Journal of
Cybersecurity and Privacy

MDPI

Review

Security in Cloud-Native Services: A Survey

Theodoros Theodoropoulos [1,*], Luis Rosa [2], Chafika Benzaid [3], Peter Gray [4], Eduard Marin [5], Antonios Makris [1], Luis Cordeiro [2], Ferran Diego [5], Pavel Sorokin [4], Marco Di Girolamo [6], Paolo Barone [6], Tarik Taleb [3] and Konstantinos Tserpes [1]

[1] Department of Informatics and Telematics, Harokopio University of Athens, 17671 Athens, Greece; ttheod@hua.gr (T.T.); amakris@hua.gr (A.M.); tserpes@hua.gr (K.T.)
[2] OneSource, 3030-384 Coimbra, Portugal; luis.rosa@onesource.pt (L.R.); cordeiro@onesource.pt (L.C.)
[3] Faculty of Information Technology and Electrical Engineering, University of Oulu, 90570 Oulu, Finland; chafika.benzaid@oulu.fi (C.B.); tarik.taleb@oulu.fi (T.T.)
[4] CloudSigma, 6300 Zug, Switzerland; peter.gray@cloudsigma.com (P.G.); pavel.s.github.private@gmail.com (P.S.)
[5] Telefonica Research, 28050 Madrid, Spain; eduard.marinfabregas@telefonica.com (E.M.); ferran.diegoandilla@telefonica.com (F.D.)
[6] Hewlett Packard Enterprise, 20063 Milan, Italy; marco.digirolamo@hpe.com (M.D.G.); paolo.barone@hpe.com (P.B.)
* Correspondence: ttheod@hua.gr

Abstract: Cloud-native services face unique cybersecurity challenges due to their distributed infrastructure. They are susceptible to various threats like malware, DDoS attacks, and Man-in-the-Middle (MITM) attacks. Additionally, these services often process sensitive data that must be protected from unauthorized access. On top of that, the dynamic and scalable nature of cloud-native services makes it difficult to maintain consistent security, as deploying new instances and infrastructure introduces new vulnerabilities. To address these challenges, efficient security solutions are needed to mitigate potential threats while aligning with the characteristics of cloud-native services. Despite the abundance of works focusing on security aspects in the cloud, there has been a notable lack of research that is focused on the security of cloud-native services. To address this gap, this work is the first survey that is dedicated to exploring security in cloud-native services. This work aims to provide a comprehensive investigation of the aspects, features, and solutions that are associated with security in cloud-native services. It serves as a uniquely structured mapping study that maps the key aspects to the corresponding features, and these features to numerous contemporary solutions. Furthermore, it includes the identification of various candidate open-source technologies that are capable of supporting the realization of each explored solution. Finally, it showcases how these solutions can work together in order to establish each corresponding feature. The insights and findings of this work can be used by cybersecurity professionals, such as developers and researchers, to enhance the security of cloud-native services.

Keywords: cybersecurity; cloud-native services; survey; key aspects; features; solutions

check for updates

Citation: Theodoropoulos, T.; Rosa, L.; Benzaid, C.; Gray, P.; Marin, E.; Makris, A.; Cordeiro, L.; Diego, F.; Sorokin, P.; Girolamo, M.D.; Security in Cloud-Native Services: A Survey. *J. Cybersecur. Priv.* **2023**, *1*, 1–36. https://doi.org/

Received: 19 August 2023
Revised: 6 October 2023
Accepted: 17 October 2023
Published:

1. Introduction

Cloud-native services [1] refer to a paradigm shift in the design, development, deployment, and management of software applications within the cloud computing paradigm. This approach leverages the inherent capabilities of cloud infrastructure to optimize the performance, availability, scalability, and efficiency of applications. At its core, cloud-native services encompass a set of principles and practices that enable the creation of highly resilient, scalable, and portable applications. These services are built upon containerization, which encapsulates individual application components and their dependencies into lightweight, isolated containers. This containerization allows for efficient resource utilization [2], facilitates rapid deployment, and ensures consistent behavior across different environments.

These types of services constitute a contemporary paradigm in software development and deployment, deeply rooted in the adoption of DevOps [3] principles, a scientific approach that prioritizes collaboration and automation between development and operations teams. DevOps practices such as Continuous Integration and Continuous Deployment (CI/CD) [4] play a pivotal role, in automating the integration, testing, and deployment of code changes to ensure software quality and rapid feature deployment with reduced risks. Furthermore, Infrastructure as Code (IaC) [5], an essential component of DevOps, enables the provisioning and management of infrastructure resources through code, promoting consistency and reproducibility. Scientifically, this approach aligns infrastructure management with version control and automation, fostering a more efficient, reliable, and scalable software development and deployment process within the dynamic realm of cloud-native services.

Cloud-native services are built using microservice architectures [6], where applications are decomposed into smaller, loosely coupled, and independently deployable services. Each service focuses on a specific business functionality and can be developed, deployed, scaled, and updated independently. This architecture promotes flexibility, agility, and scalability, allowing organizations to rapidly deliver new features and adapt to changing demands [7]. Microservice architectures leverage containers to package and isolate applications and their dependencies. Containers provide a lightweight and portable runtime environment, ensuring consistency across different computing environments [8]. They enable easy scaling, deployment, and management of applications, allowing organizations to efficiently utilize resources and achieve fast startup times. Orchestration platforms, such as Kubernetes [9], play a crucial role in managing and automating the deployment, scaling, and management of containers in a cloud-native environment. They provide features like service discovery, load balancing, and self-healing, ensuring that applications run smoothly and reliably [10]. Orchestration platforms abstract away the underlying infrastructure complexities, allowing developers to focus on building and deploying applications without having to worry about the infrastructure details.

Cloud-native services leverage the scalability and resilience that are associated with cloud infrastructures [11]. They utilize auto-scaling capabilities to automatically adjust resources based on workload demands, ensuring optimal performance and cost-efficiency. Load balancers distribute incoming traffic across multiple instances of services, ensuring high availability and fault tolerance [12]. Distributed data storage and caching mechanisms are employed to handle large volumes of data and provide fast and efficient access [13]. Furthermore, cloud-native services require robust monitoring and observability capabilities to ensure optimal performance and detect issues. Monitoring tools [14] collect and analyze metrics, logs, and traces from various components of the cloud-native environment, providing insights into the health, performance, and behavior of services. Observability facilitates troubleshooting, performance optimization, and proactive management of applications [15]. Such architectural paradigms are characterized by an overly distributed nature.

However, this distributed nature, despite its numerous benefits, introduces unique cybersecurity challenges. These services are provided through a network of servers, creating a large attack surface for potential attackers. Due to that, they are susceptible to all sorts of threats such as malware, DDoS, or Man-in-the-Middle (MITM) attacks [16,17]. Furthermore, many of these services process sensitive data, including user data and proprietary information, which are often stored in the cloud and must be protected from unauthorized access [18]. On top of that, the dynamic and scalable nature of cloud-native services makes maintaining a consistent security posture challenging. The services must be able to scale quickly to meet user demand, but this requires the deployment of new instances and infrastructure, which can introduce new vulnerabilities. Due to these facts, there is an ever-increasing need to establish efficient security solutions [19,20] that are capable of mitigating the potential security threats associated with cloud-native services, while being aligned with the aforementioned characteristics of cloud-native services. In order to do so, it is of paramount importance to identify and analyze the desired features in terms of cybersecurity that cloud-native services should have.

Despite the fact that there have been numerous surveys that explore various aspects of security (such as requirements, challenges, and solutions) in the cloud [21–36], there have been none that focus on security from the perspective of cloud-native services. In fact, although these two terms are closely related, there are some distinct differences. Security in cloud-native services specifically focuses on the security considerations related to applications and services that are designed and built to leverage the capabilities of cloud-native architectures. These services are characterized by their use of containerization, microservices, serverless computing, and orchestration platforms like Kubernetes. Furthermore, security in cloud-native services addresses the unique challenges and requirements that arise from the adoption of cloud-native architectures, such as container security [37], service mesh security [38], and the dynamic nature of deployments. It encompasses the security of individual cloud-native components, their interactions, and the overall security of the cloud-native environment. As a matter of fact, on top of some aspects such as access management [39], data security [40], security monitoring [41], and incident response [42] that are of paramount importance in the context of cloud security, any attempt at establishing security in cloud-native services shall also incorporate:

- **Container [43] and Microservice [44] Security:** This involves securing the containerized components, such as Docker containers, used in cloud-native applications. It includes measures like vulnerability scanning, secure container image management, and runtime protection to mitigate container-specific risks, such as container escape attacks or compromised container images. Furthermore, security in cloud-native services considers the security of individual microservices and their interactions. This includes securing communication channels between microservices, implementing access controls and authentication mechanisms for inter-service communication, and ensuring proper authorization and data protection between microservices.
- **Orchestration Platform Security [45]:** Security considerations extend to the orchestration platforms used in cloud-native environments, such as Kubernetes. This involves securing the Kubernetes control plane, implementing secure configuration practices, and protecting critical components like the etcd data store. It also includes securing container orchestration and deployment processes to prevent unauthorized access or unauthorized changes to deployments.
- **DevSecOps Practices [46]:** Security in cloud-native services embraces the integration of security practices throughout the development, deployment, and operations lifecycle. It emphasizes embedding security as an integral part of the development process and implementing security automation, Continuous Security Testing, and security monitoring as part of the DevSecOps approach.

Table 1 encapsulates the gaps in contemporary surveys that motivated us to construct this work. After extensive research, we were able to identify only a single work [47] that covers areas similar to the ones that are being explored in the context of this survey. That work, however, did not go into much detail regarding aspects that are of paramount importance in the context of securing cloud-native services such as DevSecOps Practices and Orchestration Platform Security. We, instead, chose to dedicate a quite significant portion of this survey to exploring key features and solutions that relate to these aspects. Furthermore, our work is based on a unique structure of exploratory analysis that maps the key aspects of security in cloud-native services to the corresponding features, and these features to numerous contemporary solutions. Figure 1 illustrates this mapping process that constitutes the core of this work.

This work commences by presenting the key aspects of cybersecurity in cloud-native service. Then, we identify and explore the challenges and features, as well as map the latter with the aforementioned aspects. The rest of this work is dedicated to exploring in great detail a plethora of contemporary solutions when leveraged are capable of manifesting them.

This exploration includes several prominent enabling technologies, as well as a detailed analysis of how these technologies can be combined alongside others in order to establish autonomic and cognitive security management. Furthermore, it also includes various solutions that correspond to the secure software development lifecycle paradigm. Moreover, for each of these solutions, we identify various candidate open-source technologies that are capable of supporting the realization of the corresponding solution. Finally, this work concludes by mapping the explored solutions to the aforementioned features and presenting how these solutions can work together in order to establish each corresponding feature.

Thus, the main scientific contributions of this work are:

- It is the first survey that is dedicated to exploring security in cloud-native services.
- It serves as a uniquely structured mapping study that maps the key aspects of security in cloud-native services to the corresponding features, and these features to numerous contemporary solutions.
- It includes the identification of various candidate open-source technologies that are capable of supporting the realization of each explored solution.
- It showcases how these solutions can work together in order to establish each corresponding feature.

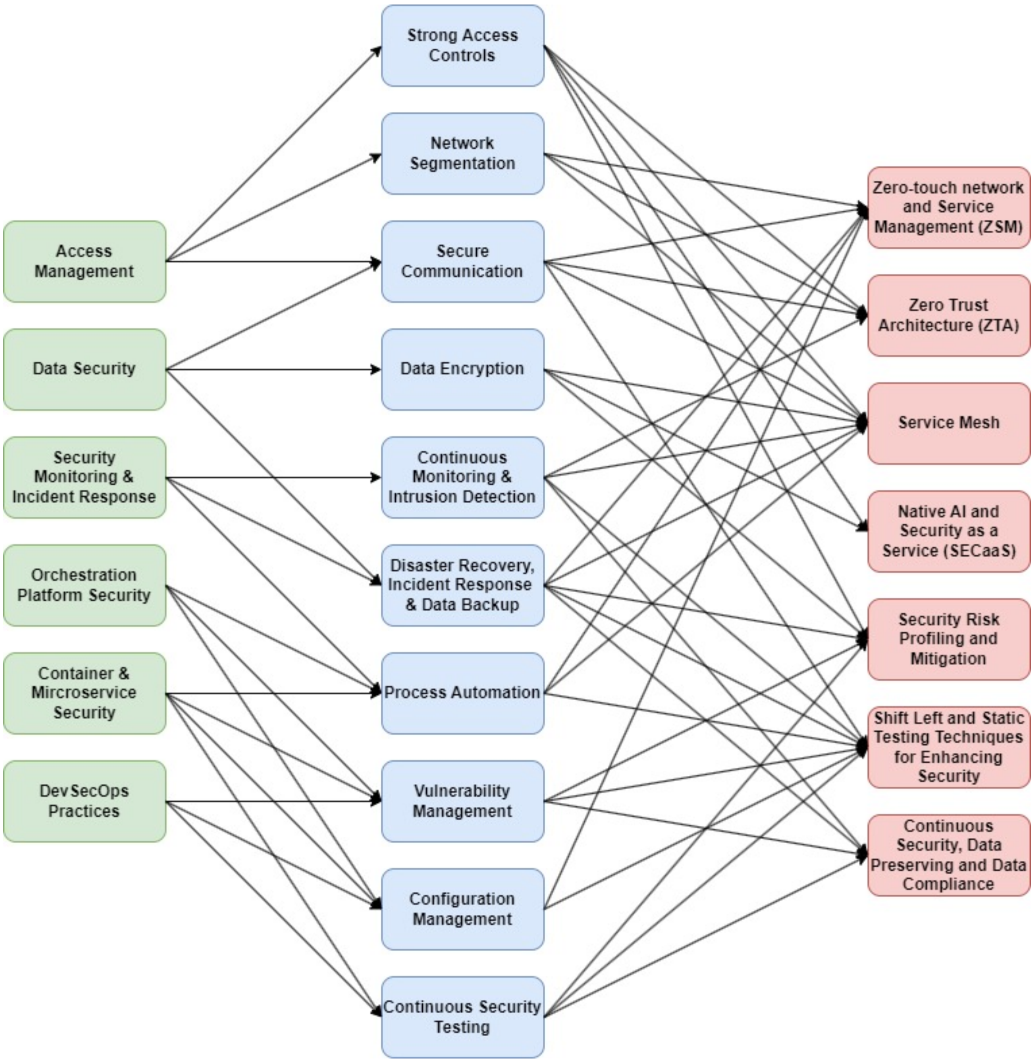

Strong Access Controls
Network Segmentation
Secure Communication
Data Encryption
Continuous Monitoring & Intrusion Detection
Disaster Recovery, Incident Response & Data Backup
Process Automation
Vulnerability Management
Configuration Management
Continuous Security Testing
Access Management
Data Security
Security Monitoring & Incident Response
Orchestration Platform Security
Container & Mircroservice Security
DevSecOps Practices
Zero-touch network and Service Management (ZSM)
Zero Trust Architecture (ZTA)
Service Mesh
Native AI and Security as a Service (SECaaS)
Security Risk Profiling and Mitigation
Shift Left and Static Testing Techniques for Enhancing Security
Continuous Security, Data Preserving and Data Compliance

Figure 1. A systematic mapping process that involves key aspects, features, and solutions for security in cloud-native services. Key aspects are colored green, features are colored blue, and solutions are colored red for illustration purposes.

Table 1. Research gaps in prior works.

	Key Aspects of Security in Cloud-Native Services					
Prior Surveys	**Access Management**	**Data Security**	**Security Monitoring and Incident Response**	**Container and Microservice Security**	**Orchest. Platform Security**	**DevSecOps Practices**
Secure cloud infrastructure: a survey on issues, current solutions, and open challenges [21].	✓	✓	✓	×	×	×
Security in cloud computing: opportunities and challenges [22].	✓	✓	✓	×	×	×
A survey on security challenges in cloud computing: issues, threats, and solutions [23].	✓	✓	✓	×	×	×
A survey on secure cloud: security and privacy in cloud computing [24].	✓	✓	✓	×	×	×
Secure cloud computing for critical infrastructure: a survey [25].	✓	✓	✓	×	×	×
State-of-the-art survey on cloud computing security challenges, approaches, and solutions [26].	✓	✓	×	×	×	×
A comprehensive survey on security in cloud computing [27].	✓	✓	×	×	✓	×
A survey of security issues for cloud computing [28].	✓	✓	✓	×	×	×
A survey on cloud security issues and techniques [29].	✓	✓	×	×	×	×
Cloud computing security: a survey [30].	✓	✓	✓	×	×	×
A survey on cloud computing security: issues, threats, and solutions [31].	✓	✓	✓	×	×	×
A survey of cloud computing security challenges and solutions [32].	✓	✓	✓	×	×	×
Cloud computing security challenges and solutions: a survey [33].	✓	✓	×	×	×	×
Cloud computing security: a survey of service-based models [34].	✓	✓	✓	×	×	×
Cloud security threats and solutions: a survey [35].	✓	✓	×	×	×	×
Understanding the challenges and novel architectural models of multi-cloud native applications: a systematic literature review [36].	✓	✓	✓	✓	×	✓

Towards achieving these goals, the paper is structured in the following manner. Section 2 is dedicated to briefly presenting the security challenges that are present in cloud-native services. Section 3 is dedicated to identifying the key features for security in cloud-native services and mapping them to the aforementioned key aspects. Section 4 analyzes various technological enablers for cybersecurity in cloud-native services. Section 5 is dedicated to showcasing how these technological enablers can be combined in order to establish autonomic and cognitive security management. Section 6 is dedicated to the secure software development lifecycle. This section includes areas of secure software development lifecycle that are vital in the context of cloud-native services, such as Security Risk Profiling and Mitigation, shift left and static testing, as well as Continuous Security, Data Preserving, and Data Compliance techniques for enhancing security. Section 7 is dedicated to showcasing how the various, previously explored solutions can collaborate to implement each respective feature. Finally, Section 8 summarizes the merits of this work. The findings of this work derived from the exploratory analysis [48] on cybersecurity in cloud-native services were conducted in the frame of the CHARITY [49] project. The insights and findings of this work are expected to be quite useful for cybersecurity professionals and developers who may leverage them in order to enhance the security and privacy of cloud-native services.

2. Challenges for Security in Cloud-Native Services

Cloud-native services offer significant advantages such as scalability, flexibility, and cost-efficiency. However, establishing secure cloud-native services is quite difficult given the fact that there are numerous inherent challenges. This section is dedicated to briefly exploring some of them. In the frame of cloud-native services, the processing and storage of sensitive data present additional cybersecurity challenges. Furthermore, safeguarding user data and proprietary information is crucial to maintain trust, comply with privacy regulations, and prevent potential legal, monetary, and reputational consequences. Furthermore, the dynamic and scalable nature of cloud-native services introduces unique challenges when it comes to maintaining a consistent security posture; although scalability is a key advantage, it also necessitates careful consideration of security implications. The notion that the distributed nature of cloud-native services introduces specific cybersecurity challenges is supported by the examination of the types of attacks that they are prone to [50].

- **Malware Attacks [51]:** The distributed nature of cloud-native services makes them vulnerable to malware attacks. Malicious software can infiltrate one part of the infrastructure and spread across the interconnected components, potentially causing data breaches, unauthorized access, or disruption of services. Additionally, malware can be designed to exploit specific weaknesses in cloud-native architectures, compromising the security and integrity of the entire system.
- **Man-in-the-Middle (MITM) Attacks [52]:** The distributed nature of cloud-native services also increases the risk of Man-in-the-Middle attacks. In such attacks, an attacker intercepts and alters the communication between two parties without their knowledge. This can lead to data leakage, unauthorized access, or manipulation of sensitive information. MITM attacks can exploit vulnerabilities in communication channels, weak encryption protocols, or compromised certificates.
- **DDoS Attacks [53]:** Cloud-native services are also susceptible to Distributed Denial of Service (DDoS) attacks, where a large number of compromised devices flood the network or services with excessive traffic, overwhelming the infrastructure and causing service disruptions. These attacks, which are also referred to as Cloud Zombie attacks [54], can be devastating for cloud-native services as they rely heavily on network connectivity and availability [55]. Furthermore, there have been certain variations of this type of attack that are designed to exploit the autoscaling capabilities of cloud environments [56]. These variations include Yo-Yo attacks [57], which rely

on producing periodic bursts of traffic to oscillate the auto-scaling system between scale-out and scale-in status inducing economic loss to the tenant.

In order to better understand the intricacies that are intertwined with tackling such attacks, it is worth examining Link Flooding Attacks (LFAs) [58], which have emerged as another prominent example of DDoS Attacks. LFAs aim to disrupt the network connectivity of numerous users by congesting critical network links. Essentially, adversaries seek to overwhelm these links by injecting a substantial volume of flows that simultaneously traverse a specific set of core network links. The challenge lies in the fact that adversaries can employ low-volume, individual flows that blend with regular traffic, making it arduous for network operators to implement efficient countermeasures against LFAs. Research conducted by Meier et al. [59] included launching an LFA on a specific link without any knowledge of the network topology necessitates adversaries to transmit five times more flows compared to when they possess such information. Similarly, the number of flows required to execute an LFA against a target link increases exponentially when the topology remains unknown. These findings clearly indicate that having a certain level of network topology knowledge is a crucial requirement for carrying out such attacks effectively, efficiently, and discreetly.

At first glance, it may seem logical to assume that maintaining the confidentiality of the network topology would raise the cost of executing successful LFAs. This defense strategy aligns with the approach adopted by Internet Service Providers (ISPs), who consider their network topology as sensitive information. However, researchers have demonstrated that existing path-tracing tools like traceroute can be utilized to infer the previously unknown network topologies of ISPs, including their forwarding behavior and flow distribution patterns. This means that adversaries can employ these techniques to conduct LFAs with greater efficiency and effectiveness. In recent years, various proactive countermeasures have been proposed to mitigate LFAs by presenting a virtual (false) network topology. These countermeasures aim to hide potential bottleneck links and nodes while preserving the utility of path-tracing tools. However, a study [60] that after analyzing three state-of-the-art proactive network obfuscation defenses: NetHide [59], Trassare et al.'s solution [61], and LinkBait [62] identified four common weaknesses that significantly compromise the security and utility of the exposed virtual topologies. Such instances are indicative of the fact that tackling such attacks is a multifaceted endeavor that requires very careful examination.

Furthermore, previous works [63] have shown that adversaries with access to a container (running inside a host that is part of a large cluster) can exploit vulnerabilities at various layers to conduct very powerful attacks. Of these vulnerabilities, the most dangerous ones are those that target the OS kernel of the host where the (malicious) container resides. By sending maliciously crafted syscalls, adversaries can trigger memory leaks, write arbitrary contents into shared files with the host, or gain elevated privileges in the host, among others. One notable example is the weakness found in the waitid system call, which enabled adversaries to run a privilege escalation attack to gain access to the host. The root cause behind the existence of these types of attacks lies in the excessive default privileges granted to containers, whereas several Linux kernel security modules, such as AppArmor [64] and SELinux [65], have been introduced to restrict container capabilities; none of these modules adequately address the fundamental question of minimizing container privileges. To tackle this, it is of paramount importance to leverage tools that are capable of automatically identifying the minimum set of privileges that containers need for executing their applications correctly while minimizing their interactions with the OS kernel.

Unfortunately, the level of threat that such attacks pose is greatly enhanced by the fact that cloud-native services rely on a network of servers and interconnected components, creating a larger attack surface for potential cyber threats. Attackers can exploit vulnerabilities in different layers of the cloud infrastructure, including the underlying hardware, hypervisors, virtual machines, containers, and the cloud provider's management interfaces.

This architectural design results in an Increased Attack Surface [66] that greatly jeopardizes the ability to establish secure cloud-native services.

3. Key Features for Security in Cloud-Native Services

In order to mitigate the aforementioned cybersecurity challenges, parties utilizing cloud-native services should incorporate the following features that shall be explored in this section. After carefully examining the corresponding scientific literature, the authors of this work were able to construct Table 2 that illustrates how these features correspond to the aforementioned key aspects for security in cloud-native services. These features are the following:

Table 2. Mapping of key aspects to features.

Key Aspects	Features
Access Management	Strong Access Controls Network Segmentation Secure Communication
Data Security	Secure Communication Data Encryption Disaster Recovery, Incident Response, and Data Backup
Security Monitoring and Incident Response	Disaster Recovery, Incident Response, and Data Backup Continuous Monitoring and Intrusion Detection Process Automation
Orchestration Platform Security	Process Automation Vulnerability Management Configuration Management
Container and Microservice Security	Process Automation Vulnerability Management Configuration Management Continuous Security Testing
DevSecOps Practices	Vulnerability Management Configuration Management Continuous Security Management

- **Strong Access Controls [67]:** Implementing strict access controls, including authentication, role-based access control, and least privilege principles helps prevent unauthorized access and reduces the impact of potential breaches. Strong access controls are crucial in cloud-native services to ensure the security, confidentiality, and integrity of data and resources. Access controls govern the authentication and authorization processes, determining who can access specific resources and what actions they can perform. Here are the key reasons why strong access controls are important in cloud-native services:
- **Network Segmentation [68]:** Network segmentation is a security practice that involves dividing a network into smaller, isolated segments to enhance the security and control of cloud-native services. Each segment, known as a network zone or subnet, contains a specific set of resources with defined access controls. Properly segmenting the cloud-native infrastructure into isolated networks and subnets can limit the lateral movement of attackers and contain the impact of potential breaches.
- **Data Encryption [69]:** Ensuring that data are encrypted both at rest and in transit helps protect sensitive information from unauthorized access. Data encryption plays a vital role in ensuring the security and privacy of data in cloud-native services. As organizations increasingly adopt cloud computing and migrate their infrastructure and applications to the cloud, the need for robust data protection mechanisms becomes even more critical.

- **Secure Communication [70]:** Utilizing secure communication protocols, such as HTTPS, and regularly updating cryptographic libraries and certificates enhances the integrity of data transfers. Secure communication in cloud-native services involves implementing measures to protect data and ensure the confidentiality, integrity, and authenticity of information exchanged between components within the cloud-native environment.
- **Continuous Monitoring [71] and Intrusion Detection [72]:** Employing robust monitoring and intrusion detection systems enables the early detection of potential threats, allowing for timely response and mitigation. This includes monitoring network traffic, analyzing log data, and leveraging threat intelligence to identify and respond to suspicious activities. Monitoring the security of dynamically scaling cloud-native services is essential but can be challenging. The increased number of instances, constant changes in the infrastructure, and complex network connections make it difficult to maintain visibility and detect potential security incidents. Implementing robust security monitoring tools and practices, such as centralized logging, real-time threat detection, and anomaly detection, can help identify security breaches and enable prompt incident response.
- **Disaster Recovery [73], Incident Response [74], and Data Backup [75]:** Establishing comprehensive disaster recovery plans and incident response procedures ensures that organizations can quickly recover from cybersecurity incidents and minimize their impact on operations. Regularly testing and updating these plans is crucial to maintaining their effectiveness. Furthermore, cloud-native services should have regular backup schedules, reliable backup storage solutions, and well-documented recovery procedures.
- **Process Automation [76]:** Process automation is essential for enhancing cybersecurity in cloud-native services. It reduces human error, ensures consistency, and standardizes security practices. Automation enables quick detection and response to security events, supports scalability [77], and facilitates continuous compliance. It also integrates threat intelligence for proactive threat hunting and faster incident response. Overall, automation strengthens security, protects data, and mitigates risks in the dynamic cloud-native environment. As such, in the context of facilitating secure cloud-native services, process automation refers to areas such as service orchestration, mitigation of incidents, security management, security checks, and static application testing.
- **Vulnerability Management [78]:** With the dynamic nature of cloud-native environments, vulnerability management becomes complex. Traditional vulnerability management practices may not be sufficient due to the large number of instances and rapid deployment cycles. Identifying and patching vulnerabilities in a timely manner becomes crucial to preventing potential exploits. Continuous vulnerability scanning, automated patch management [79], and integration with security tools and processes can help address this challenge.
- **Configuration Management [80]:** Cloud-native services rely heavily on configurations that define their behavior and security settings. However, managing and enforcing consistent configurations can be challenging in a dynamic and scalable environment. Configuration drift, where instances deviate from their desired configurations over time, can lead to security vulnerabilities. Implementing configuration management tools and practices that automate configuration enforcement and regularly verify compliance can help mitigate this challenge.
- **Continuous Security Testing [81]:** Regular security testing [82], including vulnerability assessments, penetration testing, and security code reviews, should be integrated into the development lifecycle of cloud-native services. Adopting a DevSecOps approach, where security is incorporated throughout the software development and deployment processes, helps identify and address vulnerabilities early on and ensures that security measures are continuously tested and validated.

To overcome the challenges associated with maintaining a consistent security posture in dynamic and scalable cloud-native environments, organizations should prioritize security considerations throughout their cloud-native development and deployment processes. Emphasizing automation, configuration management, vulnerability management, and Continuous Security monitoring helps ensure that security controls are consistently applied, vulnerabilities are promptly addressed, and potential security incidents are detected and responded to in a timely manner. The remainder of this work is dedicated to exploring numerous solutions that can be leveraged in order to establish the aforementioned features.

4. Enabling Technologies for Cybersecurity in Cloud-Native Services

As services continue, the shifting to cloud-native environments, adopting security measures and practices tailored to this environment becomes essential. This section explores the role of some of the most prominent enablers for the security of the next generation of cloud-native services, including the zero-touch network and service management (ZSM), the zero-trust architecture (ZTA), service mesh, Native AI, and SECaaS.

4.1. Zero-Touch Network and Service Management (ZSM)

In cloud-native environments, it is crucial to automate the detection and mitigation of network anomalies intelligently. Moreover, since they are typically multi-tenancy environments, a failure in a given service/level can harm the rest of the environment. Proposed by ETSI, the Zero-touch network and Service Management (ZSM) specification [83] defines an End-to-End (E2E) management reference architecture that aims to provide a more flexible and automated approach for E2E service orchestration in multi-domain deployments. In the context of Zero-touch Network and Service Management, E2E refers to a comprehensive approach that encompasses the entire lifecycle of network and service management, from initial provisioning to ongoing operations and maintenance, without requiring manual intervention. The E2E principle in ZSM aims to automate and streamline the management of network infrastructure, services, and associated operations, reducing human involvement and minimizing errors. Key aspects of this approach in ZSM include:

- **Automated Provisioning:** It focuses on automating the provisioning process, starting from the initial setup of network devices and services. It involves automating the configuration, deployment, and activation of network resources and services without the need for manual intervention. Automated provisioning ensures consistent and reliable deployment, reducing the potential for human errors and misconfigurations.
- **Service Orchestration:** It emphasizes the orchestration of services across the entire network infrastructure. It involves automated coordination and management of various network functions and services, ensuring seamless integration and interoperability between different components. Service orchestration helps optimize resource utilization, enhance service delivery, and enable efficient scaling and elastic provisioning of services.
- **Automated Operations and Maintenance:** It promotes automation in ongoing operations and maintenance tasks. This includes automating routine management tasks, such as monitoring, performance management, fault detection, and troubleshooting. By automating these tasks, organizations can reduce manual effort, improve operational efficiency, and respond quickly to network incidents or performance issues.
- **Closed-Loop Automation:** It involves closed-loop automation, where automated processes continuously monitor the network, collect data, analyze it, and take proactive actions based on predefined policies and rules. Closed-loop automation enables self-healing capabilities, where network issues or service degradation can be automatically identified and remediated without human intervention.
- **Analytics and Intelligence:** It leverages analytics and intelligence to gain insights from network data and make data-driven decisions. By applying machine learning and artificial intelligence techniques, organizations can analyze network performance,

user behavior, and security data to optimize resource allocation, detect anomalies, and enhance overall network operations and security.

- **Security and Compliance:** It includes incorporating security and compliance measures throughout the network and service management lifecycle. This involves integrating security controls, such as access controls, encryption, and threat detection into automated processes. It also ensures compliance with relevant regulations and standards throughout the management and operation of network services.

ZSM specification addresses how a set of management services (and their respective capabilities) can enable a more consistent and standardized method to manage the entire life-cycle of services spanning over that multi-domain scenarios [84]. Indeed, ZSM defines the concept of a Management Domain (MD) by encompassing the existing management functions of each domain and an integration fabric, to expose such capabilities and facilitate the communication between service consumers and producers. Moreover, ZSM reference architecture includes an E2E Service Management Domain (and a cross-domain integration fabric) to support the overall E2E orchestration capabilities. Furthermore, ZSM specification builds upon the principle of closed-loop management automation, where feedback-driven processes (e.g., OODA loop model) can be leveraged to realize fully automated (and intelligent) management functionalities [85]. ZSM-like architectures are not only beneficial for achieving a more automated and intelligent service orchestration [86] but they can also be leveraged to facilitate the deployment and lifecycle management of ancillary security mechanisms in cloud-native environments (e.g., purpose-built security probes, DevSecOps toolchains, etc.).

4.2. Zero-Trust Architecture (ZTA)

Moreover, in the past, the traditional perimeter security model was based on the "trust but verify" approach. As long as components and equipment belong to a specific network or segmented group, they are considered trustworthy. In such a model, network segmentation, used to establish a trust zone, was assumed to be sufficient. The focus was on whether or not to authorize access to network segments through perimeter security mechanisms such as firewalls. The perimeter model can protect against threats from outside but does not prevent unauthorized lateral movement. Moreover, the implicit perimeter trust model, based on the segmentation of virtual or physical perimeters, does not fit the increasingly complex and dynamic cloud computing environments [87]. Instead, it is imperative to adopt a security model where trust is never granted by default and the principle of least privilege is followed to grant only the minimum necessary privileges, thus limiting the visibility and accessibility of assets [88]. One of the purposes of the zero-trust model is to prevent unauthorized lateral movement as no implicitly trusted zones exist (e.g., no internal network segments are trusted by default). Zero trust intends to provide a more granular and dynamic segmentation of the different resources (i.e., even when network traffic belongs to the same network, it should undergo a validation process) [89].

According to NIST, the zero-trust architecture must follow the basic zero-trust principles, namely [90]: data, assets, and services are considered resources; communications should be secured regardless of whether networks are considered enterprise-owned or non-enterprise-owned; access to resources is only granted per session; trust should be evaluated before access is granted; access is determined by multiple aspects including the observable state of the user and the service requesting asset; no resource is inherently trusted, there is always an assessment for each access request, which implies monitoring the integrity of the resources.

Figure 2 illustrates the difference between a perimeter security model and the zero-trust model. In the classical perimeter security model, the authentication focuses on outsider communication to the local area network (LAN), assuming the network is trusted, whereas in the zero-trust model, the network is never trusted and each communication should go through an authentication process. In a service-based architecture and an edge/cloud environment, this means that the enforcement of security policies should occur in all network communications between services (and containers).

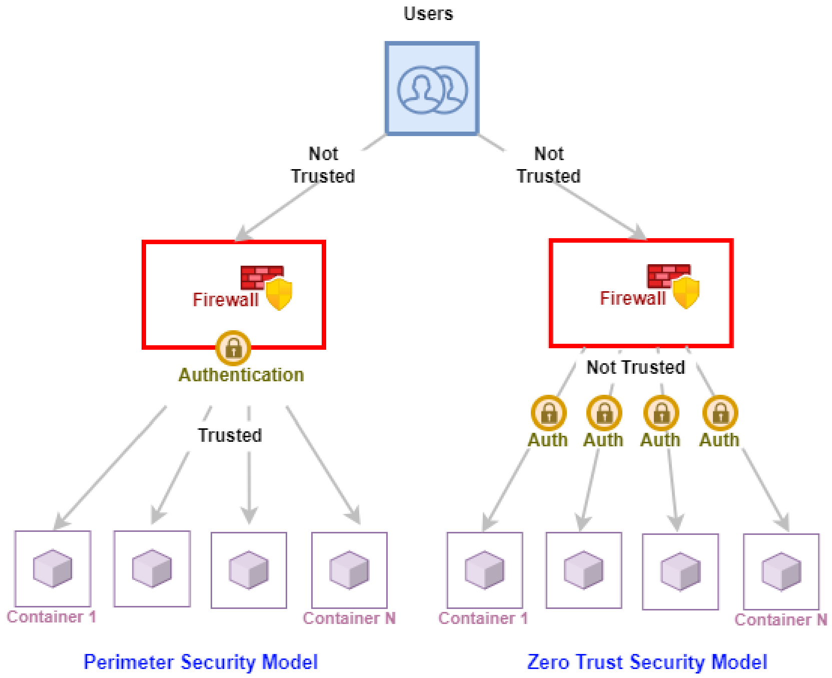


Figure 2. Comparison between perimeter security model and zero-trust model.

Zero trust does not aim to replace the security perimeter model but complements it with a more granular approach to enforce security policies. Whilst zero trust is often linked to authentication and authorization policy enforcement, before granting access to resources, here, its use is based on the idea of having the means for continuous monitoring of all the network traffic (i.e., both north–south and east–west traffic) within a cloud-native environment. Such a continuous evaluation of network traffic is crucial to the timely detection of cyberattacks (e.g., due to compromised components or API abuses) and thus complements additional policy enforcement strategies, fulfilling the zero trust overall principle of "always verify".

4.3. Service Mesh

The rise of microservice-based architectures is not without its challenges. Despite all the benefits and flexibility, such architectures are composed of numerous and dynamic components that need to communicate and interact with each other. Hence, one of the biggest challenges is how to deal with the exponential and complex relationship between all the moving parts that form a cloud-native application.

Service mesh [91] is an increasingly widely used solution to address challenges of complex and exponential relationships between components [92]. The service mesh concept has gained a lot of strength to achieve network and resource observability; a key feature within any cloud-native environment. An example of this approach, Eunji Kim et al. [93] used Istio and Envoy for data collection and full visibility into infrastructure resources, network traffic, and application behavior in the environment. Similarly, the INSPIRE-5Gplus https://www.inspire-5gplus.eu/inspire-5gplus-high-level-architecture/ (accessed on 15 August 2023) project adopted Istio in implementing the integration fabric of its 5G security management framework [94].

Service mesh is an infrastructure layer for handling service-to-service communication without imposing changes to services [92]. Service meshes provide a more cloud-native and comprehensive strategy to control network traffic, apply distinct policies, and cope with the complexity and dynamism of service communications. Service mesh addresses such challenges by shifting common orchestration-related functions from the application logic to the infrastructure layer. The immediate benefit is the simplification of applications that would otherwise need to include them. Moreover, by abstracting common functions to the infrastructure layer, one might envision a more standard approach to implementing them. As a summary, service meshes are used to provide the following key features:

- **Service Registry and discovery**: Mechanisms used to facilitate the process of discovery and registry of new components and services.
- **Load balancing**: The ability to balance network traffic depending on aspects such as latency, infrastructure, and health status.
- **Fault tolerance**: The ability to redirect requests to an alternate instance when the original is degraded or not available.
- **Traffic monitoring**: The ability to monitor all the network traffic and key metrics between the mesh of microservices.
- **Encryption, Authentication, and Access Control**: The possibility of dynamically encrypting the network communication on the fly as well as the possibility to authorize and authenticate network communication between services.

Although conceptually not restricted, the rise of service meshes is tied to cloud-native applications [95]. As depicted in Figure 3, a service mesh is constituted of two planes: data and control planes. The data plane is composed of a set of proxies known as sidecars, co-located in each microservice. The sidecars are proxies that intermediate the communication with other proxies. The combination of several proxies forms a mesh network intercepting the communication between microservices, mediating and controlling all network communications, and collecting telemetry. Together, they have full visibility over all microservice communications, being able to perform health checking, filtering, routing, load balancing, service discovery, authentication, authorization, collecting telemetry, etc. The control plane is responsible for the overall orchestration of sidecar behavior. Herein, the proxies provide the capabilities to configure the components in order to collect telemetry and enable observability and also to apply policies for routing traffic [96]. The service mesh concept is used as an architectural approach to enforce security policies on top of microservices network traffic.

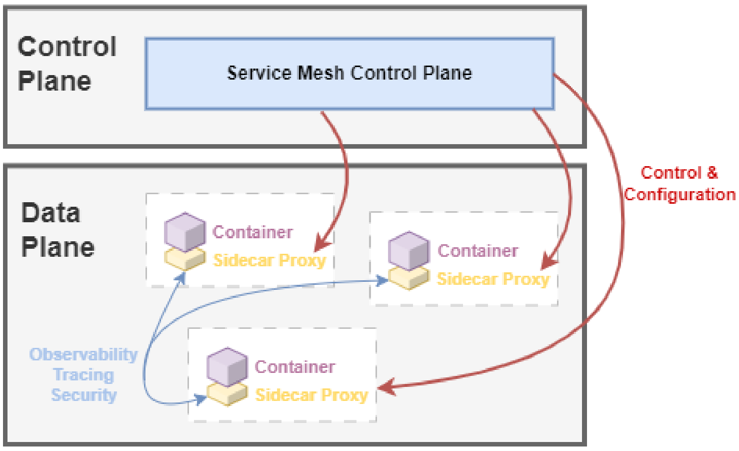


Figure 3. Data and control planes of a service mesh, adapted from https://www.nginx.com/blog/what-is-a-service-mesh/ (accessed on 15 August 2023).

Although this approach has many benefits, they have not yet been fully investigated in the available literature. Chandramouli et al. [97] address this issue in a NIST publication. This publication presents some additional implementation guidance on security solutions for cloud-native environments, more specifically, how proxy-based service mesh components can collectively form a security infrastructure to support microservices. The topic of anomaly detection in cloud-native environments has new challenges and new relevance, as discussed in several works. Harlicaj et al. [98] discussed this problem for the detection of web-based attacks on microservices, using Kubernetes and Istio. Similarly, in [99] the authors proposed an anomaly detection mechanism in API traffic to improve its security in microservice environments. Characteristics such as bandwidth, number of consecutive requests were some characteristics analyzed, concluding that such a mechanism is more accurate than a rule-based anomaly detection mechanism. On the other hand, both works do not cover automated mitigation of incidents (e.g., based on policies).

Istio is a realization of an open-source service mesh platform that establishes control over the way microservices share data between them. It comprises a set of layered distributed applications providing traffic management, security, and observability at the service mesh level. Istio exposes APIs to enforce access control at mesh, namespace, and service levels. As a result of applying policies over a Kubernetes network, it becomes possible to configure security for service communications at the network and application layers. Istio APIs also provide support for integration with other components, such as telemetry or policy systems. The sidecars are deployed as Envoy proxies supporting compatibility issues. Envoy provides built-in functionality to cope with challenges, such as retries, delays, circuit breakers and fault injection, dynamic service discovery and load balancing, traffic management and routing, security policies, and rate limiting. Istio offers security advantages through strong identity, transparent TLS encryption, powerful policy, and authentication, authorization, and auditing (AAA) tools to protect services and data exchanged between them. However, another feature to highlight is its support for heterogeneous environments, including virtual machines (VMs), Kubernetes, and multi-domain settings [100].

It is not only a challenge to efficiently observe and understand, for instance, all the network traffic of an environment, but it is also crucial to have the means to apply mitigation measures and security policies effectively. One example of policy enforcement in cloud-native environments is the architecture proposed by Loıc Miller et al. [101] that uses Istio combined with the Open Policy Agent (OPA) to execute policies. It should be noted that although this architecture presents key ideas focused on the authorization of communications through the set of defined policies, it does not present ways of detecting anomalies. OPA is an open-source, general-purpose policy engine that unifies the implementation of policy enforcement procedures across IT environments, such as the ones involving cloud-native applications. OPA enables decoupling policy decisions from software for policy enforcement, and whereas Istio policies are limited to networks, OPA allows a more comprehensive strategy to implement distinct policies and provide more control over deployments and containers.

4.4. Native AI and Security as a Service (SECaaS)

This section discusses the role of Native AI and the benefits of the Security as a Service (SECaaS) [102] model as an approach to support applications' security and privacy requirements. Native AI represents a disruptive paradigm where AI capabilities are leveraged to realize or augment the existing capabilities of a system [103]. Security-wise, this provides many advantages for bolstering threat detection and prevention [104–106]. AI security mechanisms can facilitate the analysis of user behavior, system interactions, and network traffic [107] within applications to identify anomalies or suspicious activities that may indicate potential security breaches. Native AI enhances access control mechanisms by harnessing cutting-edge technologies like biometric recognition, voice authentication, and gaze tracking, ensuring robust verification of user identities. Similarly, the SECaaS model has been used for multiple purposes such as Identity and Access Management (IAM), Information Security, Network Security, Intrusion Supervision, or Encryption [108,109]. SECaaS offers an opportunity to incorporate some of these mechanisms into applications' runtimes without design time intrusions [110]. The SECaaS model is also a more cost-efficient solution that allows next-generation developers and application providers to shift the onus of security functions to infrastructure providers. This way, various security and privacy mechanisms will be further investigated to take advantage of the hybrid edge/cloud network, compute, and storage resources and the notion of closed loops to have more autonomous and intelligent security coverage.

5. Autonomic and Cognitive Security Management

The amalgamation of many of the technologies that were presented during the previous section of this work shall serve as the building blocks for the emergence of frameworks that are capable of establishing cybersecurity in cloud-native services. This section is dedicated to exploring such a framework that is based on paradigms, such as the enabling technologies that were previously explored. Modern applications are being designed into smaller, more manageable microservices due to a plethora of requirements, such as portability, scalability, or reliability, in the context of cloud-native environments. From a security standpoint, such an emerging paradigm raises new challenges in terms of managing the volume and complexity of cloud-native applications. As such, this demands intelligent and automated solutions to lower the burden assigned to humans in the context of managing the security of cloud-native applications. With the objective of establishing autonomous secure management of resources in various domains, a framework that adheres to the key design principles of ETSI ZSM was proposed in [19], as depicted in Figure 4. This framework introduces AI-powered closed loops with various different scopes, from node level to end-to-end and inter-slice level. Thus, it allows the rapid and effective detection and mitigation of security threats close to the source, which prevents their proliferation in the network.

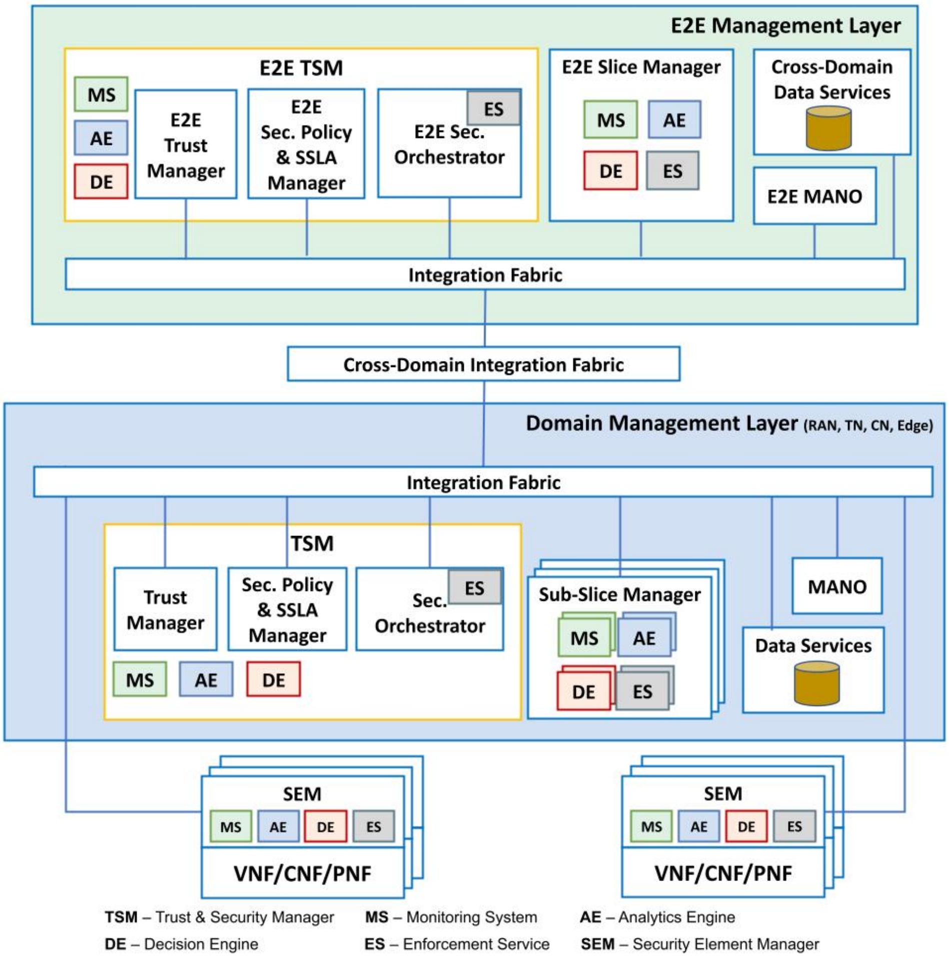


Figure 4. Architecture of an autonomic and cognitive security management framework, adapted from [19].

The framework enhances E2E security management capabilities across various domains in a hierarchical manner. It builds upon the domain vision presented in [111] and introduces AI-driven closed loops with varying scopes, ranging from individual network nodes to end-to-end connections and inter-slice levels. This fine-grained approach enables efficient and rapid detection and mitigation of security threats at their source, preventing their spread throughout the network. It is important to note that this framework aligns with the aforementioned fundamental design principles of ETSI ZSM. It achieves this by supporting the segregation of security management concerns and adopting a service-based architecture. This architecture allows authorized consumers to access and utilize the provided security management services through an integration fabric. The integration fabric facilitates the registration, discovery, and invocation of security management services. It also facilitates communication between these services and other management services. The framework leverages historical data and knowledge generated by various security management services, which are stored and made available through data services within the same domain or across multiple domains. In the following sections, we outline the main functional components of this framework.

In this architecture, the closed-loop automation [112] manifests in the context of four functions, Monitoring System (MS), Analytics Engine (AE), Decision Engine (DE), and Enforcement Service (ES). MS takes responsibility for gathering, preprocessing, and presenting security-related data obtained from the managed entity. AE offers services that enable the identification or prediction of potential security anomalies and attacks, as well as determining the root causes behind observed security incidents, utilizing the collected data. DE determines the most effective mitigation policy required to address the detected or predicted security issue, ensuring the desired level of security is maintained. ES allows triggering/updating implementations of specific Virtual Security Functions (VSFs) [113], such as vFirewall [114] and vIDS through the management and orchestration platform (MANO). Each network function is associated with a Security Element Manager (SEM) that will be responsible for managing security within its scope. As mentioned, in closed loops, cognitive resources are incorporated for security analysis and decision-making. Furthermore, to increase the cognitive level of the environment, AI/ML techniques can be implemented in MS and ES.

In order to have greater security and mitigation at different levels, multiple closed loops can be coordinated at different levels. These loops are managed and orchestrated by the “Trust & Security Manager” (TSM),which comprises three functional modules, the “Security Orchestrator”, “Security Policy and SSLA Manager”, and “Trust Manager”. The “Security Orchestrator” is responsible for designing, instantiating, and managing the runtime lifecycle of circuits. Security Policy and SSLA Manager are responsible for violating SSLAs (Security Service Level Agreements) [115] and security policies defined by external entities. Trust Manager is responsible for the continuous assessment of the reliability of the network services and associated circuits (this trust is calculated based on the trust attributes specified in the Trust Level Agreement).

Based on this framework, some open-source solutions that enable zero-touch security management in environments were analyzed. A cloud-native architecture [116], based on stateless microservices implemented as containers, is a technology recognized for being suitable for the cost efficiencies, flexibility, and scalability required in the operation and management of environments. In this sense, the concept of Platform-as-a-Service (PaaS) [117] emerges, a cloud-native architecture layer that allows developers to implement, run, and manage different applications without the complexity of configuring and maintaining the cloud. In this follow-up, Kubernetes appears as a de facto standard for the implementation and orchestration of applications in containers, which allows scalability, high availability, and fault tolerance features.

For the implementation of the Fabric Integration features, including facilitating interoperability and communication between management services (within and between domains), service mesh solutions such as Istio [118] or Linkerd [119] were analyzed, as well as an event streaming platform, like Apache Kafka [120]. On the one hand, the service mesh will allow the management of traffic between services, while enhancing security and observability. The event streaming platform will handle asynchronous communications between applications and services. In addition, the use of the event streaming platform is also important for security use cases, including monitoring, analyzing, and reacting to security threats in real time. In this sense, the use of Istio and Kafka as candidates for integration fabric implementation was considered.

In the context of management and orchestration platforms, the Open Network Automation Platform (ONAP) [121] and Open Source MANO (OSM) [122] are highlighted. ONAP provides a framework for real-time, policy-driven orchestration, management, and automation of network services and edge computing. This platform includes different ecosystems, such as POLICY, CLAMP, and DCAE, which together support closed-loop automation. POLICY provides policy creation and validation capability. CLAMP designs and manages closed-control loops. DCAE (Data Collection, Analytics, and Events) collects and analyzes data. On the other hand, OSM allows modeling and automating the life cycle of network functions, network services, and network slices through MON and POL modules. MON leverages monitoring tools to collect VNFs and infrastructure metrics. POL is a policy management module. Another tool to highlight is Ansible https://www.ansible.com (accessed on 15 August 2023), which allows production-level automation in a cloud-native environment, and as such, allows increasing the automation capabilities of NFVM and NFVO in the management and orchestration of network functions and services.

To provide services in AI and analytics, the Platform for Network Data Analytics (PNDA) http://pnda.io/ (accessed on 15 August 2023) and Acumos AI Platform https://www.acumos.org/ (accessed on 15 August 2023) were analyzed. The first is a scalable big data analytics platform for networks and services that brings together multiple technologies (e.g., Kafka). PNDA was used to enable closed loop control for an ETSI NFV environment. In addition, ONAP is integrating the PNDA as part of the DCAE to provide its analysis services to the ecosystem. On the other hand, Acumus AI Platform allows you to create, share, and deploy AI/ML models, capable of packaging ML models into microservices in portable containers [123]. An "Acumos-DCAE Adapter" is developed to integrate ML models from an Acumos catalog to the ONAP DCAE.

Through these open-source solutions, the architecture represented in Figure 5 was designed. In such a configuration, Kubernetes can act as VIM and CISM. NSMF, NSSMF, NFVO, and NFVM functions can be provided by ONAP or OSM. Regarding the closed loop, MS and AE functions can be implemented using the MON and DCAE modules of OSM and ONAP, respectively, or directly using open-source monitoring tools (e.g., Prometheus [124] and ELK [125]) and analytics platforms (e.g., PNDA and Accumos). Integration fabric will collaborate with management functions deployed as services through the combination of features from Istio and Kafka. In turn, through Envoy sidecar proxies, the management functions are connected to form a service mesh managed by Istio. Synchronous communication between services can be enabled via Istio, whereas asynchronous communication can be performed via Kafka.

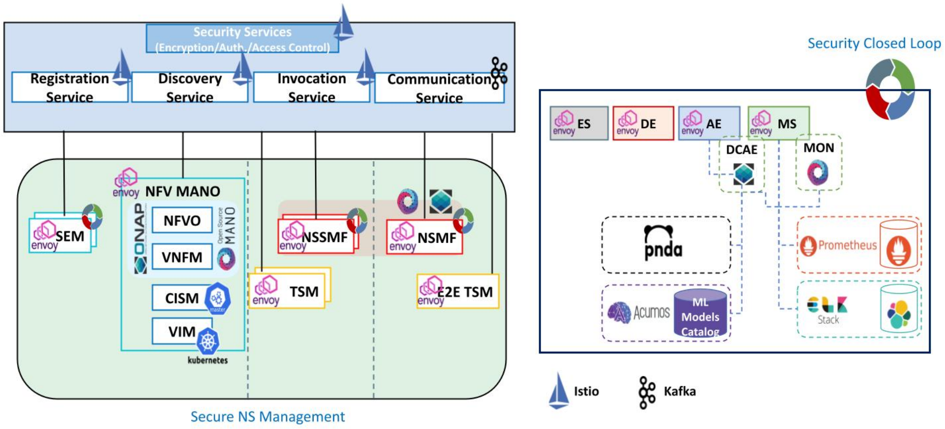


Figure 5. Architecture of the security framework with the enablers and tools, adapted from [19].

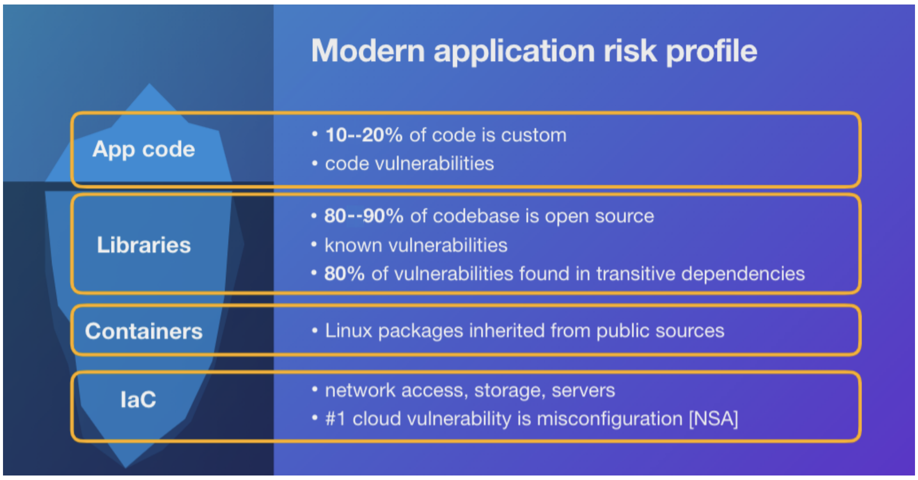
6. Secure Software Development Lifecycle (SDLC)

Cloud-native services rely on cloud infrastructure to store and process user data, including sensitive information such as personal identification, location, and behavior. This represents a potentially large attack surface for cyber attacks, such as data breaches and malware infections, which could compromise the confidentiality, integrity, and availability of services and user data. To address these security challenges, cloud providers and vendors must collaborate to establish a robust and comprehensive security framework that covers the entire service lifecycle, from design to deployment and maintenance. This framework should consider services' unique security requirements and risks and leverage state-of-the-art security technologies and practices, such as encryption, authentication, access control, and risk management.

When speaking about the attack surface, one refers to the collection of all the points where an unauthorized user or an attacker could potentially exploit a vulnerability to gain access to an organization's systems, data, or assets. The attack surface includes all the hardware, software, networks, and interfaces exposed to external or internal threats. The attack surface can be divided into three main categories:

- **Physical attack surface [126]:** This includes all the physical devices, such as servers, laptops, smartphones, routers, and IoT devices, that an attacker could target to gain unauthorized access.
- **Network attack surface [127]:** This includes all the network infrastructure, such as firewalls, routers, switches, and gateways, that an attacker could target to penetrate the organization's network and access its data and systems.
- **Software attack surface [128]:** This includes all the software components, such as applications, operating systems, and databases, that an attacker could target to exploit a vulnerability and gain unauthorized access.

In this section, we shall concentrate mainly on the Secure Software Development Life Cycle (SDLC). According to [129], data breaches from attackers outside the organization typically amount to about 55% of cases, whereas the remaining 45% are because of misconfiguration or other mistakes. This is a significant statistic and should be considered when aiming to improve the security posture of applications. We must first examine the modern application risk profile to establish a proactive approach. Figure 6 shows a typical cloud-native applications stack.



Modern application risk profile

App code
- **10--20%** of code is custom
- code vulnerabilities

Libraries
- **80--90%** of codebase is open source
- known vulnerabilities
- **80%** of vulnerabilities found in transitive dependencies

Containers
- Linux packages inherited from public sources

IaC
- network access, storage, servers
- #1 cloud vulnerability is misconfiguration [NSA]

Figure 6. Modern application stack and risk profile.

Modern cloud-native services leverage technologies such as microservices, containers, and serverless computing. With the increased adoption of cloud-native architecture, the risk profile of the applications stack has also evolved and presents some new and unique security challenges. A cloud-native architecture can be broken down into the following layers:

- **Application code [130]:** custom code may only make up to 10–20% of the codebase. Any code vulnerabilities must be identified.
- **Libraries [36]:** Open source code typically represents up to 80–90% of the code base, whereas open-source software can offer many benefits, such as rapid development and cost-effectiveness, and can also introduce risks. Using open-source code exposes the application to the vulnerabilities that exist in open-source libraries, which attackers can exploit.
- **Containers [131]:** Linux packages inherited from public sources. Cloud-native applications rely on many third-party services, which can also introduce security risks. These third-party services may have vulnerabilities or may not have adequate security controls in place. The use of microservices and container technologies in cloud-native applications has increased the complexity of the codebase. This complexity can lead to security gaps, making identifying vulnerabilities and security risks difficult.
- **Infrastructure-as-Code (IaC) [132]:** with IaC, developers use code to describe the desired configuration of their infrastructure, including servers, networks, storage, and other resources, and deploy the infrastructure automatically using tools like AWS Cloudformation, Terraform, or Puppet.

6.1. Security Risk Profiling and Mitigation

To mitigate such risks, it is essential to implement robust security measures throughout the application development lifecycle. These measures may include using secure coding practices, regularly patching and updating open-source libraries, performing vulnerability assessments [133], penetration testing, implementing access controls, and using security tools such as intrusion detection and prevention systems.

There are some specific things one can do to improve security posture. Firstly, one can add some security gateways between the development stages to help catch earlier vulnerabilities, as depicted in Figure 7. By scanning the code for security issues at different stages of the SDLC and detecting security vulnerabilities early, security issues can be addressed before they become more difficult and expensive to fix. Automated security checks can be set up to reduce the manual effort required for security testing and allow developers to focus on other critical tasks. Finally, security gateways can be integrated with other tools in the SDLC, such as static analysis tools, continuous integration, and continuous deployment tools that can help create a more streamlined and automated security process.

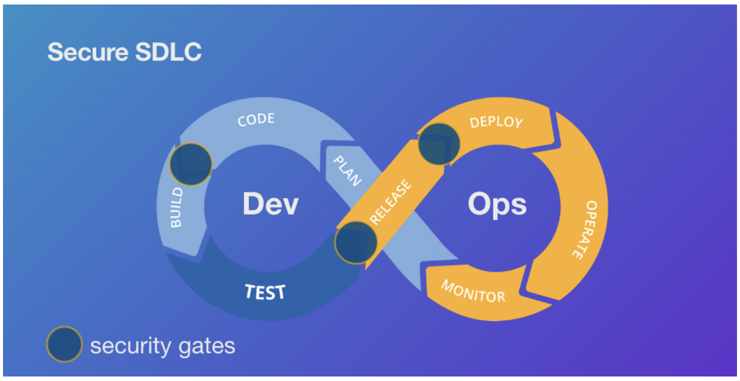
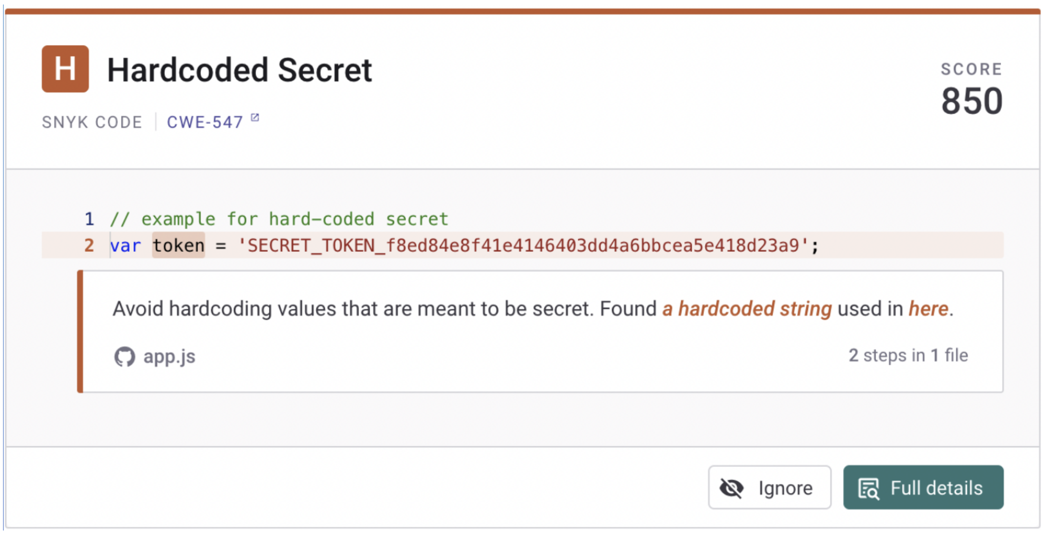


Figure 7. Secure SDLC.

The examination of the modern application risk profile presented in Figure 6, assists towards elaborating more on each layer's specific security measures.

Application Code: Static Application Security Testing (SAST) [134] methodologies can be used to identify security vulnerabilities in the source code of an application (custom code). SAST tools, depicted in Figure 8 can be run using the editor or an Integrated Development Environment (IDE) to scan the application's source code, looking for common vulnerabilities such as SQL injection, cross-site scripting (XSS), buffer overflows, and other security issues.



H Hardcoded Secret

SNYK CODE | CWE-547

SCORE
850

```
// example for hard-coded secret
var token = 'SECRET_TOKEN_f8ed84e8f41e4146403dd4a6bbcea5e418d23a9';
```

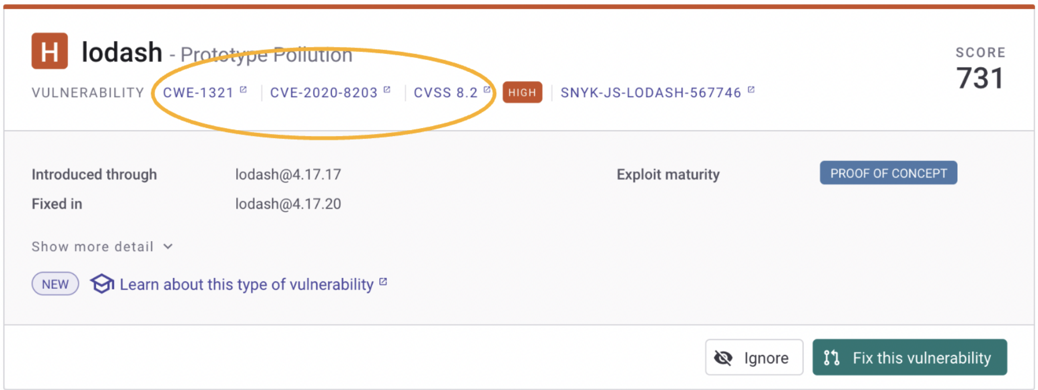
Avoid hardcoding values that are meant to be secret. Found *a hardcoded string* used in *here*.

app.js 2 steps in 1 file

Figure 8. Static Application Security Testing (SAST).

SAST tools use pattern-matching and rule-based techniques to identify security vulnerabilities in the source code. During a SAST scan, the tool generates a report listing all the security vulnerabilities found in the source code. The report includes details on the location of the vulnerability, the type of vulnerability, and recommendations on how to fix the issue. SAST is also useful for enforcing secure coding practices and standards, as the tool can identify code that does not adhere to best practices and provide recommendations for improvement.

Open Source Libraries: Nowadays, applications often rely on numerous dependencies and third-party libraries. Each of them increases the security risk of the application. Software Composition Analysis (SCA) [135] is an approach to identify any vulnerabilities, bugs, or other potential issues that could affect the application's overall performance. SCA, depicted in Figure 9, is ideal for analyzing open-source or third-party integrations and their dependencies to ensure that they do not introduce any security risks.



H lodash - Prototype Pollution

VULNERABILITY | CWE-1321 | CVE-2020-8203 | CVSS 8.2 | HIGH | SNYK-JS-LODASH-567746

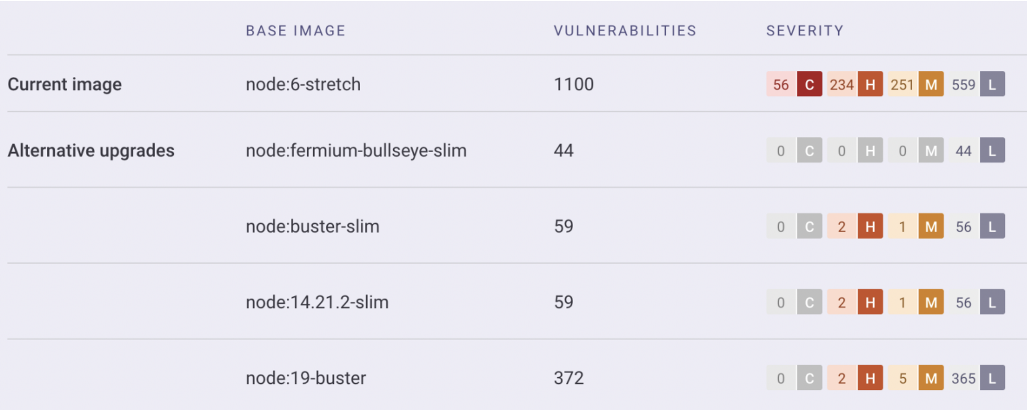
SCORE 731

Introduced through	lodash@4.17.17	Exploit maturity	PROOF OF CONCEPT
Fixed in	lodash@4.17.20		

Figure 9. Software Composition Analysis (SCA).

SCA tools can automate the analysis process and provide developers with detailed reports on the quality and security of their software components. These tools can also provide recommendations for improving the overall quality and security of the software application. SCA tools can significantly simplify the process by referencing the Common Vulnerabilities and Exposures (CVE) database [136], which is essentially a standardized list of publicly known vulnerabilities and security issues. Each CVE entry provides a unique identifier, a brief vulnerability description, and any relevant references and solutions. Moreover, it is essential to ensure any third-party or open-source packages are actively being used and maintained by the community. Lastly, tools such as WhiteSource Bolt or Black Duck are pivotal in searching for known vulnerabilities [137]. Again, their scope is typically narrowed to the search of issues in databases of known vulnerabilities.

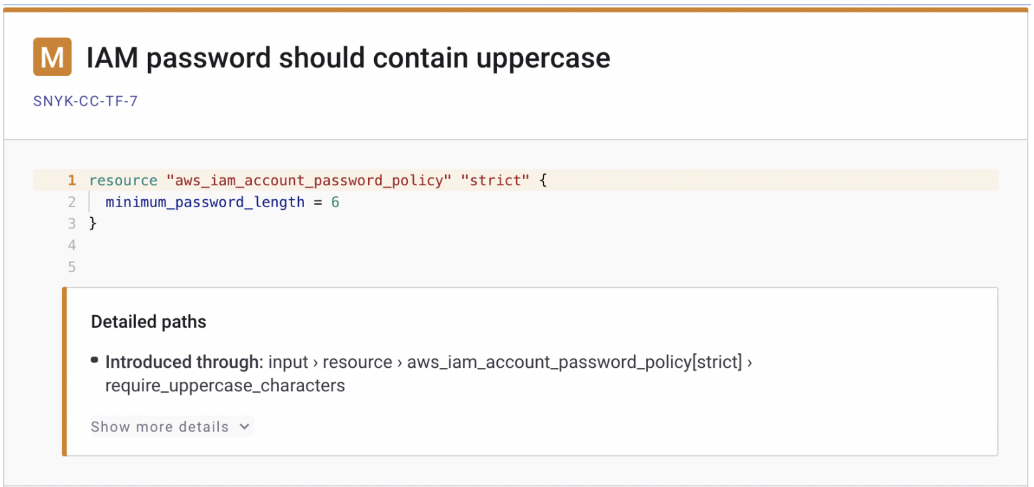
Containers: Containers are a form of virtualization that allows for lightweight and portable deployment of applications, but they also introduce new attack surfaces and security risks. First of all, containers are built using base images, which can contain vulnerabilities that attackers can exploit. In addition, the container runtime environment can be vulnerable to attacks if not configured securely. Furthermore, container orchestration platforms [138], such as Kubernetes, can introduce security risks if not configured securely. Containers may be subject to compliance requirements, such as HIPAA or PCI DSS. An example of container security is depicted in Figure 10.



	BASE IMAGE	VULNERABILITIES	SEVERITY
Current image	node:6-stretch	1100	56 C 234 H 251 M 559 L
Alternative upgrades	node:fermium-bullseye-slim	44	0 C 0 H 0 M 44 L
	node:buster-slim	59	0 C 2 H 1 M 56 L
	node:14.21.2-slim	59	0 C 2 H 1 M 56 L
	node:19-buster	372	0 C 2 H 5 M 365 L

Figure 10. Container security.

Infrastructure-as-Code (IaC): Here, we refer to provisioning and managing infrastructure resources using machine-readable definition files rather than manual processes; although IaC provides many benefits like automation, version control, repeatability, and consistency [139], security should be a key consideration when implementing IaC since any vulnerability or weakness in IaC code can be automatically propagated across all infrastructure instances. An example of IaC is depicted in Figure 11.



M IAM password should contain uppercase

SNYK-CC-TF-7

```
resource "aws_iam_account_password_policy" "strict" {
  minimum_password_length = 6
}


```

Detailed paths

- **Introduced through:** input › resource › aws_iam_account_password_policy[strict] › require_uppercase_characters

Show more details ⌄

Figure 11. Infrastructure-as-Code.

Several principles should be considered, including:

- The principle of least privilege: To ensure that each user has only the minimum access required to perform their tasks.
- Define who is authorized to run the scripts and who is not.
- Limit the permissions of authorized IaC users to what is necessary to perform their tasks.
- IaC scripts should ensure that the permissions granted to the various resources created are limited to what is required.
- Network segmentation: The resources and their related dependencies are all secured within a private subnet.
- Data encryption: A technique used to authenticate and encrypt data sent between two services. Mutual Transport Layer Security (mTLS) ensures that traffic is secure and trusted in both directions between a client and server, providing an additional layer of security for users who log in to a network or applications.

Secure SDLC [140] is critical towards building secure, reliable, resilient software to withstand cyber threats and attacks. By incorporating security into every phase of the software development process, we can minimize the risk of security incidents and, at the same time, comply with regulatory requirements.

It is of major importance to understand how these mechanisms should be implemented in the context of cloud-native environments. This has been an increasingly important research topic for cloud-native applications, especially in the case of applications that are characterized by traits such as the vast amount of involved data (e.g., audio/video), the simultaneous presence of multiple users, and low-latency & high-bandwidth requirements.

6.2. Shift Left and Static Testing Techniques for Enhancing Security

One prominent challenge is the detection of vulnerabilities before attackers can exploit them. These vulnerabilities may be due to incorrect infrastructure configurations, vulnerabilities inherited from container-based images, or vulnerabilities in the application code. With this in mind, Shift Left approach [141] should be considered. Shift Left in the context of Secure SDLC refers to incorporating security measures and testing earlier in the development process. This way, security concerns are identified and addressed proactively as early as possible, rather than waiting until later stages or after the software has been released. By using Shift Left practices, security shall become a fundamental aspect of the development process rather than an afterthought. This shall help to prevent security vulnerabilities and flaws from being introduced into the codebase, reducing the risk of security breaches and other issues down the line. This approach also promotes better communication within the development team and ensures alignment of security goals and requirements, leading to more effective security solutions. Furthermore, the concept of Configuration as Code (CaC) [142], often employed in Shift Left practices, simplifies and automates configuration provisioning and management, facilitating the integration of security checks directly into code or configuration files. Finally, this approach assists towards complying with the requirements of the various regulatory and standardization bodies.

Static Application Security Testing (SAST) [143] techniques are used to perform a static (application at rest) scan of an application component's source to assess the general code quality and detect potential security vulnerabilities. This technique is limited to the application code and does not examine environment or run-time-related vulnerabilities. Issues are detected at the early stages of the software development life-cycle, reducing the overall impact and the cost of mitigation.

On the other hand, Static Container Image Security Testing [144] is used for statically scanning container images of application microservice components to search for known vulnerabilities, typically by parsing through image packages or other dependencies. Automated static application testing combined with container testing techniques allows the detection of a broad range of different vulnerabilities. They can be tightly integrated with the basic DevOps pipeline or put aside as an additional set of tools. Several open security assurance tools and software packages can potentially be part of a DevSecOps cycle for application developers. Their selection is based on aspects like license type, capabilities, open project activity and maturity, documentation, and the possibility of being extended.

In this regard, various tools can be employed for Shift Left and automating the process and vulnerability search. Static Application Security Testing (SAST) tools like Checkmarx [145] allow verifying source code against known bad patterns that risk the security of applications. Likewise, Dynamic Application Security Testing (DAST) [146] tools OWASP Zed Attack Proxy (ZAP) [147] can be used to dynamically assess the running behavior of applications as an attacker would. Moreover, tools like Microsoft Credential Scanner (CredScan) are useful for identifying credential-related vulnerabilities and leaks in source code and configuration files (e.g., default passwords, SQL connection strings, or exposed private keys).

Furthermore, tools like Curiefense https://www.curiefense.io/ (accessed on 15 August 2023) and Falco https://falco.org/ (accessed on 15 August 2023) are also fundamental to detecting and reporting unexpected application runtime behaviors quickly. Falco, a cloud-native threat detection engine, relies on monitoring Linux system calls in containers to flag unexpected behaviors (e.g., privilege escalation events, suspicious reads/writes to known directories). To accomplish that, Falco uses a set of predefined rules. Similarly, Curiefense is a cloud-native tool capable of monitoring HTTP API requests and detecting suspicious behaviors on HTTP traffic between containers. Despite all the benefits of such tools, most existing tools mainly rely on using hard-coded rules, which might limit them to only known vulnerabilities.

Cloud-native security can be split into four key levels, the so-called "4C's": the security of cloud, clusters, containers, and code. Cloud security [148] concerns the security of the underlying cloud environment. It is essential to ensure the principle of least privilege to control, amongst others, the (network) access to the environment APIs, cloud provider APIs, and nodes. The authors of [149] include the protection of the components of the cluster (the applications). Their access should include authentication and admission control mechanisms. For instance, network and pod policies are used as an admission control method for controlling the allowed traffic and pod operations. Container security [150] refers to implementing the best security practices to ensure containers are free from vulnerabilities. This might include the search for vulnerabilities, image signing to enforce the usage of trustable images, and avoiding (detecting) running containers as privileged users. Code security [151] encompasses methods to verify third-party libraries that may be used, all the code and dynamic testing, the limitation of allowed ports, and the usage of TLS or mTLS in all network communications. The Center for Internet Security (CIS) Kubernetes Benchmarks https://www.cisecurity.org/benchmark/kubernetes (accessed on 15 August 2023) provides guidelines for cybersecurity best practice recommendations for configuring Kubernetes. The main goal is to provide a set of recommendations for Kubernetes to improve security against threats.

Moreover, in accordance with the zero-trust model that was explored earlier, each component should only be given the least amount of privileges it needs. For instance, this might help to contain a lateral movement in the case of a compromised element. Nevertheless, this is a significant challenge in cloud-native environments when considering all heterogeneous assets used to support cloud-native applications (e.g., enforcing a common security strategy from the infrastructure and orchestration layers up to the various containers constituting the application).

In the same way, encryption-based techniques for protecting data access are both a sensible choice and a mandatory requirement in several scenarios. For instance, the network communications between containers should be encrypted. This can be natively supported in the application logic or realized, as stated before, with the concept of service meshes. Istio is a cloud-native service mesh implementation that aims to support the orchestration of microservices which, together with Envoy proxies, allows the implementation of various traffic management techniques, including the dynamic enforcing of mutual TLS between service communications. Tools such as Kube-bench [152] and InSpec https://www.chef.io/products/chef-inspec (accessed on 15 August 2023) can be used to perform such security assessments [153]. Kube-bench allows automating the implementation of CIS Benchmarks, offering recommendation actions for detected issues. Similarly, InSpec allows the automation of infrastructure testing, auditing of applications, and policy conformance. InSpec compares the actual state of the systems with the desired state, and whenever it detects deviations, issues a report. Likewise, Kube-hunter, a vulnerability scanning tool, aims to increase the awareness and visibility of security controls in Kubernetes environments.

6.3. Continuous Security, Data Preserving, and Data Compliance

In cloud-native environments, implementing Continuous Security mechanisms by enforcing analyzing and monitoring [154] paradigms is crucial. Even so, security monitoring approaches should not compromise the availability of the applications. Tools such as Falco are pivotal to monitoring many aspects of cloud-native environments. Likewise, a network security solution can be achieved by leveraging the capabilities of Istio and OPA [155].

Cloud environments do not target scenarios that involve a single user, but rather highly complex multi-tenancy environments. Hence, it is also crucial to guarantee that only authorized people shall have access to this environment and consequently access to specific data or services. Having a proper authentication method and fine control of the required privileges is also important. The concept of service meshes combined with a policy engine can be used to provide such identification and access capabilities. Namely, Istio and OPA offer a highly integrated solution that reconciles the benefits of a service mesh but also offers a more centralized and comprehensive approach to enforcing security policies across different security levels.

In every environment, it is important to consider that components can eventually be compromised, so it is necessary to have a backup and recovery plan in case of failure. Furthermore, it is important to test the backup and recovery plan by ensuring that the involved people who are charged with this responsibility know all the steps to follow and that the backup was executed correctly and without any issues. One of the tools that can be used to help the process of doing a backup is Amanda (Advanced Maryland Automatic Network Disk Archiver) [156], an open-source software that allows configuring a unique backup server master to do a backup of various hosts in the network to tape drivers, disks, or optical media.

Lastly, Data Compliance [157] and security share a strong interdependency. Data Compliance measures ensure that organizations handle, store, process, and transmit data securely and with privacy in mind. Hence, it's crucial to adopt the best practices and act according to legislation. Moreover, cloud environments can be composed of geographically distributed environments where there is a need to be compliant with regulatory standards of cloud usage in accordance with various industry guidelines and local/national/international laws. Cloud compliance ensures that cloud computing services meet compliance requirements in the case of processing personal information. To provide security assessments, tools like Kube-bench can be used to run multiple tests. Whenever there is a test that fails, a recommendation to remediate the issue is provided. Alternatively, InSpec also allows testing and auditing applications and infrastructure.

7. Solution Synergy

Establishing cybersecurity in cloud-native services is a multifaceted endeavor where various technologies and methodologies converge synergistically. This section is dedicated to showcasing how the various solutions that were explored in the previous sections of this study work in synergy in order to facilitate the emergence of the features that are required for establishing cybersecurity in cloud-native services. Table 3 depicts the mapping of key features to the corresponding solutions. It is worth mentioning that each solution corresponds to multiple key features. This happens due to the fact that each of these solutions constitutes an amalgamation of various distinct technological paradigms, each of which aims at facilitating different security features. Furthermore, each key feature is associated with multiple solutions, because each solution enables the emergence of a key feature at a different level by utilizing different means and methodologies.

Table 3. Mapping of features to solutions.

Key Features	Solutions
Strong Access Controls	Service Mesh
	Zero-Trust Architecture (ZTA)
	Native AI and Security as a Service (SECaaS)
	Security Risk Profiling and Mitigation
Network Segmentation	Zero-Touch Network and Service Management (ZSM)
	Zero-Trust Architecture (ZTA)
	Service Mesh
Data Encryption	Service Mesh
	Native AI and Security as a Service (SECaaS)
	Security Risk Profiling and Mitigation
Secure Communication	Zero-Touch Network and Service Management (ZSM)
	Zero-Trust Architecture (ZTA)
	Service Mesh
	Shift Left and Static Testing Techniques for Enhancing Security

Table 3. *Cont.*

Key Features	Solutions
Continuous Monitoring and Intrusion Detection	Zero-Trust Architecture (ZTA)
	Service Mesh
	Autonomic and Cognitive Security Management
	Shift Left and Static Testing Techniques for Enhancing Security
	Continuous Security, Data Preserving, and Data Compliance
Disaster Recovery, Incident Response, and Data Backup	Zero-Touch Network and Service Management (ZSM)
	Service Mesh
	Security Risk Profiling and Mitigation
	Shift Left and Static Testing Techniques for Enhancing Security
	Continuous Security, Data Preserving, and Data Compliance
Process Automation	Zero-Touch Network and Service Management (ZSM)
	Service Mesh
	Security Aware Orchestration
	Shift Left and Static Testing Techniques for Enhancing Security
Vulnerability Management	Security Risk Profiling and Mitigation
	Shift Left and Static Testing Techniques for Enhancing Security
	Continuous Security, Data Preserving, and Data Compliance
Configuration Management	Zero-Touch Network and Service Management (ZSM)
	Shift Left and Static Testing Techniques for Enhancing Security
Continuous Security Testing	Security Risk Profiling and Mitigation
	Continuous Security, Data Preserving, and Data Compliance
	Shift Left and Static Testing Techniques for Enhancing Security

7.1. Strong Access Controls

Together, ZTA, service meshes, Native AI and SECaaS, and Security Risk Profiling and Mitigation establish a holistic approach to strong access controls that emphasizes continuous verification, least privilege, and adaptive responses, enhancing the organization's overall security posture. ZTA redefines access controls by adopting a "never trust, always verify" approach, ensuring stringent verification of identity, device health, and context for all entities. Service meshes complement this philosophy by providing granular access control mechanisms for microservice-based applications, enforcing secure communication between authorized services. Native AI and SECaaS continuously monitor behaviors, promptly adjusting access permissions based on real-time threat detection, and fortifying strong access controls against evolving threats. Concurrently, Security Risk Profiling and Mitigation assesses vulnerabilities and risks, guiding access control decisions, and ensuring that access controls align with identified security concerns, thereby reducing the attack surface.

7.2. Network Segmentation

Service meshes ETSI ZSM and ZTA are interconnected elements that contribute to network segmentation and bolster network security and management. Service meshes excel at securing and governing communication between microservices within applications, granting service-level segmentation within the application architecture. ETSI ZSM serves as a comprehensive framework and standard for automating network and service management, enabling the efficient orchestration of network segmentation policies and ensuring consistent application of security measures across network segments. Zero-trust architecture, a security paradigm that assumes no inherent trust, can be applied to both service-to-service communication (inside service meshes) and network-level access control (within network segmentation), thereby establishing a robust security posture where trust is never taken for granted, and access control is meticulous. The convergence of these three concepts facilitates a unified approach to network segmentation, fortifying security and management capabilities by securing communication at both the application and network layers while automating and orchestrating network segmentation.

7.3. Secure Communications

ETSI ZSM, ZTA, service mesh, and Shift Left and Static Testing Techniques all play vital roles in enhancing the security of communications within modern network and application ecosystems. ZSM's automation framework ensures that network resources are optimized and security policies are consistently enforced, actively contributing to the establishment of secure communication channels. Meanwhile, ZTA, with its fundamental principle of distrust by default, strengthens secure communications through rigorous identity verification and continuous authentication, ensuring only authorized entities can access and exchange data. Service mesh further reinforces secure communications by applying authentication, authorization, and encryption mechanisms to inter-microservice communication within applications, guaranteeing data confidentiality during transit. Lastly, Shift Left and Static Testing Techniques, when integrated into the development process, proactively identify and rectify security vulnerabilities, thereby establishing secure communication as an inherent aspect of application design and development, fortifying the overall security posture across diverse IT environments.

7.4. Data Encryption

Service meshes, Native AI, SECaaS, and Security Risk Profiling and Mitigation all have a role in fortifying data encryption strategies to bolster data security. Within the context of data encryption, service meshes serve as a protective layer for data in transit within microservice-based applications, ensuring secure communication through mechanisms like TLS or mTLS. Native AI, when integrated into security solutions, contributes to data encryption by detecting anomalies in encryption key usage or protocol utilization, continuously monitoring and adapting encryption policies to evolving threats. SECaaS providers offer encryption solutions for various data storage and communication channels, simplifying the implementation of robust encryption measures without requiring extensive in-house expertise. Meanwhile, Security Risk Profiling and Mitigation efforts identify encryption as a pivotal control to mitigate specific security vulnerabilities and risks, particularly concerning data protection and compliance. Collectively, these elements converge to create a comprehensive approach to data encryption, enhancing overall data security across diverse IT environments.

7.5. Continuous Monitoring and Intrusion Detection

ZTA, service mesh, Continuous Security practices, Data Preserving and Data Compliance measures, and Shift Left and Static Testing Techniques all play integral roles in the context of Continuous Monitoring and Intrusion Detection. ZTA's fundamental principle of mistrusting all entities aligns seamlessly with continuous monitoring efforts, where all network activity, both internal and external, undergoes vigilant scrutiny for anomalies and potential intrusions. Meanwhile, service mesh offers insights into communication patterns within microservice applications and enforces security policies like encryption, contributing to enhanced intrusion detection capabilities. Continuous Security practices encompass the ongoing assessment of security, including monitoring network traffic, system logs, and user activities to swiftly identify and respond to threats. Data Preserving and Compliance measures help safeguard data integrity and compliance during monitoring, ensuring data remain secure and compliant. Lastly, Shift Left and Static Testing Techniques, integrated into the development process, serve to preemptively uncover and address vulnerabilities, reducing potential intrusion points and bolstering the overall efficacy of Continuous Monitoring and Intrusion Detection. Together, these elements collectively fortify an organization's security posture by providing the necessary tools and strategies for detecting, mitigating, and preventing security breaches in a continuously evolving threat landscape.

7.6. Disaster Recovery, Incident Response, and Data Backup

ETSI ZSM, service mesh, Security Risk Profiling and Mitigation, Shift Left and Static Testing Techniques for Enhancing Security, Continuous Security, Data Preserving, and Data Compliance collectively form an integrated approach to disaster recovery, incident response, and data backup. ETSI ZSM's automation framework ensures resilient network and service configurations, expediting recovery during disasters. Service mesh offers enhanced security within microservices applications, facilitating anomaly detection during incident investigations and ensuring service availability during recovery efforts. Security Risk Profiling identifies vulnerabilities and risks that could lead to incidents, informing proactive mitigation measures. Shift Left and Static Testing Techniques reduce the incidence of vulnerabilities in applications, contributing to incident prevention. Continuous Security monitoring provides early detection of incidents, whereas Data Preserving measures like backup and compliance ensure data integrity and availability for disaster recovery and incident response. These elements collectively fortify an organization's ability to withstand and recover from disruptions, fostering a robust security and business continuity posture.

7.7. Process Automation

ETSI ZSM, service mesh, Shift Left, and Static Testing Techniques for Enhancing Security are interconnected in their roles within process automation, particularly in the context of bolstering security within organizational workflows. ETSI ZSM, functioning as a network and service management automation framework, streamlines processes by efficiently allocating resources and enforcing security policies, minimizing manual intervention, and mitigating potential human errors. Service mesh further enhances automation by automating security aspects of microservice-based communication, ensuring consistent policy application across services, and simplifying complex application environments. Shift Left and Static Testing Techniques, integrated early in the software development lifecycle, contribute to process automation by automating security testing, swiftly identifying and rectifying vulnerabilities before deployment, expediting security assessments, and reducing the necessity for manual code scrutiny. Together, these components drive process automation by automating various network, service, and security tasks, ultimately enhancing efficiency, consistency, and security throughout an organization's operations and development lifecycle.

7.8. Vulnerability Management

Security Risk Profiling and Mitigation, Shift Left, and Static Testing Techniques for Enhancing Security, Continuous Security, Data Preserving, and Data Compliance are intricately interwoven with vulnerability management, creating a cohesive strategy for identifying, prioritizing, and mitigating vulnerabilities. Vulnerability Identification is fortified through early detection in the software development lifecycle (Shift Left) and ongoing vulnerability scanning and real-time monitoring (Continuous Security). Security Risk Profiling aids in prioritization based on the risk posed by identified vulnerabilities, directing vulnerability management efforts toward the most critical threats. Mitigation strategies benefit from Shift Left practices, ensuring security controls and configurations are rigorously tested before deployment, whereas Data Preserving and Data Compliance measures safeguard sensitive information during mitigation. Continuous Monitoring offered by Continuous Security maintains a vigilant stance, identifying new vulnerabilities and promptly incorporating them into vulnerability management processes. Together, these elements establish a comprehensive approach to vulnerability management that efficiently addresses vulnerabilities, prioritizes them based on risk, and maintains data security and compliance throughout the vulnerability lifecycle.

7.9. Configuration Management

Shift Left and Static Testing Techniques for Enhancing Security, in conjunction with ETSI ZSM, play a pivotal role in fortifying Configuration Management's security and reliability aspects. Shift Left practices advocate the integration of security assessments into the early stages of the software development lifecycle, employing static code analysis and other static testing methods to uncover vulnerabilities and security flaws. These techniques are equally applicable to Configuration Management, where they ensure the early detection and remediation of security-related issues in configuration files and settings. Meanwhile, ETSI ZSM automates network and service management, encompassing configuration management for network devices and services. This automation facilitates the consistent application of security configurations, such as access controls and firewall rules, reducing the risk of misconfigurations that could lead to security breaches. Collectively, Shift Left, Static Testing Techniques, and ETSI ZSM synergize to bolster Configuration Management, ensuring that security configurations are validated, accurate, and compliant, thus enhancing the overall security and reliability of IT environments.

7.10. Continuous Security Testing

Security Risk Profiling and Mitigation, Shift Left and Static Testing Techniques for Enhancing Security, Continuous Security, Data Preserving, and Data Compliance are all interconnected components of Continuous Security Testing. Security Risk Profiling and Mitigation contribute by assessing vulnerabilities and prioritizing them based on risk, guiding testing efforts to address the most critical threats. Shift Left practices ensure early integration of security testing, including static code analysis, into the development process, reducing vulnerabilities introduced in code changes. Continuous Security encompasses real-time threat detection and vulnerability scanning during testing, ensuring continuous assessment of security. Data Preserving measures secure data during testing whereas Data Compliance ensures regulatory adherence. Together, they form a holistic approach to maintaining an ongoing, effective, and compliant security testing process, minimizing security risks and vulnerabilities throughout the software development lifecycle.

8. Conclusions

Cybersecurity is not a one-time activity, but rather a continuous process that requires constant learning and improvement. Indeed, the security of cloud-native services is a fast-paced and ever-evolving landscape, requiring constant adaptation to cope with emerging threats and vulnerabilities. In light of that, the main motivation behind this work is to identify and examine desired cybersecurity key aspects and features that are of vital importance in the context of establishing security in cloud-native services. Additionally, in this article, we also discussed existing open-source solutions and enabling technologies to implement these features effectively. Furthermore, we performed a systematic review and exploratory analysis that maps the pivotal aspects of security in cloud-native services to the corresponding features, and these features to their contemporary solutions. This exploratory analysis included various fundamental solutions for establishing cybersecurity in cloud-native services such as ZSM, ZTA, service meshes, Native AI and SECaaS, Security Risk Profiling and Mitigation, Shift Left and Static Testing, as well as Continuous Security, Data Preserving, and Data Compliance techniques for enhancing security. Finally, we showcased how these solutions can operate in synergy in order to establish the various features.

DevOps and SecDevOps emphasize collaboration between development, operations, and security teams by breaking down silos and encouraging communication and teamwork. Automating key processes such as testing, deployment, and configuration management, DevOps can help accelerate software delivery. Using CI/CD pipelines, it is possible to catch and fix bugs and other issues early in the development process. Integrating a Shift Left approach can help to identify security vulnerabilities early and in some cases prevent security issues from arising in the first place. Recent approaches and concepts such as ETSI ZSM, ZTA, and service meshes are also rooted in the idea of built-in mechanisms capable of supporting a more comprehensive, automated and fine-grain security posture through the observability, segmentation, and analysis of various security dimensions. Finally, emerging advances in AI are set to disrupt the way threats and vulnerabilities are detected and prevented. AI-based anomaly detection systems can play a crucial role in the analysis of large amounts of data. AI-powered predictive systems can help to predict security issues based on historical data and trends.

The insights and findings of this work can be used by cybersecurity professionals, such as developers and researchers, to enhance the security of cloud-native services. It is worth pointing out that this work examines security in cloud-native services from a rather broad perspective. However, each type of cloud-native service is intertwined with a plethora of functional and QoS requirements. Such requirements are bound to have a significant impact on the security measures that are being deployed. Thus, any attempt at applying the insights and findings that emerged from this work shall be accompanied by a detailed analysis of such requirements and intricacies that are associated with the respective type of cloud-native service. Such scientific investigations are capable of giving birth to further research that aims at applying the findings of this work in a more service-type-specific manner.

Author Contributions: T.T. (Theodoros Theodoropoulos), L.R., C.B., P.G., E.M., and A.M. worked on the conceptualization and methodology of the work, and L.C., F.D., P.S., M.D.G., P.B., T.T. (Tarik Taleb), and K.T. developed the final solution and the original draft preparation. All authors have read and agreed to the published version of the manuscript.

Funding: This project received funding from the EU's Horizon 2020 program under Grant Agreement No 101016509 (CHARITY). This paper reflects only the authors' views and the Commission is not responsible for any use that may be made of the information it contains.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gannon, D.; Barga, R.; Sundaresan, N. Cloud-native applications. *IEEE Cloud Comput.* **2017**, *4*, 16–21.
2. Huang, S.Y.; Chen, C.Y.; Chen, J.Y.; Chao, H.C. A Survey on Resource Management for Cloud Native Mobile Computing: Opportunities and Challenges. *Symmetry* **2023**, *15*, 538.
3. Azad, N.; Hyrynsalmi, S. DevOps critical succes factors—A systematic literature review. *Inf. Softw. Technol.* **2023**, *157*, 107150.
4. Thatikonda, V.K. Beyond the Buzz: A Journey Through CI/CD Principles and Best Practices. *Eur. J. Theor. Appl. Sci.* **2023**, *1*, 334–340.
5. Kumar, M., Mishra, S., Lathar, N. & Singh, P. Infrastructure as Code (IaC): Insights on Various Platforms. *Advances In Intelligent Systems And Computing*. pp. 439-449 (2023)
6. Alshuqayran, N.; Ali, N.; Evans, R. A systematic mapping study in microservice architecture. In Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, 4–6 November 2016; pp. 44–51.

7. Ramu, V. Performance Impact of Microservices Architecture. *The Review Of Contemporary Scientific And Academic Studies*. **3** (2023,6), https://doi.org/10.55454/rcsas.3.06.2023.010
8. Kosińska, J.; Zieliński, K. Enhancement of Cloud-native applications with Autonomic Features. *J. Grid Comput.* **2023**, *21*, 44.
9. Poulton, N. *The Kubernetes Book*; Nigel Poulton Ltd., Cheshire, UK, 2023.
10. Senjab, K.; Abbas, S.; Ahmed, N.; ur Rehman Khan, A. A survey of Kubernetes scheduling algorithms. *J. Cloud Comput.* **2023**, *12*, 1–26.
11. Taleb, T.; Boudi, A.; Rosa, L.; Cordeiro, L.; Theodoropoulos, T.; Tserpes, K.; Dazzi, P.; Protopsaltis, A.I.; Li, R. Toward Supporting XR Services: Architecture and Enablers. *IEEE Internet Things J.* **2022**, *10*, 3567–3586.
12. Theodoropoulos, T.; Makris, A.; Violos, J.; Tserpes, K. An Automated Pipeline for Advanced Fault Tolerance in Edge Computing Infrastructures. In Proceedings of the 2nd Workshop on Flexible Resource and Application Management on the Edge, Minneapolis, USA, 1 July 2022, pp. 19–24.
13. Makris, A.; Psomakelis, E.; Theodoropoulos, T.; Tserpes, K. Towards a Distributed Storage Framework for Edge Computing Infrastructures. In Proceedings of the 2nd Workshop on Flexible Resource and Application Management on the Edge, Minneapolis, USA, 1 July 2022, pp. 9–14.
14. Logeshwaran, J.; Ramesh, G.; Aravindarajan, V. A secured database monitoring method to improve data backup and recovery operations in cloud computing. *BOHR Int. J. Comput. Sci.* **2023**, *2*, 1–7.
15. Theodoropoulos, T.; Makris, A.; Psomakelis, E.; Carlini, E.; Mordacchini, M.; Dazzi, P.; Tserpes, K. GNOSIS: Proactive Image Placement Using Graph Neural Networks & Deep Reinforcement Learning. In Proceedings of the 2023 IEEE 16th International Conference on Cloud Computing (CLOUD), Chicago, IL, USA, 4 July 2023; pp. 120–128.
16. Benzaid, C.; Boukhalfa, M.; Taleb, T. Robust Self-Protection Against Application-Layer (D)DoS Attacks in SDN Environment. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea, 25–28 May 2020; pp. 1–6.
17. Javadpour, A.; Ja'fari, F.; Taleb, T.; Benzaid, C. Reinforcement Learning-based Slice Isolation Against DoS/DDoS Attacks in Beyond 5G Networks. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 3930–3946. https://doi.org/10.1109/TNSM.2023.3254581.
18. Theodoropoulos, T.; Makris, A.; Boudi, A.; Taleb, T.; Herzog, U.; Rosa, L.; Cordeiro, L.; Tserpes, K.; Spatafora, E.; Romussi, A.; et al. Cloud-based xr services: A survey on relevant challenges and enabling technologies. *J. Netw. Netw. Appl.* **2022**, *2*, 1–22.
19. Benzaid, C.; Taleb, T.; Song, J. AI-based Autonomic & Scalable Security Management Architecture for Secure Network Slicing in B5G. *IEEE Netw.* **2022**, *36*, 165 – 174. https://doi.org/10.1109/MNET.104.2100495.
20. Benzaid, C.; Taleb, T.; Phan, C.T.; Tselios, C.; Tsolis, G. Distributed AI-based Security for Massive Numbers of Network Slices in 5G & Beyond Mobile Systems. In Proceedings of the 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), Porto, Portugal, 8–11 June 2021; pp. 401–406. https://doi.org/10.1109/EuCNC/6GSummit51104.2021.9482418.
21. Alghofaili, Y.; Albattah, A.; Alrajeh, N.; Rassam, M.A.; Al-Rimy, B.A.S. Secure cloud infrastructure: A survey on issues, current solutions, and open challenges. *Appl. Sci.* **2021**, *11*, 9005.
22. Ali, M.; Khan, S.U.; Vasilakos, A.V. Security in cloud computing: Opportunities and challenges. *Inf. Sci.* **2015**, *305*, 357–383.
23. Tabrizchi, H.; Kuchaki Rafsanjani, M. A survey on security challenges in cloud computing: Issues, threats, and solutions. *J. Supercomput.* **2020**, *76*, 9493–9532.
24. Kumar, S.N.; Vajpayee, A. A survey on secure cloud: Security and privacy in cloud computing. *Am. J. Syst. Softw.* **2016**, *4*, 14–26.
25. Younis, Y.A.; Kifayat, K. Secure cloud computing for critical infrastructure: A survey. *Liverp. John Moores Univ. United Kingd. Tech. Rep.* **2013**, 1, 599–610.
26. Shahzad, F. State-of-the-art survey on cloud computing security challenges, approaches and solutions. *Procedia Comput. Sci.* **2014**, *37*, 357–362.
27. Ramachandra, G.; Iftikhar, M.; Khan, F.A. A comprehensive survey on security in cloud computing. *Procedia Comput. Sci.* **2017**, *110*, 465–472.
28. Khan, M.A. A survey of security issues for cloud computing. *J. Netw. Comput. Appl.* **2016**, *71*, 11–29.
29. Sharma, S.; Gupta, G.; Laxmi, P. A survey on cloud security issues and techniques. *arXiv* **2014**, arXiv:1403.5627.
30. Khalil, I.M.; Khreishah, A.; Azeem, M. Cloud computing security: A survey. *Computers* **2014**, *3*, 1–35.
31. Singh, S.; Jeong, Y.S.; Park, J.H. A survey on cloud computing security: Issues, threats, and solutions. *J. Netw. Comput. Appl.* **2016**, *75*, 200–222.
32. Hussein, N.H.; Khalid, A. A survey of cloud computing security challenges and solutions. *Int. J. Comput. Sci. Inf. Secur.* **2016**, *14*, 52.
33. Basu, S.; Bardhan, A.; Gupta, K.; Saha, P.; Pal, M.; Bose, M.; Basu, K.; Chaudhury, S.; Sarkar, P. Cloud computing security challenges & solutions—A survey. In Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 8–10 January 2018; pp. 347–356.
34. Parast, F.K.; Sindhav, C.; Nikam, S.; Yekta, H.I.; Kent, K.B.; Hakak, S. Cloud computing security: A survey of service-based models. *Comput. Secur.* **2022**, *114*, 102580.
35. Butt, U.A.; Amin, R.; Mehmood, M.; Aldabbas, H.; Alharbi, M.T.; Albaqami, N. Cloud security threats and solutions: A survey. *Wirel. Pers. Commun.* **2023**, *128*, 387–413.

36. Alonso, J.; Orue-Echevarria, L.; Casola, V.; Torre, A.I.; Huarte, M.; Osaba, E.; Lobo, J.L. Understanding the challenges and novel architectural models of multi-cloud native applications–a systematic literature review. *J. Cloud Comput.* **2023**, *12*, 1–34.
37. Wong, A.Y.; Chekole, E.G.; Ochoa, M.; Zhou, J. On the Security of Containers: Threat Modeling, Attack Analysis, and Mitigation Strategies. *Comput. Secur.* **2023**, *128*, 103140.
38. Karakaş, B. & Others Enhancing Security in Communication Applications Deployed on Kubernetes: Best Practices and Service Mesh Analysis. (2023) Available online: https://aaltodoc.aalto.fi/handle/123456789/122929 accessed on 15 August 2023).
39. Indu, I.; Anand, P.R.; Bhaskar, V. Identity and access management in cloud environment: Mechanisms and challenges. *Eng. Sci. Technol. Int. J.* **2018**, *21*, 574–588.
40. Yang, P.; Xiong, N.; Ren, J. Data security and privacy protection for cloud storage: A survey. *IEEE Access* **2020**, *8*, 131723–131740.
41. Elsayed, M.; Zulkernine, M. Towards security monitoring for cloud analytic applications. In Proceedings of the 2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing,(HPSC) and IEEE International Conference on Intelligent Data and Security (IDS), Omaha, NE, USA, 3–5 May 2018; pp. 69–78.
42. Ozer, M.; Varlioglu, S.; Gonen, B.; Adewopo, V.; Elsayed, N.; Zengin, S. Cloud incident response: Challenges and opportunities. In Proceedings of the 2020 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 16–18 December 2020; pp. 49–54.
43. Sultan, S.; Ahmad, I.; Dimitriou, T. Container security: Issues, challenges, and the road ahead. *IEEE Access* **2019**, *7*, 52976–52996.
44. Mateus-Coelho, N.; Cruz-Cunha, M.; Ferreira, L.G. Security in microservices architectures. *Procedia Comput. Sci.* **2021**, *181*, 1225–1236.
45. Islam, C.; Babar, M.A.; Nepal, S. Architecture-centric support for integrating security tools in a security orchestration platform. In Proceedings of the Software Architecture: 14th European Conference, ECSA 2020, L'Aquila, Italy, 14–18 September 2020; Proceedings 14; Springer: Berlin/Heidelberg, Germany, 2020; pp. 165–181.
46. Zaydi, M.; Nassereddine, B. DevSecOps practices for an agile and secure it service management. *J. Manag. Inf. Decis. Sci.* **2020**, *23*, 134–149.
47. Rahaman, M.S.; Islam, A.; Cerny, T.; Hutton, S. Static-Analysis-Based Solutions to Security Challenges in Cloud-Native Systems: Systematic Mapping Study. *Sensors* **2023**, *23*, 1755.
48. Cloud for Holography and Cross Reality (CHARITY). D2.1: Edge and Cloud Infrastructure Resource and Computational Continuum Orchestration System Report 2022. Available online: https://www.charity-project.eu/deliverables (accessed on 15 August 2023).
49. Makris, A.; Boudi, A.; Coppola, M.; Cordeiro, L.; Corsini, M.; Dazzi, P.; Andilla, F.D.; Rozas, Y.G.; Kamarianakis, M.; Pateraki, M.; et al. Cloud for holography and augmented reality. In Proceedings of the 2021 IEEE 10th International Conference on Cloud Networking (CloudNet), Cookeville, TN, USA, 8–10 November 2021; pp. 118–126.
50. Shah, Y.; Sengupta, S. A survey on Classification of Cyber-attacks on IoT and IIoT devices. In Proceedings of the 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 28–31 October 2020; pp. 0406–0413. https://doi.org/10.1109/UEMCON51285.2020.9298138.
51. Alenezi, M.N.; Alabdulrazzaq, H.; Alshaher, A.A.; Alkharang, M.M. Evolution of malware threats and techniques: A review. *Int. J. Commun. Netw. Inf. Secur.* **2020**, *12*, 326–337.
52. Conti, M.; Dragoni, N.; Lesyk, V. A survey of man in the middle attacks. *IEEE Commun. Surv. Tutorials* **2016**, *18*, 2027–2051.
53. Kumari, P.; Jain, A.K. A comprehensive study of DDoS attacks over IoT network and their countermeasures. *Comput. Secur.* **2023**, *127*, 103096.
54. Panchal, A.C.; Khadse, V.M.; Mahalle, P.N. Security issues in IIoT: A comprehensive survey of attacks on IIoT and its countermeasures. In Proceedings of the 2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN), Lonavala, India, 23–24 November 2018; pp. 124–130.
55. Bremler-Barr, A.; Brosh, E.; Sides, M. DDoS attack on cloud auto-scaling mechanisms. In Proceedings of the IEEE INFOCOM 2017-IEEE Conference on Computer Communications, Atlanta, GA, USA, 1–4 May 2017; pp. 1–9.
56. Bremler-Barr, A.; Czeizler, M. Tandem Attack: DDoS Attack on Microservices Auto-scaling Mechanisms. In Proceedings of the IEEE INFOCOM 2023-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hoboken, NJ, USA, 20 May 2023; pp. 1–2.
57. Kashi, M.M.; Yazidi, A.; Haugerud, H. Mitigating Yo-Yo attacks on cloud auto-scaling. In Proceedings of the 2022 14th IFIP Wireless and Mobile Networking Conference (WMNC), Sousse, Tunisia, 17–19 October 2022; pp. 46–53. https://doi.org/10.23919/WMNC56391.2022.9954300.
58. ur Rasool, R.; Wang, H.; Ashraf, U.; Ahmed, K.; Anwar, Z.; Rafique, W. A survey of link flooding attacks in software defined network ecosystems. *J. Netw. Comput. Appl.* **2020**, *172*, 102803.
59. Meier, R.; Tsankov, P.; Lenders, V.; Vanbever, L.; Vechev, M.T. NetHide: Secure and Practical Network Topology Obfuscation. In Proceedings of the USENIX Security Symposium, California, USA, 18 August 2018.
60. Kim, J.; Marin, E.; Conti, M.; Shin, S. EqualNet: A secure and practical defense for long-term network topology obfuscation. *Network and Distributed Systems Security (NDSS) Symposium 2022)* **2022**. https://dx.doi.org/10.14722/ndss.2022.23154.
61. Trassare, S.T.; Beverly, R.; Alderson, D. A technique for network topology deception. In Proceedings of the MILCOM 2013-2013 IEEE Military Communications Conference, San Diego, CA, USA, 18–20 November 2013.

62. Wang, Q.; Xiao, F.; Zhou, M.; Wang, Z.; Li, Q.; Li, Z. Linkbait: Active Link Obfuscation to Thwart. *arXiv* **2017**, arXiv:1703.09521.
63. Martin, A.; Raponi, S.; Combe, T.; Di Pietro, R. Docker ecosystem–vulnerability analysis. *Comput. Commun.* **2018**, *122*, 30–43.
64. Gruenbacher, A.; Arnold, S. AppArmor Technical Documentation, 2007. Available online: https://lkml.iu.edu/hypermail/linux/kernel/0706.1/0805/techdoc.pdf accessed on 15 August 2023).
65. McCarty, B. SELinux, 2005. Available online: https://www.oreilly.com/library/view/selinux/0596007167/ accessed on 15 August 2023).
66. Yee, G.O. Modeling and reducing the attack surface in software systems. In Proceedings of the 2019 IEEE/ACM 11th International Workshop on Modelling in Software Engineering (MiSE), Montreal, QC, Canada, 26–27 May 2019; pp. 55–62.
67. Qiu, J.; Tian, Z.; Du, C.; Zuo, Q.; Su, S.; Fang, B. A survey on access control in the age of internet of things. *IEEE Internet Things J.* **2020**, *7*, 4682–4696.
68. Mhaskar, N.; Alabbad, M.; Khedri, R. A formal approach to network segmentation. *Comput. Secur.* **2021**, *103*, 102162.
69. Gupta, S.; Sacchetti, T.; Crispo, B. End-to-End Encryption for Securing Communications in Industry 4.0. In Proceedings of the 2022 4th IEEE Middle East and North Africa COMMunications Conference (MENACOMM), Amman, Jordan, 6–8 December 2022; pp. 153–158.
70. Zdun, U.; Queval, P.J.; Simhandl, G.; Scandariato, R.; Chakravarty, S.; Jelic, M.; Jovanovic, A. Microservice security metrics for secure communication, identity management, and observability. *ACM Trans. Softw. Eng. Methodol.* **2023**, *32*, 1–34.
71. Kott, A.; Arnold, C. The promises and challenges of continuous monitoring and risk scoring. *IEEE Secur. Priv.* **2013**, *11*, 90–93.
72. Ayyagari, M.R.; Kesswani, N.; Kumar, M.; Kumar, K. Intrusion detection techniques in network environment: A systematic review. *Wirel. Netw.* **2021**, *27*, 1269–1285.
73. Tamimi, A.A.; Dawood, R.; Sadaqa, L. Disaster recovery techniques in cloud computing. In Proceedings of the 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, Jordan, 9–11 April 2019, pp. 845–850.
74. Loukasmäki, H. Cyber Incident Response in Public Cloud: Implications of modern cloud computing characteristics for cyber incident response, 2023. Available online: https://www.theseus.fi/handle/10024/803156 accessed on 15 August 2023).
75. Suguna, S.; Suhasini, A. Overview of data backup and disaster recovery in cloud. In Proceedings of the International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, India, 27–28 February 2014; pp. 1–7.
76. Pandey, N., Kumar, K., Saini, G. & Mishra, A. Security issues and challenges in cloud of things-based applications for industrial automation. *Annals Of Operations Research.* (2023,3), https://doi.org/10.1007/s10479-023-05285-7
77. Sheganaku, G.; Schulte, S.; Waibel, P.; Weber, I. Cost-efficient auto-scaling of container-based elastic processes. *Future Gener. Comput. Syst.* **2023**, *138*, 296–312.
78. Fatima, A.; Khan, T.A.; Abdellatif, T.M.; Zulfiqar, S.; Asif, M.; Safi, W.; Al Hamadi, H.; Al-Kassem, A.H. Impact and Research Challenges of Penetrating Testing and Vulnerability Assessment on Network Threat. In Proceedings of the 2023 International Conference on Business Analytics for Technology and Security (ICBATS), Dubai, United Arab Emirates, 7–8 March 2023; pp. 1–8.
79. Zheng, J.; Okamura, H.; Dohi, T. Pull-Type Security Patch Management in Intrusion Tolerant Systems: Modeling and Analysis. In *Maintenance Management-Current Challenges, New Developments, and Future Directions*; IntechOpen: London, United Kingdom, 2023.
80. Schroeter, J.; Mucha, P.; Muth, M.; Jugel, K.; Lochau, M. Dynamic configuration management of cloud-based applications. In Proceedings of the 16th International Software Product Line Conference-Volume 2, New York, USA, 2 –7September 2012, pp. 171–178.
81. Kumar, R.; Goyal, R. When security meets velocity: Modeling continuous security for cloud applications using DevSecOps. In *Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 415–432.
82. Engström, V.; Johnson, P.; Lagerström, R.; Ringdahl, E.; Wällstedt, M. Automated Security Assessments of Amazon Web Services Environments. *ACM Trans. Priv. Secur.* **2023**, *26*, 1–31.
83. ETSI. Zero-touch network and service management (ZSM); General Security Aspects. *Ref. Archit. Eur. Telecommun. Stand. Inst. (ETSI)* **2021**, *1*, 1. Available online: https://www.etsi.org/deliver/etsi_gr/ZSM/001_099/010/01.01.01_60/gr_ZSM010v010101p.pdf accessed on 15 August 2023).
84. Benzaid, C.; Taleb, T. AI-driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions. *IEEE Netw.* **2020**, *34*, 186–194. https://doi.org/10.1109/MNET.001.1900252.
85. Theodoropoulos, T.; Violos, J.; Tsanakas, S.; Leivadeas, A.; Tserpes, K.; Varvarigou, T. Intelligent Proactive Fault Tolerance at the Edge through Resource Usage Prediction. *arXiv* **2023**, arXiv:2302.05336 .
86. Theodoropoulos, T.; Makris, A.; Kontopoulos, I.; Violos, J.; Tarkowski, P.; Ledwoń, Z.; Dazzi, P.; Tserpes, K. Graph neural networks for representing multivariate resource usage: A multiplayer mobile gaming case-study. *Int. J. Inf. Manag. Data Insights* **2023**, *3*, 100158.
87. DeCusatis, C.; Liengtiraphan, P.; Sager, A.; Pinelli, M. Implementing zero trust cloud networks with transport access control and first packet authentication. In Proceedings of the 2016 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 18–20 November 2016; pp. 5–10.
88. Sanders, M.; Yue, C. Automated least privileges in cloud-based web services. In Proceedings of the Proceedings of the fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies, San Jose California, USA, 14 October 2017, pp. 1 – 6.

89. Mehraj, S.; Banday, M.T. Establishing a zero trust strategy in cloud computing environment. In Proceedings of the 2020 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 22–24 January 2020; pp. 1–6.
90. Kerman, A.; Borchert, O.; Rose, S.; Tan, A. Nist Special Publication 1800-35E-Implementing a zero trust architecture. *Natl. Inst. Stand. Technol. (NIST)* **2020**, *1*, 1. Available online: https://www.nccoe.nist.gov/sites/default/files/2022-12/zta-nist-sp-1800-35e-preliminary-draft.pdf (accessed on 15 August 2023).
91. Hussain, F.; Li, W.; Noye, B.; Sharieh, S.; Ferworn, A. Intelligent service mesh framework for api security and management. In Proceedings of the 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 17–19 October 2019; pp. 0735–0742.
92. Li, W.; Lemieux, Y.; Gao, J.; Zhao, Z.; Han, Y. Service mesh: Challenges, State of the Art, and Future Research Opportunities. In Proceedings of the 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), San Francisco, CA, USA, 4–9 April 2019; pp. 122–1225.
93. Kim, E.; Han, J.; Kim, J. Visualizing Cloud-Native AI+ X Applications employing Service Mesh. In Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 21–23 October 2020.
94. Benzaid, C.; Alemany, P.; Artych, R.; Asensio, R.; Chollon, G.; Kalalas, C.; de Oca, E.M.; Palma, N.P.; Zarca, A.M.; Pascual, H.R.; et al. INSPIRE-5Gplus's White Paper on Intelligent Security Architecture for 5G and Beyond Networks, Version 2.0, 2022 Available online: https://5g-ppp.eu/wp-content/uploads/2022/11/INSPIRE-5Gplus_White_Paper_HLA_2.0.pdf (accessed on 15 August 2023).
95. Morgan, W. Service mesh: A critical component of the cloud native stack, 2017. Available online: https://www.cncf.io/blog/2017/04/26/service-mesh-critical-component-cloud-native-stack/#:~:text=tl%3Bdr%3A%20A%20service%20mesh,of%20the%20cloud%20native%20stack. (accessed on 15 August 2023).
96. Theodoropoulos, T.; Kafetzis, D.; Violos, J.; Makris, A.; Tserpes, K. Multi-Agent Deep Reinforcement Learning for Weighted Multi-Path Routing. In Proceedings of the 3rd Workshop on Flexible Resource and Application Management on the Edge, Orlando, Florida, USA, 20 June 2023; pp. 7–11.
97. Chandramouli, R.; Butcher, Z. NIST Special Publication 800-204A-Building secure microservices-based applications using service-mesh architecture. *NIST Spec. Publ.* **2020**, *1*, 1. Available online: https://csrc.nist.rip/external/nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-204A.pdf (accessed on 15 August 2023).
98. Harlicaj, E. Anomaly Detection of Web-Based Attacks in Microservices. Master's Thesis, Aalto University, Espoo, Finland, 2021. Available online: https://aaltodoc.aalto.fi/bitstream/handle/123456789/109316/master_Harlicaj_Eljon_2021.pdf(accessed on 15 August 2023).
99. Baye, G.; Hussain, F.; Oracevic, A.; Hussain, R.; Kazmi, S.A. API security in large enterprises: Leveraging machine learning for anomaly detection. In Proceedings of the 2021 International Symposium on Networks, Computers and Communications (ISNCC), Dubai, United Arab Emirates, 31 October–2 November 2021; pp. 1–6.
100. Benmerar, T.Z.; Theodoropoulos, T.; Fevereiro, D.; Rosa, L.; Rodrigues, J.; Taleb, T.; Barone, P.; Tserpes, K.; Cordeiro, L. Intelligent Multi-Domain Edge Orchestration for Highly Distributed Immersive Services: An Immersive Virtual Touring Use Case. In Proceedings of the 2023 IEEE International Conference on Edge Computing and Communications (EDGE), Chicago, IL, USA, 2–8 July 2023; pp. 381–392.
101. Miller, L.; Mérindol, P.; Gallais, A.; Pelsser, C. Towards secure and leak-free workflows using microservice isolation. In Proceedings of the 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), Paris, France, 7–10 June 2021; pp. 1–5.
102. Wang, W.; Yongchareon, S. Security-as-a-service: A literature review. *Int. J. Web Inf. Syst.* **2020**, *16*, 493–517.
103. Massimo Iovene, Leif Jonsson Defining AI native: A key enabler for advanced intelligent telecom networks, 2023. Available online: https://www.ericsson.com/en/reports-and-papers/white-papers/ai-native(accessed on 15 August 2023).
104. Lovén, L., Leppänen, T., Peltonen, E., Partala, J., Harjula, E., Porambage, P., Ylianttila, M. & Riekki, J. EdgeAI: A Vision for Distributed, Edge-native Artificial Intelligence in Future 6G Networks. (2019), Available online: https://api.semanticscholar.org/CorpusID:232030033(accessed on 15 August 2023).
105. Bao, S.; Sun, W.; Xu, H. A Native Intelligent and Security 6G Network Architecture. In Proceedings of the 2022 IEEE/CIC International Conference on Communications in China (ICCC Workshops), Foshan, China, 11–13 August 2022; pp. 395–400.
106. Sarker, I.H.; Furhad, M.H.; Nowrozy, R. Ai-driven cybersecurity: An overview, security intelligence modeling and research directions. *SN Comput. Sci.* **2021**, *2*, 1–18.
107. Theodoropoulos, T.; Maroudis, A.C.; Violos, J.; Tserpes, K. An encoder-decoder deep learning approach for multistep service traffic prediction. In Proceedings of the 2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService), Oxford, UK, 23–26 August 2021; pp. 33–40.
108. Musa, K.M. Evaluating Security-as-a-Service (SECaaS) Measures to Increase the Qual-ity of Cloud Computing. *Int. J. Sci. Eng. Appl. (IJSEA)* **2017**, *6*, 350–359.
109. Fatima, B.M.D.I.M. Security-as-a-service in Cloud Computing (SecAAS). *Int. J. Comput. Sci. Inf. Secur. (IJCSIS)* **2017**, *15*, *2*.
110. Torkura, K.A.; Sukmana, M.I.; Cheng, F.; Meinel, C. Leveraging cloud native design patterns for security-as-a-service applications. In Proceedings of the 2017 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, USA, 3–5 November 2017.

111. Benzaid, C.; Alemany, P.; Ayed, D.; Chollon, G.; Christopoulou, M.; Gür, G.; Lefebvre, V.; de Oca, E.; Munoz, R.; Ortiz, J.; et al. White paper: Intelligent security architecture for 5g and beyond networks. *INSPIRE-5Gplus*, **2020**. Available online: https://zenodo.org/records/4288658(accessed on 15 August 2023).
112. Johnson, J. Automating the OODA loop in the age of intelligent machines: reaffirming the role of humans in command-and-control decision-making in the digital age. *Def. Stud.* **2023**, *23*, 43–67.
113. Moradi, N.; Shameli-Sendi, A.; Khajouei, A. A scalable stateful approach for virtual security functions orchestration. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 1383–1394.
114. Salva-Garcia, P.; Chirevella-Perez, E.; Bernabe, J.B.; Alcaraz-Calero, J.M.; Wang, Q. Towards automatic deployment of virtual firewalls to support secure mMTC in 5G networks. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019; pp. 385–390.
115. Nugraha, Y.; Martin, A. Cybersecurity service level agreements: Understanding government data confidentiality requirements. *J. Cybersecur.* **2022**, *8*, tyac004.
116. Laszewski, T.; Arora, K.; Farr, E.; Zonooz, P. *Cloud Native Architectures: Design High-Availability and Cost-Effective Applications for the Cloud*; Packt Publishing Ltd.: Birmingham, United Kingdom, 2018.
117. Schneider, M.; Abeck, S. Engineering Microservice-Based Applications Using an Integration Platform as a Service. In Proceedings of the 2023 IEEE International Conference on Service-Oriented System Engineering (SOSE), Athens, Greece, 17–20 July 2023; IEEE Computer Society: Los Alamitos, CA, USA, 2023; pp. 124–129. https://doi.org/10.1109/SOSE58276.2023.00021.
118. Calcote, L.; Butcher, Z. *Istio: Up and Running: Using a Service Mesh to Connect, Secure, Control, and Observe*; O'Reilly Media: Newton, Massachusetts, United States, 2019.
119. Khatri, A.; Khatri, V. *Mastering Service Mesh: Enhance, Secure, and Observe Cloud-Native Applications with Istio, Linkerd, and Consul*; Packt Publishing Ltd.: Birmingham, United Kingdom, 2020.
120. Raptis, T.P.; Passarella, A. A Survey on Networked Data Streaming with Apache Kafka. *IEEE Access* **2023**. https://doi.org/10.1109/ACCESS.2023.3303810.
121. Debeau, E.; Quintuna-Rodriguez, V. ONAP: An open source toolkit for zero touch automation. In *Design Innovation and Network Architecture for the Future Internet*; IGI Global: Pennsylvania, USA, 2021; pp. 212–249.
122. OSM, E. OpenSourceMANO. Available online: https://osm.etsi.org/8 (accessed on 15 August 2023).
123. Zhao, S.; Talasila, M.; Jacobson, G.; Borcea, C.; Aftab, S.A.; Murray, J.F. Packaging and sharing machine learning models via the acumos ai open platform. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; pp. 841–846.
124. Turnbull, J. *Monitoring with Prometheus*; Turnbull Press: Brooklyn, New York, USA, 2018.
125. Lahmadi, A.; Beck, F. Powering monitoring analytics with elk stack. In Proceedings of the 9th International Conference on Autonomous Infrastructure, Management and Security (Aims 2015), Ghent, Belgium, 22–25 June, 2015
126. Rizvi, S.; Orr, R.; Cox, A.; Ashokkumar, P.; Rizvi, M.R. Identifying the attack surface for IoT network. *Internet Things* **2020**, *9*, 100162.
127. Zhang, M.; Wang, L.; Jajodia, S.; Singhal, A. Network attack surface: Lifting the concept of attack surface to the network level for evaluating networks' resilience against zero-day attacks. *IEEE Trans. Dependable Secur. Comput.* **2018**, *18*, 310–324.
128. Theisen, C.; Munaiah, N.; Al-Zyoud, M.; Carver, J.C.; Meneely, A.; Williams, L. Attack surface definitions: A systematic literature review. *Inf. Softw. Technol.* **2018**, *104*, 94–103.
129. IBM. Cost of a Data Breach Report 2022. Available online: https://www.ibm.com/downloads/cas/3R8N1DZJ (accessed on 15 August 2023).
130. Chernyshev, M.; Baig, Z.; Zeadally, S. Cloud-Native Application Security: Risks, Opportunities, and Challenges in Securing the Evolving Attack Surface. *Computer* **2021**, *54*, 47–57.
131. Souppaya, M.; Morello, J.; Scarfone, K. *Application Container Security Guide*; Technical Report; Gaithersburg, Maryland, United States, 2017.
132. Spielmann, D.; Sokolowski, D.; Salvaneschi, G. Extensible Testing for Infrastructure as Code. In Proceedings of the Companion Proceedings of the 2023 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity (SPLASH Companion'23), Cascais , Portugal, 22–27 October 2023. https://doi.org/10.1145/3618305.3623607.
133. Shaikh, F.A.; Siponen, M. Information security risk assessments following cybersecurity breaches: The mediating role of top management attention to cybersecurity. *Comput. Secur.* **2023**, *124*, 102974.
134. Li, J. Vulnerabilities mapping based on OWASP-SANS: A survey for static application security testing (SAST). *arXiv* **2020**, arXiv:2004.03216.
135. Zhao, L.; Chen, S.; Xu, Z.; Liu, C.; Zhang, L.; Wu, J.; Sun, J.; Liu, Y. Software Composition Analysis for Vulnerability Detection: An Empirical Study on Java Projects. In Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23), San Francisco, CA, USA, 3–9 December 2023.
136. Vulnerabilities, C. Common Vulnerabilities and Exposures (CVE) Database. Available online: https://cve.mitre.org/(accessed on 15 August 2023).

137. Imtiaz, N.; Thorn, S.; Williams, L. A comparative study of vulnerability reporting by software composition analysis tools. In Proceedings of the Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Bari, Italy, 11–15 October 2021; pp. 1–11.
138. Theodoropoulos, T.; Makris, A.; Korontanis, I.; Tserpes, K. GreenKube: Towards Greener Container Orchestration using Artificial Intelligence. In Proceedings of the 2023 IEEE International Conference on Service-Oriented System Engineering (SOSE). IEEE Computer Society, Athens, Greece, 17–20 July 2023; pp. 135–139.
139. Staron, M.; Abrahão, S.; Penzenstadler, B.; Hochstein, L. Recent Research Into Infrastructure as Code. *IEEE Softw.* **2023**, *40*, 86–88.
140. de Vicente Mohino, J.; Bermejo Higuera, J.; Bermejo Higuera, J.R.; Sicilia Montalvo, J.A. The application of a new secure software development life cycle (S-SDLC) with agile methodologies. *Electronics* **2019**, *8*, 1218.
141. Pitchford, M. The 'Shift Left' Principle. *New Electronics*. **54**, 18-21 (2021,9), https://doi.org/10.12968/s0047-9624(22)60234-7
142. Rahman, A.; Partho, A.; Morrison, P.; Williams, L. What questions do programmers ask about configuration as code? In Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering, Gothenburg, Sweden, 29 May 2018; pp. 16–22.
143. Cankar, M.; Petrovic, N.; Pita Costa, J.; Cernivec, A.; Antic, J.; Martincic, T.; Stepec, D. Security in DevSecOps: Applying Tools and Machine Learning to Verification and Monitoring Steps. In Proceedings of the Companion of the 2023 ACM/SPEC International Conference on Performance Engineering, New York, NY, USA, 15–19 April 2023; pp. 201–205.
144. Duarte, A.; Antunes, N. An empirical study of docker vulnerabilities and of static code analysis applicability. In Proceedings of the 2018 Eighth Latin-American Symposium on Dependable Computing (LADC), Foz do Iguacu, Brazil, 8–10 October 2018; pp. 27–36.
145. Jacob, M. Checkmarx Announces First GenAI-powered AppSec Platform, Empowering Developers and AppSec Teams to Find and Fix Vulnerabilities Faster–Global Security Mag Online 2023. Available online: https://www.globalsecuritymag.com/Checkmarx-Announces-First-GenAI-powered-AppSec-Platform-Empowering-Developers.html (accessed on 15 August 2023).
146. Sönmez, F.Ö.; Kiliç, B.G. Holistic web application security visualization for multi-project and multi-phase dynamic application security test results. *IEEE Access* **2021**, *9*, 25858–25884.
147. Jobin, T.; Kanjirapally, K.; Babu, K.S.; Scholar, P. Owasp Zed Attack Proxy. In Proceedings of the National Conference on Emerging Computer Applications (NCECA), Kottayam, India, 17 June 2021, p. 106.
148. Mallisetty, S.B.; Tripuramallu, G.A.; Kamada, K.; Devineni, P.; Kavitha, S.; Krishna, A.V.P. A Review on Cloud Security and Its Challenges. In Proceedings of the 2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 5–7 January 2023; pp. 798–804. https://doi.org/10.1109/IDCIoT56793.2023.10053520.
149. German, K.; Ponomareva, O. An Overview of Container Security in a Kubernetes Cluster. In Proceedings of the 2023 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), Yekaterinburg, Russia, 15–17 May 2023; pp. 283–285.
150. Maruszczak, A.; Walkowski, M.; Sujecki, S. Base Systems for Docker Containers-Security Analysis. In Proceedings of the 2022 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 22-24 September 2022; pp. 1–5.
151. Sonnekalb, T.; Heinze, T.S.; Mäder, P. Deep security analysis of program code: A systematic literature review. *Empir. Softw. Eng.* **2022**, *27*, 2.
152. Banerjee, K.; Agarwall, D.; Bali, V.; Sharma, M.; Prajwal, S.S.; Arsh, M. A Survey on Kubernetes Policy Report Custom Resource Definition Kube-Bench Adapter. In *Advances in Data and Information Sciences*; Springer, Singapore, 2022; pp. 315–322. https://doi.org/10.1007/978-981-19-5292-0_29.
153. Sedano, W.K.; Salman, M. Auditing Linux Operating System with Center for Internet Security (CIS) Standard. In Proceedings of the 2021 International Conference on Information Technology (ICIT), Amman, Jordan, 14–15 July 2021; pp. 466–471. https://doi.org/10.1109/ICIT52682.2021.9491663.
154. Korontanis, I.; Makris, A.; Theodoropoulos, T.; Tserpes, K. Real-time Monitoring and Analysis of Edge and Cloud Resources. In Proceedings of the 3rd Workshop on Flexible Resource and Application Management on the Edge, Orlando, Florida, USA, 20 June 2023; pp. 13–18.
In Proceedings of the 3rd Workshop on Flexible Resource and Application Management on the Edge, Orlando, USA, 20 June 2023; pp. 7–11.
155. Tan, J.; Ensuring component dependencies and facilitating documentation by applying Open Policy Agent in a DevSecOps cloud environment, 2022. Available online: https://aaltodoc.aalto.fi/handle/123456789/117364 (accessed on 15 August 2023).
156. Team, A. Advanced Maryland automatic network disk archiver (Amanda), (1992–present). Available online: https://www.amanda.org/ (accessed on 15 August 2023).
157. Ferreira, M.; Brito, T.; Santos, J.F.; Santos, N. RuleKeeper: GDPR-Aware Personal Data Compliance for Web Frameworks. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–25 May 2023; pp. 2817–2834.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.