SECURITY GUIDANCE FOR 5G CLOUD INFRASTRUCTURES

Part IV:
Ensure Integrity of Cloud Infrastructure

2021

TLP:WHITE

DISCLAIMER OF ENDORSEMENT

The guidance in this document is provided "as is." In no event shall the United States Government be liable for any damages arising in any way out of the use of or reliance on this guidance. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the United States Government, and this guidance shall not be used for advertising or product endorsement purposes. All trademarks are the property of their respective owners.

PURPOSE

NSA and CISA developed this document in furtherance of their respective cybersecurity missions, including their responsibilities to develop and issue cybersecurity specifications and mitigations. This information may be shared broadly to reach all appropriate stakeholders.

CONTACT

Client Requirements / Inquiries: Enduring Security Framework nsaesf@cyber.nsa.gov

Media Inquiries / Press Desk:

- NSA Media Relations, 443-634-0721, MediaRelations@nsa.gov
- CISA Media Relations, 703-235-2010, CISAMedia@cisa.dhs.gov

TABLE OF CONTENTS

BACKGROUND

The Enduring Security Framework (ESF) hosted a 5G study group comprised of government and industry experts over the course of eight weeks during the summer of 2020 to explore potential threat vectors and vulnerabilities inherent to 5G infrastructures. At the conclusion of the study, the group recommended a three-pronged approach to explore this threat space[1]:

1. Identify and assess threats posed to 5G;
2. Determine what standards and implementations can achieve a higher baseline of 5G security; and
3. Identify risks inherent to the cloud that affect 5G security.

In support of this task, the ESF established a 5G Cloud Working Panel to engage with experts across government and industry to document 5G cloud security challenges, threats, and potential mitigations, to include guidance, standards, and analytics. The result of this collaboration is a four-part series of publications that addresses the third task identified by the 5G study group: applying a threat-based approach to identify and mitigate risks in 5G networks that derives from the use of cloud technologies, and providing mitigations that can be applied to harden 5G cloud infrastructures.

SCOPE

This four-part series builds on the ESF *Potential Threat Vectors to 5G Infrastructure* white paper, released in May 2021, which focused specifically on threats, vulnerabilities, and mitigations that apply to the deployment of 5G cloud infrastructures.[2]

Although all 5G network stakeholders can benefit from this guidance, the recommendations are intended for service providers and system integrators that build and configure 5G cloud infrastructures. This includes core network equipment vendors, cloud service providers, integrators, and mobile network operators. The audience for each set of recommendations will be identified throughout the series, providing a layered approach to building hardened 5G cloud deployments.

[1] The ESF is a cross-sector working group that operates under the auspices of Critical Infrastructure Partnership Advisory Council (CIPAC) to address threats and risks to the security and stability of U.S. national security systems. It is comprised of experts from the U.S. government as well as representatives from the Information Technology, Communications, and the Defense Industrial Base sectors. The ESF is charged with bringing together representatives from private and public sectors to work on intelligence-driven, shared cybersecurity challenges.

[2] ESF, *Potential Threat Vectors to 5G Infrastructure*, 2021. https://www.nsa.gov/news-features/press-room/Article/2601078/nsa-odni-and-cisa-release-5g-analysis-paper

5G Cloud Security Challenge Overview

5G networks are being designed to handle the bandwidth, compute, and storage requirements that will be required for a predicted massive increase in network capacity as well as connected devices. For scalability, resilience, and agility, 5G networks leverage cloud infrastructures, both in the radio access network, core, and network edge. Cloud technologies underpin the implementation of virtual networking in 5G, enabling the dynamic allocation and management of networks for specific use cases, mobile network operators, or customers.

A characteristic of cloud infrastructure that presents a significant security challenge in 5G is multitenancy, the use of a shared physical infrastructure by multiple cloud infrastructure customers, e.g., mobile network operators. Multitenancy highlights the need to harden and securely configure technologies that isolate the workloads (e.g., virtualization/ containerization) for each of those customers. In addition, cloud providers and mobile network operators may share security responsibilities in a manner that requires the operators to take responsibility to secure their tenancy "in the cloud." An additional factor creating security challenges is the increasing deployment of a multi-cloud deployment model in 5G with diverse and evolving architectures and design approaches used by wireless carriers.

5G Threat

Among the threat vectors presented in the *Potential Threat Vectors to 5G Infrastructure* analysis paper, several pertained to 5G cloud infrastructure, including *Software/ Configuration*, *Network Security*, *Network Slicing*, and *Software Defined Networking*.

5G networks, which are cloud-native, will be a lucrative target for cyber threat actors who wish to deny or degrade network resources or otherwise compromise information. To counter this threat, it is imperative that 5G cloud infrastructures be built and configured securely, with capabilities in place to detect and respond to threats, providing a hardened environment for deploying secure network functions. It is also important that 5G network functions be implemented using security best practices. This four-part series will address the former, providing guidance on hardening 5G cloud infrastructure deployments that are driven by threat information. This approach supports the May 2021 Presidential Executive Order on *Improving the Nation's Cybersecurity*, which called for secure products and services and enabling easier detection of unexpected behaviors and actions.[3]

5G Cloud Security Guidance

Based on preliminary analysis and threat assessment, the Cloud Working Panel concluded that the top 5G cloud infrastructure security challenges could be divided into a four-part

[3] Executive Office of the President, *Executive Order on Improving the Nation's Cybersecuritys*, 2021. https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity

series that addressed different aspects of securing 5G clouds, facilitating the application of broad sets of mitigations.

- **Part I: Prevent and Detect Lateral Movement**: Detect malicious cyber actor activity in 5G clouds and prevent actors from leveraging the compromise of a single cloud resource to compromise the entire network.

- **Part II: Securely Isolate Network Resources**: Ensure that there is secure isolation among customer resources with emphasis on securing the container stack that supports the running of virtual network functions.

- **Part III: Protect Data in Transit, In-Use, and at Rest**: Ensure that network and customer data is secured during all phases of the data lifecycle (at-rest, in transit, while being processed, upon destruction).

- **Part IV: Ensure Integrity of Infrastructure**: Ensure that 5G cloud resources (e.g., container images, templates, configuration) are not modified without authorization.

Zero Trust is the concept that perimeter defenses are no longer sufficient to secure a network, and that there should always be an assumption that a threat actor has established a foothold in the network[4]. This four-part series will document best practices that strive to bring a Zero Trust mindset into 5G cloud endpoints and growing multi-cloud environments. All actions should be explicitly verified and monitored. Although the best practices documented in this series do not constitute a complete Zero Trust template for securing 5G cloud infrastructures, if the best practices are applied, a 5G cloud environment will have made significant strides toward the implementation of Zero Trust principles.

[4] NIST Special Publication 800-207. Zero Trust Architectures.
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

ENSURE INTEGRITY OF CLOUD INFRASTRUCTURE

The trend in the industry has been to deploy 5G core using virtualized functions on modern microservices based architecture that provides rapid enablement of services. It is imperative that the underlying 5G cloud infrastructure platform on which microservices are deployed or orchestrated have been designed with built in security and continue operating as intended. The intent of Part IV of this series is to provide guidance on platform integrity, microservices infrastructure integrity, launch time integrity, and build time security.

PLATFORM NODE INTEGRITY

Servers, storage, and network devices form the cloud infrastructure platform on which the cloud native 5G core is deployed. Existing mitigations of threats against the nodes are often rooted in firmware or software, making them vulnerable to the same attack strategies. For example, if the firmware can be successfully exploited, then the firmware-based security controls can most likely be circumvented in the same fashion.

Audience: Cloud Providers, Mobile Network Operators, OEMs selling the equipment

Guidance/Mitigation:

These devices have low level firmware running on variety of critical components such as BIOS, disk drive controller firmware, baseboard management controller firmware, SmartNICs, packet processing chips, crypto off load engines and miscellaneous micro controllers required for operation of the devices. Such firmware typically needs an update capability, and lack of this capability potentially leaves the component and its platform vulnerable to rootkit attacks.

NIST has released SP 800-193 focused on providing guidelines to equipment providers. Network designers and operators should pick devices which provide NIST SP 800-193 guided protection, detection and recovery of all rootkit-able firmware[5].

To help ensure a secure and trusted runtime environment, we want to assert that the clusters only run trusted nodes, rooted in hardware root of trust (Trusted Platform Module (TPM)). With hardware as the core root of trust for measurement to establish a chain of measurement for each component, the trust can be extended to higher layers of the software stack. This is consistent with the concept of the chain of trust (CoT) – a method where each software module in a system boot process is required to measure the next module before transitioning control.

UEFI Secure Boot is the most common example of CoT bridging between Hardware anchored secureboot to launch of operating system. Linux then extends the CoT using Integrity Measurement Architecture (IMA) and SSH Key Management (SKM).

NIST guidance on hardware root of trust and attestation is available with NISTIR 8320 Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud 4 and Edge Computing Use Cases[6].

CONTAINER PLATFORM INTEGRITY

Moving up from the hardware device level, ensuring the integrity of the container stack (worker nodes, Kubernetes cluster and containers) is critical for preventing attacks and denying cyber actors the ability to persist. Container stacks should be built to provide assurance in the integrity of the stack, ensuring that the container platform is operating as expected. This section covers integrity of the nodes that support Kubernetes (K8s) clusters, as well as the K8s clusters themselves.

Audience: Cloud Providers, Mobile Network Operators, OEMs selling the equipment

Guidance/Mitigation: The following best practices help ensure the integrity of the container platform:

Harden and optimize operating system for running containers

Ensure that the node operating system that the container platform runs on is hardened against attacks from the container platform and from within the cloud. Follow best practice guidance on securely configuring the operating system, ensuring that the operating system is stripped down to reduce the attack surface and regularly patched to protect against known vulnerabilities. Consider using an operating system that is optimized for running container

[5] NIST SP 800-193 Platform Firmware Resiliency Guidelines. https://csrc.nist.gov/publications/detail/sp/800-193/final

[6] NISTIR 8320 Hardware-Enabled Security: Enabling a Layered Approach to Platform Security for Cloud 4 and Edge Computing Use Cases. https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8320-draft.pdf

workloads. That regularly inspect hosts for exposures, vulnerabilities, and deviations from best practices.

Implement an immutable infrastructure and automate the replacement of worker nodes

Maintain services on nodes (virtualized hosts) in the development environment and update the service in the production environment by maintaining a golden (configured and clean) set of worker node images. Build the nodes from a common, hardened image. Replacing rather than updating the nodes in the production environment is key to an immutable infrastructure, which improves security by avoiding configuration drift and inconsistencies in deployed services. Integral to the concept of Infrastructure as Code, an immutable infrastructure enables deployments of pre-configured, grouped resources (compute, network, storage), key to secure automation. Combined with the attestation requirement, an immutable infrastructure makes it more difficult for attackers to maintain persistence in the container stack.

Maintain a golden (configured and clean) set of worker node images and implement patches and updates on the golden images. Replace running nodes with the golden images when available, rather than patching/updating the nodes while they are in operation.

Harden Kubernetes clusters

Use configurations that hardens a K8s cluster against known attacks. Ensure that a container platform's K8s cluster is hardened by following security guidelines and by regularly running kube-bench against the cluster to detect when the cluster falls out of compliance[7].

Minimize direct access to worker nodes

Opening up direct access to worker nodes greatly increases the risk of cluster compromise. Minimize this risk by disabling direct access (via SSH or other protocols) and using an agent-based system for node maintenance and troubleshooting.

Deploy worker nodes in private subnets unless external access is needed

Worker node subnets should be on private subnets (no access to the Internet) unless explicitly required (e.g., web server). Exposing worker nodes to the Internet greatly increases the threat and attack surface against the container resources, opening opportunities for cyber actors to compromise the nodes and maliciously manage the container resources.

Container placement and orchestration based on container platform integrity

Container platform information and verified firmware and configuration measurements that are retained within an attestation service can be used for policy enforcement in a variety of use cases. One example is orchestration scheduling. Cloud orchestrators, such as Kubernetes, provide the ability to label worker nodes in their database with key value attributes. The attestation services can publish trust and informational attributes to

[7] Securing Kubernetes. https://www.cisecurity.org/benchmark/kubernetes/

orchestrator databases for use in workload scheduling decisions. In addition, the orchestration system should provide visibility into the attestation state of the machines.

LAUNCH TIME INTEGRITY

Before launching a container, we first ensure that the underlying container platform is still trusted. This verification also includes ensuring that monitoring and other runtime controls and policies are active. For example, a Trusted Execution Environment (TEE) with loadable container-specific policies may be provided and enabled by the platform.

With the integrity of the container platform assured, the integrity of each container must be verified before launch. The first step is to assure that the container was loaded from a trusted source, such as a trusted images store. Then the integrity of the container itself is verified.

Finally, the container is launched. At this point, the container's execution will be securely monitored according to the policies specific to the container and container platform. The stack should be constantly monitored using analytics or other means that provide ongoing proof of the secure state of the stack as containers are launched and terminated.

Audience: Cloud Providers, Mobile Network Operators, OEMs selling the equipment

Guidance/Mitigation:

Built-in security with simplified assessment capabilities to standard benchmarks will help facilitate scaling security management, eliminating custom configurations for each entity. This is detailed in Scalable Remote Attestation for Systems, Containers, and Applications[8].

CONTAINER ENCRYPTION/ DECRYPTION OF IMAGES ON TRUSTED PLATFORMS

Consumers who place their workloads in the cloud or on the edge are typically forced to accept that their workloads are secured by their cloud service providers without insight or knowledge as to what security mechanisms are in place. The ability for users to encrypt their workload images can provide at-rest cryptographic isolation to help protect consumer data and intellectual property. When the runtime node service receives the launch request, it can detect that the image is encrypted and make a request to retrieve the decryption key. This request can be passed through an attestation service to a key broker with proof that the platform has been attested. The key broker can then verify the attested platform report and release the key back to the Cloud Service Provider and node runtime services. At that time, the node runtime can decrypt the image and proceed with the normal workload execution. The disk encryption kernel subsystem can provide at-rest encryption for the workload on the platform.

[8] Scalable Remote Attestations for Systems, Containers, and Applications. https://datatracker.ietf.org/doc/draft-moriarty-attestationsets/

Encrypted Container Images is a capability introduced in container build tools and runtimes that allow the encryption and decryption of container images. This is based on the Open Container Initiative container standard and ensures confidentiality of container images as soon as they leave the pipeline, until they are run on a trusted compute node with access to the decryption keys. In an event of a registry compromise, this ensures that the confidentiality of container images stay intact, and we can cryptographically associate trust with images.

Figure 1 shows a generic workflow to implement container encryption, decryption and execution on trusted platforms (that have demonstrated integrity) with customer-controlled keys.

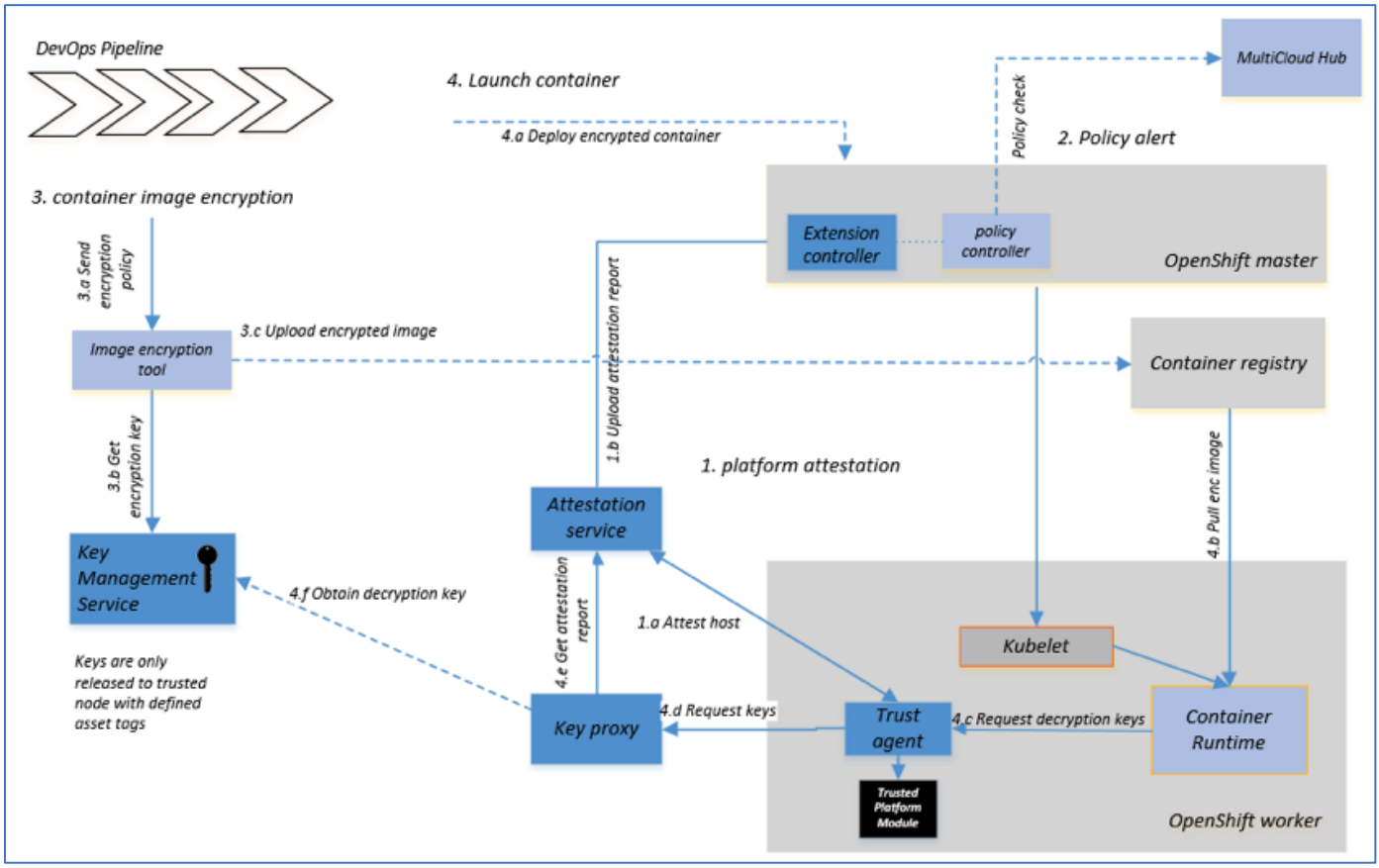

Figure 1: Generic workflow implementing container encryption, decryption, and execution

As hardware-based Trusted Execution Environment (TEEs) become more pervasive and broadly enabled for use with container platforms and runtimes, it would be advisable (and recommended) to consider running containers in TEEs to reduce the attack surface for containers, and to keep the Service providers and malicious insiders outside the Trusted Computing Base. See the TEE discussion in *Part II: Securely Isolate Network Resources* of this series for more information.

CONTAINER IMAGE HYGIENE

The container image is your first line of defense against an attack. An insecure, poorly constructed image can allow an attacker to escape the bounds of the container and gain access to the host. Once on the host, an attacker can gain access to sensitive information or move laterally within the cluster.

Audience: Cloud Providers, Mobile Network Operators, OEMs selling the equipment

Guidance/Mitigation: The following best practices will help mitigate risk of this happening.

Build images from scratch or create minimal de-fanged images

Reducing the attack surface of a container image should be a primary aim when building images. The ideal way to do this is by creating minimal images that are devoid of binaries that can be used to exploit vulnerabilities. As an example, if your container software has a mechanism to create images from scratch, it should be used. Programming languages can create a static linked binary that can be directly referenced in the container file. Creating containers in this manner ensures that the image consists of only the application, greatly reducing extraneous attack surface.

If unable to build an image from scratch, developers should still seek to reduce the attack surface inside a container by removing extraneous binaries from the container image. Inspecting container images using an application that allows the developer to see the contents of each image layer. Remove all binaries with `setuid` and `setgid` bits, as they can be used to escalate privilege, and consider removing all shells and utilities such as `nc` and `curl` that can be used for nefarious purposes.

Third-party software should also undergo an enterprise security review that includes code inspection, threat modeling, and penetration testing to identify and mitigate risks.

Use multi-stage builds

Using multi-stage builds is a way to create minimal images. Oftentimes, multi-stage builds are used to automate parts of the Continuous Integration (CI) cycle. For example, multi-stage builds can be used to lint your source code or perform static code analysis. This affords developers an opportunity to get near immediate feedback instead of waiting for a pipeline to execute. Multi-stage builds are attractive from a security standpoint because they allow you to minimize the size of the final image pushed to your container registry. Container images devoid of build tools and other extraneous binaries improves your security posture by reducing the attack surface of the image.

Scan container images for vulnerabilities regularly

Container images can contain binaries and application libraries with vulnerabilities. The best way to safeguard against exploits is by regularly scanning images and quickly mitigating identified vulnerabilities. Additionally, knowing where images with vulnerabilities have

been deployed can help forensic efforts when vulnerabilities are exploited. Several commercial offerings provide scanning and other advanced capabilities.

Restrict access to container image repositories

Create policies that restrict development teams to access only to repositories with which each team should interact.

Create a set of curated container images

Rather than allowing developers to create their own images, security administrators can create a set of vetted images providing different application stacks for developers. By doing so, developers can forego learning how to compose container specifications and concentrate on writing code. As changes are merged into a Master, a CI/CD pipeline can automatically compile the asset, store it in an artifact repository, and copy the artifact into the appropriate image before pushing it to an image repository.

Alternatively, security administrators can create a set of base images from which developers create their own container images. Base images should be vetted and regularly scanned for vulnerabilities. Additionally, administrators should ensure that the image was published by a reliable entity such as the developer of a reputable product.

Add the USER directive to run as a non-root user

As mentioned in the pod security section, of *Part II: Securely Isolate Network Resources* of this series, avoid running container as root. While you can configure this as part of the podSpec, it is a good habit to use the `USER` directive. The `USER` directive sets the UID to use when running `RUN`, `ENTRYPOINT`, or `CMD` instruction that appears after the `USER` directive.

Lint your container images

Linting can be used to verify that a container image adheres to a set of predefined guidelines, such as the inclusion of the `USER` directive and image tagging. There are tools and resources available that can help to verify common best practices and administrator defined requirements. Linting can be incorporated into a CI pipeline to reject builds that violate the organization's policy.

Use immutable tags with your images

Some image repositories support immutable tags. This forces you to update the image tag on each push to the image repository. This can thwart an attacker from overwriting an image with a malicious version without changing the image's tags. Additionally, it gives you a way to identify an image easily and uniquely.

Sign your container images

As an example, Docker adds digests to the image manifest that allow an image's configuration to be hashed and the hash to be used to generate an ID for the image. When image signing is enabled, the container (Docker) engine verifies the manifest's signature of each layer, ensuring that the content was produced from a trusted source and no tampering has occurred. Image signing enhances supply chain security, through the verification of digital signatures. Kubernetes provides a dynamic admission controller to verify that an image has been signed.

Update the packages in your container images

You should update the packages used in container images to ensure you have the most up-to-date and secure packages. As an example, include RUN `apt-get update && apt-get upgrade` in your Docker files to upgrade the packages in your images. Although upgrading requires you to run as root, this occurs during image build phase. The application doesn't need to run as root. You can install the updates and then switch to a different user with the USER directive. If your base image runs as a non-root user, switch to root and back; don't solely rely on the maintainers of the base image to install the latest security updates.

Run apt-get clean to delete the installer files from `/var/cache/apt/archives/`. You can also run `rm -rf /var/lib/apt/lists/*` after installing packages. This removes the index files or the lists of packages that are available to install.

CONCLUSION

5G networks are being designed and deployed to handle the bandwidth, compute, and storage requirements that will be required for a predicted massive increase in network capacity as well as connected devices. This makes 5G standalone core networks, which are cloud-native, a lucrative target for cyber threat actors who wish to deny or degrade network resources or otherwise compromise information. A secure foundation ensures the necessary mitigations are implemented and carried forward, further strengthening the network, services, and data.