Collaborative DevSecOps in a Service Provider Environment

ATIS-I-0000082 | March 2021

Abstract

Telecom service providers are increasingly migrating their software development, deployment and lifecycle environments to a cloud native architecture. This migration dovetails with a second trend: the use of DevOps for continuous delivery models and automation of system management. When security expertise and responsibilities are tightly integrated within the DevOps processes, the result is DevSecOps. Together, cloud native and DevOps/DevSecOps enable the creation of loosely coupled systems that are scalable, resilient, manageable, observable and secure.

Cloud native and DevOps trends have been led by enterprise IT. However, service providers have unique requirements that may make it impractical to simply import enterprise cloud native architectures and practices. Specifically, the service provider environment often has more stringent requirements for security, resiliency, availability, scalability and performance due to a larger, more diverse customer base that may be covered by a variety of SLA requirements. In addition, service provider networks must meet these more stringent requirements given a highly diverse environment where many different vendors and integrators must closely collaborate.

This report explores the unique challenges associated with collaborative DevSecOps in a service provider cloud native environment. It also provides best practices for creating and maintaining a secure environment.

Foreword

As a leading technology and solutions development organization, the Alliance for Telecommunications Industry Solutions (ATIS) brings together the top global ICT companies to advance the industry’s business priorities. ATIS’ 150 member companies are currently working to address network reliability, 5G, robocall mitigation, smart cities, artificial intelligence-enabled networks, distributed ledger/blockchain technology, cybersecurity, IoT, emergency services, quality of service, billing support, operations and much more. These priorities follow a fast-track development lifecycle from design and innovation through standards, specifications, requirements, business use cases, software toolkits, open source solutions and interoperability testing.

Copyright Information

Table of Contents

1 Executive Summary

Network service provider environments include deployment scenarios where large, complex systems are created through the integration of a variety components provided by a multitude of vendors. *Section 3 – Use Cases* discusses examples of these deployment scenarios in more detail.

Service providers are increasingly leveraging cloud native software development, deployment and lifecycle environments to implement the use cases noted above. Cloud native technologies enable service providers to build and run scalable applications in modern, dynamic environments using containers, service meshes, microservices, immutable infrastructure and declarative APIs to capture the key benefits of a cloud native architecture. Combined with DevOps, continuous delivery models and automation, this approach enables the creation of loosely coupled systems that are scalable, resilient, manageable and observable.

In a cloud native DevOps model, development and operations teams are no longer "siloed." Instead, they are merged into a collaborative team where the engineers work across the entire application lifecycle, from architecture, design, development and test to deployment and ongoing operations. DevSecOps refers to scenarios where security expertise and responsibilities are also tightly integrated within the DevOps processes. DevSecOps incorporates security culture, practices and tools to drive visibility, collaboration and agility, ensuring security in each phase of the DevOps pipeline. This focus on security and performance is particularly important in network service provider environments given the unique needs in providing wide-area communications services. Specifically, these networks are characterized by such factors as:

Complex Network Models: Unlike many IT environments and applications, service provider workloads generally exist in a complex and expansive network environment (the WAN) and may require sophisticated network models to support needed network capabilities.

Service Function Chaining: In IT environments, applications and services generally represent discrete and independent workloads. However, service provider applications are often comprised of complex combinations of services that may have interactions (e.g., policy, billing or legal intercept purposes) and must be configured together as an application through which traffic needs to be correctly steered based on dynamic information.

High-Performance Data Plane: Service provider networks require very high-scale data transfer and packet inspection capabilities associated with the large amount of traffic traversing through the network.

Enhanced Platform Awareness: The scale, diversity and service level agreement (SLA) driven reliability requirements for service provider networks demand a high level of platform awareness to efficiently provide topology, fault and performance management functions.

The successful integration of security aspects into a DevOps model affects project governance, organizational culture, processes and technology.

Collaborative DevSecOps refers to these deployment scenarios where a DevSecOps execution path must also deal with the continuous integration of third-party software from multiple vendors. Thus, the normal DevSecOps process must also collaborate with multiple third parties.

Successful collaborative DevSecOps environments require a cultural shift to support:

- Greater team integration by getting people to know one another (across teams) and building relationships that can be leveraged to better understand the questions and key issues faced by each function. In some cases, it may make sense to have certain people explicitly belong to multiple teams.
- Improved communication channels to enable and encourage interaction between the teams. This may include use of best-in-class collaboration tools.
- Improved management processes to foster knowledge sharing and a blame-free culture, as well as process changes that better support collaboration and information sharing.

Every software lifecycle may have its own suite of development and management processes. However, there are several common process attributes that are useful in most service provider DevSecOps projects. Specifically, robust DevSecOps process suites will generally support:

- Close collaboration through all phases of the development cycle, as well as collaborative process design strategy utilizing multidisciplinary teams.
- Continuous automated testing and test-driven development where software requirements are converted to test cases before software is fully developed. This includes tracking all software development by repeatedly testing the software against all test cases.

- A high degree of automation to support continuous build, integration, delivery, deployment, operation and monitoring.
- Continuous process adaption and improvement capabilities inherent in each process

In addition, DevSecOps processes can be stressed when collaborators are from separate, independent companies or organizational domains. As such, it is useful when the collaborative DevSecOps environment can enforce both process and technology alignment between the various actors participating in the system's creation and management.

Supply chain threats and their associated attacks may occur at each stage in the DevOps development lifecycle, including:

- System design
- Implementation/code/integration
- Iterative testing
- Deployment
- Ongoing support and update

Collaborative DevSecOps environments must include various technical controls, process controls and management techniques to minimize the risk of supply chain attacks.

To mitigate attacks, the DevSecOps practice may focus on specific areas of attention, including:

- Common infrastructure constructs such as a robust microservices-based infrastructure architecture with built-in security capabilities such as a library of hardened containers meeting specific security requirements.
- Uniform configuration management mechanisms to enable consistent, controllable and maintainable overall system configuration.
- A common set of automation, testing and monitoring/logging tools to ensure each module of the system maintains a consistent level of assurance relative to integrity of operation and information security.
- Ongoing monitoring and verification of the integrity of the development environment itself.

Key run-time services that support a DevSecOps focus include:

- A logging agent to push logs to a platform-centralized logging service. Including a common logging service with a consistent output data format is an important factor in understanding system behavior and successful root cause analysis of events.
- A container policy enforcement function to verify that container hardening is preserved and complies with requirements.
- Runtime defense checks that can provide both signature-based and behavior-based detection and can send notifications when there is anomalous behavior.
- Vulnerability management capabilities.
- Verification of container-level zero trust mechanisms.

The use of zero trust controls between containers should also be considered to ensure security in highly collaborative environments. Key aspects of zero trust at the container level include mutual Transport Layer Security authentication (mTLS), an encrypted communication tunnel between containers, strong identities per data center server cluster using certificates and use of "allow" lists.

Automation, testing and monitoring/logging are key capabilities to help ensure the integrity and security of a software development and integration project. Automation should be applied to all aspects of the development cycle that are automatable. Automation provides many advantages, including:

- Fast execution of the task to be automated.
- Minimization of manual interaction, which limits "human mistakes" inherent in any manual process.
- The ability to continuously enhance the automation to incorporate new findings (e.g., detection of new vulnerabilities).
- Repeatability and completeness of the automated task.

Automation can generally be applied to the build process itself, testing functions and security specific functions where software scans can run continuously to identify potential problems that may be inserted at any point in the development process.

Application of collaborative DevSecOps principles and practices can create a more secure and efficient environment for network service provider use cases.

2 Introduction

This report discusses the unique challenges associated with collaborative DevSecOps in a service provider cloud native environment. It also provides a set of recommended practices and techniques that can be used to create a secure environment. Most industry work on DevSecOps has assumed an enterprise-like cloud native environment that may not fully consider a service provider network's unique requirements and aspects.

Specifically, the service provider environment often has more stringent requirements for security, resiliency, availability, scalability and performance due to a larger, more diverse customer base that may be covered by a variety of SLA requirements.

Given the complexity and multi-tenancy attributes of service provider functions and services, the deployment model often utilizes multi-vendor implementations. These have unique characteristics and functionality that require ongoing development from a variety of vendors and integrators.

This collaborative, DevSecOps development model challenges the traditional integration models between development (Dev) and Operations (Ops) to ensure highly secure network operations.

2.1 What is Cloud Native?

Cloud computing can be characterized as a networked infrastructure where applications can access compute resources independent of the physical infrastructure or location of that infrastructure. These attributes provide a very scalable and flexible application environment. Cloud native is a software development, deployment and lifecycle environment where all aspects of the process — as well as the software architecture and design — have been optimized for deployment in a cloud environment. The benefits of cloud native architecture, design, tools and methodologies are not limited to pure cloud environments. Instead, they can also be realized in a variety of scenarios, including public and private clouds, and virtualized and dedicated bare metal deployments.

Cloud native is a "fresh start" to create software systems that support high velocity, open, scalable, platform-agnostic systems that are maintainable and efficient. Figure 2.1 summarizes the core concepts and key benefits.

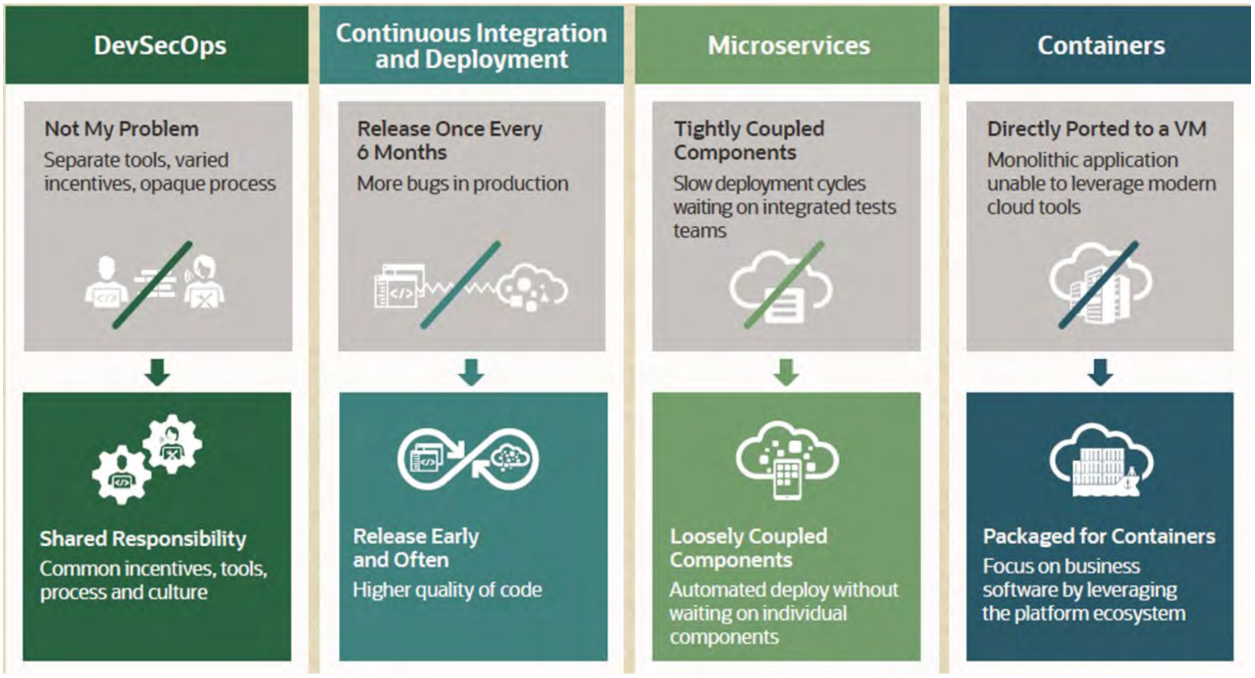
DevSecOps
Continuous Integration and Deployment
Microservices
Containers
Not My Problem
Separate tools, varied incentives, opaque process
Release Once Every 6 Months
More bugs in production
Tightly Coupled Components
Slow deployment cycles waiting on integrated tests teams
Directly Ported to a VM
Monolithic application unable to leverage modern cloud tools
Shared Responsibility
Common incentives, tools, process and culture
Release Early and Often
Higher quality of code
Loosely Coupled Components
Automated deploy without waiting on individual components
Packaged for Containers
Focus on business software by leveraging the platform ecosystem

Figure 2.1 Cloud Native Core Concepts (Source: Oracle)

Cloud native technologies will enable service providers to build and run scalable applications in modern, dynamic environments. Containers, service meshes, microservices, immutable infrastructure and declarative APIs, combined with DevOps and continuous delivery models, exemplify the cloud native approach. These techniques enable loosely coupled systems that are scalable, resilient, manageable and observable. When combined with robust automation, they allow software developers to make high-impact changes frequently and predictably, which minimizes unintended negative system impacts and downtime.

The following principles characterize successful cloud native operations:

System Immutability: Everything is code, including software and configuration. All changes are made through a process of "Continuous Integration and Continuous Deployment" (CI/CD), where they are deployed as immutable artifacts. No manual configurations or customizations are allowed. This makes it easier to implement the principle of least privilege as there is no need to run scripts in the production environment. Also, any changes not coming though the delivery pipeline can be considered malicious.

Automate Everything: All aspects of build, test, verification and deployment are automated. This includes activities such as backup, recovery and password/key rotation. Fully automating the DevOps pipeline, including verification and testing, removes much of the potential for human error, allows changes to be applied to the environment with confidence and enables rapid repair.

Disposability: All services are transient and treated as short lived. Instead of focusing on never failing, services are designed to go up and down quickly without service interruption. Regular repaving (i.e., re-deployment) of the environment ensures failed or failing services are removed and new ones deployed.

Externalized Configuration: Configuration — including passwords, credentials and location of backing stores — is decoupled from the software image. Like software, configuration can be treated as a build artifact in a controlled and versioned manner. Versioned configuration enables development and production parity as an artifact and can eliminate costly operational errors.

Logs as Event Streams and Constant Telemetry: Everything needed to debug or diagnose any functional, operational or security issue must be in logs, traces or metrics data. These are treated as a stream of time-ordered events and stored in a centralized collector outside the system, where better threat monitoring, forensics and diagnostics can occur through event correlation or analyzing the aggregated and holistic view.

Delegated Governance: Some shared aspects of the environment are centrally managed, such as networking, identity management or infrastructure. But in a true DevOps fashion, teams delivering a service are responsible for operating the service. This is allowed only with strict governance and enforced through checks in the CD pipelines, which provides greater control over rate of change. Business agility is improved as applications have better visibility into the operations of their service, with tighter feedback loops that ultimately improve quality.

Independent Lifecycle: Independently upgrading, scaling and deploying each microservice is paramount for supporting other cloud native principles, as well as minimizing the amount of change in the system at a given time. Furthermore, such decoupling makes other principles, such as repaving easier, and promotes easier isolation of issues.

The Cloud Native Computing Foundation[1] is an industry organization that supports this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects.

2.1.1 Why Cloud Native for Service Provider Deployments?

In October 2012, the seminal Network Functions Virtualization (NFV) white paper was published. It introduced the concept of NFV and announced the formation of the Network Functions Virtualization Industry Specification Group (NFV ISG) within ETSI. Since then, the communications industry has pursued a transformation initiative called NFV to significantly reduce capital expenditures and operating expenses for service providers while also providing them with flexibility and agility to bring new services to market more quickly.

However, after years of effort, many in the industry believe that NFV has fallen far short of these expectations. Although a significant and excellent body of technical work has gone into the development of NFV standards, this work was targeted specifically for service provider environments and did not take full advantage of current enterprise-centric cloud environments (i.e., cloud native).

Cloud native describes how internet companies such as Amazon, Facebook and Netflix design and deploy cloud-based software that is highly efficient, agile and scalable. By taking advantage of industrywide cloud native experience and technologies, service providers can better transform their networks to leverage the advantage seen in the web-scale industry.

For example, the use of micro-services and containers can provide efficiency and agility for many service provider services. Containers are lightweight, standalone, executable packages of software code that share an operating system, such as Linux, and can run large distributed applications. Containers have the bare minimum software that is needed to run an application, so they can be more efficient than virtual machines (VMs). Additionally, careful decomposition of network services into microservices with containers provides operational autonomy for these services and applications. Use cases range from IoT network support that does not need complex functionality to numerous smartphone applications that use data. A well-designed microservice/container-based architecture is well-suited to services and applications that will require mixing and matching different combinations of network capabilities (i.e., in containers). This architecture also facilitates the ability to scale network function capacity up or down to

[1] Cloud Native Computing Foundation – https://www.cncf.io/

ensure the correct service levels and network resources for each use case. As such, cloud native environments using microservices and containers hold promise for faster development and deployment cycles for network infrastructure.

Unfortunately, there may be no easy path to cloud native software from existing codebases and current virtual network functions. To take full advantage of cloud native technologies and processes, cloud native network functions (CNFs) must be designed specifically for the cloud and represent a clean break with the past. One option for managing this transition is by starting with the introduction of new network technology, such as 5G. The 5G mobile packet core network is particularly well-suited to the cloud native approach.

5G services are expected to dramatically increase mobile data traffic. But with existing unlimited data plans, revenue may increase at a substantially lower rate. This necessitates major cost reductions to sustain the business. In addition, delivering future 5G services in new vertical markets, such as industrial automation, may rely on new techniques, including network slicing across core and edge clouds. Verticals can leverage 5G/cloud native features such as short time-to-market (e.g., CI/CD, mix and match proprietary code with open source) and "as-a-service" usage to keep vertical services competitive. These new network structures can be better automated and managed using a cloud native approach.

2.1.2 Challenges of Cloud Native Service Provider Deployments

A carrier-grade enhancement to the service provider cloud native platform may be needed in many network use cases to provide enhanced performance compared to what is possible with classical service provider platforms. Many factors contribute to the need for these enhancements, such as:

Complex Network Models: Unlike many IT environments and applications, service provider workloads generally exist in a complex and expansive network environment (the WAN) and may require sophisticated network models to support needed network capabilities. These may include multi-homing, QoS capabilities and other network management and monitoring services needed to manage this large, complex network environment. For example, an NFV VNF Component (VNFC) can be connected to different networks (e.g., network slices, public and private network instances), which can belong to different layers such as the control, management and data planes. Each of these planes or layers has different connectivity requirements associated with different QoS requirements, management requirements and network isolation domains.

Service Function Chaining: In IT environments, applications and services generally represent discrete and independent workloads. However, service provider applications are often comprised of complex combinations of services that may have interactions (e.g., policy, billing or legal intercept purposes). As a result, they must be configured together as an application through which traffic needs to be correctly steered based on dynamic information. Management of this environment often requires an explicit definition of the network topology and distinct ingress and egress network ports. In addition, the required support of multiple networks per microservice places unique requirements on containers and associated micro services.

High-Performance Data Plane: Service provider networks require very high-scale data transfer and packet inspection capabilities associated with the large amount of traffic traversing through the network. Cloud native compute platforms are suitable for IT applications as they provide easy, reliable networking and generally good throughput. However, the existing cloud native data handling capabilities are often inadequate to handle the high data plane traffic requirements associated with high-scale WAN networking services and applications. In these cases, additional platform capabilities are needed to ensure deterministic performance required in these high-scale networks. Data performance characteristics include:

- Ultra-low latency consistent with 5G requirements driving the need for low packet processing times.
- High reliability consistent with 5G requirements and to meet various stringent SLAs for a wide variety of customers and applications.
- Minimal jitter to support predictability and stability of the packet flow through elements.
- Guaranteed and prioritized bandwidth mechanisms to support service provider QoS-sensitive applications and workloads.
- Low cost and energy usage per bit driving low CPU consumption.

Network-virtualization-based architectures have driven the development of several data plane acceleration techniques to address these requirements. One class of solutions bypasses the CPU kernel network stack and moves the data processing into the user space. Specific examples include:

- Data Plane Development Kit (DPDK)[2] consists of libraries to accelerate packet processing workloads running on a wide variety of CPU architectures. DPDK is designed to run on

[2] https://www.dpdk.org/

x86, POWER and ARM processors, generally in Linux userland, with a FreeBSD port available for a subset of DPDK features. DPDK is licensed under the Open Source BSD License.

- Vector Packet Processing (VPP) platform is an extensible, open source framework that offers the functionality of network switches or routers. Vector processing handles multiple packets at a time with low latency. This open source, Linux Foundation backed framework is part of the Fast Data Project (FD.io)[3]. VPP uses the DPDK device drivers and libraries for many of its layer 1 functions, although this functionality is separated into an optional plugin-in for VPP.

These functions or similar capabilities must be fully integrated into cloud native constructs.

Enhanced Platform Awareness: The scale, diversity and SLA-driven reliability requirements for service provider networks demand a high level of platform awareness to efficiently provide topology, fault and performance management functions. This requires container management frameworks that provide a greater awareness of the underlying platform's capabilities. In addition, the automation and use of AI-driven management tools relies on greater visibility of platform KPIs and controls to enforce network policies. Examples include:

- Standardized platform KPIs to monitor performance in a consistent manner.
- CPU pinning to avoid unpredictable latency and host CPU overcommit by dedicating CPUs to the network services and applications.
- Memory management controls to improve the utilization of compute resources for memory intensive services.
- Environment aware scheduling to provide contextual knowledge about the application and network policies, environment properties and network topology.

In some cases, new placement constraints and controls are needed. For example, 5G edge compute workloads must be distributed to the network edge to be closer to users and thus minimize latency. Consequently, container management frameworks need to be extended to support the deployment and the management of services and applications across multiple datacenters, geographical locations and administrative domains while maintaining a unique control plane.

[3] https://wiki.fd.io/view/VPP/What_is_VPP%3F

2.2 What is DevSecOps?

In a DevOps model, development and operations teams are no longer "siloed." Instead they are merged into a collaborative team where the engineers work across the entire application lifecycle, from architecture, design, development and test to deployment and ongoing operations. DevSecOps refers to scenarios where security expertise and responsibilities are also tightly integrated within the DevOps processes. DevSecOps incorporates security culture, practices and tools to drive visibility, collaboration and agility, ensuring security in each phase of the DevOps pipeline.

The successful integration of security aspects into a DevOps model affects project governance, organizational culture, processes and technology. For example, in a DevSecOps environment, project governance might include security metrics such as "number of open security defects," along with traditional measures of defects, resolution times and deployment success rates.

The organizational culture must also adapt to break down silos between security and DevOps teams to instill cyber awareness into the entire DevOps staff. The engagement of security staff with DevOps teams can, for example, facilitate security awareness by periodically briefing DevOps teams on current threats, exploits and breaches. Generally, a more centralized security function is required in most organizations to provide a central point of contact for security-related issues and reports and to ensure a consistent and robust security posture across the product line.

DevOps processes are also impacted to drive real-time security risk rationalized feedback in design and coding decisions. In addition, security considerations will affect a variety of lifecycle processes to include a security focus.

DevSecOps also impacts the technology in automating recurring security tasks and hardening the development pipeline while protecting the toolchain and infrastructure.

2.2.1 Collaborative DevSecOps in Cloud Native Service Provider Environments

Network service provider environments include many deployment scenarios where large, complex systems are created through the integration of many different components provided by many different vendors. *Section 3 – Use Cases* discusses some of these deployment scenarios in more detail.

Collaborative DevSecOps refers to these deployment scenarios where a DevSecOps execution path must deal with the continuous integration of third-party software from multiple vendors. Thus, the normal DevSecOps process must also collaborate with multiple third parties. As Figure 2.2 illustrates, collaborative DevSecOps requires both process and technology alignment between the various actors participating in the creation and management of the system.

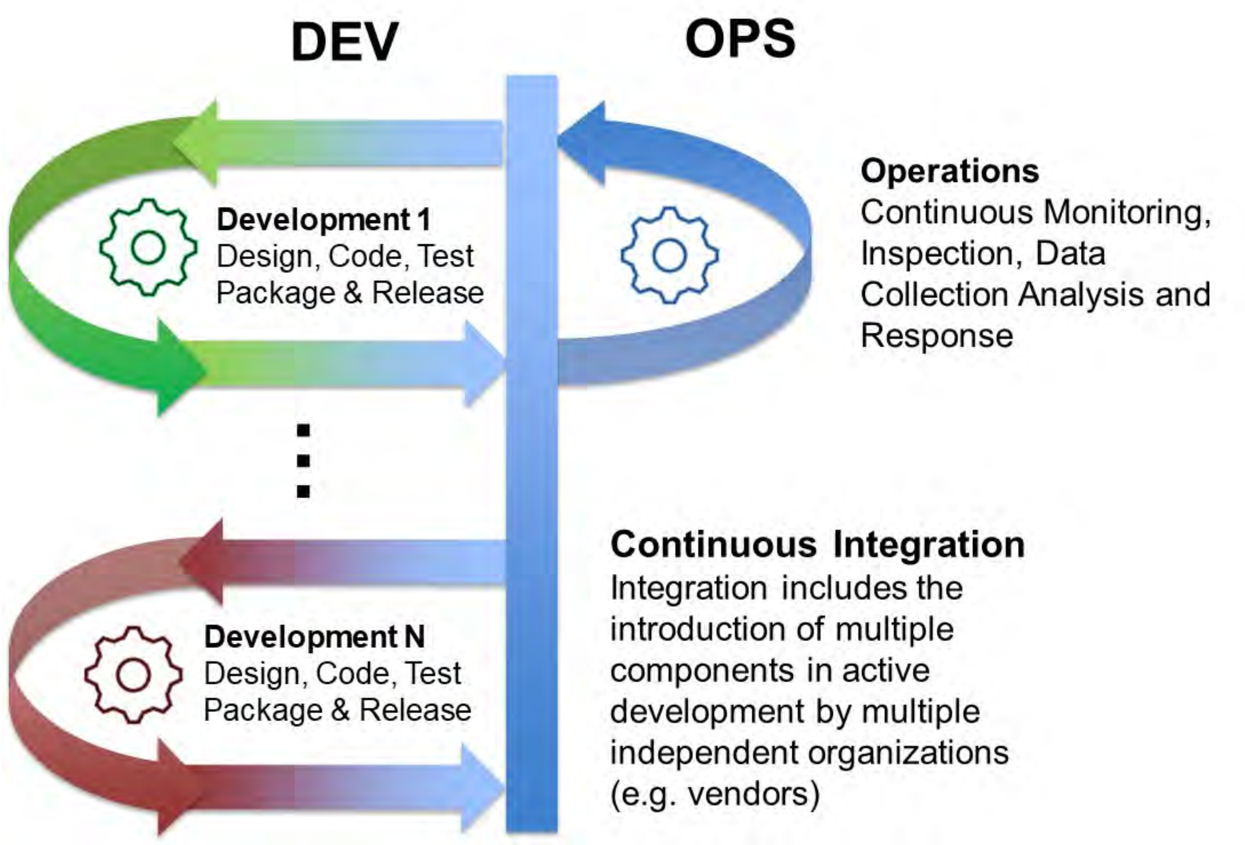
DEV
OPS
Development 1
Design, Code, Test
Package & Release
Operations
Continuous Monitoring, Inspection, Data Collection Analysis and Response
Development N
Design, Code, Test
Package & Release
Continuous Integration
Integration includes the introduction of multiple components in active development by multiple independent organizations (e.g. vendors)

Figure 2.2 Collaborative DevSecOps

Process alignment includes the establishment of "rules of engagement" between the actors to clearly define how they communicate and share critical data. For example, details about which monitoring and analytics data should be shared need to be well defined. In addition, common/standard methods of defining priorities and severity need to be agreed upon. In many ways, collaborative DevSecOps results in the "rethinking" of the traditional delivery and support models and strategies to accommodate a cloud native DevOps environment.

Technology alignment is also required to provide collaborative DevSecOps. Each of the software modules/systems being integrated from the various vendors should be aligned with the overall system architecture and toolset used to operate and maintain the final system. At lower layers of the software stack, this would include clear, common API definitions, standardized logging and common software delivery methods such as Docker image repository.

Most important from a security perspective is that all software integrated across the multi-vendor ecosystem should conform to a consistent and robust security strategy with common/consistent security mechanisms, including tools and reporting, implemented across the system.

3 Use Cases

3.1 5G Core Network

The recently defined 3GPP 5G core network provides an excellent platform for DevOps product lifecycle models. Unlike the 4G core architecture, the 5G core can be envisioned as a service-based architecture where the core control plane functions are all interconnected via a "service bus"-like structure. This creates a flexible platform well-suited to cloud native deployments that more easily supports new service and application extensions. These added functions can now support custom/proprietary features that may leverage the various standardized 3GPP functions.

One example of a new, service-based function defined in 3GPP is the NetWork Data Analytics Function (NWDAF), not shown in the basic 5G core architecture (Figure 3.1) for clarity. The NWDAF represents an operator-managed network analytics logical function. The 3GPP definition of NWDAF includes the following functionality:

- Support for data collection from 3GPP-defined network and application functions (NF and AF).
- Support for data collection from operations, administration and management (OAM) functions.
- NWDAF service registration and metadata exposure to NFs/AFs.
- Support for analytics information provisioning to NFs/AFs.

TS 23.288 defines the NWDAF functionality details.

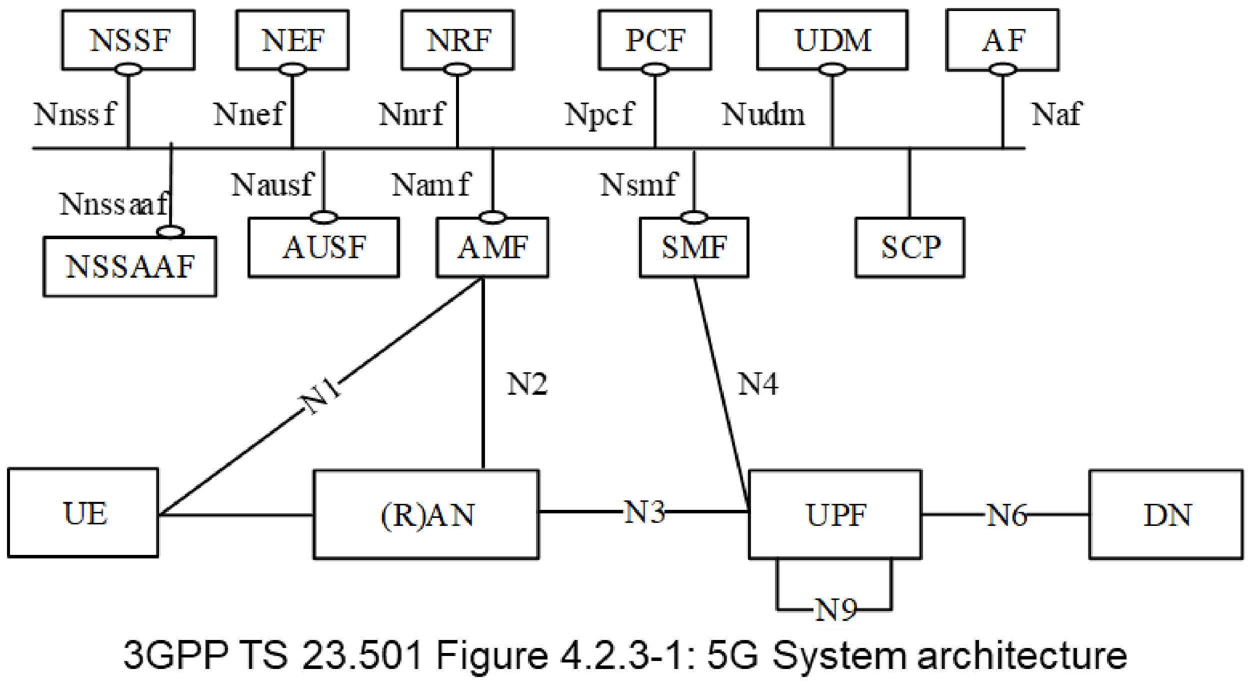

Figure 3.1 5G Core

In addition, the incorporation of network slicing capabilities into the 5G core presents additional opportunities for DevOps lifecycle models. In network sliced deployments, each network slice instance includes the Core Network Control Plane and User Plane Network Functions, as shown in Figure 3.1. Each slice is independently managed to optimize core resources for the traffic model associated with the specific slice instance. Figure 3.2 illustrates some network slice instances.

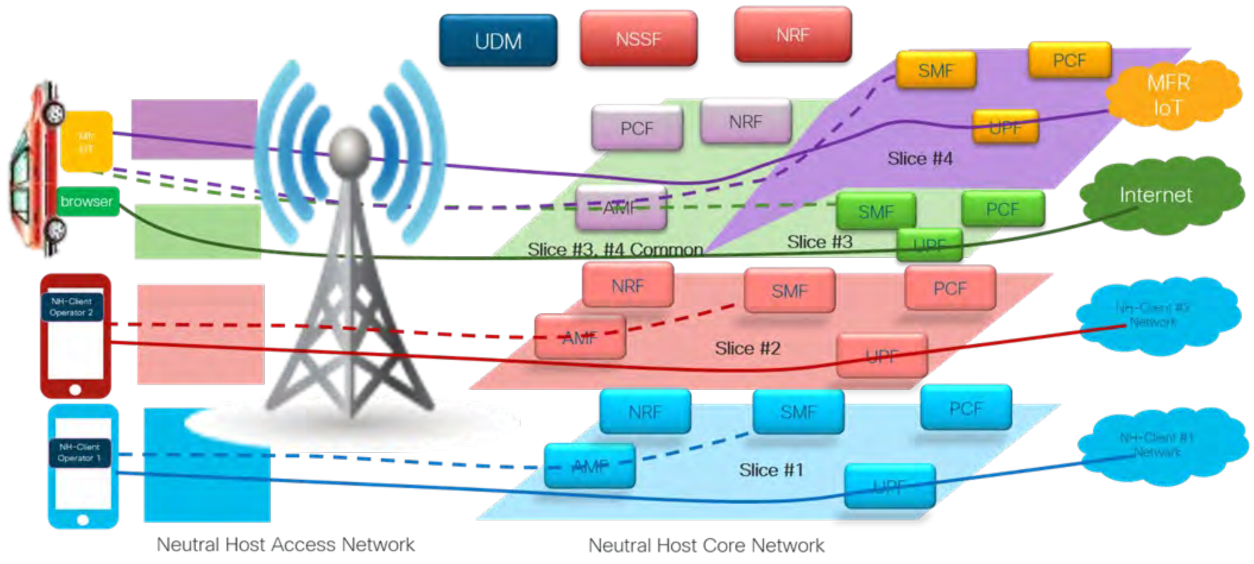

Figure 3.2 Network Slicing

These new 5G core capabilities and architectural attributes are well-suited to cloud native DevSecOps deployment models to create value-added services and applications integrated into the 5G core itself.

3.1.1 Story Highlights

As the new 5G core architecture becomes deployed as a cloud native implementation with integral support for network slicing in a service-based architecture, service providers are in an excellent position to introduce new, operator-driven features and services. Specifically, operators can now feasibly create DevOps/DevSecOps teams that would:

- Integrate a wide variety of vendor-specific components with operator-based OAM and provisioning (OAM&P) systems used to manage the 5G core and associated network.
- Create new, value-added services leveraging new architectural capabilities that better support automation/AI, custom network slices and other core services facilitated by a service-based core.
- Support a common security strategy for the 5G core to ensure secure network operations.

In this new, cloud native 5G environment, the need for collaboration is heightened as multiple vendors may be employed to provide:

- Core cloud native hardware and software components.
- Core 5G functions for existing high-volume mobile services (e.g., the basic user and control plane functions).
- Slice-specific functions. It is possible that each slice could be provided by a different vendor to get “best-of-breed” performance for each specific slice type.
- Specialized data and AI-based systems to support automation and other data/AI capabilities.

3.1.2 Business Drivers

By employing a DevSecOps environment in the new 5G core, mobile operators can now support more customized, value-added features for their customer base with improved time-to-market. These new services and features can enhance the consumer experience and provide new, innovative, enterprise-based features to attract new enterprise solutions and markets.

3.1.3 Actors

The key actor in this scenario is the mobile operator itself supporting a DevSecOps team that integrates 5G network knowledge with the OAM&P expertise required to operate and manage the network with security “built-in” to the process.

In addition, multiple vendors will be required with close collaborative working relationships to support:

- Cloud hardware and software platforms
- Core user and control plane functions
- Custom slice-specific functions
- Other specialized function (e.g., relating to data/AI and automation management)

3.1.4 Deployment Model and High-Level Architectural Context

Figures 3.1 and 3.2 provide an excellent view of the architectural context and deployment model for this scenario. This use case builds on standard 5G core network architecture constructs as defined in 3GPP TR 23.501.

3.1.5 Issue(s) Spotlight

This use case highlights the need not only for close collaboration between the actors, but also the need for a common software architecture and consistent security strategy. It is an absolute requirement that the end system be reliable, manageable and scalable in the face of continuous integration of new features and capabilities. This will require a well-defined software architecture and security framework with a common set of tools to ensure ease of integration and that all the piece parts work together to create a secure system. For example, in the cloud native context, the architecture should describe a well-defined set of containerized microservices with well-defined APIs/interfaces, a common security strategy and a common set of tools to support testing and integration addressing both operational and management functions.

3.2 Mobile/Multi-Access Edge Computing (MEC)

Edge computing[4] offers network, application and content providers cloud-computing capabilities in an IT service environment at the network edge. This edge can be very close to the edge (e.g., collocated with a cloud RAN deployment) or as far into the network as the IP edge (e.g., at the main packet core data center). This edge environment is characterized by:

- Low latency due to the edge placement.
- Core bandwidth offload because content can be placed in the edge compute platform.
- Real-time access to radio network information that applications can leverage.

Edge computing is a mature technology domain with a wide variety of ongoing standards activities.[5] This work includes:

- **ETSI ISG MEC:** The ETSI work on MEC has focused on creating an open and standardized IT service environment that allows third-party applications to be hosted at the edge of the mobile network. This environment is also capable of exposing network and context information with service-related APIs for information exposure and programmability, as well as management, orchestration and mobility-related APIs.

[4] https://www.etsi.org/technologies/multi-access-edge-computing
[5] https://www.etsi.org/images/files/ETSIWhitePapers/ETSI_wp36_Harmonizing-standards-for-edge-computing.pdf

- **3GPP SA6:** The 3GPP SA6 group has created the EDGEAPP architecture for enabling edge applications. This specifies an enabling layer to facilitate communication between application clients and applications deployed at the edge.
- **3GPP SA2:** The 3GPP SA2 group is responsible for the architecture for mobile core networks including 5G. In the context of edge computing, 3GPP SA2 defines how user traffic is routed to the appropriate application servers in the edge clouds. It also provides the means for applications to provision traffic steering rules.
- **3GPP SA5:** The 3GPP SA5 group is responsible for the management and charging aspects of 3GPP networks. In the context of edge computing, 3GPP SA5 is in the process of specifying lifecycle management of application servers in the edge cloud and charging aspects for edge services.
- **GSMA:** The GSMA organization is working to specify requirements and end-to-end high-level architecture for a unified operator platform. This can help operators make their assets and capabilities consistently available to the developers and the enterprise segment across networks and national boundaries.

MEC is well-suited to address applications that require low latency, high bandwidth and/or access to local network information to enhance application performance. Typical use cases discussed in the industry include:

- Industrial IoT applications
- Automotive applications in support of fully automated driving.
- Augmented and virtual reality (AR/VR) provide rich and immersive experience to consumers and businesses. Edge computing can provide the capability to offload compute to a low-latency edge cloud, hence driving creation of low-cost and mass-market devices.

3.2.1 Story Highlights

As the new edge-optimized applications become deployed as a cloud native implementation with an edge-based architecture, service providers are in an excellent position to introduce new operator driven features and services. Specifically, operators can now feasibly create DevOps/DevSecOps teams focused on the edge platform of choice that would:

- Integrate a wide variety of vendor-driven edge applications into a common edge platform that can be managed by the operator and integrate with operator OAM&P systems.

- Create new, value-added edge applications leveraging access to radio/core network information to optimize user and network performance.
- Integrate existing network services with edge deployments to optimize existing service capabilities (e.g., in support of custom low-latency industrial IoT applications).
- Create a consistent security strategy and implementation to ensure that both the network and the edge application are secure.

In this new, cloud native edge environment, the need for collaboration is essential as multiple vendors may be employed to provide:

- Core cloud native hardware and software edge platform components.
- Vendor-driven edge applications that provide enhanced performance and low-latency capabilities leveraging the edge platform.

3.2.2 Business Drivers

In multi-vendor, multi-application edge environment, an operator-managed DevSecOps development paradigm would speed delivery of new edge features while improving the system's overall network management. The mobile operator can now support more customized, value-added features for their customer base with improved time-to-market. These new services and features can enhance the consumer experience while also providing new, innovative, enterprise-based features to attract new business solutions and markets.

3.2.3 Actors

The key actor in this scenario is the mobile operator itself supporting a DevSecOps team that will integrate the edge applications with the OAM&P expertise required to operate and manage the network with security "built in" to the process. In addition, the operator's DevSecOps team can develop its own custom applications to address new markets and opportunities.

The network operator will require close collaboration with multiple vendors (i.e., actors) to support:

- Cloud hardware and software platforms
- Edge applications

3.2.4 Deployment Model and High-Level Architectural Context

Figure 3.3 provides an excellent view of the architectural context and deployment model for this scenario. This use case builds on standard 5G core network architecture constructs as defined in 3GPP TR 23.501 as well as MEC standards referenced in Section 3.2 above.

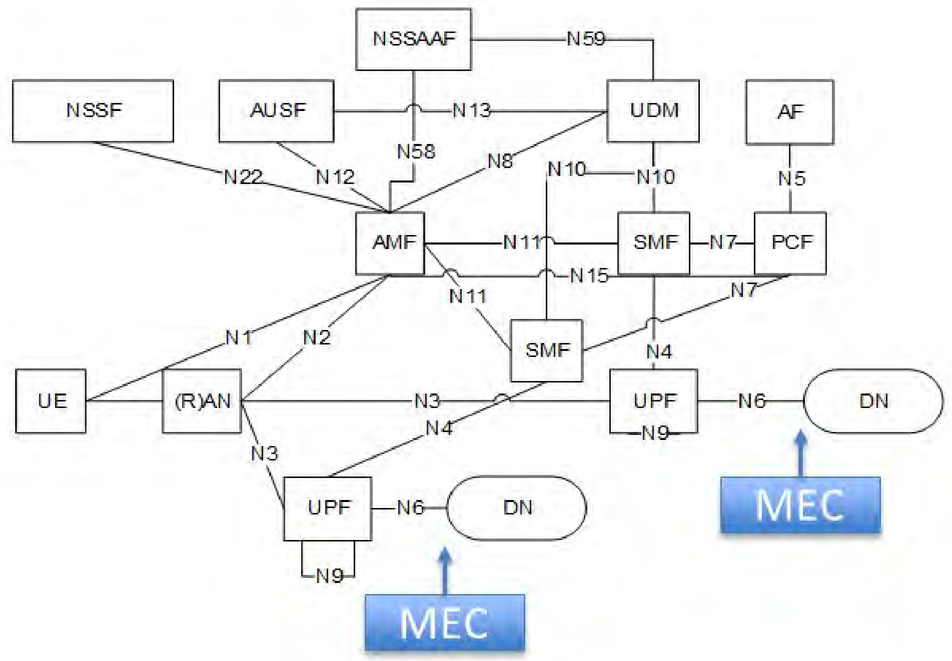

Figure 3.3 5G Core Network Architecture with Multiple UPFs and MEC

3.2.5 Issue(s) Spotlight

As in the previous use case addressing the 5G core network, this use case highlights the need for close collaboration between the actors, as well as the need for a common cloud native software architecture and consistent security strategy.

However, unlike the 5G core network deployment, MEC deployments have a much stronger applications focus. This may require interworking with various client applications on the device, as well as application servers hosted much deeper in the cloud. As such, the level of collaboration may extend well beyond APIs and container/microservice interworking toward many different device clients and cloud-based application servers across a wide variety of applications.

3.3 Network Management and Operations

Service provider networks can be characterized as large-scale, high-availability, high-performance systems distributed across a wide geographical area. They use network elements providing a wide range of different functions supplied by a wide range of vendors serving a broad customer population, from residential/individual consumers to large enterprises. Management and operations of this complex, diverse system require a high degree of automation and sophisticated tools to maintain the reliability and performance that customers expect.

Network operations and management includes functions to provide fault management, configuration (e.g., provisioning), accounting (e.g., charging), performance and security management. This set of functions has been defined by the ITU-T recommendations M.3010, M.3400 and X.700 and are commonly referred to as FCAPS. These functions need to be applied to each network element specifically and to the network as a whole. As such, the management and operations system itself can often be thought of as an integration of various element management systems with higher level network management systems that connect to other back-end systems that provide project-management-like functions for network operations (e.g., trouble ticket management, customer service management).

Given the unique deployment architectures, vendor/product set and feature/service-specific network functions, service providers must generally create their own management and operations software environment. As such, service providers engage in significant software development in the management and operations of their network. This development lends itself to collaborative DevSecOps development processes.

3.3.1 Story Highlights

Operator software development teams work to maintain their unique mix of operational software used to manage their network and services. As a result, they must deal with an ever-changing network environment where new services and infrastructure are constantly added or enhanced. The network is in a constant state of change as capacity increases and new infrastructure is installed to keep pace with customer needs for new capacity, products and services. In a competitive world, speed is of the essence to ensure that customers have the network capabilities required to efficiently run their business.

This environment is well-suited to a collaborative DevOps approach where the engineers work across the entire application lifecycle, from architecture, design, development and test to deployment and ongoing operations. Indeed, a DevSecOps focus is desired because network operations and management require a high degree of security expertise — one where responsibilities are also tightly integrated within the DevOps processes. It is essential that the network operations and management environment incorporates the security culture, practices and tools to drive visibility, collaboration and agility. This approach ensures security for the network as a whole.

The collaborative aspect of DevSecOps is also important in network operations development scenarios because the DevSecOps execution path must deal with the continuous integration of the many third-party software products from multiple vendors in the network.

3.3.2 Business Drivers

In the complex service provider environment, new services and products cannot be deployed at scale unless they can be managed at scale within the network's operations environment. As such, network operations and management become a limiting critical path component in the sales and associated revenue-generation process. Even if a new service or product is operationally ready, the service or product cannot be immediately sold to customers. First the network must enable the necessary service provisioning, fault management, performance management, charging and administration controls required for proper operation in the network. As such, speed, agility and security are key elements network management and operations to enable new revenue streams.

3.3.3 Actors

The key actor in this scenario is the mobile operator itself supporting a collaborative DevSecOps team that will work with network vendors and other operators to create and integrate the OAM&P system required to operate and manage the network with security "built in" to the process. In addition, the operator's DevSecOps team must manage the introduction of new products and services and the capacity growth in existing network infrastructure.

The network operator will require close collaboration with multiple vendors, operators and enterprise customers (i.e., actors) to support:

- The network elements that compromise the network itself, as well as the associated products and services.
- Interfaces to other cooperating operators (e.g., leased facilities that may be necessary), as well as the enterprise customers that may require more sophisticated, high-scale service interfaces.

3.3.4 Deployment Model and High-Level Architectural Context

Network service provider management and operations systems typically include:

- Business support systems (BSS), which underpin commercial activities and handle customer-facing interactions. These systems include order capture and management, customer relationship management (CRM), mediation, charging and billing, as well as call center automation.
- Operations support systems (OSS), which support management functions such as network inventory, service provisioning, network configuration and fault management. In general, an OSS covers functions such as network management systems, service delivery and service fulfillment, including the network inventory, activation and provisioning, service assurance and customer care. OSS higher level functions often rely on separate network management systems (NMS) and element management (EM) functions.
- NMS, which generally provide network configuration, performance monitoring and fault management, as well as security and accounting management on a network level.
- EM systems, which optionally exist to manage very complex network elements relative to fault, configuration, accounting, provisioning and security functions.

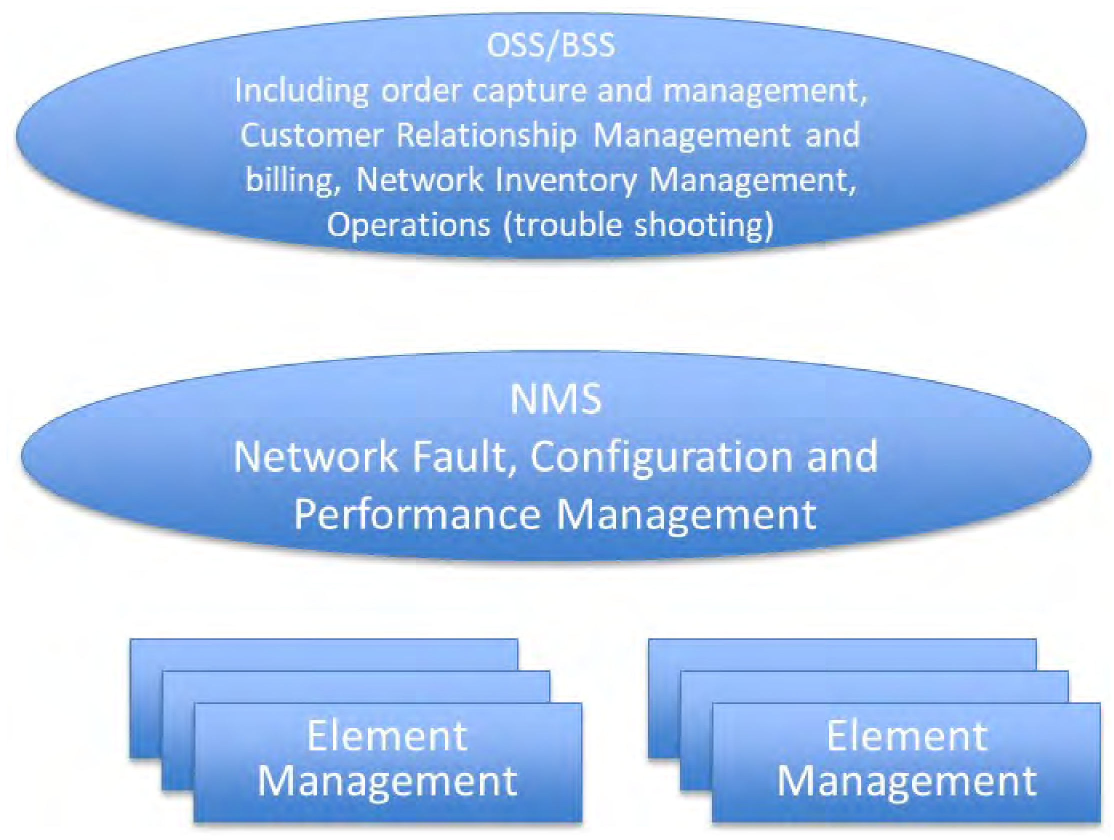
OSS/BSS
Including order capture and management, Customer Relationship Management and billing, Network Inventory Management, Operations (trouble shooting)
NMS
Network Fault, Configuration and Performance Management
Element Management
Element Management

Figure 3.4 Network Service Provider Management and Operations Systems Architecture

4 Applying DevSecOps to Collaborative Environments

4.1 Creating a Development Culture

In a traditional, non-DevOps environment, separate teams manage different phases of the project. These teams may be separated by functional responsibilities such as:

- **Business Development:** Working to ensure that the new project/product has a positive business case by meeting the needs of the company's customers. This role also includes leveraging the company's core competence and associated sales channels and customer base.
- **Marketing:** Providing both inbound marketing data to identify needs, as well as outbound marketing to educate the target customer base about the company's product and services.
- **Architecture and Design:** Working to create high-level framework requirements for the development that fits well with the existing customer deployment environment. These

requirements also leverage both technology and the company's proprietary resources in a way that meets the business goals set by the business development team.

- **Security:** Responsible for creating a security strategy and associated requirements to ensure that the developed products meet the security goals at a specific level of risk.
- **Developers/Coders:** Responsible for implementing and integrating code modules to meet the requirements established by the business and architecture/design teams.
- **Testers:** Responsible for creating and executing tests to verify that the developed product meets the stated requirements.
- **Operations:** Responsible for operating and managing the new product/system as deployed in a live network.

Depending on the specifics of the projects, other teams may also be engaged. Nevertheless, non-DevOps projects can be characterized as a flow through various specialized functions in a linear manner. Each functional team has specialized knowledge of their function but may know very little about the other functions in the end-to-end process.

This lack of knowledge and responsibility caused by the siloing of functions can create friction between the teams. Another unwelcome byproduct is rigid boundaries where information is passed "over the wall" from one function to another without a great deal of regard to issues that other functions face. This can result in incorrect assumptions and decisions along the development path, leading to rework and poor system performance. In addition, this siloing of functions creates a more serial development process, which has difficulty performing steps in parallel, thus slowing the whole development process.

DevOps (and DevSecOps when security is included) is intended to create a new, more collaborative development environment where teams can work in a more integrated fashion breaking down the traditional barriers. This enables broader understanding of the issues faced in each function. It also enables parallel development and faster decision-making as each function better understands the fundamental goals and needs of the other functions.

Ultimately, the DevOps environment requires a cultural shift to support:

- Greater team integration by getting people to know one another (across teams) and building relationships that can be leveraged to better understand the questions and key issues faced by each function. In some cases, it may make sense to have certain people explicitly belong to multiple teams.

- Improved communication channels to enable and encourage interaction between the teams. This may include use of best-in-class collaboration tools.
- Improved management processes to foster knowledge sharing and a blame-free culture, as well as process changes that better support collaboration and information sharing.

The success of any DevOps/DevSecOps initiative is often determined by the extent to which the development culture can be shifted to support the integration and collaboration needed to leverage the advantages of DevOps.

4.2 DevSecOps Development Process Considerations

4.2.1 Key Collaborative DevSecOps Process Components

DevSecOps principles can be applied in many different environments with many different process variations. Depending upon such factors as system complexity, network architecture, software design choices, risk tolerance level and system maturity, every software lifecycle may have its own suite of development and management processes. However, there are several common process attributes that are useful in most service provider DevSecOps projects.

Specifically, robust DevSecOps process suites will generally support:

- Close collaboration through all phases of the development cycle, as well as collaborative process design strategy utilizing multidisciplinary teams.
- Continuous automated testing and test-driven development where software requirements are converted to test cases before software is fully developed. This process includes tracking all software development by repeatedly testing the software against all test cases.
- A high degree of automation to support continuous build, integration, delivery, deployment, operation and monitoring.
- Continuous process adaption and improvement capabilities inherent in each process.

In addition, DevSecOps processes can be stressed when collaborators are from separate independent companies or organizational domains. This separation can limit visibility into the robustness and level of management of processes within these separate entities. As such, interworking processes must be created to ensure the quality and security of asset handoffs and support. These processes may be governed through contractual terms and conditions. However, in practice, the actual process operation will more likely use industry-standard practices and zero trust principles.

4.2.2 Impacts on the Supply Chain

Supply chain risk is defined by the US Department of Defense[6] as "the risk that an adversary may sabotage, maliciously introduce unwanted function, or otherwise subvert the design, production, distribution, manufacturing, installation, operation or maintenance of a system so as to surveil, deny, disrupt or otherwise degrade the function, use or operation of such system." As Figure 4.1 illustrates, the supply chain represents one very important threat surface of a software product. Other threat surfaces might include environmental attacks (i.e., power, HVAC, physical destruction), network-interface-based attacks (i.e., online attacks) and social engineering attacks (i.e., tricking people to do things that compromise the product's security).

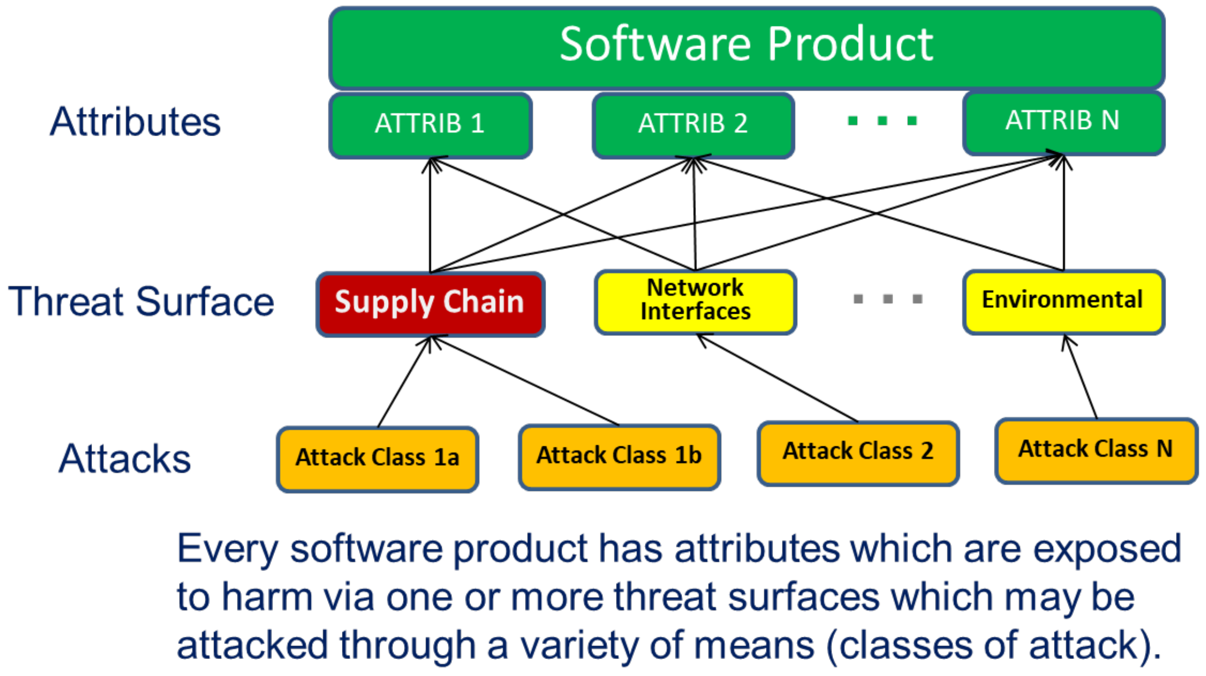

Figure 4.1 Supply Chain as a Threat Surface in DevSecOps

Specific threats and their associated attacks may occur at each stage in the DevOps development lifecycle. For example:

[6] *Department of Defense Instruction Number 5200.44, Change 3, October 2018*

SYSTEM DESIGN

- Weak/flawed security strategy and coverage
- Weak/flawed cryptography
- Default hardcoded passwords
- Failure to follow “design for security” principles (e.g., inappropriate trust boundaries, insecure update procedures)

IMPLEMENTATION/CODE/INTEGRATION

- Integration of open source with embedded vulnerabilities and/or malware
- Use of compromised software development kits (SDK) that embed vulnerabilities and/or malware
- Use of a compromised build environment
- Code insertion during the coding phase
- Certificate theft
- Credential theft

ITERATIVE TESTING

- Compromised or out-of-date test plans and/or automated test suites

DEPLOYMENT

- Compromised or “security weak” configuration files
- Stolen credentials for account access
- Code insertion

ONGOING SUPPORT AND UPDATE

- Weak/flawed update processes
- Unsigned or self-signed code signatures
- Weak credentials
- Lack of software module provenance (e.g., inability to identify when vulnerabilities are discovered in open source or other software embedded in the product)

These attack classes can be mitigated though various process controls and management techniques to minimize the possibility of a serious risk of impact based on the software product.

In its "Breaking Trust: Shades of Crisis Across an Insecure Software Supply Chain"[7] report, the Atlantic Council collected 115 instances, going back a decade, of publicly reported attacks on the software supply chain or disclosure of high-impact vulnerabilities likely to be exploited in such attacks. For each instance, the report documents the incident name, date, victim and likely source while categorizing the basic technical characteristics of each. The report then analyzes this data with, for example, the distribution of attacks and disclosures.

4.3 Collaborative DevSecOps Security Practices

Many different security practices and mechanisms have been identified to help secure a DevOps cloud native environment across the entire lifecycle, from conception to retirement. For highly collaborative environments, it is useful to require the use of specific practices and mechanisms to ensure consistency and security across the ecosystem participating in development and operations of a specific system. Areas of attention include:

- Common infrastructure constructs such as a robust microservices-based infrastructure architecture with built-in security capabilities such as a library of hardened containers meeting specific security requirements.
- Uniform configuration management mechanisms to enable consistent, controllable and maintainable overall system configuration.
- A common set of automation, testing and monitoring/logging tools to ensure each module of the system maintains a consistent level of assurance relative to integrity of operation and information security.
- Ongoing monitoring and verification of the integrity of the development environment itself.

4.3.1 Common Infrastructure Components

In highly collaborative environments, there are many infrastructure constructs available to help ensure consistency, security and ease of integration. At the core is an immutable infrastructure

[7] Atlantic Council, "Breaking trust: Shades of crisis across an insecure software supply chain," https://www.atlanticcouncil.org/in-depth-research-reports/report/breaking-trust-shades-of-crisis-across-an-insecure-software-supply-chain/

with a well-defined microservices architecture using containers. The concept of immutable infrastructure is an IT strategy in which deployed components are replaced in their entirety rather than being updated in place. Deploying immutable infrastructure requires standardization and emulation of common infrastructure components to achieve consistent and predictable results.

The first step in the creation of an immutable infrastructure is the creation of a robust microservices architecture used by all collaborating vendors and developers onto which their specific functions and services can be integrated. This microservice architecture would include the requirements associated with the network environment in which the software system is deployed. For example, the microservices architecture would define:

- How existing and well-defined network functions are partitioned into microservices and how they should communicate with each other as well as with outside entities.
- How various microservices interface to the necessary infrastructure support systems such as DNS, user authentication servers and infrastructure key management systems, as well as configuration and network management systems.
- Specific requirements on the internal structure of each microservice that may include the use of hardened containers and zero trust mechanisms for all interfaces.

One size does not necessarily fit all applications, so a library of hardened containers is often the best approach to ensure that both efficiency and security needs are met. Hardened container compliancy tests along with operational capabilities to "build in" needed security capabilities provide an excellent means to address embedded vulnerabilities.

One key service enabled by DevSecOps and the container-based Kubernetes runtime environment is called the Sidecar Container Security Stack (SCSS), which includes:

- A logging agent to push logs to a platform centralized logging service. Including a common logging service with a consistent output data format is an important factor in understanding system behavior and successful root cause analysis of events.
- Container policy enforcement function to verify that container hardening is preserved and complies with requirements.
- Runtime defense checks that can provide both signature-based and behavior-based detection and send notifications when there is anomalous behavior.
- Vulnerability-management capabilities
- Verify container-level zero trust mechanisms

The use of zero trust controls between containers should also be considered to ensure security in highly collaborative environments. Key aspects of zero trust at the container level include mutual Transport Layer Security authentication (mTLS), an encrypted communication tunnel between containers, strong identities per data center server cluster using certificates and the use of “allow” lists rather than” disallow” lists.

The NIST document *Zero Trust Architecture* (SP 800-207) contains an abstract definition of zero trust architecture (ZTA) and gives general deployment models and use cases where zero trust could improve overall information technology security posture. NIST notes that a zero trust architecture includes three key components:

- The policy engine makes the ultimate decision to grant access to the resource based on policy, input from external sources (e.g., the continuous diagnostic mitigation system or threat intelligence services) and a trusted algorithm.
- The policy administrator is responsible for establishing or shutting down the communication path between the requesting container and the target resource. Once it receives the policy engine’s ultimate approval to grant access, it commands the policy enforcement point to initiate a session through the credential, key or token used to access the resource.
- The policy enforcement point acts as the gatekeeper for the communication path responsible for enabling, monitoring and terminating sessions.

NIST also identifies a set of key zero trust components that enable zero trust operation and are also used a data source for zero trust policy decisions. These components include:

- Public key infrastructure that generates and logs certificates issued by the core common infrastructure to resources, subjects, services and applications.
- Identity management functions that provide a baseline for a higher degree of trust around entities through the use of identity proofing.
- Data access policies provide a domain of understanding and controls around what is acceptable behavior for the entities.
- A continuous diagnostic mitigation (CDM) system monitors, reports and corrects on the integrity of systems in the architecture and relies on support the threat intelligence feed(s).

- Threat intelligence feed(s) provide(s) information from internal or external sources, such as information on newly discovered attacks or vulnerabilities, to help the policy engine make access decisions.
- An industry compliance, security incident and event monitoring (SIEM) system, and network access and activity logs, serve to collectively audit and track events and activities, providing analysis or reporting about the access records.

4.3.2 Configuration Management Mechanisms

To effectively manage a system, all aspects of system configuration should be separated from the software image of that system. Configuration as code (CAC) is the set of processes and practices where configuration parameters are managed in a source repository as part of a versioned control system. In this way, application configuration resources are treated as versioned artifacts.

Keeping configuration in separate repositories enables versioning of application configuration providing visibility into who made changes and when. In addition, by using branches, changes can be isolated to development environments without affecting the production application. The use of repositories also enables traceability and tracking of which configuration version is deployed in various environments.

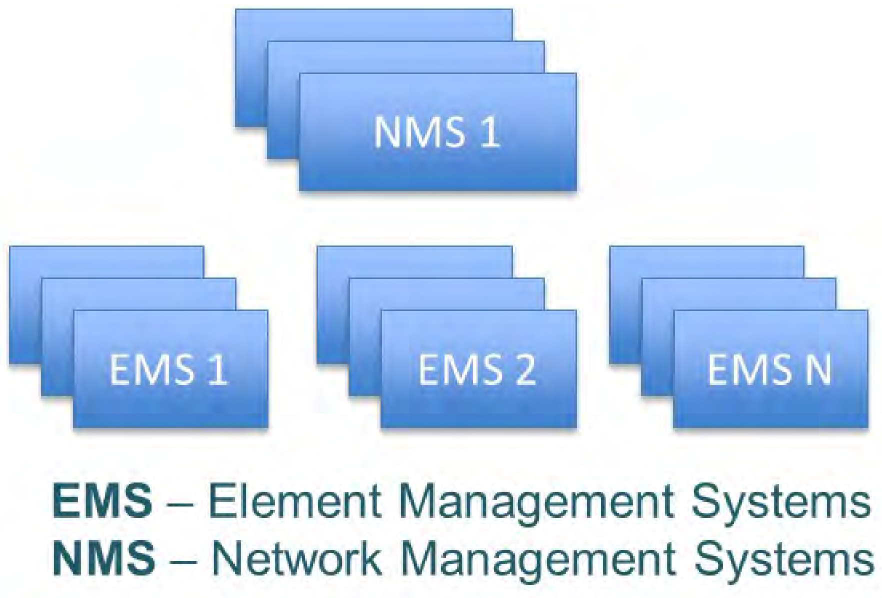
NMS 1
EMS 1
EMS 2
EMS N
EMS – Element Management Systems
NMS – Network Management Systems

Figure 4.2 EMS-NMS Hierarchy

Another effective approach in managing software configuration is the use of traditional service provider network management constructs using an FCAPS model from element up to network-level operations. This model utilizes the ITU-T Telecommunications Management Network (TMN) defined by the ITU-T recommendations M.3010, M.3400 and X.700. In this context, FCAPS refers to fault management, configuration (e.g., provisioning), accounting (e.g., charging), performance and security.

While network management tends to focus on fault, configuration and performance management, other functions (e.g., accounting/charging) are often managed via subscriber policy functions such as AAA (which has a security function), PCRF/PCF and HSS/UDM. Security management is often distributed in AAA/UDM databases, key management functions and other specialized appliances to support PKI, identity and authorization activities.

4.3.3 Automation, Testing and Monitoring/Logging Tools

Automation, testing and monitoring/logging are key capabilities to help ensure the integrity and security of a software development and integration project. Automation should be applied to all aspects of the development cycle that are automatable. Automation provides many advantages including:

- Fast execution of the task to be automated.
- Minimization of manual interaction to limit human mistakes inherent in any manual process.
- The ability to continuously enhance the automation to incorporate new findings (e.g., detection of new vulnerabilities).
- Repeatability and completeness.

Automation can generally be applied to the build process itself, testing functions and security-specific functions where software scans can run continuously to identify potential problems that may be inserted at any point in the development process.

In addition, the use of real-time system monitoring/logging is necessary for the ongoing management of the system/software health.

Typically, application/infrastructure code is designed to log various events to log files that can be monitored/scanned on an ongoing basis. In addition, external security appliances can be used to scan for anomalies (i.e., when content is encrypted).

In addition to logs and scans, it is possible to embed hooks into the code to provide real-time but independent monitoring/scanning. With embedded hooks using well-managed security libraries, deep instrumentation of the system can be executed at the runtime. Embedded security logic can provide real-time analysis and collection and enables better correlation of security events with logs for faster, more accurate anomaly detection. In addition, embedded security scan algorithms that have been segregated into separate dynamic libraries can be updated independently of the application/framework/infrastructure code.

4.3.4 Challenges of Encryption

Encryption of data in the cloud in transit and at rest are no longer nice-to-have capabilities. They are now considered a must have. However, encryption can be a double-edged sword:

- Encryption creates complications for logging and scanning software that may need access to encrypted data to effectively log an event.
- Encryption can cause performance challenges (i.e., slowing down and/or increasing the cost of transactions).
- Detecting network attacks and responding quickly in the face of data encryption can be challenging.

In many cases, encryption is either necessary for regulatory compliance or contract requirement and thus cannot be avoided.

Key to managing this issue is a robust and flexible key management system to enable access to data without compromising the security afforded by data encryption.

5 Conclusion

Network service provider environments often have more stringent requirements for security, resiliency, availability, scalability and performance given a larger and more diverse customer base that may be covered by many different yet stringent SLA requirements.

Nevertheless, many network service provider functions are well-suited for deployment in a cloud native, collaborative DevSecOps environment. Specific use cases analyzed in this report include the:

- 5G core network
- Mobile/multi-access edge computing
- Network management and operations

These use cases highlight the need for close collaboration between the actors, as well as the need for a common software architecture and consistent security strategy. It is an absolute requirement that the end system be reliable, manageable and scalable in the face of continuous integration of new features and capabilities. This will require a well-defined software architecture and security framework with a common set of tools to ensure ease of integration. It also requires that all the piece parts work together to create a secure system. For example, in the cloud native context, the architecture should describe a well-defined set of containerized microservices with well-defined APIs/interfaces, common security strategy and common set of tools to support testing and integration addressing both operational and management functions. In many cases, interworking is required with different device clients and cloud-based application servers across a wide variety of applications.

Bibliography

1. Cloud Native Computing Foundation – Landing page: https://www.cncf.io/, CNCF Cloud Native Interactive Landscape - https://landscape.cncf.io/, Cloud Native Trail Map - https://raw.githubusercontent.com/cncf/trailmap/master/CNCF_TrailMap_latest.png
2. DevSecOps Embedded Security Within the Hyper Agile Speed of DevOps, Mark G. Moore, Managing Director, Deloitte and Touche LLP, Antonio L. Bovoso, Senior Manager, Deloitte and Touche LLP, https://www2.deloitte.com/content/dam/Deloitte/global/Documents/About-Deloitte/DevSecOps-Explained.pdf
3. DoD Enterprise DevSecOps Reference Design v1.0; 12 August 2019, https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf?ver=2019-09-26-115824-583, https://software.af.mil/wp-content/uploads/2020/02/DoD-Enterprise-DevSecOps-Initiative-Introduction-v4.91.pptx
4. 5G Americas – 5G and the Cloud, https://www.5gamericas.org/5g-and-the-cloud/
5. 5G-PPP Software Network Working Group – From Webscale to Telco, the Cloud Native Journey, https://5g-ppp.eu/wp-content/uploads/2018/07/5GPPP-Software-Network-WG-White-Paper-July-2018.pdf
6. 5G Network & Service Strategies Survey – 2020 Operators Survey, https://www.lightreading.com/lg_redirect.asp?piddl_lgid_docid=758034
7. The Evolution of Security in 5G – 5G Americas, https://www.5gamericas.org/the-evolution-of-security-in-5g-2/
8. A Framework to Halt Cloud-Native Application Threats, Trend Micro Webinar, https://resources.trendmicro.com/Cloud-One-Webinar-Series-Cloud-Native-Application-Threats.html
9. Defending the Cloud - Encryption Leads the Way in Protecting Cloud Data http://assets.extrahop.com/whitepapers/SC%20Media%20Defending%20the%20Cloud%20eBook.pdf
10. For 5G, Telcos Must Break With The Past, Forbes, March 11, 2020, https://www.forbes.com/sites/forbestechcouncil/2020/03/11/for-5g-telcos-must-break-with-the-past/#74310d9f303d
11. Verizon and Ericsson turn up cloud-native and containers in a wireless core trial, Fierce Telecom, July 15, 2019, https://www.fiercetelecom.com/telecom/verizon-and-ericsson-turn-up-cloud-native-and-containers-a-wireless-core-trial
12. "Cloud Native and 5G Verticals' Services," 5G-PPP SW Network Working Group, Feb 2020, https://5g-ppp.eu/wp-content/uploads/2020/02/5G-PPP-SN-WG-5G-and-Cloud-Native.pdf
13. NIST Special Publication 800-207; Zero Trust Architecture, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207-draft2.pdf

14. Zero Trust Architecture Project, Alper Kerman, NIST (PPT in PDF) https://csrc.nist.gov/CSRC/media/Presentations/zero-trust-networks-brief/images-media/2-4Kerman%20-%20Zero%20Trust%20Architecture%20-%20NCCoE%20-%202019%20-%20ISPAB.pdf
15. Implementing a Zero Trust Architecture, March 2020, National Cybersecurity Center of Excellence, NIST, https://www.nccoe.nist.gov/sites/default/files/library/project-descriptions/zt-arch-project-description-draft.pdf
16. The Foundations of Implementing A Zero Trust Architecture, Forbes, https://www.forbes.com/sites/forbestechcouncil/2019/05/13/the-foundations-of-implementing-a-zero-trust-architecture/#d8d70ad62e9a

Annex A - Definitions, Acronyms & Abbreviations

Definitions

For a list of common communications terms and definitions, visit the ATIS Telecom Glossary, which is located at https://glossary.atis.org.

Acronyms and Abbreviations

AAA	Authentication, Authorization and Accounting
AF	Application Functions
AMF	Access (and Mobility) Management Function
ATIS	Alliance for Telecommunications Industry Solution
AUSF	Authentication Server Function
BSS	Business Support Systems
CAC	Configuration as Code
CDM	Continuous Diagnostic Mitigation
CI/CD	Continuous Integration and Continuous Development
CRM	Customer Relationship Management
DEV	Development
DPDK	Data Plane Development Kit
EM	Element Management
FCAPS	Fault, Configuration, Accounting, Performance, Security
KPI	Key Performance Indicators
MEC	Mobile/Multi-Access Edge Computing
NF	Network Function
NFV	Network Functions Virtualization
NFV ISG	Network Functions Virtualization Industry Specification Group
NMS	Network Management Systems
NSSF	Network Slice Selection Function
NWDAF	NetWork Data Analytics Function
OAM	Operations, Administration and Management
OPS	Operations
OSS	Operations Support Systems
PCF	Policy Management Function
RAN	Radio Access Network
SCP	Service Communication Function
SCSS	Sidecar Container Security Stack
SDK	Software Development Kits
SIEM	Security Incident and Event Monitoring

SLA	Service Level Agreement
SMF	Session Management Function
TMN	Telecommunications Management Network
UDM	Unified Data Manager
UE	User Equipment
UPF	User Plane Function
VNFC	VNF Component
VPP	Vector Packet Processing
ZTA	Zero Trust Architecture