opentext™

OpenText™ Fortify WebInspect and OAST on Docker

Software Version: 24.2.0
Windows® and Linux® Operating Systems

User Guide

Document Release Date: May 2024
Software Release Date: May 2024

Legal Notices

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Copyright Notice

Trademark Notices

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number
- Document Release Date, which changes each time the document is updated
- Software Release Date, which indicates the release date of this version of the software

This document was produced on May 14, 2024. To check for recent updates or to verify that you are using the most recent edition of a document, go to:

https://www.microfocus.com/support/documentation

Contents

Preface

Contacting Customer Support

Visit the Support website to:

- Manage licenses and entitlements
- Create and manage technical assistance requests
- Browse documentation and knowledge articles
- Download software
- Explore the Community

https://www.microfocus.com/support

For More Information

For more information about Fortify software products:

https://www.microfocus.com/cyberres/application-security

About the Documentation Set

The Fortify Software documentation set contains installation, user, and deployment guides for all Fortify Software products and components. In addition, you will find technical notes and release notes that describe new features, known issues, and last-minute updates. You can access the latest versions of these documents from the following Product Documentation website:

https://www.microfocus.com/support/documentation

To be notified of documentation updates between releases, subscribe to Fortify Product Announcements on the OpenText Fortify Community:

https://community.microfocus.com/cyberres/fortify/w/announcements

Fortify Product Feature Videos

You can find videos that highlight Fortify products and features on the Fortify Unplugged YouTube channel:

https://www.youtube.com/c/FortifyUnplugged

Change Log

The following table lists changes made to this document. Revisions to this document are published between software releases only if the changes made affect product functionality.

Software Release / Document Version	Changes
24.2.0	Added: • Information related to external SQL Server for Linux image version. See the following topics: • "Using an External SQL Server" on page 34 • "Configuring an External SQL Server using a Docker Compose File " on page 35 • "Configuring an External SQL Server using Helm Charts" on page 37 Updated: • Licensing information with new LIM URL format. See the following topics: • "Configuring the Environment File for CLI and API Modes" on page 17 • "Editing the Environment File for Linux" on page 28 • "Running Fortify WebInspect on Docker Windows Version with Fortify OAST" on page 53
23.2.0	Added: • Content for updating SecureBase in containers. See "Updating SecureBase in the Container" on page 39. Updated: • Docker image version numbers.
23.1.0	Updated: • Docker image version numbers.

Software Release / Document Version	Changes
22.2.0 / January 2023	Updated: • LIM settings in the Linux environment file. See "Editing the Environment File for Linux" on page 28.
22.2.0	Added: • Known limitations of WebInspect images. See "Known Limitations of WebInspect Images" on page 15. • Content related to the Linux versions. See "Using the Linux Versions" on page 26. Updated: • Overview content with information about Linux image. See "Fortify WebInspect and OAST on Docker" on page 10. • Operation modes to include Mode 0. See "Understanding the Operation Modes" on page 15. • Content related to OAST image with Red Hat version. See the following topics: ◦ "Using the OAST Docker Image" on page 43 ◦ "About the OAST Image" on page 45 ◦ "Pulling the Fortify OAST Image" on page 45 ◦ "Running the OAST Container" on page 48 • Image names with current tags throughout.

Chapter 1: Fortify WebInspect and OAST on Docker

Fortify engineers have created Fortify WebInspect images that are available for download on the Docker container platform. Windows and Red Hat image versions are available. For more information about available versions, see "Using the WebInspect Docker Images" on page 15.

Fortify OAST

Fortify engineers have created an out-of-band application security testing (OAST) server image that is available for download on the Docker container platform. The image enables you to configure local DNS service and is intended for use in networks that lack an Internet connection.

Tip: By default, Fortify WebInspect build 21.2.0.117 (or later) or Fortify WebInspect on Docker 21.2.0.118 (or later) use the OpenText public OAST server. For networks that have Internet access, configuring a local OAST infrastructure is not necessary.

Important! You must use the Fortify OAST image with Fortify WebInspect build 21.2.0.117 (or later) or Fortify WebInspect on Docker 21.2.0.118 (or later). Only these versions of Fortify WebInspect support Fortify OAST.

What is Docker?

Docker is a platform that facilitates creating, deploying, and running applications. Developers can package their application and all dependencies, including the platform and all its dependencies, into one logical package called a container or image. You can download a Docker image and run the application contained therein on a virtual machine (VM).

Benefits of Docker

Using a Docker image makes configuring the various prerequisite dependencies unnecessary, and can reduce the time it takes to deploy an instance of the application.

Docker is command-line driven, so it is easy to integrate into build processes, making Docker perfect for automation. As part of an automated build process, you can download a Fortify WebInspect image from the Docker repository, conduct a scan, and then remove the image from your VM.

For more information about Docker, visit https://www.docker.com.

Supported Docker Versions

Follow Docker recommendations for the Docker engine version to use for Windows, Red Hat Universal Base Image (UBI) 8.x x86_64, and Ubuntu 22.04 LTS x86_64 host operating systems.

Audience

This document is intended for users who are familiar with Fortify WebInspect, in particular its CLI and API, and the License and Infrastructure Manager (LIM). Users should also have experience installing, configuring, and using Docker.

Requesting Access to Fortify Docker Repository

Access to the Fortify Docker repository requires credentials and is granted through your Docker ID. To access the Fortify Docker repository, email your Docker ID to mfi-fortifydocker@opentext.com.

Setting Up Docker

Before you can run Docker containers, you must set up Docker according to the process described in the following table.

Stage	Description
1.	Download and install the appropriate Docker version on the host machine. **Note:** Follow Docker recommendations for the Docker engine version to use for Windows, Red Hat Universal Base Image (UBI) 8.x x86_64, and Ubuntu 22.04 LTS x86_64 host operating systems.
2.	Configure your machine for Docker containers.
3.	Register and start the Docker service.

For additional Docker documentation, see https://docs.docker.com/.

Related Documents

This topic describes documents that provide information about Fortify software products.

Note: You can find the Fortify Product Documentation at https://www.microfocus.com/support/documentation. Most guides are available in both PDF and HTML formats. Product help is available within the Fortify LIM product and the Fortify WebInspect products.

All Products

The following documents provide general information for all products. Unless otherwise noted, these documents are available on the Product Documentation website.

Document / File Name	Description
About Fortify Software Documentation About_Fortify_Docs_<*version*>.pdf	This paper provides information about how to access Fortify product documentation. **Note:** This document is included only with the product download.
Fortify Software System Requirements Fortify_Sys_Reqs_<*version*>.pdf	This document provides the details about the environments and products supported for this version of Fortify Software.
Fortify Software Release Notes FortifySW_RN_<*version*>.pdf	This document provides an overview of the changes made to Fortify Software for this release and important information not included elsewhere in the product documentation.
What's New in Fortify Software <version> Fortify_Whats_New_<*version*>.pdf	This document describes the new features in Fortify Software products.

Fortify ScanCentral DAST

The following document provides information about Fortify ScanCentral DAST. These documents are available on the Product Documentation website at https://www.microfocus.com/documentation/fortify-ScanCentral-DAST.

Document / File Name	Description
OpenText™ Fortify ScanCentral DAST Configuration and Usage Guide	This document provides information about how to configure and use Fortify ScanCentral DAST to conduct

Document / File Name	Description
SC_DAST_Guide_*<version>*.pdf	dynamic scans of Web applications.
OpenText™ Fortify License and Infrastructure Manager Installation and Usage Guide LIM_Guide_*<version>*.pdf	This document describes how to install, configure, and use the Fortify License and Infrastructure Manager (LIM), which is available for installation on a local Windows server and as a container image on the Docker platform.
OpenText™ Fortify WebInspect and OAST on Docker User Guide WI_Docker_Guide_*<version>*.pdf	This document describes how to download, configure, and use Fortify WebInspect and Fortify OAST that are available as container images on the Docker platform. The Fortify WebInspect image is intended to be used in automated processes as a headless sensor configured by way of the command line interface (CLI) or the application programming interface (API). It can also be run as a Fortify ScanCentral DAST sensor and used in conjunction with Fortify Software Security Center. Fortify OAST is an out-of-band application security testing (OAST) server that provides DNS service for the detection of OAST vulnerabilities.

Fortify WebInspect

The following documents provide information about Fortify WebInspect. These documents are available on the Product Documentation website at https://www.microfocus.com/documentation/fortify-webinspect.

Document / File Name	Description
OpenText™ Fortify WebInspect Installation Guide WI_Install_*<version>*.pdf	This document provides an overview of Fortify WebInspect and instructions for installing Fortify WebInspect and activating the product license.
OpenText™ Fortify WebInspect User Guide WI_Guide_*<version>*.pdf	This document describes how to configure and use Fortify WebInspect to scan and analyze Web applications and Web services. **Note:** This document is a PDF version of the Fortify WebInspect help. This PDF file is provided so you can easily print multiple topics from the help

Document / File Name	**Description**
	information or read the help in PDF format. Because this content was originally created to be viewed as help in a web browser, some topics may not be formatted properly. Additionally, some interactive topics and linked content may not be present in this PDF version.
OpenText™ Fortify WebInspect and OAST on Docker User Guide WI_Docker_Guide_<*version*>.pdf	This document describes how to download, configure, and use Fortify WebInspect and Fortify OAST that are available as container images on the Docker platform. The Fortify WebInspect image is intended to be used in automated processes as a headless sensor configured by way of the command line interface (CLI) or the application programming interface (API). It can also be run as a Fortify ScanCentral DAST sensor and used in conjunction with Fortify Software Security Center. Fortify OAST is an out-of-band application security testing (OAST) server that provides DNS service for the detection of OAST vulnerabilities.
OpenText™ Fortify License and Infrastructure Manager Installation and Usage Guide LIM_Guide_<*version*>.pdf	This document describes how to install, configure, and use the Fortify License and Infrastructure Manager (LIM), which is available for installation on a local Windows server and as a container image on the Docker platform.
OpenText™ Fortify WebInspect Tools Guide WI_Tools_Guide_<*version*>.pdf	This document describes how to use the Fortify WebInspect diagnostic and penetration testing tools and configuration utilities packaged with Fortify WebInspect and Fortify WebInspect Enterprise.
OpenText™ Fortify WebInspect Agent Installation and Rulepack Guide WI_Agent_Install_<*version*>.pdf	This document describes how to install the OpenText™ Fortify WebInspect Agent and describes the detection capabilities of the Fortify WebInspect Agent Rulepack Kit. Fortify WebInspect Agent Rulepack Kit runs atop the Fortify WebInspect Agent, allowing it to monitor your code for software security vulnerabilities as it runs. Fortify WebInspect Agent Rulepack Kit provides the runtime technology to help connect your dynamic results to your static ones.

Chapter 2: Using the WebInspect Docker Images

The following paragraphs describe the naming convention of the Fortify WebInspect images on Docker, known limitations of using the Fortify WebInspect Docker images, and the operation modes for the images in containers.

Image Naming Convention

The Fortify Docker repository uses the following naming convention for the Fortify WebInspect Windows version image:

`fortifydocker/webinspect:`*`<version>`*

The Fortify Docker repository uses the following naming convention for the Fortify WebInspect Linux version images:

`fortifydocker/dast-scanner:`*`<version.linux_os_version>`*

Tip: Although the Linux version image includes `dast-scanner` in its names, you can use the image by way of CLI commands or the API outside of an OpenText™ Fortify ScanCentral DAST environment.

For more information about the versions that are available, refer to the Readme file in the fortifydocker/webinspect repository.

Known Limitations of WebInspect Images

The following known limitations apply when using a Fortify WebInspect Docker image:

- Currently, the Fortify WebInspect Docker images do not support Kerberos authentication. Therefore, you cannot conduct a scan on a network that requires Kerberos authentication.
- When using client certificates for authentication on the Fortify WebInspect Linux versions, only `.pfx` certificates are supported and the certificates must be exportable.

Understanding the Operation Modes

The Fortify WebInspect images can run in one of five operation modes in a container as described in the following table.

Mode	Description
`0`	**Self-deploy Configuration mode.** Use this mode to unpack the environment file and Docker compose files needed to configure and run the Linux versions of the Fortify WebInspect image. **Note:** This mode applies only to the Linux versions.
`1`	**WebInspect CLI mode.** Use this mode to conduct scans using options available in the command-line interface. For an entire list of CLI options, see the "Command Line Execution" topic in the *OpenText™ Fortify WebInspect User Guide*.
`2`	**WebInspect API mode.** Use this mode to conduct scans using the endpoints available in the Fortify WebInspect REST API. After the Docker container starts, you can navigate to the following URL to browse the Swagger documentation from your local machine: http://<*hostname*>:8083/webinspect/swagger/docs/v1 If you map ports from the container to the host machine as shown in the Docker run command, you can access it using localhost as <*hostname*>. Otherwise, use the IP address of the Docker host machine.
`3`	**ScanCentral DAST mode.** Use this mode to conduct scans from the ScanCentral DAST user interface in OpenText™ Fortify Software Security Center. For more information about Fortify ScanCentral DAST, see *OpenText™ Fortify ScanCentral DAST Configuration and Usage Guide*. **Note:** This description is provided for informational purposes only. You do not configure an environment file for this mode. For more information, see "Running the Windows Container in ScanCentral DAST Mode" on page 22.
`4`	**ScanCentral DAST Utility Service mode.** Use this mode to run the Fortify WebInspect image as a ScanCentral DAST Utility Service container. **Note:** This description is provided for informational purposes only. You do not configure an environment file for this mode. You pull this image and start the container using either the Docker compose file or one of the PowerShell scripts that the Fortify ScanCentral DAST Configuration Tool CLI generates. For more information, see *OpenText™ Fortify ScanCentral DAST Configuration and Usage Guide*.

Using the Windows Version

The Windows image includes the full version of Fortify WebInspect 24.2.0 software, but is intended to be used in automated processes as a headless scanner configured by way of the command line interface (CLI) or the application programming interface (API). It can also be run as a Fortify ScanCentral DAST sensor and used in conjunction with Fortify Software Security Center.

This release of Fortify WebInspect 24.2 image is available in Windows version 1809.

Important! Before you can run the Fortify WebInspect image, you must install Microsoft update KB4561608 on the host machine. For more information, see https://support.microsoft.com/en-us/topic/june-9-2020-kb4561608-os-build-17763-1282-437af506-e3ef-a8a1-09e7-26cc94e509c7.

The Fortify WebInspect 24.2 image includes the SQL Server 2017 Express edition database.

Pulling a Windows Image for API and CLI Modes

After starting the Docker service and requesting access to the private Fortify WebInspect repository on Docker Hub, you can pull a Windows image of Fortify WebInspect from the Fortify Docker repository as described in this topic.

Note: Instructions for pulling an image apply only when running the container in API and CLI modes. To pull an image when running the container in ScanCentral DAST Mode, see "Running the Windows Container in ScanCentral DAST Mode" on page 22.

To pull the current version of the Fortify WebInspect image:

- In PowerShell, enter the following command:

```
docker pull fortifydocker/webinspect:24.2
```

Configuring the Environment File for CLI and API Modes

After you download a Fortify WebInspect image from the Docker repository, you must configure an environment (`.env`) file that defines how the image will operate. For more information, see https://docs.docker.com/compose/env-file.

In the environment file, configure the operation mode, licensing (if required), and options as described in the following sections.

Configuring the Operation Mode (Required)

You must specify a mode for the image. For more information about the modes, see "Understanding the Operation Modes" on page 15.

In the environment file, specify the operation mode as follows:

```
# WebInspect Container Mode
```

`mode=<number>`

The following example sets the image to run in WebInspect CLI mode:

```
# WebInspect Container Mode

mode=1
```

Configuring Licensing (Required for CLI and API Modes)

You must configure licensing for the image when running in CLI and API modes. Currently, licensing must be handled by a License and Infrastructure Manager (LIM). In the environment file, type the following information for your LIM installation to configure licensing for this instance of Fortify WebInspect:

```
# Licensing

limURL=http://<server-url>:<port>

limPool=<LIM_pool>

limPswd=<LIM_password>
```

Note: If using a version of the LIM prior to 24.2.0, the format is `https://<server-url>:<port>/<service-directory>` where:

- *server-url* is the site specified during LIM initialization as the root web site.
- *service-directory* is the directory specified during LIM initialization as the Service Virtual Directory name (the default is “LIM.Service” or "LIM.API").

For more information about using the LIM, see the *OpenText™ Fortify License and Infrastructure Manager Installation and Usage Guide*.

Configuring CLI Mode Options

You must configure CLI options to use WebInspect CLI mode. You can configure any of the available CLI options as scan arguments in the environment file. For the complete list of CLI options, see the "Command Line Execution" topic in the *OpenText™ Fortify WebInspect User Guide*.

In the environment file, type the following to configure the CLI options to use in the scan. Substitute *<options>* with your specific options:

```
# WebInspect CLI scan options

scanArgs=<options>
```

The following example performs a crawl-only scan of zero.webappsecurity.com and exports the results to the `zero.scan` file:

```
# WebInspect CLI scan options

scanArgs=-u http://zero.webappsecurity.com -c -es zero.scan
```

Sample CLI Environment File

The following is a sample environment file for WebInspect CLI mode to run a full audit:

```
#!-- WebInspect Docker Mode. --!
#!-- Sample configuration for CLI mode. --!

# 1 = CLI mode
mode=1

# Licensing
limURL=http://<server-url>:<port>
limPool=<LIM_pool>
limPswd=<LIM_password>

# WebInspect options - for use in scan mode
# Full audit
scanArgs=-u http://zero.webappsecurity.com -es c:\host\zero.scan

# Full audit with macro
#scanArgs=-u http://zero.webappsecurity.com -xd -es c:\host\zero.scan -
macro c:\host\zero_macro.webmacro

# Crawl only
#scanArgs=-u http://zero.webappsecurity.com -es c:\host\zero.scan -c

# Full audit with settings file and reporting
#scanArgs=-u http://zero.webappsecurity.com -s c:\host\Settings.xml -r
Vulnerability -y Standard -f c:\host\Report -gp -es c:\host\zero.scan
```

The full audit with macro, crawl only, and full audit with settings file and reporting examples are commented out in this sample file.

Configuring API Mode Options

You must configure API options to use WebInspect API mode. To conduct a scan that uses the Fortify WebInspect API, you must provide the host, port, and authentication type parameters for the API server as described in the following table.

Parameter	Description
`RCServerHost`	Specifies the hostname that the WebInspect API Server should listen on. Use `+` for all.

Parameter	Description
`RCServerPort`	Specifies the WebInspect API Server port to listen on.
`RCServerAuthType`	Specifies the WebInspect API Server authentication type. The value can be one of the following: • `None` • `Basic` • `NTLM` • `ClientCert`

In the environment file, provide the details for your Fortify WebInspect REST API using the following parameters:

```
# WebInspect API

RCServerHost=<hostname>

RCServerPort=<port_number>

RCServerAuthType=<auth_type>
```

Sample API Environment File

The following is a sample environment file for WebInspect API mode:

```
#!-- WebInspect Docker Mode. --!
#!-- Example configuration for API mode. --!

# 2 = WebInspect API mode
mode=2

# Licensing
limURL=http://<server-url>:<port>
limPool=<LIM_pool>
limPswd=<LIM_password>

# WebInspect API settings
RCServerHost=+
RCServerPort=8083

# RCServerAuthType: None, Basic, NTLM, ClientCert
RCServerAuthType=None
```

What's next?

After you have configured and saved your environment file, you can run the image in a container. Go to "Running the Windows Container in CLI and API Modes" below.

Running the Windows Container in CLI and API Modes

This topic provides a sample Docker run command for the Windows version of WebInspect in the CLI and API modes. The Docker run command uses CLI options that define the container's resources at runtime. To understand how the Docker CLI options used in the samples determine how the container is run, see "Understanding the Docker CLI Options" below.

Note: If proxy settings are required, see "Using Proxy Settings in Windows" on page 25.

Sample Docker Run Command for CLI Mode

The following example uses Docker CLI options to run the container in CLI mode:

```
docker run -d --rm -v c:/scans:c:/host --env-file ScanMode.env --memory=16g
--cpus=4 --name webinspect fortifydocker/webinspect:24.2
```

For more information about image filenames and version numbers, see "Pulling a Windows Image for API and CLI Modes" on page 17.

Sample Docker Run Command for API Mode

The following example uses Docker CLI options to run the container in API mode:

```
docker run -d --rm -p 8083:8083 --env-file APIMode.env --memory=16g --
cpus=4 --name webinspect_api fortifydocker/webinspect:24.2
```

For more information about image filenames and version numbers, see "Pulling a Windows Image for API and CLI Modes" on page 17.

Understanding the Docker CLI Options

The following table describes the Docker CLI options used in "Sample Docker Run Command for CLI Mode" above and "Sample Docker Run Command for API Mode" above.

Option	**Description**
`-d`	Runs the container in the background and displays the container ID.
`--cpus`	Specifies the number of CPUs to allocate to the container. We recommend 2 CPUs.

Option	Description
`--env-file`	Identifies the `.env` file to use. For more information, see "Configuring the Environment File for CLI and API Modes" on page 17.
`--memory`	Specifies the amount of memory to allocate to the container. We recommend 16 GB.
`-p`	Maps a port inside the container to a port on the host system. **Important!** This option is required when using WebInspect API mode.
`--rm`	Automatically removes the container when it exits.
`-v`	Maps the volume (or folder) from the container to a folder on the host system. Separate multiple folder names with a colon.

Tip: For more information and a complete list of Docker run options, see https://docs.docker.com/engine/reference/commandline/run.

Running the Windows Container in ScanCentral DAST Mode

The ScanCentral DAST Configuration Tool CLI creates and downloads the following PowerShell scripts that you can use to pull and start a new sensor container:

- `pull-and-start-sensor-container.ps1` - This PowerShell script pulls the Fortify WebInspect image from Docker Hub, and then starts the container.
- `pull-sensor-image.ps1` - This PowerShell script pulls the Fortify WebInspect image from Docker Hub, but does not start the container.
- `start-sensor-container.ps1` - This PowerShell script starts the Fortify WebInspect container, but does not pull the image.

You can find these files in the `dast-windows-start.zip` file along with the other ScanCentral DAST launch artifacts. For more information about the ScanCentral DAST Configuration Tool CLI, see the *OpenText™ Fortify ScanCentral DAST Configuration and Usage Guide*.

Sample Script

The script you use should be similar to the following example:

```
docker run -d --restart always --name scancentral-dast-sensor --hostname
   "<MyHostName>"
   -e "mode=3" -e "RCServerHost=+" -e "RCServerPort=8089"
   -e "RCServerUseHTTPS=false" -e "RCServerAuthType=none"
```

```
    -e "DASTApiRootUrl=http://<IP_Address>:<Port>/api"
    -e "AllowNonTrustedServerCertificate=true"
    -e "ServiceToken=QgitxRErVP5Eh7hr2Bnuig=="
    -e "ScannerDescription=<MyDASTSensorName>"
    -e "ScannerPoolId=0" --memory=8g --cpus=2 fortifydocker/webinspect:24.2
```

About the ServiceToken

The `ServiceToken` is the encrypted "Sensor Service Token" that is set by the administrator using the ScanCentral DAST Configuration tool. This value should be protected.

Caution! A change to the `ServiceToken` value by the configuration tool requires all sensor containers to be updated with the new value.

About the ScannerDescription

The `ScannerDescription` enables you to provide a description for the sensor service. This value is displayed in the Description column of the Sensors list table on the ScanCentral DAST tab in Fortify Software Security Center. The description is shown in the sample script as an environment variable, but you can also provide it in the `appsettings.json` file.

About the ScannerPoolId

The `ScannerPoolId` is the sensor pool ID number in Fortify Software Security Center. If you need to hand-edit the `ScannerPoolId` in your script, you can find the sensor pool ID number in Fortify Software Security Center on the **SCANCENTRAL** > **DAST** > **Sensor Pools** page. Select the sensor pool in the list and view the Pool ID in the detail panel.

Tip: Setting `ScannerPoolId` to `0` automatically allocates the sensor to the Default pool.

Using PowerShell Scripts

These PowerShell scripts offer the following options:

- Use one script to pull the Fortify WebInspect image and then start the container.
- Use two scripts: one to pull the image, and then another to start the container.

You use the script or scripts on the host where you want to run the Fortify WebInspect container.

Using One Script

Use the following process to use a single PowerShell script to pull the image and start the container.

Stage	Description
1.	Copy the `pull-and-start-sensor-container.ps1` file to the host where you want to run the Fortify WebInspect container.
2.	On this same host, start Windows PowerShell ISE as Administrator. For more information about using PowerShell, refer to your Windows PowerShell documentation.
3.	To avoid errors regarding non-digitally signed scripts, run the contents of the `pull-and-start-sensor-container.ps1` script: 1. Copy the contents from the `pull-and-start-sensor-container.ps1` script. 2. Paste the contents in the PowerShell ISE script pane. 3. Click the **Run Selection** icon. **Note:** Alternatively, you can set the execution policy to allow all scripts, and then run the script as follows: `& "<drive>:<path_to_script>\pull-and-start-sensor-container.ps1"` For more information about setting the execution policy, refer to your Windows PowerShell documentation. The Fortify WebInspect image is pulled and the container is started.

Using Two Scripts

Use the following process to use separate pull and start PowerShell scripts.

Stage	Description
1.	Copy the following files to the host where you want to run the Fortify WebInspect container: • `pull-sensor-image.ps1` • `start-sensor-container.ps1`
2.	On this same host, start Windows PowerShell ISE as Administrator. For more information about using PowerShell, refer to your Windows PowerShell documentation.
3.	Pull the image. To avoid errors regarding non-digitally signed scripts, run the contents of the `pull-sensor-image.ps1` script:

Stage	Description
	1. Copy the contents from the `pull-sensor-image.ps1` script. 2. Paste the contents in the PowerShell ISE script pane. 3. Click the **Run Selection** icon. **Note:** Alternatively, you can set the execution policy to allow all scripts, and then run the script as follows: `& "`*`<drive>`*`:`*`<path_to_script>`*`\pull-sensor-image.ps1"` For more information about setting the execution policy, refer to your Windows PowerShell documentation. The Fortify WebInspect image is pulled.
4.	Start the container. To avoid errors regarding non-digitally signed scripts, run the contents of the `start-sensor-container.ps1` script: 1. Copy the contents from the `start-sensor-container.ps1` script. 2. Paste the contents in the PowerShell ISE script pane. 3. Click the **Run Selection** icon. **Note:** Alternatively, if you set the execution policy to allow all scripts as described in Stage 3, you can run the script as follows: `& "`*`<drive>`*`:`*`<path_to_script>`*`\start-sensor-container.ps1"` The Fortify WebInspect container is started.

Using ScanCentral DAST

When using the Fortify WebInspect sensor in ScanCentral DAST Mode, you manage the sensor, configure scan settings, and conduct scans in ScanCentral DAST. For more information about using ScanCentral DAST, see the *OpenText™ Fortify ScanCentral DAST Configuration and Usage Guide*.

Using Proxy Settings in Windows

You cannot pass proxy settings directly to the WebInspect Windows container through command line arguments or in the .env file. However, you can use the following process to use proxy settings for a scan.

Stage	Description
1.	Create a custom WebInspect settings file that includes the proxy settings.
2.	Save the file on the Docker host machine.

3.	Use the following options:
	• The `-s` WebInspect CLI option as a scan argument (`scanArgs`) in the .env file to pass the settings file, as shown in the following example:
	`scanArgs=-u http://zero.webappsecurity.com/`
	`   -s c:\host\CustomSettings.xml -es c:\host\zero.scan -xd`
	• The `-v` Docker CLI option in the Docker run command to map the folder with the settings to a folder in the container, as shown in the following example:
	`docker run -v c:/widocker:c:/host --env-file config.env`
	`   fortifydocker/webinspect`

Using the Linux Versions

The Linux image is available for the Ubuntu Linux distribution and for the Red Hat Linux distribution. Each image launches the following possible containers:

- wi application for scan logic (also called a scanner)
- Datastore for scan data
- WebInspect script engine (WISE) for JavaScript execution and Web Macro Recorder macro playbacks

  Note: This WISE does not automatically scale for scan scaling. It is a single script engine that runs as a sibling container for the sensor. For more information about scan scaling, see the *OpenText™ Fortify ScanCentral DAST Configuration and Usage Guide*.

- 2FA server to synchronize two-factor authentication requests (used only if the scan is configured to playback a two-factor authentication login macro). For more information about conducting scans using two-factor authentication, see the following documents:
  - *OpenText™ Fortify ScanCentral DAST Configuration and Usage Guide*
  - *OpenText™ Fortify WebInspect User Guide*
  - *OpenText™ Fortify WebInspect Tools Guide*

Process Overview for Using Linux Version

The Fortify WebInspect Linux version image uses a Docker compose YAML file to configure and start the sensor container and its auxiliary containers. An environment file and several Docker compose file templates are embedded in the scanner image with logic for unpacking the configurations onto the Docker host. You can then modify the environment file as needed and start the required mode for the sensor container.

The following table describes the process.

Stage	Description
1.	Pull the Docker image. For more information, see "Pulling a Linux Image for API and CLI Modes" below.
2.	Unpack the environment file and Docker compose files. For more information, see "Unpacking the Configuration Files for Linux" below.
3.	Edit the environment file. For more information, see "Editing the Environment File for Linux" on the next page.
4.	Run the image in a container. For more information, see the following topics: • "Running the Linux Container in API or CLI Mode" on page 30 • "Running the Linux Container in ScanCentral DAST Mode" on page 32

Pulling a Linux Image for API and CLI Modes

After starting the Docker service and requesting access to the private Fortify WebInspect repository on Docker Hub, you can pull a Linux image of Fortify WebInspect from the Fortify Docker repository as described in this topic.

Note: Instructions for pulling an image apply only when running the container in API and CLI modes. To pull an image when running the container in ScanCentral DAST Mode, see "Running the Windows Container in ScanCentral DAST Mode" on page 22.

To pull the current version of the Fortify WebInspect image:

- At the terminal prompt on the Red Hat Linux Docker host machine, enter the following command:

```
docker pull fortifydocker/dast-scanner:24.2.ubi.8
```

Unpacking the Configuration Files for Linux

An environment file, Helm charts, and several Docker compose (YAML) file templates are embedded in the scanner image. You must use the Docker run command in Mode 0 to unpack the files and copy them to the configuration directory on the Docker host. Mode 0 unpacks the files to the configuration directory and displays instructions on how to start the Fortify WebInspect scanner.

To unpack the compose file:

- At the terminal prompt on the Red Hat Linux Docker host machine, enter the following commands:

```
mkdir -p "$HOME/widocker" && \
docker run -e "WI_MODE=0" -v "$HOME/widocker:/etc/wi/docker-configs" \
--rm fortifydocker/dast-scanner:24.2.ubi.8
```

About the Helm Charts

You can use the Helm charts for deployment in Kubernetes. For more information about using Helm deployment, see the `readme.txt` file that is included in the Helm charts.

About the YAML Files

Several YAML files are included in the Linux images. The `docker-compose.yaml` file is the base Docker compose file, and it configures and starts the Linux sensor in Mode 2. The `docker-compose.mode3.yaml` file is an extension of the first YAML file. It removes the LIM configuration, adds configuration for ScanCentral DAST, and starts the Linux sensor in Mode 3. The `docker-compose.mode4.yaml` file is an extension of the first YAML file, and starts the container in Mode4. For more information about these modes, see "Understanding the Operation Modes" on page 15.

Advanced users may modify these files with additional configuration changes, such as for more complex network configurations. However, these changes are not described in this document.

Editing the Environment File for Linux

If you plan to use the scanner in API or CLI Mode, then you must edit the LIM settings in the environment file. You can also change the scanner port that is exposed. The default port setting is `8089`.

If you plan to use the scanner in ScanCentral DAST Mode, then you must edit the ScanCentral DAST API URL and service token in the environment file.

Tip: You can locate the environment file at `~/widocker`. A `readme.txt` file in the same location provides instructions on how to configure and run the WebInspect scanner container.

Editing for Mode 1 or 2

To edit the LIM information in the environment file for Mode 1 or 2:

1. At the terminal prompt on the Linux Docker host machine, enter the following command:
   ```
   cd ~/widocker
   ```
2. Enter the following command:
   ```
   nano .env
   ```

The environment file is opened for editing.

3. Locate the following lines:

```
#[5]. License and Infrastructure Manager (LIM):
# Applicable for WI_MODE 1 and 2
LIM=http://<HOST_OR_IP:PORT>/Lim.API/
LIM_POOL_NAME=<NAME>
LIM_POOL_PSWD=<PASSWORD>
```

4. Continue according to the following table:

For this variable...	Provide this information...
`LIM`	The LIM URL, which uses the format `https://`*`<server-url>`*`:`*`<port>`*. **Note:** If using a version of the LIM prior to 24.2.0, the format is `https://`*`<server-url>`*`:`*`<port>`*`/`*`<service-directory>`* where: • *server-url* is the site specified during LIM initialization as the root web site. • *service-directory* is the directory specified during LIM initialization as the Service Virtual Directory name (the default is "LIM.API").
`LIM_POOL_NAME`	The name of the LIM pool to use for licensing the scanner.
`LIM_POOL_PSWD`	The password for the LIM pool.

For more information about using the LIM, see the *OpenText™ Fortify License and Infrastructure Manager Installation and Usage Guide*.

5. Optionally, if you need to change the scanner port that is exposed, locate in the following line in the `#[2]. Exposed services configuration:` section of the environment file and change the port setting:

```
FORTIFY_SCANNER_EXPOSED_PORT=8089
```

6. Optionally, if you are using the Linux image in conjunction with a local OAST server, locate the following lines and provide the local WebInspect OAST domain name:

```
#[4]. Local Fortify OAST server domain name (optional):
FORTIFY_LOCAL_OAST_SERVER=
```

For more information about OAST, see "Using the OAST Docker Image" on page 43.

7. Save your edits.

Editing for ScanCentral DAST Mode

To edit the ScanCentral DAST information in the environment file:

1. At the terminal prompt on the Linux Docker host machine, enter the following command:
   ```
   cd ~/widocker
   ```
2. Enter the following command:
   ```
   nano .env
   ```
   The environment file is opened for editing.
3. Locate the following lines:
   ```
   #[6]. WebInspect Scanner DAST Sensor:
   # Applicable for WI_MODE 3
   DAST_API_ROOT_URL=http://<HOST_OR_IP:PORT>
   DAST_SERVICE_TOKEN=<TOKEN>
   DAST_SCANNER_POOL_ID=0
   DAST_SCANNER_TYPE=Fixed
   DAST_INSECURE=true
   ```
4. Continue according to the following table:

For this variable...	Provide this information...
`DAST_API_ROOT_URL`	The URL and port where the DAST API service runs.
`DAST_SERVICE_TOKEN`	The shared secret that all DAST sensors must use to authenticate with the DAST API. **Tip:** You can find this token in the `service-token.txt` file that is part of the ScanCentral DAST launch artifacts.

   For more information about the ScanCentral DAST launch artifacts, see the *OpenText™ Fortify ScanCentral DAST Configuration and Usage Guide*.
5. Save your edits.

Running the Linux Container in API or CLI Mode

After you have edited the environment file with the required changes for Mode 1 or 2, you can use the Docker compose file to start the four Linux containers that comprise the Fortify WebInspect sensor and create the internal communication network for the containers.

To start the containers:

- At the terminal prompt on the Linux Docker host machine, enter the following command:
  ```
  docker compose up -d
  ```

The database, WISE, scanner, and 2FA server containers are started, and the internal communication network is created.

Viewing the Running Containers

By default, the scanner container name is `widocker-scanner-1`. The prefix `widocker-` is the default directory where the Docker compose YAML files are unpacked. If you unpack the files to a different directory, the prefix will not be `widocker-`. The suffix `-1` is automatically applied by Docker to allow multiple instances of the container to be started from the Docker compose file. Fortify does not support starting the same container multiple times, so the suffix is always `-1`.

To view the running containers:

- At the terminal prompt on the Linux Docker host machine, enter the following command:

  ```
  docker ps
  ```

  The list of containers should include the following names:

  ```
  widocker-scanner-1
  widocker-datastore-1
  widocker-wise-1
  widocker-twofa-1
  ```

Viewing the Scanner Console Logs

Optionally, you can check the scanner console logs to troubleshoot issues with the containers.

To check the logs:

- At the terminal prompt on the Linux Docker host machine, enter the following command:

  ```
  docker compose logs <container_name>
  ```

Tip: When using the default YAML files, you can use `scanner` as the container name without the prefix or suffix as described in "Viewing the Running Containers" above.

```
docker compose logs scanner
```

Using the Exposed Scanner Port for CLI Mode

You can use the exposed Fortify WebInspect scanner port number to configure and run scans by way of `wi.exe`. The following example shows how to conduct a scan using `wi.exe` on the scanner container:

```
docker exec <container_name> wi -u <url>
```

For more information about using `wi.exe`, see *OpenText™ Fortify WebInspect User Guide*.

Accessing the Scanner Swagger UI

To access the Fortify WebInspect Swagger UI of the scanner container, you must know the IP address for the exposed interface.

To view the IP addresses and locate the interface for the Swagger UI:

1. At the terminal prompt on the Linux Docker host machine, enter the following command:
   ```
   ifconfig
   ```
   A list of interfaces appears.
2. Select the IP address for the scanner container.
3. Copy and paste the IP address into a browser using the following format:
   ```
   <ip_address>:<port>/webinspect/swagger/index.html
   ```

For more information about using the Fortify WebInspect Swagger UI, see *OpenText™ Fortify WebInspect User Guide*.

Running the Linux Container in ScanCentral DAST Mode

The ScanCentral DAST Configuration Tool CLI creates and downloads the following bash shell scripts that you can use to pull the image, start the sensor containers, and create the internal communication network:

- `pull-and-start-sensorcontainer.sh` - This shell script pulls the Fortify WebInspect Linux image from Docker Hub, and then starts the database, WISE, scanner, and 2FA server containers, and creates the internal communication network.
- `pull-sensor-image.sh` - This shell script pulls the Fortify WebInspect Linux image from Docker Hub, but does not start the containers or create the internal communication network.
- `start-sensor-container.sh` - This shell script starts the database, WISE, scanner, and 2FA server containers, and creates the internal communication network, but does not pull the Fortify WebInspect Linux image.

You can find these files in the `dast-linux-start.tar.gz` file along with the other ScanCentral DAST launch artifacts. For more information about the ScanCentral DAST Configuration Tool CLI, see the *OpenText Fortify ScanCentral DAST Configuration and Usage Guide*. For information about executing bash scripts, refer to your Linux distribution documentation.

Alternatively, you can use the `docker-compose.mode3.yaml` file that you unpacked from the scanner image as described in "Using the Compose File" below.

Using the Compose File

After you have edited the environment file with the required changes for Mode 3, you can use the Docker compose file to start the four Linux containers that comprise the Fortify WebInspect sensor and create the internal communication network for the containers.

To start the containers:

- At the terminal prompt on the Linux Docker host machine, enter the following command:
  ```
  docker compose -f docker-compose.yaml -f docker-compose.mode3.yaml up -d
  ```

The database, WISE, scanner, and 2FA server containers are started, and the internal communication network is created.

Viewing the Running Containers

By default, the scanner container name is `widocker-scanner-1`. The prefix `widocker-` is the default directory where the Docker compose YAML files are unpacked. If you unpack the files to a different directory, the prefix will not be `widocker-`. The suffix `-1` is automatically applied by Docker to allow multiple instances of the container to be started from the Docker compose file. Fortify does not support starting the same container multiple times, so the suffix is always `-1`.

To view the running containers:

- At the terminal prompt on the Linux Docker host machine, enter the following command:

  ```
  docker ps
  ```

  The list of containers should include the following names:

  ```
  widocker-scanner-1
  widocker-datastore-1
  widocker-wise-1
  widocker-twofa-1
  ```

Viewing the Scanner Console Logs

Optionally, you can check the scanner console logs to troubleshoot issues with the containers.

To check the logs:

- At the terminal prompt on the Linux Docker host machine, enter the following command:

  ```
  docker compose logs <container_name>
  ```

Tip: When using the default YAML files, you can use `scanner` as the container name without the prefix or suffix as described in "Viewing the Running Containers" above.

```
docker compose logs scanner
```

Using ScanCentral DAST

When using the Fortify WebInspect sensor in ScanCentral DAST Mode, you manage the sensor, configure scan settings, and conduct scans in ScanCentral DAST. For more information about using ScanCentral DAST, see the *OpenText™ Fortify ScanCentral DAST Configuration and Usage Guide*.

Stopping the Linux Containers

When you have finished conducting scans, you can stop the containers and remove them.

To stop the containers:

- At the terminal prompt on the Linux Docker host machine, enter the following command:

  ```
  docker compose down
  ```
  The database, WISE, scanner, and 2FA server containers are stopped. The containers and wi_net communications network are removed.

Using Proxy Settings in Linux

You cannot pass proxy settings directly to the WebInspect Linux container through command line arguments or in the .env file. However, you can use the following process to use proxy settings for a scan.

Stage	Description
1.	Create a custom WebInspect settings file that includes the proxy settings.

2.	Do one of the following:
	• Use the WebInspect API to upload the scan settings with proxy.
	• Use scan setting overrides that apply proxy settings for the scan.
	• Copy the scan settings into scanner container as shown in the following example:
	`docker cp <directory_path>/<scan-settings>.xml widocker-scanner-1:/etc/wi/.widata/shared/Settings`
	Use the scan settings in the command line as follows:
	`docker exec widocker-scanner-1 wi -s /etc/wi/.widata/shared/Settings/scan-settings.xml`

Using an External SQL Server

Using an external SQL Server is an optional configuration, depending on the operation mode of the sensor container.

External SQL Server Not Recommended for ScanCentral DAST Mode

By default, ScanCentral DAST moves each completed scan into the ScanCentral DAST scan database, which is outside of the sensor and removes everything from the sensor container. In this mode, the local SQL Express database does not accumulate any scan results, but instead works like a temporary cache for the current scan, which is purged automatically when the scan is completed.

Note: ScanCentral DAST configuration includes an optional setting that enables you to save scans in the sensor container. Even with this setting enabled, however, each scan has its own database file with a 10 GB limit. You can have an unlimited number of scans until Docker allocates the entire Docker volume disk partition size.

Optional External SQL Server for CLI and API Modes

When using the default local SQL express database in CLI and API modes, the sensor creates a separate `*.mdf` database file for each new scan. Creating a separate database for each scan allows multiple scans and avoids the 10 GB limit for a single SQL Express scan database. Additionally, the Docker engine provides Docker volumes that help to configure where each volume is stored, and space is only limited by the disk partition size where the Docker volume is located.

Facts About External SQL Servers

If you feel an external scan database might be needed, consider the following facts when making your determination:

- Each sensor should use its own database on the external SQL Server. A sensor's database should not be used by other sensors. The same best practice applies to the standard WebInspect desktop configuration that uses an external scan database.
- Network latency communication to the external SQL Server might be worse than with the local SQL Server Express and might affect scan duration.
- The external shared SQL Server might be loaded by parallel tasks from other clients, which can cause interferences and adversely affect scan duration.
- The sensor is a light version of Fortify WebInspect, and it does not contain functionality for upgrading scan databases from previous versions. With each sensor using its own database on the external SQL Server, you must manage scan database upgrades manually using an upgrade script from a Fortify WebInspect desktop version.

Options for Configuring an External SQL Server

If you decide to use an external SQL Server, you can configure the external SQL Server settings in either the Docker compose file or the Helm charts. For more information, see the following topics:

- "Configuring an External SQL Server using a Docker Compose File " below
- "Configuring an External SQL Server using Helm Charts" on page 37

Configuring an External SQL Server using a Docker Compose File

When using a Docker compose file to start the sensor container, you must edit the database settings in the environment file and the Docker compose (YAML) file for API or CLI mode.

Editing the Environment File for an External SQL Server

In addition to the edits described in "Editing the Environment File for Linux" on page 28, you must edit the database password in the environment file.

To update the database password:

1. Locate the following line in the `#[3]. Internal services configuration:` section of the environment file and type the remote SQL Server sa password for the setting:

```
FORTIFY_SCANNER_DB_PSWD=<SA_Password>
```

2. Save your edits.

Editing the Docker Compose File for an External SQL Server

To use the sensor in API or CLI Mode connected to an external SQL Server, then you must edit the `docker-compose.yaml` file.

To edit the Docker compose file:

1. Locate the local datastore service content and add a number sign (#) at the start of each line to comment out the lines as shown here.

```
version: "3"
services:
    #datastore:
    #    image: ${FORTIFY_SCANNER_DB_IMAGE:-mcr.microsoft.com/mssql/...
    #    restart: unless-stopped
    #    environment:
    #      - ACCEPT_EULA=Y
    #      - MSSQL_PID=Express
    #      - SA_PASSWORD=${FORTIFY_SCANNER_DB_PSWD}
    #    networks:
    #      - wi_net
    #    volumes:
    #      - scandata:/etc/wi/.widata/user/ScanData
    #    logging:
    #      driver: none
```

2. Locate the scanner service environment variables and make the following edits:
    a. Remove the number sign (#) from the start of the line for the `WI_SCANDB` environment variable and edit its arguments as follows:
        - For `Server`, type the IP address or hostname of the external SQL Server.
        - For `Database`, type a database name to be created and used by the sensor.
    b. Add a number sign (#) at the start of the line for the `WI_SQLEXPRESS` environment variable to comment out the line.

```
scanner:
    ...
    environment:
      ...
      #Single shared Scan DB:
      - WI_SCANDB=Server=<IP_Address>;Database=<DB_Name>;User
Id=sa;Password=${FORTIFY_SCANNER_DB_PSWD};
      #Multiple separated MDF Files DB:
          #- WI_SQLEXPRESS=Data Source=datastore;User
Id=sa;Password=${FORTIFY_SCANNER_DB_PSWD};
```

3. Save your edits.

Running the Container with an External SQL Server

After you have edited the environment file and the Docker compose file with the required changes for an external SQL Server, you can use the Docker compose file to start the Linux containers.

To start the containers:

- At the terminal prompt on the Linux Docker host machine, enter the following command:

  ```
  docker compose up -d
  ```
  The WISE, scanner, and 2FA server containers are started, and the internal communication network is created. The database container is not started because scan data is written to the sensor's database on the external SQL Server.

Configuring an External SQL Server using Helm Charts

When using Helm charts to start the sensor container, you must edit the database settings in the `scanner-k8s-config.yml` file.

Editing the Configuration File

To edit the `scanner-k8s-config.yml` file:

1. At the terminal prompt on the Linux Docker host machine, enter the following commands:

   ```
   cd "$HOME/widocker"
   tree .
   ```

   The commands return a directory structure similar to the following:

   ```
   ├── docker-compose.mode3.yaml
   ├── docker-compose.mode4.yaml
   ├── docker-compose.yaml
   ├── .env
   ├── helm-charts
   │   ├── Chart.yaml
   │   ├── .helmignore
   │   ├── readme.txt
   │   ├── templates
   │   │   ├── scanner-k8s-config.yml
   │   │   └── scanner-pull-secret.tpl
   │   └── values.yaml
   ```

```
└── readme.txt
```

2. Enter the following command:

   ```
   nano "./helm-carts/templates/scanner-k8s-config.yml"
   ```
   The helm chart file is opened for editing.
3. Locate the local datastore container content and add a number sign (#) at the start of each line to comment out the lines as shown here.

   ```
   containers:
         #- name: datastore
         #  image: "{{ index .Values.images.scandb .Values.images.base }}"
         #
         #  imagePullPolicy: {{ .Values.images.pull.policy }}
         #  volumeMounts:
         #  - name: scandata
         #    mountPath: /etc/wi/.widata/user/ScanData
         #  env:
         #  - name: ACCEPT_EULA
         #    value: "Y"
         #  - name: MSSQL_PID
         #    value: "Express"
         #  - name: SA_PASSWORD
         #    valueFrom:
         #      secretKeyRef:
         #        name: "{{ .Values.scanner.name }}-secrets"
         #        key: datastore
   ```
4. Locate the scanner container environment variables and make the following edits:
   a. Remove the number sign (#) from the start of the line for the `WI_SCANDB` environment variable and edit its arguments as follows:
      - For `Server`, type the IP address or hostname of the external SQL Server.
      - For `Database`, type a database name to be created and used by the sensor.
      - For `Password`, type the SQL Server sa password.

   b. Add a number sign (#) at the start of the line for the `WI_SQLEXPRESS` environment variable to comment out the line.

   ```
       #Single shared Scan DB:
          - name: WI_SCANDB
            value: "Server=<IP_Address>;Database=<DB_Name>;User
   Id=sa;Password=<SA_Password>;"
          #Multiple separated MDF Files DB:
          #- name: WI_SQLEXPRESS
   ```

```
        #     value: "Data Source=127.0.0.1;User Id=sa;Password=$(WI_
SCANDB_SECRET);"
```

5. Save your edits.

Running the Container with an External SQL Server

After you have edited the configuration file with the required changes for an external SQL Server, you can run the Helm installation following the instructions in the `helm-charts/readme.txt`.

Updating SecureBase in the Container

SecureBase is Fortify's database of adaptive agents, vulnerability checks, and policy information. The database is updated regularly with the latest threats and improvements to vulnerability detection. Each Fortify WebInspect image includes a SecureBase that was up-to-date at the time the image was created. However, you can use the SecureBase Manager CLI tool to ensure that you have the latest content.

Note: You can also use the SecureBase update endpoints in the Fortify WebInspect REST API. For information on how to access the Fortify WebInspect REST API Swagger UI, refer to the *OpenText™ Fortify WebInspect User Guide*

Important! The SecureBase Manager CLI tool applies only check and policy updates to the database. It does not update the Fortify WebInspect version.

Updating Linux Containers

For Linux, the SecureBase Manager CLI tool is called **sbm** and is included in the scanner container. Therefore, you must use the tool after all required Linux containers have been started. For more information, see "Running the Linux Container in API or CLI Mode" on page 30.

To launch the SecureBase Manager CLI tool in Linux:

- At the terminal prompt on the Red Hat Linux Docker host machine, enter the following command:

  `docker compose exec scanner sbm`
  The SecureBase Manager CLI tool starts and displays available options. For more information, see "Understanding the SecureBase Manager CLI Tool Options" on the next page.

Use the options as shown in the following syntax:

`docker compose exec scanner sbm -<option>`

For example, to check for an updated version of SecureBase, use the following text:

```
docker compose exec scanner sbm -ck
```

If updates are available, then to download and install the updates, use the following text:

```
docker compose exec scanner sbm -up
```

Updating Windows Containers

For Windows, the SecureBase Manager CLI tool is an executable file named **SecureBaseManager.exe** and is located in the Fortify WebInspect installation directory. By default, the installation directory is:

```
C:\Program Files\Fortify\Fortify WebInspect
```

Tip: Although the tool is designed for use in a Docker container, you can also use it in a classic Windows desktop installation.

To launch the SecureBase Manager CLI tool on Windows:

1. At the command prompt, use the `cd` command to change the current working directory to the directory where the tool is installed.
2. Enter the following command:

   `SecureBaseManager.exe`
   The SecureBase Manager CLI tool starts and displays available options. For more information, see "Understanding the SecureBase Manager CLI Tool Options" below.

Use the options as shown in the following syntax:

`SecureBaseManager.exe -`*`<option>`*

For example, to check for an updated version of SecureBase, use the following text:

```
SecureBaseManager.exe -ck
```

If updates are available, then to download and install the updates, use the following text:

```
SecureBaseManager.exe -up
```

Understanding the SecureBase Manager CLI Tool Options

The following table describes the tool options.

Option	Description
`-h`	Displays the help.
`-ck`	Checks for an updated version of SecureBase. This option returns results similar to the following: `* Smartupdate Server  :`

Option	Description

	```
	https://smartupdate.fortify.microfocus.com/
	* Engine Version      : 4.31.00
	* Last Updated Date   : 5/15/2024 21:02:57
	* Language Id         : 1, (en-us)
	* Pending Enabled     : False
	* Status              : New Updates Available
	```
	 These results are explained as follows:
	• `Smartupdate Server` – Identifies the OpenText server used for obtaining check and policy updates.
	• `Engine Version` – Indicates the engine version of the updater application.
	• `Last Updated Date` – Indicates the date and time the SecureBase was last updated. This information comes from the SmartUpdate server.
	• `Language Id` – Indicates the language that is currently in use for the local SecureBase content. Language numbers are shown under "-ls" below.
	• `Pending Enabled` – Indicates whether checks that are pending release should be included in the updated content.
	• `Status` – Indicates whether the database content is current. Possible values are `New Updates Available` and `Already up to date`.

`-up`	Updates SecureBase.
`-fr`	Disables Docker interactive mode.
`-ps`	For advanced users only, specifies a proxy server for accessing the SmartUpdate server. Specify the address and port in the following format: `-ps <address>:<port>`
`-pn`	Includes data for checks that are pending release. **Important!** Do not use this option unless instructed to do so by Fortify Support.
`-su`	Specifies the SmartUpdate server to access. Specify the server in the following format: `-su https://<smartupdate.address>/` **Important!** Do not use this option unless instructed to do so by Fortify Support.
`-ls`	Lists the available languages for SecureBase content. This option returns results similar

Option	Description

	to the following:
	`► 1: en-us, English            | Installed`
	`  2: ja-jp, Japanese           |`
	`  3: ko-kr, Korean             |`
	`  4: zh-cn, Simplified Chinese  |`
	`  5: zh-tw, Traditional Chinese |`
	`  6: es-es, Spanish            | Installed`
	`  7: pt-br, Portuguese         |`
	In the preceding example, English and Spanish versions are installed. The ► symbol indicates that English is the version that is currently in use.

`-ln`	Switches SecureBase content to another language. Specify the desired language in the following format: `-ln` *`<language_number>`* Language numbers are shown under "-ls" on the previous page. **Important!** If the desired language has not been downloaded, then you must download a new SecureBase in the target language from the SecureBase server. Depending on your network, it may take up to 30 minutes to complete. During the process, the existing SecureBase is temporarily removed from its default location and replaced with an empty SecureBase. Do not attempt to run any scans until this process has completed. After the language is downloaded and installed, switching between the languages takes very little time.

Chapter 3: Using the OAST Docker Image

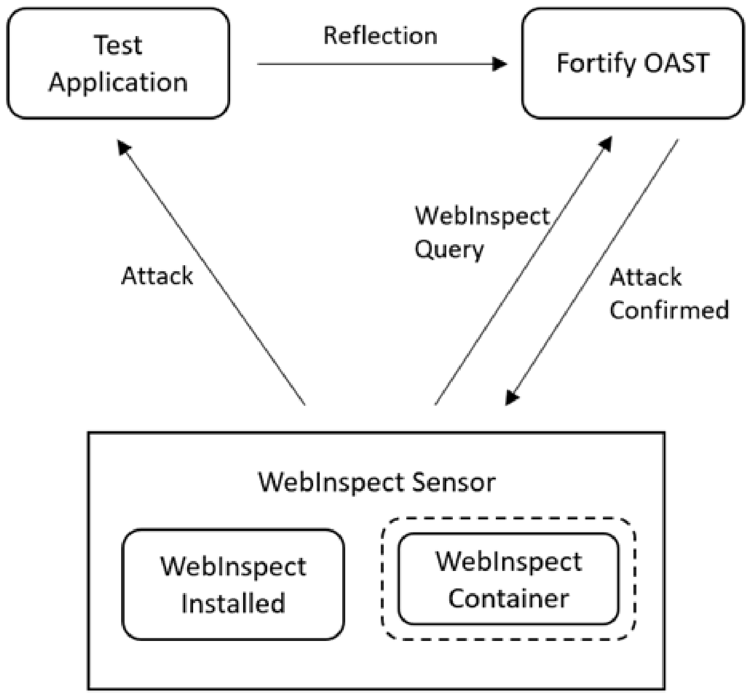
The following paragraphs describe how the Fortify OAST on Docker image works and how to configure and run it in a container.

How Fortify OAST Works

OAST vulnerabilities do not reflect back to Fortify WebInspect, making them difficult to detect with traditional DAST scanning. The Fortify OAST server provides DNS service for the detection of out-of-band attack vulnerabilities, such as Log4Shell, Generic Deserialization, and General JNDI Injection. You configure and use the server with a desktop version of Fortify WebInspect or with WebInspect on Docker.

With the Log4Shell vulnerability, if WebInspect is able to detect the vulnerability, then the application server under test will send a DNS lookup to the Fortify OAST server. Fortify WebInspect will then query the Fortify OAST server to determine whether it received the DNS lookup. If the Fortify OAST server received it, then the application server is susceptible to the vulnerability.

The following diagram illustrates how Fortify WebInspect works with Fortify OAST during a scan to detect the Log4Shell vulnerability.



Ubuntu and Alpine

This document assumes that the Ubuntu 22.04 LTS x86_64 operating system is used on the Docker host machine for the Alpine image version. In this document, Ubuntu refers to the Docker host machine and Alpine refers to the Docker image.

Understanding the Configuration Process

The following table describes the process of configuring and using Fortify OAST in conjunction with a Fortify WebInspect scan.

Stage	Description
1.	Prepare a Linux VM machine and install the appropriate Docker Engine. This machine will be the host for the Fortify OAST image. **Note:** Follow Docker recommendations for the Docker engine version to use for Red Hat Universal Base Image (UBI) 8.x x86_64 and Ubuntu 22.04 LTS x86_64 host operating systems.
2.	Pull the Fortify OAST Docker image. See "Pulling the Fortify OAST Image" on the next page.
3.	Configure settings on the Linux Docker host machine to disable the embedded DNS server and use a manual DNS configuration. For Ubuntu, see "Configuring the Ubuntu Linux Docker Host Machine" on page 46. For Red Hat, refer to your Red Hat documentation for manually configuring the `/etc/resolv.conf` file.
4.	Run the Fortify OAST container. See "Running the OAST Container" on page 48.
5.	Do one of the following: • Configure Fortify WebInspect to use the Fortify OAST server. See "Configuring Fortify WebInspect for OAST" on page 51. • Pass environment variables in the Docker run command to start your Fortify WebInspect Windows container and use the Fortify OAST server. See "Running Fortify WebInspect on Docker Windows Version with Fortify OAST" on page 53. • Edit the environment file for the Linux images and use the Docker compose file to start the Linux container with the local OAST server. See "Editing the Environment

Stage	Description
	File for Linux" on page 28 and "Running the Linux Container in API or CLI Mode" on page 30.
6.	Do one of the following: • Configure the target web application to use the Fortify OAST server. See "Configuring the Target Application for OAST " on page 54. • Run the target application container with the Fortify OAST server. See "Running the Target Application in Docker with OAST" on page 55.

About the OAST Image

The Fortify OAST image runs on a Linux VM Machine and is available for the Ubuntu Linux distribution and for the Red Hat Linux distribution. It provides DNS service for the detection of OAST vulnerabilities, and it is intended for use in networks that lack an Internet connection.

Image Naming Convention

The Fortify Docker repository uses the following naming convention for the Fortify OAST image:

```
fortifydocker/fortify-oast:<version.linux_os_version>
```

The latest Alpine image version that is available as of this writing is:

```
fortifydocker/fortify-oast:24.2.alpine.3.17
```

The latest Red Hat image version that is available as of this writing is:

```
fortifydocker/fortify-oast:24.2.ubi.8
```

Note: These image versions include the Alpine or Red Hat Linux operating system build number that is used in the image.

For more information about the version that is available, refer to the Readme file in the fortifydocker/fortify-oast repository.

Pulling the Fortify OAST Image

After starting the Docker service, you can pull an image of Fortify OAST from the Fortify Docker repository as described in this topic.

Pulling the Alpine Image

To pull the current version of the Fortify OAST image:

- At the terminal prompt on the Ubuntu Linux Docker host machine, enter the following command:

```
docker pull fortifydocker/fortify-oast:24.2.alpine.3.17
```

Pulling the Red Hat Image

To pull the current version of the Fortify OAST image:

- At the terminal prompt on the Red Hat Linux Docker host machine, enter the following command:

```
docker pull fortifydocker/fortify-oast:24.2.ubi.8
```

Configuring the Ubuntu Linux Docker Host Machine

You must determine if the ports needed by the Fortify OAST server are currently in use. If they are, you must edit the default settings in the `resolved.conf` file on the Ubuntu Linux Docker host machine to disable the embedded DNS server and use a manual DNS configuration. Afterward, you must reboot the host machine.

Checking Port Usage

The Fortify OAST server requires the following ports:

- 443/TCP
- 53/TCP
- 53/UDP

By default, the Ubuntu OS allocates its local DNS resolver to ports 53/TCP and 53/UDP. However, port 443 is not allocated.

To check whether these required ports are used on the Ubuntu Linux Docker host machine:

- At the terminal prompt on the host machine, enter the following command:

```
netstat -antu
```

If port 443 is in use, either find the server that is using it and free the port or use an Ubuntu OS with default settings. If ports 53/TCP and 53/UDP are in use, you can free them as described in "Editing the Configuration File to Free Ports" on the next page.

Editing the Configuration File to Free Ports

Ubuntu 20.04 may allocate 53/TCP and 53/UDP ports by default for the `systemd-resolved` system service that provides network name resolution on the local DNS server. You can reconfigure them in the `resolved.conf` file.

To edit the configuration file:

1. At the terminal prompt on the Ubuntu Linux Docker host machine, enter the following command:

   ```
   sudo nano /etc/systemd/resolved.conf
   ```

   The following example shows the `resolved.conf` file contents.

   ```
   [Resolve]
   #DNS=
   #FallbackDNS=
   #Domains=
   #LLMNR=no
   #MulticastDNS=no
   #DNSSEC=no
   #Cache=yes
   #DNSStubListener=yes
   ```

2. Remove the number sign (#) from the start of the line for `DNS`.
3. After `DNS=`, enter the IP address for your working primary local network DNS server.
4. Remove the number sign (#) from the front of the line for `DNSStubListener`.
5. Change the `DNSStubListener` setting to `no`.

   The updated `resolved.conf` file should resemble the following example.

   ```
   [Resolve]
   DNS=<ip_address>
   #FallbackDNS=
   #Domains=
   #LLMNR=no
   #MulticastDNS=no
   #DNSSEC=no
   #Cache=yes
   DNSStubListener=no
   ```

6. Save your changes.

Creating a Symbolic Link

At this point, the Ubuntu embedded DNS server is disabled. You must configure Ubuntu Linux to use manual DNS configuration to resolve hostnames. For manual DNS configuration, Linux reads settings from `/etc/resolv.conf`. Therefore, you must create a symbolic link to this file from the `systemd` service that provides network name resolution to local applications.

To create the symbolic link:

- At the terminal prompt on the Ubuntu Linux Docker host machine, enter the following command:

```
sudo ln -sf /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

The following table describes the options used in the command.

Option	Description
`-s`	Creates a symbolic link instead of a hard link.
`-f`	Removes existing files from the destination directory.

Rebooting the Ubuntu Linux Docker Host Machine

After configuring the changes, you must reboot the Ubuntu Linux Docker host machine. Refer to your Ubuntu documentation for details.

Running the OAST Container

After your DNS configurations are complete and the host machine has been rebooted, you can run the OAST container.

Running the Alpine Version

To run the container:

- At the terminal prompt on the Ubuntu Linux Docker host machine, enter the following commands:

```
mkdir -p "<host_path>/certs"
docker run -d \
    --name <string> \
    --restart unless-stopped \
    -p 443:443 \
    -p 0.0.0.0:53:53/tcp \
    -p 0.0.0.0:53:53/udp \
    -e "WIH_DOMAIN=<domain_name>" \
    -e "WIH_IPv4_PUBLICIP=<ip_address>" \
    -v "<host_path>/certs:/etc/wihorizon/certs" \
    --log-opt max-size=20m \
    --log-opt max-file=5 \
    fortifydocker/fortify-oast:24.2.alpine.3.17
```

Tip: The backslash (\) indicates the end of line for the Linux OS.

Running the Red Hat Version

To run the container:

- At the terminal prompt on the Red Hat Linux Docker host machine, enter the following commands:

```
mkdir -p "<host_path>/certs"
docker run -d \
    --name <string> \
    --restart unless-stopped \
    -p 443:443 \
    -p 0.0.0.0:53:53/tcp \
    -p 0.0.0.0:53:53/udp \
    -e "WIH_DOMAIN=<domain_name>" \
    -e "WIH_IPv4_PUBLICIP=<ip_address>" \
    -v "<host_path>/certs:/etc/wihorizon/certs" \
    --log-opt max-size=20m \
    --log-opt max-file=5 \
    fortifydocker/fortify-oast:24.2.ubi.8
```

Understanding the Run Command Options

The following table describes the options used in the run command.

Option	Description
`-d`	Runs the container in the background and prints the container ID.
`--name`	Specifies the name of your Fortify OAST container. Any string is valid. Examples in this table use `wihorizon`.
`--restart unless-stopped`	Restarts the container unless the container is manually stopped.

Option	Description
`-p 443:443`	Publishes the container's main TCP ingress port to the host.
`-p 0.0.0.0:53:53/udp`	Publishes the container's UDP DNS server port to the host.
`-p 0.0.0.0:53:53/tcp`	Publishes the container's TCP DNS server port to the host.
`-e "WIH_DOMAIN=<domain_name>"`	Configures the local domain name. For example: `-e "WIH_DOMAIN=local-fortify-oast.net"`
`-e "WIH_IPv4_PUBLICIP=<ip_address>"`	Configures the local IP address for the Ubuntu Linux Docker host machine that is exposed to the Fortify WebInspect sensor.
`-e "WIH_API_PORT=8443"`	Optionally, if your security policy prevents services from running on ports below 1024, you may add this option to the command and publish the assigned port using the `-p` option.
`-v "<host_path>/certs:/etc/wihorizon/certs"`	Adds a volume for a Fortify OAST auto-generated certificates directory. This directory safeguards the certificates in case the Fortify OAST container needs to be removed or upgraded. For example: `-v "$HOME/.wihorizon/certs:/etc/wihorizon/certs" \`
`--log-opt max-size=20m`	Limits the Docker log file size to the specified number of megabytes. This setting prevents log files from consuming too much disc space.
`--log-opt max-file=5`	Limits the number of Docker log files to the specified number. When the number is reached, Docker removes the oldest log file and starts a new one.

Configuring Fortify WebInspect for OAST

You can use the Fortify OAST server with a classic Fortify WebInspect installation or with the Fortify WebInspect on Docker image. This topic describes the required configuration changes to support the Fortify OAST server with a classic installation. For information using Fortify OAST with the Fortify WebInspect container, see "Running Fortify WebInspect on Docker Windows Version with Fortify OAST" on page 53.

Configuring Access to the Fortify OAST Server

You must configure either your network or your sensor to provide access to the Fortify OAST server.

To configure access, do one of the following:

- Add the domain name that you configured for the `WIH_DOMAIN` option to your local DNS server.
- Edit the host file on the Fortify WebInspect machine to point to the Docker host IP address that you configured for `WIH_PUBLICIP` option.

For more information, see "Running the OAST Container" on page 48.

Verifying Access to the Fortify OAST Server

Verify that the Fortify OAST server works on the Fortify WebInspect machine.

To verify access:

- In PowerShell on the sensor machine, enter the following command:

  `nslookup 00000000-0000-0000-0000-000000000000.<WIH_DOMAIN> <WIH_DOMAIN>`

  Using the example from "Running the OAST Container" on page 48, the command would be as follows:

  ```
  nslookup 00000000-0000-0000-0000-000000000000.local-fortify-oast.net
  local-fortify-oast.net
  ```

  If the Fortify OAST server works, you should see `127.0.0.1` as the resolved address, as shown in the following example:

  ```
  Server:         local-fortify-oast.net
  Address:        <WIH_IPv4_PUBLICIP>#53

  Name:   00000000-0000-0000-0000-000000000000.local-fortify-oast.net
  Address: 127.0.0.1
  ```

Verify the Fortify OAST Docker Logs

Verify that the Fortify OAST server is logging its connection to the sensor in the Docker container log file.

To verify log files:

- At the terminal prompt on the Ubuntu Linux Docker host machine, enter the following commands:

  `docker logs <fortify_oast_container_name>`

  Using the example from "Running the OAST Container" on page 48, the command would be as follows:

  ```
  docker logs wihorizon
  ```

  You should see output similar to the following:

  ```
  0/00, 00:00:00  00 | [DNS] Detected correlation GUID 00000000-0000-0000-
  0000-000000000000 p0
  ```

Configure Fortify WebInspect to Use the Local Domain

You must configure the sensor to use the local domain name that you configured for the `WIH_DOMAIN` option.

To use the local domain:

1. Close all Fortify WebInspect instances.
2. On the sensor machine, open the command line prompt and navigate to the Fortify WebInspect installation directory.

   Tip: By default, the installation directory is `C:\Program Files\Fortify\Fortify WebInspect\`.

3. At the command prompt, enter the following command:

   `WIConfig.exe -WIOASTServerAddress "<WIH_DOMAIN>"`

   Using the example from "Running the OAST Container" on page 48, the command would be as follows:

   ```
   WIConfig.exe -WIOASTServerAddress "local-fortify-oast.net"
   ```

Running Fortify WebInspect on Docker Windows Version with Fortify OAST

You can pass environment variables in the Docker run command to start your Fortify WebInspect to use the Fortify OAST server that you configured.

To start the Windows sensor with Fortify OAST:

- In PowerShell on the sensor machine, enter the following command:

```
docker run -d `
       --name <container_name> `
       -p 8089:8089 `
       -e 'mode=<number>' `
       -e 'limURL=http://<server-url>:<port>' `
       -e 'limPool=<string>' `
       -e 'limPswd=<string>' `
       -e 'RCServerHost=+' `
       -e 'RCServerPort=<port_number>' `
       -e 'RCServerAuthType=None' `
       -e 'WIOASTServerAddress=<WIH_DOMAIN>' `
fortifydocker/webinspect:24.2
docker exec webinspect cmd /c "echo <WIH_IPv4_ADDRESS> <WIH_DOMAIN> >>
C:\Windows\System32\drivers\etc\hosts"
```

Understanding the Run Command Options

The following table describes the options used in the run command.

Option	Description
`-d`	Runs the container in the background and prints the container ID.
`--name`	Specifies the name of your Fortify WebInspect container. Any string is valid. Examples in this table use `webinspect`.
`-p 8089:8089`	Publishes the Fortify WebInspect REST API port to the host.
`mode`	Specifies the operation mode for the container. For more information, see "Understanding the Operation Modes" on page 15.
`limURL`, `limPool`, `limPswd`	Configures licensing. For more information, see "Configuring

Option	Description
	Licensing (Required for CLI and API Modes) " on page 18.
`RCServerHost`, `RCServerPort`, `RCServerAuthType`	Configures access to the Fortify WebInspect API server. For information about the API server options, see "Configuring API Mode Options" on page 19.
`WIOASTServerAddress`	Configures Fortify WebInspect to use the local Fortify OAST server. Using the example from "Running the OAST Container" on page 48, the command would be as follows: `-e 'WIOASTServerAddress=local-fortify-oast.net' ``

Configuring the Target Application for OAST

You must configure the target web application to use the Fortify OAST server for DNS lookup requests from Fortify WebInspect or run the target application container with Fortify OAST. This topic describes the required changes to the target application. For information about running the target application container, see "Running the Target Application in Docker with OAST" on the next page.

Adding the Local Domain Server

Add the local domain name that you configured for the `WIH_DOMAIN` option to the web application network as the primary DNS server or add it as a primary DNS server for the machine that hosts the target web application. In a Linux OS, for example, you can edit the `/etc/resolv.conf` file by adding the Fortify OAST server as the primary DNS server and using the real network DNS server as secondary:

```
nameserver <WIH_IPv4_ADDRESS>
nameserver <dns_server_ip_address>
```

Verifying Application Access to the Fortify OAST Server

Verify that the Fortify OAST server works for the target web application machine.

To verify access:

- At the terminal prompt on the web application machine, enter the following command:

```
nslookup 00000000-0000-0000-0000-000000000000.<WIH_DOMAIN>
```

Using the example from "Running the OAST Container" on page 48, the command would be as follows:

```
nslookup 00000000-0000-0000-0000-000000000000.local-fortify-oast.net
```

If the Fortify OAST server works, you should see `127.0.0.1` as the resolved address, as shown in the following example:

```
Server:         local-fortify-oast.net
Address:        <WIH_IPv4_PUBLICIP>#53

Name:   00000000-0000-0000-0000-000000000000.local-fortify-oast.net
Address: 127.0.0.1
```

Verify the Fortify OAST Docker Logs

Verify that the Fortify OAST server is logging its connection to the target web application in the Docker container log file.

To verify log files:

- At the terminal prompt on the Ubuntu Linux Docker host machine, enter the following command:

  ```
  docker logs <fortify_oast_container_name>
  ```

  Using the example from "Running the OAST Container" on page 48, the command would be as follows:

  ```
  docker logs wihorizon
  ```

  You should see output similar to the following:

  ```
  0/00, 00:00:00  00 | [DNS] Detected correlation GUID 00000000-0000-0000-
  0000-000000000000 p0
  ```

Running the Target Application in Docker with OAST

If the target web application resides in a Docker image, you can run the target application container with Fortify OAST.

To start the application with Fortify OAST:

- At the terminal prompt on the web application container, enter the following commands:

  ```
      docker run -d \
      --name <container_name> \
      --restart unless-stopped \
  ```

```
       -p <port>:<port> \
       --dns <WIH_IPv4_ADDRESS> \
       --dns <ip_address> \
       --dns <ip_address> \
       --log-opt max-size=20m \
       --log-opt max-file=5 \
  <docker_repo>/<application_name>:latest
```

Understanding the Run Command Options

The following table describes the options used in the run command.

Option	**Description**
`-d`	Runs the container in the background and prints the container ID.
`--name`	Specifies the name of your test application container.
`--restart unless-stopped`	Restarts the container unless the container is manually stopped.
`-p <port>:<port>`	Publishes the container's port to the host.
`--dns`	Indicates the various DNS servers to use. The first entry adds the Fortify OAST server as the primary DNS server. Each of the following `--dns` entries are real network DNS servers that respond to all regular nameserver queries so that the container environment can perform all required nslookups.
`--log-opt max-size=20m`	Limits the Docker log file size to the specified number of megabytes. This setting prevents log files from consuming too much disc space.
`--log-opt max-file=5`	Limits the number of Docker log files to the specified number. When the number is reached, Docker removes the oldest log file and starts a new one.

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email.

Note: If you are experiencing a technical issue with our product, do not email the documentation team. Instead, contact Customer Support at https://www.microfocus.com/support so they can assist you.

If an email client is configured on this computer, click the link above to contact the documentation team and an email window opens with the following information in the subject line:

Feedback on User Guide (Fortify WebInspect and OAST on Docker 24.2.0)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to fortifydocteam@opentext.com.

We appreciate your feedback!