Hardening OpenShift Containers to complement Incident Handling

Kurtis Holland
Hardening OpenShift Containers to complement Incident Handling

GIAC (GCIH) Gold Certification

Author: Kurtis Holland, Kurtis.Holland@Gmail.com
Advisor: Christopher Walker, CISSP
Accepted: September 27, 2018

Abstract

Incident Responders are always faced with not knowing if they have adequate information on a server is appropriately security controls hardened or susceptible to attack. There is no such thing as 100% security. You're under attack and now are scrambling to understand your risks and threat surface should a hacker gain a foot hold in your environment. You want a mix of commercial and open source tools in place to manage this threat. This paper will dive into the processes and demonstrate a design using tools available for managing Linux controls for Open Shift containers and how you scan the multiple products and layers involved in the development operations processes. The guess work by Incident Handlers will be minimized and a simple "eyes on glass" solution for the entire environment will be at your disposal so you can assess the software inventory, version levels, security scan reports, and assist identification and containment options.
1. Introduction

“Lions and tigers, and bears, oh my!” was what Dorothy from the Wizard of Oz exclaimed hearing scary noisy in the Dark Forest on her way to the Emerald City. (Wizard of Oz, 1939). Today, almost 80 years later, the Dark Forest could be compared to the “Internet” which is full of very good content, as well as the darker content, that is tools and software to start campaigns to disrupt or compromise others using malware, ransomware, vulnerability discovery, hacking exploit, or denial of service (DoS) for fun or for profit. You only need to know where to look and download to run the software as is, or customize for your specific objectives depending on your goals. As a security Incident Handler, you will spend a lot of time on Investigation, Containment, and Eradication when an event leads to an incident. As part of the Lessons Learned, the focus needs to be on addressing your Software Development Life Cycle and the tools and processes that go into the technology.

1.1. The Threat Landscape

You don’t have to look very far to see that businesses and individuals are putting up applications and services that are vulnerable to compromise. There are many reasons for this ranging from inexperience of the developers or system engineers, immaturity of the technology, inexperienced end-users, as well as the rush to market skipping key deployment requirements to properly secure the application infrastructure to gain a competitive business advantage. There are many data aggregators and security firms that collect information on security events and incidents leading to loss services or data, such as Mandiant (FireEye), the Ponemon Institute, and Verizon Enterprises.

An interesting fact from the Mandiant M-Trends 2017 Report, is that the global median dwell time for first evidence of a data breach into your environment before detection is averaging approximately 101 days. The numbers in the U.S. are lower at about 75 days and higher in EMEA and APAC. This allows an attacker a long time to establish command and control, persistent access, and to target many areas of your environment. Companies experiencing data breaches have significant
gaps in their infrastructure, their product life cycles, and their technical skills. These breaches, according to the Ponemon Institute cost on average $3.86 Million dollars, or an average cost of about $148 per data record. Even after remediation, you are approximately 27.9% likely to be breached again.

1.2. Defense in Depth Strategy

Each year Verizon publishes a Data Breach Report, (Verizon DBIR, 2018) that covers the analysis of data breaches across all sectors of business. The attacks vary from malware, phishing, and ransomware to Web Application hacking, Denial of Service, Espionage, and numerous other attacks. Unfortunately, the statistics year over year hold that roughly 75% of the attacks are from external actors, and roughly 25% internal. Nation States make of a big part of the advanced persistent threat and espionage attacks. For fun and profit attacks are ever present and often have a political motive. This leaves the individual organization few options but to deploy defense in depth architectures and processes for protecting both the external and internal facing environments.

A mature risk and security managed organization is going to have a layered security approach. Often, data breaches or access to your environment are not just based on user error such as with a malware or phishing attack against an end-user, but with a vulnerability exploited within the application or infrastructure. The Department of Homeland Security within their “Recommended Practice: Improving Industrial Control System Cyber-Security with Defense-in-Depth Strategies” (U.S. Department of Homeland Security, 2016) paper outlines several security functions crossing multiple design boundaries. Even though the focus is on specific industrial control systems, it outlines the overall process an organization must implement to manage risk and implement adaptive and layered security within the operating environment.

The Risk Management process is a continuous cycle, so the process of developing software for a CI/CD pipeline should be model on a continuous assessment, patch/update, package, deploy and assessment cycle. To make this successful, all
levels of management need to be supportive of the process, not just Incident Response team, nor Security and Risk Management teams. A disruptive task for DevOps is to delay new feature releases to clean up prior mistakes. The process flow should integration patch life cycle management and testing into the integration, build, orchestration, and deployment process.

Figure 1.2: Defense in Depth – Homeland Security, 2016

Patch Management, Secure Systems Development, along with Testing and Audit Methodologies have a direct impact on overall software hardening for applications deployed on Open Shift Container platforms or standard operating systems. Software development when aligned with a formal Security Life Cycle approach as outlined in NIST SP 800-39 along with the NIST Framework for Improving Infrastructure Cyber Security. (NIST, 2018) will lead to higher quality, lower risks, and rapid corrections when vulnerabilities are detected.

2. Security of Open Shift Containers

2.1. DevOps Continuous Integration and Deployment

To minimize the risk of future software related incidents, the goal is to start with changes to the Development Operations Continuous Integration (CI) and Continuous Deployment (CD) life-cycle processes. The components used by developers are almost always never standardized, freely shared among multiple

Author: Kurtis.Holland@Gmail.com

Author Retains full rights.
sources, and original creditor of where the source code came from is lacking. When working with open source from the Internet, such as sourceforge.net, github.com, gitlab.com, docker.io, and other Internet sources, not only must the security be validated, but your also need to understand the license for usage. Ensure that you build in gates or check points into your life cycle to scan the code going in to the repositories, scan the compiled instances, and scan it during promotion.

The CI/CD pipeline tools must be seamlessly integrated with Developers not to be a burden in their productivity, but to improve the overall product delivered. Focus on products that allow integration with the development, design, packaging, and deployment of the product with proper integrity and ownership tracking within your internal repositories.

Tools for scanning the code structure and security, as well as tools for deployment from your desktop to a runtime environment, such as a dedicated development/integration test environment, a certification or pre-production environment as well as a production run-time environment. Key requirements of your pipeline require the delivery of stable and secure software with the ability to easily replace and upgrade when necessary.

For containers, this means the software is also immutable, in that it never changes during runtime, and that it has permanence, in that you know what content it contains. The best way to patch or upgrade containers in the CI/CD pipeline is to simple replace the entire container by promoting a new instance.

### 2.2. Software Secure Design

Developers do not always write applications using the same methodologies, the same languages, or on the same platform. Various integrated development environments exist for Java and .NET for both local compilation or centralized build environments. It is important for your environment to select the proper tools for static code analysis, dynamic code analysis, as well as validate any open source license dependencies when you go the “build you own” route. Commercially
purchased products have similar requirements, but will require different compliance testing when you don’t have the source code.

When source code is scanned by security tools designed to find flaws, the reward will be many pages of detail on each vulnerability. Common Vulnerability Scoring System (CVSS) (FIRST.org, 2018) is used by many vendors to risk score the found flaw as a qualitative low, medium, or high. Vendor provided code, as well as many open source product packages that have previously reported issues will also have risk scoring based on a common vulnerabilities and exploits (CVE) when problems are reported and documented by CVE Numbering Authorities (CNA).

Many patches or bug fixes released by 3rd parties will have the CVE listed using the format “CVE-YYYY-NNNN” where YYYY is the year and NNNN the CVE identifier. Scoring with be 0 to 10.0 with Low, Medium, High, and Critical status. The three products that will be discussed next are commercially available, as well as some open source versions with limited functionality. The integrate will with tools such as Jenkins, Maven, and Team City, and others, for integrating with your CI/CD pipeline and for creating a code nexus for placing artifacts in the various stages of development and testing.

SonarQube, Veracode SAST and DAST and Sonatype Lifecycle and Repository Server are three products for scanning Java applications, Web Application Programmable Interfaces (APIs), and the Source Code License and version information. Expect some initial push back from the idea of DevOps, until technical leaders become comfortable with its processes, including scans and pipeline sequences that segregate their hands-on and automate the promotion processes.

2.2.1. SonarQube

Even with the benefit of prewritten source code you have pulled from the Internet, or pulled from your corporate re-use repository, you are not guaranteed it will be free from errors. Continuous Integration and Deployment (CI/CD) is successful with tools and process. Orchestration and Builds can be automated for efficiency, quality, operability, and security. SonarQube (SonarQube 2018) is a candidate to integrate with your initial DevOps pipeline. This product currently
supports 20+ languages, for continuous inspection, bug detection, go/no-go gates for permitting or blocking deployments all to assist with improving security and stability of the source code delivered. A project profile and dashboard will provide ease of version tracking and at-a-glance status during each scanning cycle.

![SonarQube Scanning Architecture Overview](SonarQube, 2018)

**2.2.2. Veracode SAST and DAST**

Veracode provides several products for static and dynamic testing of applications. (Veracode.com, 2018). Extensive support to analyze generated packages for both static or dynamic security testing is available. Languages support includes Java, C/C++, Visual Studio .NET, COBOL, RPG, JavaScript, ASP.NET, Classic ASP, PHP, Python, Ruby on Rails, plus Mobile IOS and Android. The current release of the scanning policy leverages the OWASP 2017 Web Application design best practices (OWASP.org, 2018), and Veracode offers Secure Development training for multiple products and developer skill levels.

The Static Analysis or SAST is a very detailed and flexible scanner for packaged EAR, WAR, and JAR packages to validate code structure, detection of known vulnerabilities, security, encryption, credential management, DLLs and other code dependencies. You can upload packages to the Software as a Service (SaaS) web site manually or directly from your Integrated Development Environment (IDE)
for Microsoft Visual Studio, or Eclipse as well as integrated API for build tools such as Maven (Apache Maven, 2018), or Jenkins (Jenkins, 2018). Results of the code scan have summary and detailed reporting, including code examples on remediation reducing the mean-time-to-repair (MTTR) of the vulnerabilities.

A Dynamic Analysis or DAST is a scan run against a running Web Application Uniform Resource Locator (URL) with an authorized credential to walk the web structure and validate code structure and any form input with report-only or attempt to exploit suspected vulnerabilities. You would do this routinely on your pre-production or certification systems as soon as you reach the level of functionality desired as part of your development process. Like the SAST testing, you build your profiles, projects, and policies on the SaaS portal and schedule scanning when appropriate. There is also an option to deploy a DAST service on your internal network to address internal-facing applications not exposed to 3rd party external networks.

Figure 2.2.2. Veracode SAST Continuous Integration (Veracode, 2018)
2.2.3. Continuous Integration

DevOps teams as part of their modular code development methodology will be taking code piece by piece and assembling it into larger and larger pieces. Multiple programmers will be contributing their components to the overall project. Consider the Continuous Integration and Build process in Figure 2.2.3 below. DevOps teams create their projects in a code repository such as a Git Bitbucket pipeline (Bitbucket 2018) which has a user interface assessable via a “web hook” for source code projects. Jenkins is used to pull the software and push into Maven to orchestrate the build and deploy into Open Shift’s temporary registry. From there is it pulled into a temporary Open Shift project for acceptance tests, and when successful, the source and image are tagged and pushed with other artifacts into Sonatype Nexus repository for staging. Sonatype IQ Life Cycle server (Sonatype 2018) will then scan the artifacts for compliance. When successful, Sonatype will promote the Staging artifacts into a Production Repository, and notification results will be sent back to Jenkins which will launch the orchestration and deployment part of the pipeline.

Continuous Integration

2.2.4. Sonatype Nexus Lifecycle and Repository

Once you start up a software project, you need a development pipeline where you can store your various artifacts to deploy in various stages of the product maturity. For source code, not only do you need the code quality you get from SonarQube or Veracode, you need the ability to validate licensing, version
signatures, CVE validation, and notifications of new vulnerabilities and which code is affected. Sonatype brings both a repository, a policy, and a reporting process to scan software for compliance to both security and legal requirements for product licensing. With an appropriate policy configuration, you can determine what you do or do not want to permit in your code base for both CVE severity or Open Source license categories.

Sonatype Nexus Lifecycle or IQ (Sonatype, 2018) provides the visibility throughout your software lifecycle to manage software policies, integrate with your developer tools such as Maven, Jenkins, Eclipse, IntelliJ, Visual Studio and others. Rest API's provided allow for custom integrations with orchestration engines and eGRC tools. All the effort to build and scan software requires a repository for the packaged binary artifacts outside your normal Git bitbucket repository. Nexus Repository is such a product allowing the ability to place images for manual or automatic deployment through provided API's bringing all the CI/CD pipeline tools together in the reference architecture suitable for DevSecOps.

![Sonatype IQ Server Policy Violations](image)

**Figure 2.2.4 Sonatype IQ Server Policy Violations**
2.2.5. Continuous Deployment

As you see from the process of the DevOps Continuous Integration, it is all about the integration, build and packaging of the runtime artifacts. Once the builds are successful, the next phase is to orchestration the code deployment. This is where you leverage our buddy Jenkins and new tools such as Ansible Tower (Red Hat Ansible 2018) and Cloud Forms (Red Hat Cloud Forms 2018) as part of the Open Shift Container deployment. The Cloud Forms and OpenSCAP components will be discuss further in Section 2.3. Jenkins in this use-case will be initiated by a Software Change Management environment.

Jenkins will trigger an Ansible process to execute the playbooks that are created as part of the build process and packaged in Git and Sonatype. Ansible will look in the temporary repository for the Container to deploy, and if not found, pull from Sonatype. Jenkins will then run a series of queries to obtain the status of the scan Cloud Forms performs on the container. A successful scan will result in an Open Shift promotion to a run-state. Ansible then initiates some basic tests. Jenkins will notify the Service Management system that a success deployment has occur, or has failed.

Figure 2.2.4 Continuous Deployment
2.3. Container Security Layers

The DevOps teams have now packaged an artifact and are in the process of deploying into the OpenShift environment. A security in depth strategy will layer the security in your infrastructure, and for the software components, layer the controls that harden the configuration and interoperation of the various microservices running on the container infrastructure nodes. The software developed for deployment will need a secure platform to run on. Red Hat Enterprise Linux (Red Hat, 2018), was developed from open source and over the past several years has matured in functionality and security.

The system engineers that manage the OCP infrastructure will work with the DevOps team for securing access to the running OpenShift environment for CI/CD deployments of the secure image, often called the “Gold Image”. The application image in a container will have limited access to host Linux OS services, and with OpenShift, deployment on SELinux is the most common method. Persistent storage will be allocated properly secured only for the targeted container.

2.3.1. Container Security Guidelines

The basics for running multiple containers on the same physical (or virtual Linux) node is to make sure they are properly isolated from each other. Red Hat OpenShift has a security guide (Red Hat, 2018) outlining processes for the secure build, network, platform, storage, and monitoring YAML configs and Ansible Tower playbooks. With SELinux you have segregation by non-root containers, unique namespaces, mandatory access control (MAC), control groups (CGROUPs) for isolation of CPU, memory, disk I/O, network), and Secure Computing Mode (seccomp) to restrict available system calls.

OpenShift is based on Docker so most of the build processes leverage Dockerfile command structure for the build process. Proper naming conventions, tagging, management of credentials, environment variables, establishing network ports, and how you add or link resources such as storage need to be planned. It is key that you limit microservices to individual containers, avoid root, avoid sshd, and
ensure proper persistent data is setup. The challenge with numerous DevOps projects will be ensuring the configuration and Dockerfile parameters do not overlap or leverage excessive user permissions that could compromise the stability or security of the cluster.

Usage of normal credentials in a container will be problematic as /etc/passwd does not exist, so specifying users for running a container will have a numeric value, not a name. Passing parameters to or from a container for credential authentication or authorization will leverage Kubernetes secrets either attached to persistent storage, or as part of an API call, and properly managed for key and password rotation frequency.

Figure 2.3.1. Red Hat Open Shift Architecture

2.3.2. OpenSCAP – Let us Vulnerability Scan

The Red Hat Security Guide (Red Hat, 2018) includes integration with the Open Security Content Automation Protocol (SCAP) that was designed and tailored to inspect your operating system configuration for compliance. The
product comes with a Workbench for customizing and tailoring the policy; the
OSCAP scanning utility; a Script Check Engine (SCE) for custom written checks; and
a Security Guide. The SCAP tool uses standard extensible checklist descriptors
(XCCDF) and vendor neutral Open Vulnerability and Assessment language (OVAL).
Asset reporting and metrics are provided at the end of the scan in XML and HTML
formats. OpenSCAP leverages a Security Technical Implementation Guide (STIG)
for defining best practice controls for baseline configurations. The SCAP Workbench
is the tool to configure or customize your STIG policy prior to setting up automation
to scan platforms – either from a stand-alone server, or from Red Hat Satellite.

Before running the command on a remote system, you need to fetch the Red
Hat CVE list from https://access.redhat.com and setup remote SSH keys between
the source and target platform. A sudo command for the account used for remote
access from a central scan engine, must be able to run the /usr/bin/oscap utility as
‘root’, using ssh-keys and no password challenge. A potential risk, however, is
running the account as root via sudo – make sure you restrict permissions to only
run the oscap program. The below example is a fully expanded SSG XCCDF Example
7-8 that uses a PCI policy, RHEL SSG configuration, and tailoring of custom checks.
Output reports are sent to an archival location for review.

- oscap-ssh --sudo oscan@target-server 22 xccdf veal --fetch-remote-resources --data
stream-id scap_org.open-scap_datastream_from_xccdf_ssg-rhel7-xccdf-1.2.xml --xccdf-
id scap_org.open-scap_cref_ssg-rhel7-pcidss-xccdf-1.2.xml --tailoring-file /scap-
security-guide-0.1.38/ssg-rhel7-ds-tailoring.xml --profile
xccdf_org.ssgproject.content_profile_pci-dss_centric_customized --oval-results --results
/oscap/reports/target-server.xccdf-results.xml
--results-are /oscap/reports/target-server.arf.xml --report /oscap/reports/target-
server.report.html /scap-security-guide-0.1.38/ssg-rhel7-ds.xml

The output will return three results with an HTML report containing the summary
and overview of the rules passage or failure. Figure 2.3.2 is the score card of a scan.
As you can see from a default OpenSCAP PCI report for a non-hardened distribution, there are numerous system configuration setting to adjust, such as Aide, Auditd, SSH defaults, Password defaults etc. as these rely mostly on customer specific settings. There is a feature in OpenSCAP that would allow remediation of a finding to that expected in the policy and that is to specify the –remediate command. A better solution would be to leave this to the system administrators by importing the scan ARF output in an Enterprise Governance Risk Compliance (eGRC) tool such as RSA’s Archer creating a Security Finding to be assigned to the appropriate System Administrators group to address through normal change management procedures.
2.3.3. Scanning Containers with Cloud Forms

OpenSCAP has an option to run directly against a Docker container, however, if you’re going to manage multiple Red Hat Clusters on Public and Private Clouds, Red Hat has a tool called Cloud Forms. Cloud Forms integrated with the Master nodes using API keys with a role to monitor, detect, and scan containers placed in the deployment registry used by Open Shift in staging deployments. When setup with a daily schedule to pull the current CVE list from Red Hat, and scan the container repository, you get a consistent real-time view of new CVE’s to existing containers to alert and report on non-compliant images.

For the DevOPS CI/CD pipeline as discussed in Figure 2.2.4, Jenkins and Ansible are running the container deployment pipeline. When Cloud Forms completes the Container scan in the temporary Docker registry it will tag the image Compliant or Non-Compliant. Jenkins can get this status from a query to the OCP Cluster Manager for the image it is looking for by running this command:

- `oc get istag <image>:<tag> --template='{{index .image.metadata.annotations "images.openshift.io/deny-execution" }}'`
- From there, it will return **true/false** or **empty**. If it returns **true**, the scan has failed with a Non-Compliant result and denying execution of the container, or if it returns **false**, the scan has passed. If the value is **empty** the scan hasn’t been completed yet.

You can also configure Cloud Forms to alert but not block the promotion in Development clusters via policy and allow the CI/CD DevOps process to flow unabated allowing flexibility with remediation timelines. For higher promotions into Production – set the policy to block HIGH or Critical severity detections.

Features within Cloud Forms allow you to schedule periodic scans of containers in the repository to track against the daily CVE updates downloaded from [https://access.redhat.com](https://access.redhat.com). Trending and analysis reporting is available within the database available via the user interface or through API’s. Below is a sample report from a development environment just getting started with the integration, build, and test deployment of a new container package. The OpenScap details will
flag High, Medium, Low issues, so don’t be discouraged on new DevOps where it will take some time to work out the CVSS and CVE issues on a new build.

![Cloud Forms Scan report of Container Images](image)

Figure 2.3.3. Cloud Forms Scan report of Container Images

### 2.3.4. Infrastructure Scanning utilizing Satellite with OpenSCAP

The DevOps teams need their scanning tools and repositories to track software projects, versions, and compiled artifacts. The System Administrators for managing servers and operating systems need viable tools of their own. Management of server assets, tracking their vulnerabilities, and promoting updates is a similar process to DevOPS CI/CD pipeline. The architecture of Satellite provides for establishing a lifecycle and promotion of system and security configuration changes using a product called “puppet”, which is like Ansible Tower, a playbook of configurations that are scripted and deployed through automation to improve accuracy, speed, and minimize manual errors.

Managing assets involves a scripted playbook which is used as part of an orchestration of changes to apply to a server or group of servers. The design supports both private and public cloud deployments from a central database and remote orchestration and caching servers called “capsules” allowing for local data center deployments. As part of this architecture, OpenSCAP scans can be launched using the same Red Hat SSG profiles and policies as done with the standalone OpenSCAP deployments. With Satellite server and configuration management, along with the patch vulnerability remediation, you are providing a stable...
infrastructure for your applications to run on, both standard virtualization as well as OpenShift containers.

Since Satellite is used for lifecycle management, it is also a very convenient Incident Handling tool if you need to find every system vulnerable to a specific CVE, and all the demographics registered for that server in the configuration management database (CMDB). Satellite’s architecture as shown in Figure 2.3.3 below is designed to have a high available central database, orchestration engine and user interface. Remote capsules, or proxies are place in the data centers closets to the assets you are going to manage. Puppet, a tool very like Ansible allows you to create modules to perform specific scripting for configuration, such as Apache, MySQL, Tomcat, JBoss, RPM deployments, YUM installs, and so forth. Access to the managed services is established with encryption keys and credentials ensuring a reliable, security and deployment with the utmost accuracy and integrity.

![Figure 2.3.4. Red Hat Satellite 6 – Central and Remote Capsules (Red Hat, 2018)](image)

**2.3.5. Infrastructure Logging and Monitoring**

The security hardening so far discussed surround the static scans of the initial code, the packaged software, the infrastructure operating system, and the packaged Container. A well-orchestrated and repeatable lifecycle for code builds and
deployments will greatly increase your overall application stability and security. What has not been discussed is what tools or processes are recommended for scanning runtime environment in near real-time for issues or anomalies that might occur? Along with application and system event logs, you need tools to monitor the system and application configuration and detect unscheduled changes.

A feature of Satellite, and Cloud Forms is that you can execute the playbooks and deployment scripts in read-only mode not making any changes. This works well for configuration and properties files, but not as effective on binaries. For the thousands of system files, and those deployed with an application, your need a cryptographic hash code of the file and a database to manage known system and application software. A SHA256 or HMAC256 cryptographic hash code on the binary code or system libraries is necessary to validate file integrity monitoring (FIM). There are options for commercial or open source deployment to choose from. Products such as Tripwire (commercial), and Open Source Security (OSSec) and Advanced Intrusion Detection Environment (AIDE) have been around industry for many years. A properly configured FIM tool with near real-time tamper detections, logging, and alerts is an invaluable tool for system security, and incident indications of compromise.

3. Conclusion

As an Incident Handler, the last thing you want to be faced with at 02:00 in the morning during an outage is wondering the status of your application or operating environment., Conducting mock incident drills periodically and asking questions surrounding current patch status, CVE gaps, or susceptibility to a threat is a good way to help drive the need to deploy infrastructure for your DevOps CI/CD processes and System Administration life cycles. If you don’t know by looking at a dashboard, or running a simple report what the snapshot of your environment looks like, then immediately start a project like what was outlined above.

Selecting the proper reference architecture to support the DevOps process and ease the transition into full DevSecOps in an operating environment is a key risk and

Author: Kurtis.Holland@Gmail.com
security threat minimization goal. It is important that part of a defense in depth strategy that you properly assess the risk of your applications and infrastructure all while looking for the cost-benefit analysis of tools to assist in the automation. With proper executive sponsorship, making changes to your DevOps life cycle will not have to be driven as an emergency following a security incident. You will be able to sell this architecture, or something very similar in how you improve quality, reduce outages, security threats, and improve efficiency and time to market. Ensure that you include tools to assist in the integration, the build, the deployment, and the run-time of your applications. Static and Dynamic testing is an absolute must.

In summary, hardening your application containers and infrastructure is more than just a security best practice. Layer your infrastructure defenses with technology, process, training, and governance. Build and design your software development pipeline to complement your DevOps teams and ease their ability to securely deploy newly designed products. Yes, it will be rough road to travel at first, but that will ease after you get most of the non-compliant software out of our code repositories and replaced with newly patched and scanned content. Make friends with your developers and understand the business needs to push the envelope and risk with new technology. Do not make a mistake and scan once or twice and get your product into production and declare victory. The war against hackers is an ever-vigilant campaign fought battle by battle as each new threat or vulnerability occurs.
References

Metro-Goldwyn-Mayer. (1939), The Wizard of Oz


SonarQube (2018), Architecture and Integration, Retrieved May 15, 2018 from https://docs.sonarqube.org/display/SONAR/Architecture+and+Integration


Author: Kurtis.Holland@Gmail.com

Author Retains full rights.
Hardening Open Shift Containers to Complement Incident Handling


Bitbucket (2018), Git Pipeline, Retrieved May 15, 2018 from [https://bitbucket.org/product/features/pipelines](https://bitbucket.org/product/features/pipelines)


Author: Kurtis.Holland@Gmail.com  
Author Retains full rights.
Hardening Open Shift Containers to Complement Incident Handling


# Upcoming SANS Training

Click here to view a list of all SANS Courses

<table>
<thead>
<tr>
<th>Event Name</th>
<th>City, Country</th>
<th>Dates</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Zurich February 2020</td>
<td>Zurich, CH</td>
<td>Feb 24, 2020 - Feb 29, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Secure India 2020</td>
<td>Bangalore, IN</td>
<td>Feb 24, 2020 - Feb 29, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Manchester February 2020</td>
<td>Manchester, GB</td>
<td>Feb 24, 2020 - Feb 29, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Jacksonville 2020</td>
<td>Jacksonville, FLUS</td>
<td>Feb 24, 2020 - Feb 29, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Secure Japan 2020</td>
<td>Tokyo, JP</td>
<td>Mar 02, 2020 - Mar 14, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>Blue Team Summit &amp; Training 2020</td>
<td>Louisville, KYUS</td>
<td>Mar 02, 2020 - Mar 09, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Northern VA - Reston Spring 2020</td>
<td>Reston, VAUS</td>
<td>Mar 02, 2020 - Mar 07, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Munich March 2020</td>
<td>Munich, DE</td>
<td>Mar 02, 2020 - Mar 07, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS St. Louis 2020</td>
<td>St. Louis, MOUS</td>
<td>Mar 08, 2020 - Mar 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Dallas 2020</td>
<td>Dallas, TXUS</td>
<td>Mar 09, 2020 - Mar 14, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Prague March 2020</td>
<td>Prague, CZ</td>
<td>Mar 09, 2020 - Mar 14, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>Wild West Hackin Fest 2020</td>
<td>San Diego, CAUS</td>
<td>Mar 10, 2020 - Mar 11, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Doha March 2020</td>
<td>Doha, QA</td>
<td>Mar 14, 2020 - Mar 19, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Norfolk 2020</td>
<td>Norfolk, VAUS</td>
<td>Mar 16, 2020 - Mar 21, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS FOR585 Rome March 2020 (In Italian)</td>
<td>Rome, IT</td>
<td>Mar 30, 2020 - Apr 04, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Training at RSA Conference 2020</td>
<td>OnlineCAUS</td>
<td>Feb 23, 2020 - Feb 24, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Books &amp; MP3s OnlyUS</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>

Last Updated: February 22nd, 2020