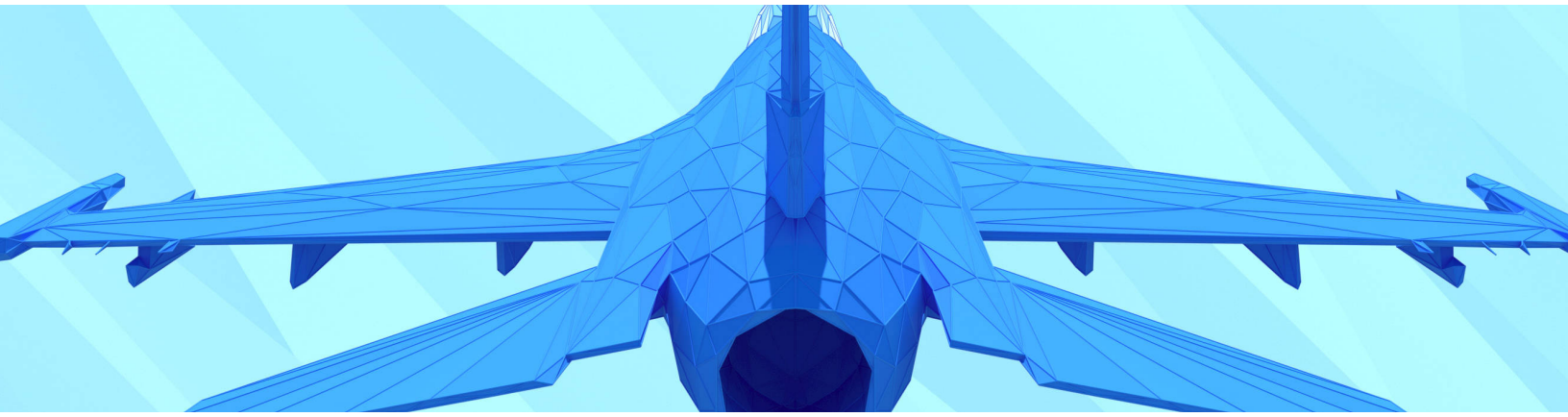
Employing a hardening methodology is fundamentally important due to the inherent nature of information systems not being secure out-of-the-box. By implementing a hardening guide, the security of information systems, network, and configuration are enhanced, decreasing the surface area for attackers. Hardening systems also apply maintenance standards for software updates and vulnerability triage, ensuring proper and secure management of software artifacts composing these information systems.

Conclusion

The U.S. Department of Defense (DoD) is, in many ways, leading the field in container security and providing a roadmap for a broader US Federal rollout.

The unimpeachable standards of the DoD and other agencies have highlighted early on that containerization demands a new approach to security. Traditional scanning tools and a manual approach to container security simply do not meet the stringent requirements for defense systems and Federal-grade security.

It is clear that containerized development requires security tools that integrate with the DevOps process and CI/CD pipelines. Without this, it becomes impossible for security teams to meet Federal requirements without wiping out many of the efficiency gains of container-based development.



About Anchore

Based out of Santa Barbara, California Anchore provides a set of tools that provide visibility, transparency, and control of your container environment. The Anchore Professional Services team helps users leverage Anchore to analyze, inspect, scan, and apply custom policies to container images within custom CI/CD pipelines.

info@anchore.com
anchore.com