The Lighthouse

By Lew Folkerth, Principal Reliability Consultant

Using Advanced IT Technologies in an OT Environment Part 2 - Containers

In my previous article I discussed some recommended principles to follow when adopting new technologies into your Operational Technology (OT) environment. In this article I provide considerations for adopting container technology for use in OT systems.

Virtual Systems Old and New

NERC recently published three Compliance Monitoring and Enforcement Program (CMEP) Practice Guides related to the use of virtualization technologies. The practice guides provide guidance to CMEP staff on how to assess the use of virtual systems, virtual storage and virtual networks in a CIP environment. I've seen many Entities use these three virtualization technologies successfully. You need to be careful to adopt these technologies in a way that doesn't compromise CIP compliance. The three CMEP Practice Guides should be useful in that effort.

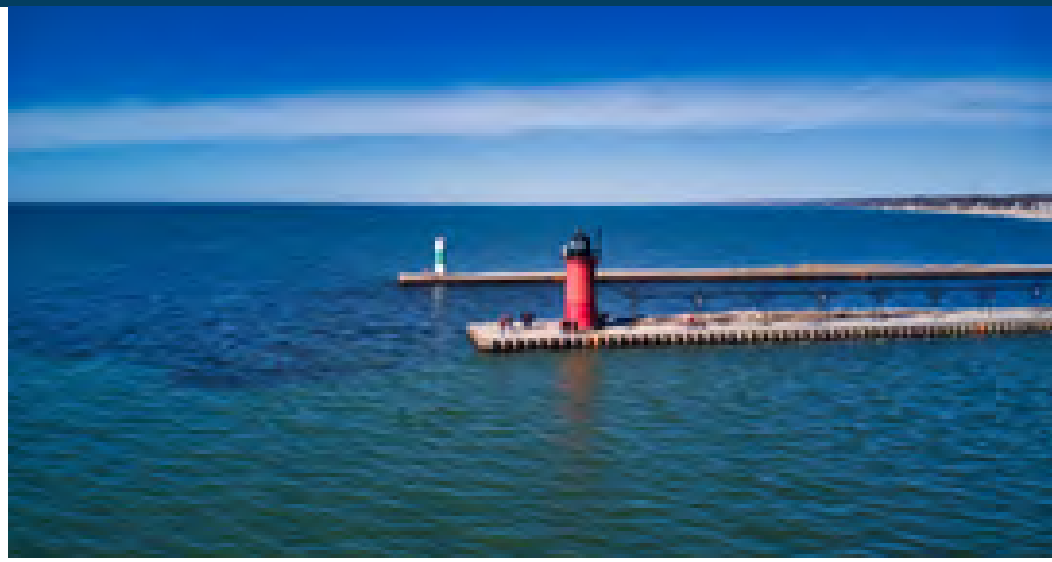
In this recurring column, I explore various questions and concerns related to the NERC Critical Infrastructure Protection (CIP) Standards. I share my views and opinions with you, which are not binding. Rather, this information is intended to provoke discussion within your entity. It may also help you and your entity as you strive to improve your compliance posture and work toward continuous improvement in the reliability, security, resiliency and sustainability of your CIP compliance programs. There are times that I also may discuss areas of the Standards that other entities may be struggling with and share my ideas to overcome their known issues. As with lighthouses, I can't steer your ship for you, but perhaps I can help shed light on the sometimes stormy waters of CIP compliance.

Recently, I've seen Entities begin to use a newer virtualization technology called **containers** or **containerization**. Containers are a way of encapsulating an application program to isolate it from other applications and to make deployment of the application easier and faster.

What are Containers?

Containers can be thought of as a lightweight form of virtualization. The technology uses features of the operating system to isolate the application in each container from the operating system and other containers. A container reduces the overhead required for a full virtual system, enabling more efficient and flexible use of computing assets.

Each container begins with building a container **image**. The image will include an application and can include everything the application needs to run, such as libraries, programming language runtimes, utilities and configuration settings. Once an image is built, it cannot be modified. If an image needs modification, a new image must be built.



South Haven N & S Pier Lights, South Haven, MI – Photo: L Folkerth

Container images that are in production typically reside in a **registry**, which is a method of storing and controlling approved container images. Once an image is built and tested, it is **pushed** to a registry. To run an application, it is **pulled** (copied) from the registry and executed.

Don't confuse containers with other types of virtualization. Containers can be, and frequently are, used in conjunction with virtual machines, but containerization is a separate technology that should be evaluated on its own merits.

Lew's Principles for Adopting IT Technologies in an OT Environment

1. Clearly identify the IT technology to be implemented
2. Objectively assess the benefits
3. Objectively assess the risks
4. Perform a risk/benefit analysis in addition to a cost/benefit analysis

Continued on page 13

Continued from page 12

Benefits of Containers

Packaging – The primary benefit of using containers is that application code is packaged with all the dependencies the application relies upon, such as libraries and runtimes. This keeps applications from interfering with each other. A patch or update to a runtime for one application might, without warning, break another application that uses the same runtime. For example, use of the Java language is common in development of real-time systems. Programs written in Java are compiled to an intermediate format known as bytecode. Bytecode is executed by the Java Virtual Machine (JVM) runtime. Java programs can be very sensitive to the version of the JVM being used. By placing the appropriate JVM in each container image, you can be certain that an incompatible version of the JVM will never be used.

Encapsulation – A container can expose only needed and expected network ports when it runs. If an application unexpectedly opens a new port, that port will not be accessible outside the container unless it is specifically permitted.

Abstraction – You can think of containers as systems at the application level, since each container includes dependencies needed for the application to run. The application and its dependencies can be tested and deployed as a unit.

Agility – Containers permit more flexibility in where and how applications are run. If one server becomes heavily loaded, containers can easily be started on other servers, which will reduce the load on the original server.

Immutability – Container images are **immutable**, meaning their contents cannot be changed after they are built. This helps prevent unauthorized changes to an application. Applications in running containers may become compromised, but if an application in a container image is compromised it will no longer run. This means that any malware that compromises a running container will not be able to persist through a restart of that container, making the malware's task more difficult.

Risks of Containers

Authenticity – Public container registries are convenient and easy to use, but these registries may not fully examine the software in the containers to ensure the software is free of malware. In this case, ease of use equates to higher security risk. Also, when building container images locally, building tools are usually configured by default to pull dependency software from public repositories.

Proliferation – Once software developers begin using containers, the use of containers tends to proliferate. This may lead to issues with change management and version control.

Environment – Containers can be configured to run on most modern operating systems. This may lead to containers being run in unintended environments.

Configuration – While containers help simplify some software deployment tasks, containers come with their own complexities. This may permit insecure configurations without the knowledge of the system owners.

Containers in a CIP Environment

If you're going to use containers in a CIP environment, you should carefully architect your internal controls to ensure you maximize the system's reliability, resilience and security while remaining within the bounds of compliance. The items below may serve as a starting point for your considerations.

→ BES Cyber Systems Identification

When you begin planning to bring containers into a CIP environment you will need to decide how you will identify the containers. The most popular method of identifying container images is through the baselines of CIP-010-3 R1, Configuration Change Management. Each container image can be documented by applying a unique identifier to the container image and then identifying the name and version of each software component in the image. The documentation should also include the Cyber Asset that each container created from the image is permitted to use.

→ Software Inventory and Version Control

You should maintain a list of container images authorized to run in your CIP environment. For each image, keep track of when the image was built; when, where and how it was tested; when it entered into service; what image was superseded by it; and change authorization records.

→ Software Integrity and Authenticity

One of the popular features of containers is the ease with which they can be built and deployed. There are public repositories with thousands of pre-built images ready for use by executing a single "pull" command. While easy and convenient, this functionality does not play well in a CIP environment. You must

Continued on page 14

Continued from page 13

be able to document the source of each software component in a container image. Each component must have integrity and authenticity (identification of software source) records in compliance with CIP-010-3 R1 Part 1.6. In addition, make sure you protect the registry where your container images are stored.

→ Patch Management

Each software component in a container image should be tracked by your patch management system. Because container images are immutable, whenever any component needs to be patched you will follow your process to create a new image.

→ Anti-Malware

After a container image is built, it should be scanned by your anti-malware tools. As the container image cannot be changed, malware cannot infect the image without causing the image to become invalid. Of course, container images should continue to be scanned as new malware signatures are published, but be sure to prevent a malware detection from quarantining files within the image, or that image will no longer be valid. If the malware detection is valid, you should immediately rebuild the image with clean components.

→ Vulnerability Management

Your processes for CIP-010-3 R3, Vulnerability Assessments, should be implemented as part of the testing process for each container image.

→ Audit Considerations

Containers are a new technology in CIP environments. You should let your Audit Team Lead (ATL) know you are using containers as early in the audit process as feasible. The ATL can then ensure the audit team has adequate preparation before reviewing your evidence. This should allow the audit to proceed normally, rather than needing to bring your audit team up to speed on the particular container implementation you are using during the busiest part of the audit.

→ Beyond the Standards

Advanced technologies need advanced security. Consider adding additional protections to your containers beyond the minimums required by the CIP Standards. For example, software defined networking permits much more control over network traffic than what is required by CIP-005-6, Electronic Security Perimeters. This tighter level of control may be appropriate in container environments.

Since container technology isn't yet directly addressed by the CIP Standards, you should develop a set of practices you adhere to when employing containers in a CIP environment. These practices should restrict how containers are built and deployed in order to maximize reliability, resilience, security and compliance.

Conclusion

Done properly, using containers in your CIP environment can improve reliability, resilience and security and streamline some compliance processes. But as with all things CIP: document, document, document!

Requests for Assistance

If you are an entity registered within the RF Region and believe you need assistance in sorting your way through this or any compliance related issue, remember RF has the Assist Visit program. Submit an Assist Visit Request via the RF website here.

Feedback

Please provide any feedback you may have on these articles. Suggestions for topics are always welcome and appreciated.

Lew Folkerth, Principal Reliability Consultant, can be reached here.