Log 4.1: User Creation for Containers
- Log 4.1 Description: Enforce that containers do not run as the root user. Implement this by specifying a user with the USER directive in Dockerfiles or employing tools like gosu for command execution.
- Log 4.1 Remediation: Use the USER directive in Dockerfiles to define a non-root user or create a user with useradd before setting it with USER. For systems that must start as root, ensure processes switch to a non-root user via script execution.

Log 4.2: Trusted Base Images
- Log 4.2 Description: Containers should be built using trusted base images from secure sources. This can be base images you create or from reputable vendors.
- Log 4.2 Remediation: Use Docker Content Trust and conduct regular scans of images. Review the history and sources of base images to ensure their integrity and security.

Log 4.3: Minimize Installed Packages
- Log 4.3 Description: Minimize the attack surface by avoiding unnecessary packages in containers.
- Log 4.3 Remediation: Utilize minimal base images and carefully manage the inclusion of packages. Regularly review and justify the installed packages against business needs.

Log 4.4: Vulnerability Scanning and Patching
- Log 4.4 Description: Regularly scan container images for vulnerabilities and rebuild them with the necessary security patches.
- Log 4.4 Remediation: Use tools like Docker Scan or Trivy to assess vulnerabilities and update images accordingly. Re-instantiate containers from updated images to maintain security.

Log 4.5: Content Trust for Docker
- Log 4.5 Description: Enable Docker Content Trust to ensure that all operations using Docker images are verified and trusted.
- Log 4.5 Remediation: Set `DOCKER_CONTENT_TRUST=1` on all Docker operations to enforce content verification. This should be set in user profiles or Docker configuration files.

Log 4.6: HEALTHCHECK Instructions
- Log 4.6 Description: Add `HEALTHCHECK` instructions to Dockerfiles to automate the monitoring of container health.
- Log 4.6 Remediation: Implement health check commands within Dockerfiles to assess the operation of containers and define actions based on their state.

Log 4.7: Update Instructions in Dockerfiles
- Log 4.7 Description: Avoid using update commands (`apt-get update`, `yum update`) standalone in Dockerfiles to prevent using outdated packages.
- Log 4.7 Remediation: Combine update commands with installation commands and use the `--no-cache` flag during builds to ensure the latest packages are used.

Log 4.8: Setuid and Setgid Permissions
- Log 4.8 Description: Strip off setuid and setgid permissions from files within Docker images unless necessary.

- Log 4.8 Remediation: Use file permissions management commands in Dockerfiles to remove unnecessary setuid/setgid permissions, carefully preserving legitimate needs.

Log 4.9: Use COPY Over ADD
- Log 4.9 Description: Prefer `COPY` over `ADD` for transferring files into containers as it is safer and more predictable.
- Log 4.9 Remediation: Restrict Dockerfile commands to `COPY` for file transfer, avoiding `ADD` unless absolutely necessary for functionality like unpacking local tar files.

Log 4.10: Secrets in Dockerfiles
- Log 4.10 Description: Avoid embedding secrets in Dockerfiles to prevent exposure.
- Log 4.10 Remediation: Utilize Docker secrets management or external secrets management tools to handle sensitive data securely during the build process.

Log 4.11: Verified Packages Only
- Log 4.11 Description: Install only authenticated packages to ensure their integrity and origin are verified, typically using digital signatures like GPG.
- Log 4.11 Remediation: Utilize secure mechanisms, such as verifying GPG signatures of packages, to ensure the authenticity of packages before installation.

Log 4.12: Validating Signed Artifacts
- Log 4.12 Description: Artifacts, such as software packages and other digital entities, should have their cryptographic signatures verified before use.
- Log 4.12 Remediation: Implement processes to automatically verify signatures of all artifacts. This can be integrated into continuous integration/continuous deployment (CI/CD) pipelines.