Table of Contents

Introduction .................................................................................................................................................. 2

Architecture and Definitions......................................................................................................................... 2
  DoD Hardened Containers (DHC) .............................................................................................................. 2
  Container Hardening Team (DHT) ............................................................................................................. 2
  DoD Container Factory (DCF) .................................................................................................................... 2
  DoD Centralized Container Source Code Repository (DCCSCR) ................................................................ 2
  DoD Centralized Artifacts Repository (DCAR) ........................................................................................... 2
  DoD Approved CI/CD Orchestration Solution (DACCOS) ................................................................. 3
  DoD Base Container Image Approved Sources (DBCIAS) .......................................................................... 3

Container Hardening Process (CHP) .............................................................................................................. 3
  Selecting the container base image ........................................................................................................ 3
  Process ...................................................................................................................................................... 4

Example of a Hardened Container Folder Structure .................................................................................. 7
  Inside DoD Centralized Container Source Code Repository (DCCSCR): ........................................ 7
  Inside DoD Centralized Artifacts Repository (DCAR): ........................................................................ 7

Automated Hardening of New Releases ..................................................................................................... 8

Third Party Container Sharing ................................................................................................................... 8

DoD Hardened Containers Cybersecurity Requirements ............................................................................. 8
  Cybersecurity Requirements .................................................................................................................. 8
  DoD Hardening Container Risk Acceptable ............................................................................................. 9
  Active Directory / Common Access Card Integration Requirements ...................................................... 9

Appendix A .................................................................................................................................................. 10
  Security Control Inheritance (PaaS / Platform Host) ............................................................................ 10

Appendix B .................................................................................................................................................. 11
  Container Security/Remediation Considerations .................................................................................. 11
Introduction
Following the creation of the DoD Enterprise DevSecOps Initiative, Master Approach Document, this document was created to detail the DoD Container Hardening Process to ensure it meets the DoD Hardened Containers Cybersecurity Requirements. Please refer to the Master Approach Document to understand how the DSOP platform functions.

This document focuses on container security. It is understood that any application code or library must be scanned by static/dynamic code analysis tools prior to being integrated into a container and must pass those scans. This document does not describe that process.

Architecture and Definitions

DoD Hardened Containers (DHC)
A DoD hardened container is an Open Container Image (OCI) compliant image that is secured and made compliant with the DoD Hardened Containers Cybersecurity Requirements (see below). Container images should adhere to the OCI Image Format Specification to ensure portability whenever possible.

Container Hardening Team (DHT)
The Container Hardening Team is responsible for hardening DoD container according to DoD Hardened Containers Cybersecurity Requirements. The team is composed of DevOps Engineers and other container experts that have knowledge of the product being hardened and that understand the DoD Hardened Containers Cybersecurity Requirements. A single team member is called a Hardener.

A Hardener will require the following:

- Expertise in OCI compliant container images, including Docker and writing compliant Dockerfile, STIG/SCAP
- Expertise in the DoD Hardened Containers Cybersecurity Requirements
- Good understanding of Twistlock, Anchore and OpenSCAP

DoD Container Factory (DCF)
To automate the Hardening Process, the DoD will stand up a DSOP platform on a DoD approved Cloud to setup the Continuous Integration/Continuous Delivery (CI/CD) pipeline leveraged to automate the container hardening process.

DoD Centralized Container Source Code Repository (DCCSCR)
To store the Hardened Dockerfiles and associated checksums and various documentations, a source code repository will be centrally hosted so Hardeners can store their code and leverage a CI/CD pipeline.

DoD Centralized Artifacts Repository (DCAR)
To store the DoD Hardened Containers, a centralized artifacts repository will be stood up on a DoD Approved Cloud to provide access to DoD Programs to the DoD Hardened Containers.

The artifact repository must support both files (traditional artifacts) and containers. It must support container registry capabilities as well. It must provide a secure mechanism to store, track, and sign, and distribute approved containers.
DoD Approved CI/CD Orchestration Solution (DACCOS)
To build a CI/CD pipeline, an orchestration solution is needed. The DoD will select CI/CD orchestration solutions such as Jenkins, Cloudbees Jenkins and GitLab which will be containerized and hardened.

DoD Base Container Image Approved Sources (DBCIAS)
To ensure container base images are sourced from approved sources, a container base image must be downloaded from a DoD Container Base Image Approved Source.

To ensure the harden image is not “vendor locked-in” or base image becomes unavailable due to license changes or provider decisions, all hardened container images will use publicly available free and open-source software (FOSS) images, if available, and leverage the DSOP to automate the build/test/scan process of the hardening. It is important that this is as automated as possible for scalability.

Please check the “Selecting the container base image” section of this document to understand how to select your base image.

The following sources are currently approved and should be used in order of priority:

- DoD Centralized Artifacts Repository (DCAR) (trusted)
- Product Vendor proprietary repository (untrusted) – these repository can only be used if a Memorandum of Understanding (MOU) is signed between the DoD and the vendor to ensure access to base images and their updates.
- Docker Hub (untrusted)
- Red Hat Container Repository (untrusted)

A trusted repository means that the containers were already hardened by the DoD Hardening team and can be used as is.

Container Hardening Process (CHP)

Selecting the container base image

- Always prioritize the use of the base image from its vendor or its open source community so you can be directly coupled with their updates and not creating forks. Avoid creating forked Dockerfile. If you need to enhance the file, just create a Dockerfile which uses the base image with the FROM instruction.
- Modify, if need be, the FROM instruction so the Dockerfile uses the DoD-hardened/approved available base containers whenever available. Example, if Jenkins use the Docker Hub JDK image, ensure the FROM instruction is replaced by the DoD hardened JDK base image instead.
- Use base images that reuse the "closest" hardened image available to inherit its hardening: for example, if you use a Java application, use the base image of DoD hardened JDK and not the DoD hardened OS so you inherit all the hardened code of JDK as well and don’t have to re-harden JDK.
- If you have multiple source options/tags/variants for your base image, ensure you scan them with both Twistlock and Anchore to understand security risks associated with each image and select the image most closely aligned with DoD Hardened Containers Cybersecurity Requirements.
• If the base image has security flaws, first try using it as a base image in your own Dockerfile and mitigating the flaw by overriding. If that is impossible, you will have to fork the base image and remove the library/issue used to pass scan results or simply accept its risk.

• It is highly unlikely that the DoD Container environment will have internet access so if your Dockerfile pulls base images/artifacts/binaries and other files from the Internet, it is mandated to include these files inside the DCCSCR repository so the hardened container actually be used in disconnected and classified environments.

It is highly recommended to have the DoD Enterprise DevSecOps team review and agree of the selection of a base image.

Process

It is important to understand that containers will inherit many security controls of the underlying PaaS solution, assuming the underlying host OS has been hardened per agency specific security guidelines (see Appendix A).

When selecting a base image and/or hardening a container image - there are a series of best practice considerations prior to engaging the hardening pipeline (see Appendix B).

It is important to select the proper base images, see instructions above “Selecting the container base image”.

Using a DoD Approved CI/CD Orchestration Solution, the Hardener will begin the hardening process (Hardening Pipeline) through the following steps:

• Setup a new folder in the DCCSCR with the vendor name if it doesn’t already exist. Example: Open-source if open source or for example “Oracle” for the product MySQL.

• Setup a new folder in the DCCSCR with the product name if it doesn’t exist. Example: “alpine” or “mysql”.

• If a product has multiple variant options like “cli” mode or “daemon” mode or x86 vs x64 versions, various sub folders can be created for each variant.

• Define and list which version of the product, the hardened containers will initially cover/support. Example, if a Hardener needs to harden “alpine”, and the current versions of alpine supported on Docker Hub are 3.1 to 3.6, Hardener should start with 3.1 and go to the latest version - using ideally the same documentation and try to reuse the same Dockerfile hardening created across versions.

• Select a DoD Base Container Image Approved Sources to download the base image from, for each version. For example for alpine, Docker Hub could be used for each version: See https://hub.docker.com/_/alpine/.

• Setup a sub folder within DCCSCR for each new version that will contain the newly created Dockerfile and related scripts/documentation/LICENSE/README.md (see below). For example inside “Open-source” and inside “alpine” a sub folder “3.1” “3.2” etc. will be created per version.

• Within each version, starting with the lowest version supported, create a Dockerfile in the DCCSCR which will use the selected base image. The Dockerfile will use the FROM instruction that will download the base image. Additionally, the Dockerfile will have additional INSTRUCTIONS (commands) used to harden the container based on the Hardener decisions to mitigate findings and/or ensure proper DoD compliance. To know which instructions are required, the Hardener should know the product they’re hardening and make decisions based
on the Scanner findings (see below)

- It is imperative to reuse instructions as much as possible between versions so hardening can be similar across versions but it is also possible that new versions bring new features which can require additional hardening, which is why the Dockerfile must be reused from version minus 1 if it exists to avoid redundant work and leverage past work.

- Using the CI/CD orchestration tool, the DCCSCR folder content will be downloaded into the pipeline and the Dockerfile will be used to “build” the container using Docker. This can be done by create a custom Jenkinsfile for example if the CI/CD pipeline is using Jenkins.

- The CI/CD pipeline should launch the containers and ensure proper health and readiness. Most containers should have a health and readiness endpoint or provide some effective way to ensure the container is healthy and ready.

- The CI/CD pipeline will then run the required Container Hardening Scanners, scanning the Docker container image:
  - Twistlock (please leverage SCAP_ENABLED to true in Twistlock configuration whenever possible).
  - Anchore: feel free to use Anchore policies capabilities to create custom policies to enforce the Container Hardening Process. DoD (OSD) is working on creating custom policies as well including (CIS, health checks, root user, vulnerability scan, keys search, etc.).
  - Oscap (open source): should you have a ComplianceAsCode SCAP file (see: https://github.com/ComplianceAsCode) available for that product, it is imperative to use OpenSCAP (oscap-docker) command line to scan the containers for CVEs and compliance requirements. You can leverage OVAL, Data Streams or XCCDF files. Leverage the —remediation option whenever available. (This is part of the openscap-utils and scap-security-guide package)
  - Atomic scan (open source): leverage Atomic scan for automated CVE detection as well. Leverage the —remediation option whenever available.

- Based on the findings of the Container Hardening Scanners, the Hardener will add Dockerfile instructions to mitigate the findings as needed.
  - This can be disabling ports, enforcing TLS, disabling features and changing default configurations etc.
  - Additionally, containers must:
    - Whenever possible, emit logs to stdout
    - Not run with root user privileges or as root user (UID 0) (unless necessary for cybersecurity reason)
    - Leverage environment variables, configmaps, and secrets for configuration
    - It is imperative for any application/container that has User access (with a backend for example) to leverage CAC integration through SAML for example. If CAC integration is not possible, Multi Factor Authentication must be used but that will require a waiver and is NOT recommended.
    - Ensure that TLS and secure certificates are utilized to connect with other services/containers.
    - Should not have unnecessary packages in the container
    - Have setuid and setgid permissions removed
Be compliant with the DoD Hardened Containers Cybersecurity Requirements (see below);

- Once findings are acceptable or mitigated or risk accepted (through approved DoD Hardening Container Risk Acceptance (see below)), the container is signed and a checksum is created using SHA 256. This checksum will be stored in a file that will be stored into DCAR alongside the artifact. Those hashes will be used to ensure proper integrity of the containers.
- The entire content of the sub folder from the DCCSCR should be replicated into DCAR so all files can be found inside the DCAR folder as well, following the SAME folder structure than the DCCSCR.
- The CI/CD pipeline will then store the artifact and checksum inside the DoD Centralized Artifacts Repository (DCAR).
- Additionally, the Docker hardened image (artifact) will be stored on DCAR alongside the files from the DCCSRC copied and the artifact should be tagged with the product version AND tagged as “pre-production” as well until the artifact receives an authority to operate in which case it will then be tagged for “production”.
- Scripts can be used to automate launch or configuration of the containers as well as to facilitate adoption. Those scripts must be stored within the DCCSCR alongside the Dockerfile, in a sub folder called “scripts” and copied to DCAR as well. That being said, hardening should be done within the Dockerfile whenever possible.
- Similarly, documentation can be provided and will be stored on the DCCSCR in a sub folder called “documentation” and copied to DCAR following a successful build.
- Hardener should provide a README.md within the sub folder, per version, that will document how to use the hardened container and what Impact Level it is approved for. The existing documentation from the base container can be reused as well. This will be copied into the DCAR folder as well.
- All results from the Scanners (Twistlock and Anchore) should be copied into a sub folder “compliance” as HTML files into DCAR.
- The necessary Risk Management documentation created by the Hardener to document findings, false positives, accepted risks and current container risk posture should be stored into a sub folder called “accreditation” into DCCSCR which must be copied to DCAR as well.
- Additionally, whenever possible, the CHANGELOG should be added to the subfolder in the DCCSCR and DCAR.
- Additionally, examples can be added to help with the adoption of the containers, those should be stored in a sub folder called “examples” in the DCCSCR and copied to DCAR as well.
- It is important that each hardened container also has a LICENSE file alongside the artifact/Dockerfile to describe the LICENSE required to run the container, particularly for commercial products. It must be clear that a commercial license is needed to run these products from the DoD Programs. That LICENSE file should be duplicated on DCAR as well. The README.md should also indicate if a product requires a commercial license.
- Any changes in the DCCSCR folder should ensure files are copied into DCAR. If Dockerfile changed, the new artifact must be built and scanned to be sent to DCAR following process described above.
Example of a Hardened Container Folder Structure

Inside DoD Centralized Container Source Code Repository (DCCSCR):

- “oracle” (vendor name)
- “mysql” (product name)
  - “v.8.0” (product version)
    - “Dockerfile” (file using the base approved image and hardening the container by using Dockerfile INSTRUCTIONS.
    - “README.md” (Read me file to describe the container, warn if this a commercial product or not etc.)
    - "scripts" (scripts folder) (optional)
      - any scripts such as .sh or other
    - "documentation" (documentation folder) (optional)
      - any PDF used for documentation
    - "compliance" (compliance folder)
      - any SCAP file or OVAL or similar compliance file.
    - "accreditation" (accreditation folder) (optional)
      - PDF file created for Risk Management of the container
    - examples (examples folder) (optional)
      - Any documentation example that explains how to use the container
    - LICENSE (license text file describing license of the product, if commercial a POC must be listed for vendor questions)
    - CHANGELOG (contain changes from previous version)
  - v.5.7
    - COMPLETE SAME FOLDER STRUCTURE than v.8.0
  - v.5.6
    - COMPLETE SAME FOLDER STRUCTURE than v.8.0
  - v.5.5
    - COMPLETE SAME FOLDER STRUCTURE than v.8.0

Inside DoD Centralized Artifacts Repository (DCAR):

Bold show custom files created by CI/CD pipeline not present in DCCSCR.

- “oracle” (vendor name)
- “mysql” (product name)
  - “v.8.0” (product version)
    - “mysql-v.8.0-hardened-container.tar (tar version of the hardened container built”
    - “mysql-v.8.0-checksum” (checksum of the untar version of the container
    - “Dockerfile” (file using the base approved image and hardening the container by using Dockerfile INSTRUCTIONS.
    - “README.md” (Read me file to describe the container, warn if this a commercial product or not etc.)
"scripts" (scripts folder) (optional)
  • any scripts such as .sh or other
"documentation" (documentation folder) (optional)
  • any PDF used for documentation
"compliance" (compliance folder)
  • all HTML / PDF results from scanners
"accreditation" (accreditation folder) (optional)
  • PDF file created for Risk Management of the container
examples (examples folder) (optional)
  • Any documentation example that explains how to use the container
LICENSE (license text file describing license of the product, if commercial a POC must be listed for vendor questions)
CHANGELOG (contain changes from previous version)
  • v.5.7
    COMPLETE SAME FOLDER STRUCTURE than v.8.0
  • v.5.6
    COMPLETE SAME FOLDER STRUCTURE than v.8.0
  • v.5.5
    COMPLETE SAME FOLDER STRUCTURE than v.8.0

Automated Hardening of New Releases
The hardening process will be fully automated so that when new release is detected by the CI/CD pipeline, the pipeline should download new base images and run the steps mentioned above, triggered automatically, as soon as a new image is released. The pipeline should search for new base images and automatically run a new hardening pipeline and notify the team if a build fails to pass the scans.

Third Party Container Sharing
Third parties must provide complete Dockerfile and other necessary scripts or artifacts so they can be ingested into the DCCSCR and new folders can be created to ensure automation. Docker Binary files will not be accepted if Dockerfile aren’t provided so we can replicate build. The goal is to integrate existing Dockerfile into the DoD Hardening process and the DCCSCR so the new containers can benefit from the same automation.

DoD Hardened Containers Cybersecurity Requirements

Cybersecurity Requirements
  • Comply with initial and ongoing DoD Cybersecurity accreditation regulations / frameworks
    o Please make sure that the container base OS image that you are using is STIGed, assuming a DISA STIG is available. If the OS doesn’t have a STIG like Alpine for example, use the generic OS STIG.
    o NIST 800-53v4 moderate controls plus FedRAMP+ IL 6 controls
    o Risk Management Framework (RMF) process and required documentation
    o Containers must be compliant with NIST SP 800-190 and Center for Internet Security (CIS) Docker Benchmark
  • Documentation
Generate and automate necessary documentation for Risk Management
  - RMF Controls
  - Data Flows

Enable TLS on all PaaS tools that have UI or send data. HTTP will be disabled and FIPS 140.2 used.

Where authentication and authorization is required, leverage DoD ADFS CAC authentication.

Whenever possible, prohibit processes and containers from running as root.

**DoD Hardening Container Risk Acceptable**
Process created to accept specific risk of a cybersecurity findings that is not potentially fixable at the time. This will be case by case specific and will be reviewed by OSD A&S and DoD CIO.

**Active Directory / Common Access Card Integration Requirements**
Active Directory Domain Services (AD DS) provides a centralized, delegated administrative model and single sign-on (SSO) capabilities to an organization. Active Directory stores information about objects on the network and makes this information available for administrators and users to find and use. Active Directory uses a structured data store as the basis for a logical, hierarchical organization of directory information.

*It is imperative that any Impact Level 4 and above environment use CAC authentication for any UI based authentication (such as a backend or dashboard). For Non Person Entity (NPO) such as communication between services, signed certificates can be used and PKI can be leveraged.*

In addition, organizations need the ability to use Common Access Card (CAC) as the principal card to enable physical access to buildings and controlled spaces, and as well access to DoD computer network and systems.

The Single Sign On (SSO) using Common Access Card should be fully integrated with the DoD ADFS that is currently being used.
Appendix A

Security Control Inheritance (PaaS / Platform Host)

PaaS providers rely on technical concepts such as process isolation, service routing, redundancy, firewall controls, and many other security layers - typically aligning with DoD security concepts out-of-the-box.

Further security overlays should always be leverage (STIG, NIST 800-53, etc.). Ensure the selected host operating system has significant DISA STIG involvement, to include security guidance/standards available.

With a properly locked down hosting environment - containers inherit most of the security controls and benefits from infrastructure to host OS level remediation requirements. The focus then shifts to container security requirements and application security requirements (i.e. static code analysis, unit testing, library CVE testing, etc.).
Appendix B

Container Security/Remediation Considerations

Preferred approach: using OpenSCAP as it is the industry standard.

If a scan returns non-compliant result(s), always ensure to evaluate the validity of those findings. False positives are common within major host OS based containers - as the security profiles account for the all host-level controls that are likely non-applicable to a container build (i.e. GUI, CAC authentication, etc.). In addition to false positives many of the base operating system STIG requirements are not applicable in the containers.

If determined to be a true-positive, always remediate when able.

In case of false positives or non-applicable, you can either remediate to force the scanner to detect it as corrected, or preferably modify scanning profiles to mark specific components as non-applicable.

For example, there are 363 configuration settings applicable to RHEL 7 base OS. 93 of these controls are not inherited and or applicable. If any of the 93 controls need to be remediated 85 of them can be automated.

Always automate all container hardening processes leveraging CI/CD pipelines whenever possible. Manual processes should not be used.

Select a proper base image (see instructions under “Selecting the container base image” section of this document.

Example: If a container requires a base OS (such as RHEL/Alpine), the DoD hardened base image of these OS must be used.

Example: If a container needs JDK such as a Java application or a product using Java, that the JDK base image must be reused whenever possible. This ensures that hardening of JDK is centralized and consolidated across all containers by leveraging inheritance with base images.