World Journal of Advanced Research and Reviews

eISSN: 2581-9615 CODEN (USA): WJARAI

Cross Ref DOI: 10.30574/wjarr

Journal homepage: https://wjarr.com/

(Research Article)

Container runtime security detection and prevention techniques

Ramasankar Molleti *

Computer Science and Engineering, Jawaharlal Nehru Technological University, Hyderabad, India.

World Journal of Advanced Research and Reviews, 2024, 24(03), 2626-2639

Publication history: Received on 17 November 2024; revised on 26 December 2024; accepted on 28 December 2024

Article DOI: https://doi.org/10.30574/wjarr.2024.24.3.4003

Abstract

Application control is a particular layer of cloud-native security that is aimed at protection during the application's work in the container. This article provides a comprehension of container runtime security measures, including detection and prevention measures. It goes deeper into the importance of protecting container environments, mainly pointing out the outlook for dangers and risks that may occur at runtime. It also provides an overview of commodity detection techniques, such as container image analysis, runtime behavioral analysis, traffic analysis, Host and file integrity. Further, it dwells on the prevention measures involving image hardening, runtime security policies, the principle of least privilege, secure configurations and updates, patching, workload isolation and segmentation, secrets management, and using the immutable infrastructure. Also, this paper briefly discusses recent solutions such as kernel-level security features, runtime application self-protection (RASP), Sandboxing and unikernel solutions. It also goes further in explaining how automated security tools and platforms, especially open source and commercial tool platforms, can be used to improve the security of containers during runtime. In concluding the study, incident response and forensics need to be implemented across container objects, as it stipulates the need to take preventative measures for security threat detection and response.

Keywords: Container Runtime Security; Behavioral Analysis; Vulnerability Detection; Multi-layered Defense; DevSecOps; Orchestration Platforms; Kubernetes

1. Introduction

Containers are small, completely moveable, and highly autonomous executable software modules that contain the application itself, including the code, interpreter, maintenance programs, references, and configuration information [1]. Containers do not possess their own OS kernel as much as virtual machines do, though they take less resources and time to launch and operate as compared to them [2]. This efficiency results from the kind of container that utilizes the existing operating system, which leads to a reduction in overhead.

Container politicians are the lower-level programs that deal with the container life cycle. It deals with container management, including the creation, destruction and reboot of containers and the allocation and management of their resources. Some of the most used container runtimes are Docker, which has gotten attached to the word container now; contained; and CRI-O, which are most preferred in organizations as they are stable and compatible with container scheduling applications such as Kubernetes [3]. In current cloud-native architectures, containers have emerged as the new methodology for developing, deploying and managing applications. They are well suited for microservice architecture, which is a style of designing applications based on smaller, loosely coupled services that can be developed, tested and scaled independently [4, 5]. This has the benefit of providing modularity, which provides better flexibility when making changes and makes for robust applications.

* Corresponding author: Ramasankar Molleti

Containers also help the DevOps processes by creating pipelines for continuous integration and continuous deployment, known as CI/CD [6]. Containers eliminate the "it works on my machine" problem because they deploy the applications and all the necessary components, and entails the same environment in development, test, and production [7]. When working in dynamic but scalable environment, the task to keep solutions and services highly available and reliable is a crucial one, and that is where the concept of consistency starts to come into the foreground. Container technology has developed to the next level and some of the runtimes have gained popularity based on their special features and compatibility with the environment. Among these, Docker which has become one of the most famous containers run time, comes with admirable application tool sets covering both construction and packing and running of containers [3]. Such characteristics make it preferred by developers and businesses in that they do not need robust support to operate efficiently. Containerd which started as a fork of Docker has developed to be a runtime for many large container systems [8]. It is famous for its basic design, speed, and compliance with the Open Container Initiative (OCI) regulations, which is why it can be recommended for use in production [9]. There is another OCI-conformant runtime called CRI-O, which is also part of the Kubernetes project, allowing for seamless integration of the runtime with Kubernetes and offering a weightless and safe means for working with containers in the orchestration platforms [10]. Since containers are part of modern cloud-native environments, shipping their security to the forefront is a must. This article describes the trend as well as the methods necessary for implementing the means of identification and protection against vulnerabilities in the specifics of container runtimes. Thus, by having a deeper understanding of what container runtimes are and what their most significant weaknesses are, one can develop robust security concepts and countermeasures to be taken in order to protect containerized applications.

2. Literature review

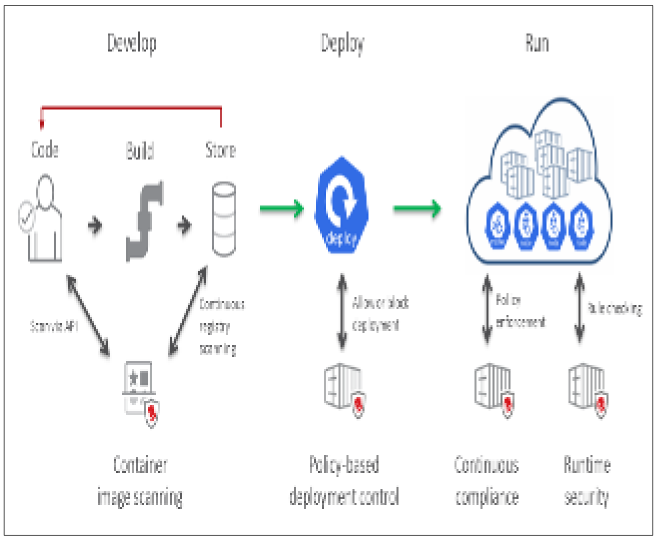

Figure 1 A General Presentation of Container Security [19]. From the Figure above, when it comes to deployment, policy-based deployment control makes sure that container images are only launched when they satisfy the security standards that the group has established. Following deployment, the team is able to periodically inspect its containers thanks to continuous compliance. At runtime, any container activity that deviates from a configurable set of rules can be seen by Runtime Security [19]

2.1. Container Security Environment

Infrastructural innovation spurred by container architecture brings a proportional security complexity that requires detailed handling to guard against vulnerabilities in the applications and data. The isolation problem is one of the main issues. Containers do not have their own operating systems, like other types of virtual machines or the full-fledged operating systems that normal virtual machines do [2]. This feature of sharing OS layer implies that if a container is used for hosting a malicious application, it could result in getting hold of the host system and other containers. These kinds of isolation issues are very essential, especially in multi-tenant environments where different containers may be owned by different users or institutions. Another kind of danger in the area of container security is image vulnerability [11]. The containers could be said to be based on images that consist of the application and all necessary dependencies. When not well handled, these images can contain weaknesses. Hence, using an outdated or configuring a base image with loopholes is likely to result in creating numerous base images loaded with security vulnerabilities that affect all the developed containers [12]. Image vulnerabilities can also derive from the use of untrusted or malicious images, thus the need to source images from reputable image registries and scan them for known vulnerabilities (see Figure 1 descriptions).

Configuration adds another layer of risk to the process of making containers secure. One can encounter misconfigurations at the time of building the container image and even when the container is already deployed and running [13]. The most frequent mistakes apparent in assessments are misconfigurations such as network, privilege, and resource settings. To illustrate, it is critical not to run containers with root privileges because if an attacker gets control over the container, it may severely impact the system [14]. Secure configuration best practices are implemented strictly at all phases of the container's life cycle.

External threats occur at runtime and are quite diverse, containing privilege escalation, lateral movements, and denial-of-service attacks. Privilege escalation arises when the attacker achieves more privilege than he should, which at times leads to the attacker controlling the entire host system [15]. Lateral movement implies the attacker's ability to move through the network to other containers or services, as there is basically poorly constructed network segmentation and bad access controls. DoS attacks aim to interrupt and cause service outages by overloading resources and compromising service availability. To protect containerized environments against these runtime hazards, strong detection and preventive techniques are required.

Coming to the next factors, it is imperative to mention that regulatory and compliance issues also form vital components of container security. There are rules and statutes that must be followed, including GDPR, HIPAA, and PCI DSS, which prescribe certain security measures and procedures that have to be put in place [16]. This has to do with the ability to adequately embrace the right access controls, encryption, auditing and monitoring to correspond to the regulations that are set. Noncompliance may lead to legal consequences, fines, and other consequences that harm the organisation's image.

2.2. Container Runtime Architecture

It is necessary to start the security analysis of container runtimes with understanding of the architecture of such systems. Container runtimes are made up of a number of subroutines that are concatenated in order to execute the container management lifecycle. They are made up of the container engine, the image manager, the executors, and the networks [17]. The container engine is analogous to the processes involved in the creation and running of containers, while the image manager is involved in the storage and retrieval of the containers. The process manager is responsible for the running of containers, and the network manager is responsible for setting and managing the network of the containers [18].

Different container engines, like Docker, Containerd, and CRI-O, have different security features. The most commonly used container engine is Docker; Docker Swam, which has a rich set of surrounding environments but is more exposed to attacks because it provides a wide range of functionalities [23]. The Demonbotnet works under the root user, which may become an attractive target for attackers [21]. Containerd is somewhat simpler and lighter than Dockers; it has weaker adm appeals, but on the other hand, it eliminates most of the possible attack vectors with the help of extremely accurate privilege and access control (refer to Figure 2). CRI-O, developed for Kubernetes, focuses on security and conformance to the OCI specifications, which are perfect for a secured, orchestrated environment.

Container orchestration platforms such as Kubernetes add an extra layer of complexity and security features to the deployment, scaling, and management of containerized applications; however, it is an extensive piece of software with a lot of functions, and its main configuration is open to security issues [22]. Hence, misconfigured Kubernetes clusters may put crucial services on the internet where just about anyone may be able to access the services and even breach the cluster. Network segmentation, role-based access control (RBAC), and pod security policies are a few examples of the robust security measures that may need to be enforced in addition to the default Kubernetes settings [24]. In a YAML file, a Role object must be defined in order to create a role in Kubernetes. The resources, the permitted verbs (actions), and the API group are all included in the Role object. For instance, only the "get" and "list" operations on pods in the core API group are permitted by the following role [25].

```
apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:
```

```
namespace: default
name: pod-reader
rules:
- apiGroups: [""]
resources: ["pods"]
 verbs: ["get", "list"]
```

After defining the Role object, you can use the following command to create the role in your Kubernetes cluster (assuming the YAML file was saved as role.yaml):

```
kubectl apply -f role.yaml
```

It also warrants discussing the links to security from a view of runtime architecture as regards the interaction of containers with the host system or with other containers. The kernel structure model is efficient, but extremely careful measures to contain it must be implemented, coupled with strict measures to enable it to run in a separate space. As for runtime security, mechanisms like namespaces, cgroups, and seccomp profiles let us maintain the level of security. Namespaces create isolation by sharing process IDs, network interfaces and file systems, and cgroups reduce the utilization possibilities of the systems in a certain container [25, 26]. Using seccomp profiles, system calls that a container may perform are limited to the ones not commonly used in attacks [26]. Just to reiterate, gaining a strong understanding of the design of container run time and its constituent elements is important if one is to be able to secure this component of the cloud infrastructure. Thus, by understanding the weaknesses and threats, organizations can implement sustainable measures for shielding container environments for integrity, confidentially and availability of applications and data.

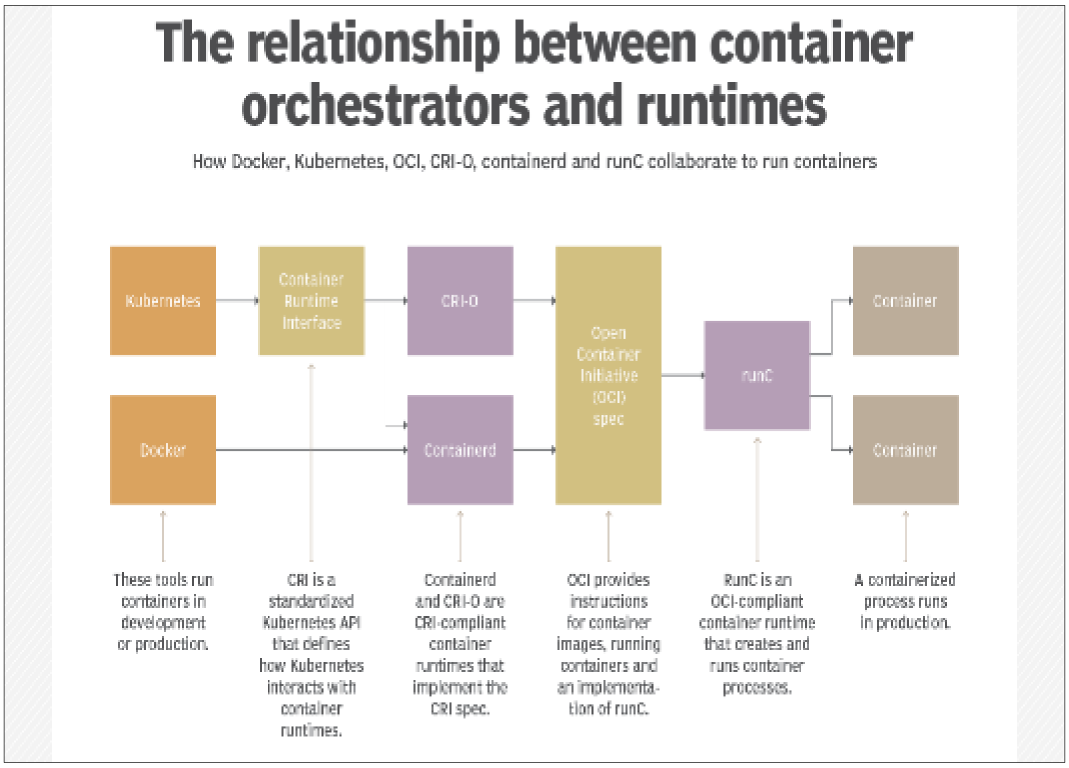
The relationship between container orchestrators and runtimes
How Docker, Kubernetes, OCI, CRI-O, containerd and runC collaborate to run containers
Kubernetes
Container Runtime Interface
CRI-O
Open Container Initiative (OCI) spec
runC
Container
Docker
Containerd
Container
These tools run containers in development or production.
CRI is a standardized Kubernetes API that defines how Kubernetes interacts with container runtimes.
Containerd and CRI-O are CRI-compliant container runtimes that implement the CRI spec.
OCI provides instructions for container images, running containers and an implementation of runC.
RunC is an OCI-compliant container runtime that creates and runs container processes.
A containerized process runs in production.

Figure 2 A flowchart elucidating the relationship between container orchestration and runtimes [20]

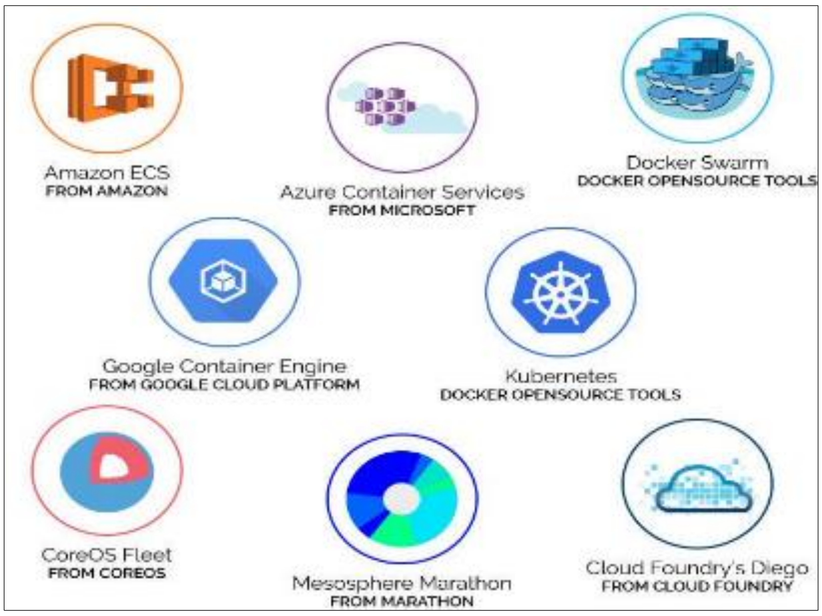

Figure 3 Some of the available tools for container orchestration available in the market [23]. For instance, as presented in the article, Docker Swarm transforms a collection of Docker engines into a single, virtual Docker engine by offering native clustering capabilities for Docker containers [23]

3. Methodology: container runtime security detection and prevention techniques

3.1. Detection Techniques

To employ effective container runtime security, it is crucial to begin with suitable detection tools, which help in preventing container breaches by identifying vulnerabilities, suspicious events, and possible threats before they lead to potentially severe consequences [27]. There are two main techniques of detection: analysis based on the identification of vulnerabilities and misconfiguration of the container image without running the code. Clair, Trivy and Anchore are the instruments that work based on the analysis of the databases for the previously identified vulnerabilities to search the images for shortcomings [27]. These tools inspect the software components within the container to determine if there are any old packages, misconfigured system or any other threats. With that, it is easy to detect evils at the image level in a way that eliminates their chance to fester in production assets. The second detection technique is called runtime behavioral analysis, which is concerned with the examination of dynamics in containers. Falco and Sysdig are such tools that monitor system calls, files' access, and network connections for suspicious behaviors, which may imply an intrusion [28]. For instance, if a container begins to read files, it should not access or establish calls to the network. Such tools will let the administrators know that the container is a security threat. Of all the program analysis methods, runtime behavioral analysis is again more effective for identifying zero-day attacks and other complex attacks that cannot be revealed by static analysis.

Besides, monitoring and analysing the traffic are critical for identifying malicious activities on the network and possible attacks. Cilium (check Figure 4) and Calico are available as solutions to monitor the networking within the context of containerization and observe abnormal activities, including suspicious traffic flows, illegitimate access, and data leakage [29]. It also pointed out that such tools assist in the detection of threats such as lateral movement, in which attackers attempt to extend the infection to other containers in the network.

From Figure 4, the Extended Berkeley Packet Filter (eBPF) permits safe, fast and programmable modification of the network packets, to which the developers can attach micro-programs that are linked with low-level Linux kernel triggers for different networking objectives. Cilium uses eBPF to build a high-speed and adaptable construct of network and application policies for security on the Linux operating system. It makes the application have its own security protocol executions near its edges, diminishing the avenues of attack and strengthening network traffic observation [30]. This comprises an agent that is located on every cluster node, a central command platform, and the distribution of an information plane. eBPF programs are administered by the agent, whereas the command platform and information plane are used for management and security considerations.

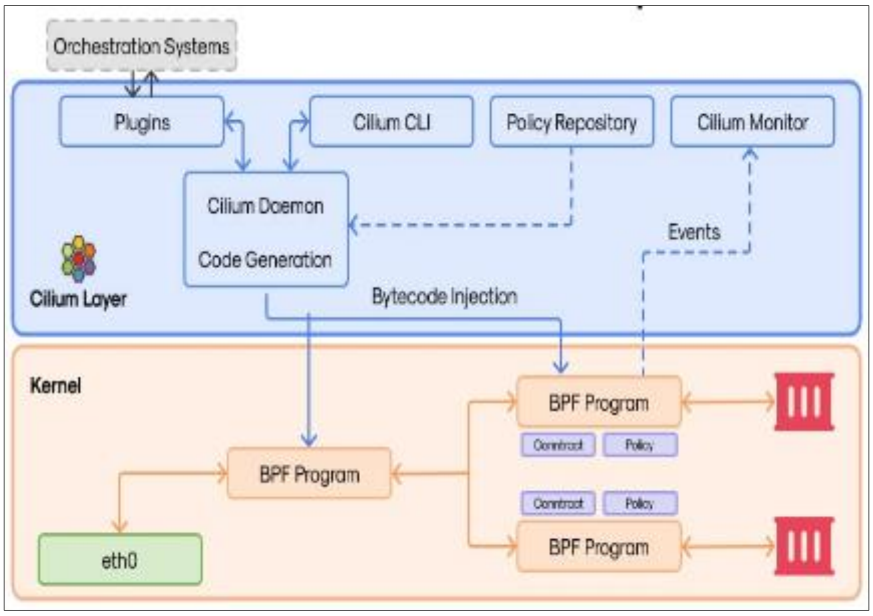

Figure 4 Cillium for Network Security [30]

Host monitoring is another kind of detection that is done at the host level and concentrates on the host system. Third, AuditD and Osquery are the tools that allow for observing what is happening on the host and whether there are signs of a security threat [31]. While Osquery offers an interface akin to SQL for accessing system state data, AuditD keeps detailed logs of system events. Organizations that employ host monitoring can identify attempts to take advantage of host-level vulnerabilities, such as unauthorized modifications to important system files or suspicious process activity. Another security control that is important is the monitoring of changes to files in a container and the host OS. For example, Tripwire and Advanced Intrusion Detection Environment (AIDE) are programs that scan file systems for changes, and if a program tries to write into any unauthorized section, this is easily detected [32]. Through setting and overseeing a mean file state, as well as notifying the administrators when the state is different, these tools are considered to support maintaining the integrity and, therefore, protection of the containerized environment. It is also worth mentioning that logs and anomalies play an important role in the transition to security events. Therefore, there is a possibility of collecting logs from the hosts, containers and network devices to understand the potential threats. Some of the tools used in log analysis can help in drawing out patterns as well as novelties, which, when referred to as risks, can be highly useful in the timely detection and prevention of incidents.

3.2. Prevention Techniques

Prevention strategies are oriented toward minimizing threats and opportunities for attacks before they occur. It is one of the primordial layers of prevention, which means shielding the container images via measures that include frequent updates, working with a limited number of base images, and practicing pre-deployment checks. System Images: Relative updates guarantee that the images have the newest security patches, and creating minimal base images cuts down on the attack surface by only installing the most necessary addenda [33]. Scanning, which can be done prior to the image's deployment in the organization with the help of programs such as Trivy, Clair, and Anchore, is another protective measure.

Another important preventive measure that must be taken is the enforcement of runtime security policies. A network policy in Kubernetes and procedures such as AppArmor and SELinux regulate the manner in which containers behave and the number of risks they pose [34]. Pod networks in Kubernetes allow the organization of traffic between pods, and network policies built on the pod networks reduce the possibility of horizontal movement. Containers and virtualization use AppArmour and SELinux, where the latter is a form of MAC, because it enforces policies on what the container is allowed to do or not do [35]. The principle of least privilege is a security best practice in the use of containers that means that containers should be run with the minimal amount of privileges. Organizations may drastically lessen the effect of a potential security breach by limiting access to only essential resources and employing non-root users inside containers. For the lowest potential risk, it is crucial to note that permissions on containers should be strictly regulated, and they should always be executed as unprivileged users. Security is another significant factor that needs to be given much consideration when it comes to dealing with the containers, as viewed from the following revelations: This includes making a list of reloads for the options the system didn't use, setting a cap on the amount of computer resources a container can use, and making sure that containers begin with safe parameters. To prevent adverse events, it is essential for organizations to eliminate all the services and features that are not crucial for their functioning. Resource control ensures that containers do not use too many resources from the host system in a way that will harm the host system, such as in a denial-of-service attack. Security in containerized environments requires frequent updating and patching of the environments. Such important processes include updating not only the container images but also the host OS and the container orchestration platform with the latest security patches [36]. It is also important to apply patches on a regular basis, as this helps prevent a system from being vulnerable to already identified flaws.

The relative aspect of network segmentation and the definite aspect of micro-segmentation are the key measures to contain the effects of a breach [37]. Implicitly dividing the network into segments also makes it difficult for an attacker to advance to the next segment; they are kept different by access controls to minimize the effect of a possible attack. Network policies that are employed by tools such as Calico and Cilium are tools that are used to enforce a very detailed degree of control within containers. Besides, secrets management entails having a way of storing and controlling secret data that includes identification numbers, passwords, and certificates used by containers. Solutions such as HashiCorp Vault and Kubernetes Secrets allow for methods of storing and pulling tokens and other sensitive information in order to only allow the containers that need to access them to do so [38]. Information security hence requires that secrets are properly managed to avoid incidences such as these and to protect such vital information. Immutable infrastructure is one of the best practices where containers are deployed and cannot be altered after provision [39]. All changes, including those in the application or its settings, mean the creation of a new container image. This helps to avoid configuration drift and maintain security, as well as being much easier in terms of having repeatable processes.

3.3. Advanced Security Mechanisms

An extra layer of security measures serves to enhance the security of a containerized environment. Kernel-level security features, including seccomp, AppArmor, and SELinux, are very important security modules that are responsible for the enforcement of security policies at the kernel level [35]. Seccomp lowers the cards that containers may make to the system, therefore limiting the possibilities of executing certain unwanted operations [26]. AppArmor and SELinux use the MAC that grants the security policies and enforces them on the computers.

Runtime application self-protection (RASP) is a relatively new security solution that enables the real-time protection of applications through program code monitoring and the possibility of preventing suspicious actions [40]. RASP operates at the application level and deploys in the application runtime environment; hence, it can detect and take action on threats at its level. It is useful when employed as an additional layer of security to counter sophisticated methods that other methods fail to notice. The other form of isolation that is offered by sandboxing technologies is where the containers are run in other isolated environments [41]. This minimizes the chance of having another container or the host system contaminated with a malware attack. Because the concept of containerization involves layering isolation, sandboxing augments and strengthens the security of containerized platforms from possible attacks. Also, unikernel solutions entail the use of applications that are scaled down to bare-bone operating systems [42]. This, in turn, reduces the attack surface and boosts security by removing components and services that are not necessary. Porno represents a logical evolution of traditional virtual machines, and it is a handy security tool in the containerization context as it is very isolated.

3.4. Automated Security Tools and Platforms

The application of automated security solutions and platforms is imperative for dealing with the security of many containers. Other tools that are open-sourced include securing and monitoring, which are offered by Falco Sysdig and Cilium [28]. These tools are currently being used in the community and possess features such as high customization and integration. There are various other commercial container security solutions, which include Aqua Security, Twistlock, which is now a part of Palo Alto Networks, and StackRox, which is now a part of Red Hat [43]. These platforms have additional features such as vulnerability scanning, runtime protection and compliance, which qualify them for enterprise deployment. Security should be incorporated right from the development process up to the deployment, and therefore, it is essential to integrate security with CI/CD pipelines. Thus, security tools can be integrated into CI/CD processes, which allow for discovering vulnerabilities, maintaining security policies, and allowing the application of security to only approved code in production. This helps to achieve the objective of shifting security to the left so that most of the problems can be discovered and fixed in the development phase.

3.5. Incident Response and Forensics in Container Environments

Security incidents must be dealt with and analyzed, which is why response and forensic functions are part of container security. Some peculiarities of containerization are the rolling nature of containers, the nature of containerization engines, and the fact that they require separate tools and approaches. Forensics of containers is basically about acquiring a dump or a copy of the part of the container as well as the host system. This comprises the gathering of logs, memory dumps, and filesystem images to analyze possible security risks. Tools like Sysdig Inspect and Falco can help in the forensic investigation process as they give a live view of what is happening at the container level and at the system level, respectively [28]. Automated incident response tactics are therefore pre-planned measures of using other tools and processes to identify, act on, and neutralise security breaches. This can entail the isolation of contaminated containers, notification to security staff, and the execution of other related actions. Truly, when an organization has put in measures for automating the response to incidents, they are assured of better protection of their networks.

Table 1 Detection and Prevention Techniques for Container Runtime Security

Technique	**Tools**	**Technique**
Detection		
Static Analysis	Clair, Trivy, Anchore	Scans container images for known vulnerabilities and misconfigurations
Runtime Behavioral Analysis	Falco, Sysdig	Monitors container behavior for anomalies
Network Traffic Analysis	Cilium, Calico	Analyzes network traffic to detect suspicious activities
Host Monitoring	AuditD, osquery	Monitors the host system for suspicious activities
File Integrity Monitoring	Tripwire, AIDE	Tracks changes to critical files
Log Analysis and Anomaly Detection	Various logging tools	Analyzes logs to identify patterns and anomalies
Prevention		
Image Hardening	Best practices	Secures images through regular updates, minimal base images, and scanning
Runtime Security Policies	Kubernetes Network Policies, AppArmor, SELinux	Enforces security policies to restrict container behavior
Least Privilege Principle	Best practices	Runs containers with minimum necessary privileges
Secure Configurations	Best practices	Configures containers with secure settings and resource limits
Regular Updates and Patching	Best practices	Keeps images, host systems, and orchestration platforms up to date
Network Segmentation	Calico, Cilium	Divides the network into smaller segments with strict access controls
Secrets Management	HashiCorp Vault, Kubernetes Secrets	Securely stores and manages sensitive information
Immutable Infrastructure	Best practices	Deploys containers in an immutable manner to ensure consistency
Advanced Security Mechanisms		
Kernel-level Security Features	seccomp, AppArmor, SELinux	Enforces security policies at the kernel level
RASP	Various RASP tools (Dynatrace Application Security, Imperva RASP, Datadog Application Security Management, e.t.c.)	Provides real-time protection for applications

Sandboxing Technologies	Various sandboxing tools (Falcon, Cuckoo, e.t.c.)	Runs containers in isolated environments
Unikernel Approaches	Various unikernel projects	Runs applications on specialized, minimalistic operating systems
Automated Security Tools		
Open-source Security Tools	Falco, Sysdig, Cilium	Provides robust security capabilities for detecting and preventing threats
Commercial Security Platforms	Aqua Security, Twistlock, StackRox	Offers comprehensive security solutions for container environments
CI/CD Integration	Various CI/CD tools	Embeds security throughout the development and deployment process
Incident Response and Forensics		
Incident Response Strategies	Various automation tools	Automates detection, response, and mitigation of security incidents
Forensic Analysis	Sysdig Inspect, Falco	Collects and analyzes evidence from containers and hosts

It is vital to note that applying all of the above mentioned detection (Table 1) and prevention measures can contribute to the improvement of security in the context of containerization and the corresponding decrease of threats related to container runtime security.

4. Results and Discussion

4.1. Container Runtime Security Detection and Prevention Significance and Advantages

That is why monitoring and protection against threats in the container's runtime is considered one of the most essential levels of cybersecurity. First of all, a stronger security stance is maintained through the ongoing examination and shielding of applications that have been containerized. Such a preventive course minimizes the number of successful cyber attacks so that one is able to safeguard the data as well as retain the system's sanctity. More surprising, as containers add value to cloud-native environments, it is crucial to build up security solutions for them since their application is crossing diverse industries. The fourth advantage is compliance with unequivocal sales. opaquen Privacy is a must-have in today's world and organizations have standard regulatory measures such as GDPR, HIPAA, and PCI-DSS put in place to enhance the security of data. Measures of Container Runtime Security meet such compliance requirements; hence, organizations avoid legal repercussions and customer trust loss. Thus, facilitated by these measures, constant audits and security checks guarantee compliance with the emerging regulations. Container runtime security, therefore, achieves a notable enhancement of operational efficiency [44]. It allows for efficient allocation of resources because, through security processes such as vulnerability scanning, behavioral analysis, and network monitoring, among others, the company can automate a number of processes. This automation frees up the number of checks and balances that would need to be carried out by the security teams in real time, constantly resulting in security teams being bogged down in redundant work. Furthermore, the early detection and prevention of security problems reduces the time taken to resolve problems and improves the systems' dependability. This is another advantage that can be considered a direct outcome of the successful implementation of container runtime security. Security breaches lower the probability of getting penalties, paying for the cost of detection of the break and the consequential damage of consumers' loss of confidence [45]. Further, most industrial processes that enhance automation and efficiency are likely to reduce operation costs. It is far more cost-effective to incorporate security into the design of organizations from the beginning than to later incur the ridiculous prices of security and breach solutions.

The elaboration and enhancement of the incident detection and prevention systems lead to better response times. Through constant system checks and the use of notifications, the security department is able to combat threats more efficiently. This swift action has a great impact on mitigating the effects of security incidents, thereby continuing business and preserving the reputation of the organization. Robust security measures, in conjunction with efficient incident response tactics, provide faster recovery and less operational disruption [46].

A specific example of this concept is container runtime security, which is particularly relevant to e-commerce platforms such as Shopify. Shopify itself also continued to grow its user base, which produced an increasing need for easily scalable and easily performant containerized applications. Though this transition had many advantages, it also created many security problems. Opportunities and threats emerged due to the dynamic characteristics of containers and the requirement of a fast and enormous deployment.

To this end, Shopify responded by devising a sound container runtime security framework in the face of these hurdles. These measures comprise Container Image Vulnerability Scanning, which is used to search for vulnerable container images before they are deployed; Behavioral Analysis of Containers, where containers are monitored for their activities; and Network Traffic Analysis, where patterns of traffic by containers are observed for anything out of the normal [47]. Also, Shopify expanded security policies like the principle of least privilege, secure configurations, regular updates and patching [48]. The consequences of these implementations were quite profound. Concerning this aspect, Shopify experienced a direct enhancement of security profiles with the fewest vulnerabilities and related threats. The adherence to the set regulations was effectively implemented, and this was an added advantage that increased customers' confidence. Strategic efficiency was improved by automating security procedures, thus freeing up time for users such as Shopify's security team to work on other issues. Security management costs were reduced, while the reaction time to security incidents was increased to prevent affecting the majority of services.

4.2. Analysis of the Effectiveness of Various Techniques

The general worth of container runtime protection strategies can be illustrated from the steps where they offer end-to-end security within the layers of the container stack. By using tools such as Clair and Trivy and running a static analysis on the images to confirm the level of vulnerability, only images without serious vulnerabilities will be launched [49]. Falco and Sysdig are examples of runtime behavioral analysis tools that can monitor and detect suspicious activities in real time, which is very helpful to protect against runtime threats. Cilium and Calico are examples of netflow analysis tools that provide insight into network behaviors, allowing for monitoring or blocking of suspicious movement. Specific tools for host monitoring include AuditD and Osquery, which assist in checking the host system to look for signs of breaches in the infrastructure. Other measures, such as file integrity monitoring and log analysis, also enhance security as they check on the integrity of certain files and find out if there are irregularities in logs, respectively.

4.3. Challenges and Limitations in Implementing Container Runtime Security

However, like any other security solutions, there are some limitations associated with container runtime security solutions. One primary challenge is the fact that security is very dynamic and distributed environments of containers are very dynamic. Containers themselves are lightweight in design and this makes it hard to have constant view and control over them. Also, the incorporation of security tools and processes into already functional activities implies a proper workflow in the organization. Another limitation is the influence on the performance of security measures. While fundamental, some of the security tools and methods can cause additional delay and workload, which in turn can undermine the application's efficiency [50]. Security should be implemented appropriately to avoid a negative impact on the functionality of a website, since some measures that enhance security may be ineffective for a website. Moreover, the threat environment is dynamic, and a security threat that may be relevant today may not be the same threat in two years' time. and as such, there is a constant need for updating and changing security postures, which entails constant spending on security personnel and equipment.

4.4. Best Practices

Container runtime security is an important area of focus and the tips to follow are best practices for any organization. This constant check is significant to identify the little security flaws that might have been noted in previous assessments yet pose a threat, hence making sure that a business continues to protect itself as well as take advantage of the best laws to use. CI/CD security involves ensuring that the security measure to be taken will be part of the development life cycle, hence being taken care of right from the development stage [51]. This cuts down on the probability of deploying contaminated containers by incorporating only credible images from reliable sources.

This includes processes like network segmentation as well as micro-segmentation, which locks down suspect areas and stops the spread of the breach within the network. Logging and monitoring are done in a comprehensive manner to allow monitoring of the containers at all times in order to detect forms of threat as they occur. Altogether, these best practices contribute to the strengthening of the security of containerized environments through the comprehensive protection of various threats.

4.5. Future Trends And Inventions

The following are the thrusts that advance container runtime security to a new level:. Machine learning and AI are used to implement as well as improve threat identification and management. These technologies are capable of sorting out data that can predict security patterns and future possible threats that can be combatted. It is observed that the implementation of Zero Trust Architectures, which enforce continuous and strict access protection models, is gradually finding its place in container security against advanced threats [52]. In addition to that, existing runtime protections are set to be improved with more robust anti-exploit capabilities, such as fine-grained access controls and spike security policies that will be in real-time. New types of containers that are coming to the market, including server-less and unikernels, offer new, sometimes troubling, forms of security. First of all, it is simply essential to observe how innovative such technologies are in their fields and, consequently, how dynamic the security concepts to safeguard them must also be. This concept is achieved through DevSecOps practices, which help bring security from a secondary consideration to the central point of the development life cycle, paving the way for a culture of security improvement

5. Conclusion

With container technologies becoming the order of application deployment and management, it is critical that there are strong means of runtime security detection and averting tactics in place. Thus, this paper has addressed a rich spectrum of protective measures, ranging from code analysis and behavior profiling to sophisticated protection layers and script-driven response capabilities. Successful security of containers therefore must incorporate preventive measures as well as a proactive and reactive system of detection.

The nature of container technologies is moving really fast, as is the threat that targets them; thus, there is always a need to be on the lookout and to learn. To sustain a healthy security position, organizations need to acquaint themselves with new threats and security solutions as they come into the market. So as years pass and larger percentages of companies adopt containerization in their cloud systems, there must be a focus on security right from the start and development, as well as deployment loops.

Thus, future work must concern the application of artificial intelligence and machine learning to improve threat detection and counteraction in containers. Also, understanding how modern advances in containerization are posing new security concerns and how zero-trust principles should be realigned for containerized operations would be important. Thus, keeping with these advancements and remaining focused on security principles, organizations can benefit effectively from container technologies while protecting their important resources and information

References

[1] IBM, “Containerization Explained | IBM,” *www.ibm.com*, 2023. https://www.ibm.com/topics/containerization

[2] GoogleCloud, “Containers vs. virtual machines (VMs),” *Google Cloud*. https://cloud.google.com/discover/containers-vs-vms

[3] Docker, “The Industry-Leading Container Runtime | Docker,” *www.docker.com*, Sep. 28, 2021. https://www.docker.com/products/container-runtime/#:~:text=Docker%20Engine%20is%20the%20industry (accessed Jul. 18, 2024).

[4] D. K. Pandiya, “Performance Analysis of Microservices Architecture in Cloud Environments,” *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 10, no. 12, pp. 264–274, Dec. 2022, Accessed: Jul. 18, 2024. [Online]. Available: https://ijritcc.org/index.php/ijritcc/article/view/10745/8231

[5] D. Huang and H. Wu, “Mobile Cloud Service Models,” *Mobile Cloud Computing*, pp. 65–85, 2018, doi: https://doi.org/10.1016/b978-0-12-809641-3.00004-1.

[6] Cloudkinetics, “CI/CD Pipeline For Container-Based Workloads | Cloud Kinetics,” *Cloudkinetics*, May 30, 2022. https://www.cloud-kinetics.com/blog/enabling-ci-cd-pipeline-for-container-based-workloads/#:~:text=A%20DevOps%20Strategy (accessed Jul. 18, 2024).

[7] C. Gaikwad, “What is Containerization? | Harness,” *Harness.io*, May 29, 2024. https://www.harness.io/harness-devops-academy/what-is-containerization#:~:text=With%20containers%2C%20developers%20can%20package (accessed Jul. 18, 2024).

[8] A. Andrei, “Docker vs. Containerd: A Quick Comparison (2023),” *DevOps Blog*, Jul. 26, 2022. https://kodekloud.com/blog/docker-vs-containerd/ (accessed Jul. 18, 2024).

[9] Open Container Initiative, “Open Container Initiative - Open Container Initiative,” *opencontainers.org*. https://opencontainers.org/

[10] CRI-O, “cri-o,” *cri-o.io*. https://cri-o.io/#:~:text=It%20allows%20Kubernetes%20to%20use (accessed Jul. 18, 2024).

[11] S. Zipkin, “4 Categories of Container Security Vulnerabilities (& Best Practices to Reduce Risk),” *Veracode*, Feb. 02, 2023. https://www.veracode.com/blog/secure-development/4-categories-container-security-vulnerabilities-best-practices-reduce-risk

[12] A. Maqsud, “Docker Container : Born To Revolt, But Now It Needs More Security,” *Medium*, Jun. 29, 2023. https://medium.com/@azizulmaqsud/docker-container-revolted-first-but-now-requires-more-security-cab3eaf29091

[13] N. Ehrman, “8 Container Security Best Practices | Wiz,” *wiz.io*, Mar. 01, 2024. https://www.wiz.io/academy/container-security-best-practices

[14] “What is Container Security | Tools, Solutions & Best Practices | Imperva,” *Learning Center*. https://www.imperva.com/learn/application-security/container-security/#:~:text=In%20addition%2C%20attackers%20with%20root (accessed Jul. 18, 2024).

[15] M. Haber, “Privilege Escalation Attack & Defense Explained | BeyondTrust,” *www.beyondtrust.com*, Mar. 02, 2021. https://www.beyondtrust.com/blog/entry/privilege-escalation-attack-defense-explained

[16] Penta Security, “A Brief Look at 4 Major Data Compliance Standards: GDPR, HIPAA, PCI DSS, CCPA,” *Penta Security Systems Inc.*, Aug. 06, 2020. https://www.pentasecurity.com/blog/4-data-compliance-standards-gdpr-hipaa-pci-dss-ccpa/

[17] Gitlab, “Docker executor | GitLab,” *docs.gitlab.com*. https://docs.gitlab.com/runner/executors/docker.html (accessed Jul. 18, 2024).

[18] E. Mell, “What Is Container Management and Why Is It Important?,” *SearchITOperations*, 2023. https://www.techtarget.com/searchitoperations/definition/container-management-software

[19] Trend, “Container Security | Trend Micro Service Central,” *docs.trendmicro.com*, 2024. https://docs.trendmicro.com/en-us/documentation/article/trend-vision-one-container-security (accessed Jul. 18, 2024).

[20] S. J. Bigelow, “A breakdown of container runtimes for Kubernetes and Docker | TechTarget,” *IT Operations*, 2021. https://www.techtarget.com/searchitoperations/tip/A-breakdown-of-container-runtimes-for-Kubernetes-and-Docker (accessed Jul. 18, 2024).

[21] Kaspersky, “What is a Botnet?,” *Kaspersky.com*, 2019. https://www.kaspersky.com/resource-center/threats/botnet-attacks

[22] H. Dhaduk, “Container Orchestration | Basics, Benefits, Tools, and Best Practices,” *Simform - Product Engineering Company*, Jun. 01, 2022. https://www.simform.com/blog/container-orchestration/

[23] F. Bashir, A. Padmanabhan, T. Tin, and devbot5S, “Container Orchestration,” *Devopedia*, Mar. 17, 2017. https://devopedia.org/container-orchestration

[24] “Securing Your Kubernetes Cluster: Kubernetes Best Practices and Strategies,” *Palo Alto Networks*. https://www.paloaltonetworks.com/cyberpedia/kubernetes-cluster-security (accessed Jul. 18, 2024).

[25] Aqua, “Kubernetes RBAC: Why You Need It and 4 Tips for Success,” *Aqua*, Sep. 2023. https://www.aquasec.com/cloud-native-academy/kubernetes-101/kubernetes-rbac/#:~:text=Kubernetes%20RBAC%20is%20a%20powerful (accessed Jul. 18, 2024).

[26] Kubernetes, “Restrict a Container’s Syscalls with seccomp,” *Kubernetes*. https://kubernetes.io/docs/tutorials/security/seccomp/

[27] Palo Alto, “What Is Container Security?,” *Palo Alto Networks*. https://www.paloaltonetworks.com/cyberpedia/what-is-container-security#:~:text=Container%20scanning%20tools%20include%20Aqua (accessed Jul. 18, 2024).

[28] Sysdig, “What is Falco?,” *Sysdig*, 2024. https://sysdig.com/learn-cloud-native/container-security/what-is-falco/#:~:text=Falco%20works%20by%20monitoring%20syscalls (accessed Jul. 18, 2024).

[29] K. Hoffman , “Cilium vs Calico vs Flannel,” *Civo.com*, 2023. https://www.civo.com/blog/calico-vs-flannel-vs-cilium (accessed Jul. 18, 2024).

[30] Wallarm, "Cilium vs. Calico: A Roadmap to Advanced Network Security," *www.wallarm.com*, 2024. https://www.wallarm.com/cloud-native-products-101/cilium-vs-calico-network-security (accessed Jul. 18, 2024).

[31] IzyKnows, "Linux auditd for Threat Detection [Final] - IzyKnows - Medium," *Medium*, Mar. 21, 2023. https://izyknows.medium.com/linux-auditd-for-threat-detection-final-9d5173706b3f (accessed Jul. 18, 2024).

[32] Anirudhadak, "AIDE (Advanced Intrusion Detection Environment) , Grub Secure," *Medium*, May 27, 2024. https://medium.com/@anirudhadak25/aide-advanced-intrusion-detection-environment-grub-secure-0278c9e31811#:~:text=AIDE%2C%20or%20Advanced%20Intrusion%20Detection (accessed Jul. 18, 2024).

[33] F. Soulis, "Hardening Container Images: Best Practices and Examples for Docker," *Medium*, Dec. 19, 2023. https://medium.com/@SecurityArchitect/hardening-container-images-best-practices-and-examples-for-docker-e941263cab13 (accessed Jul. 18, 2024).

[34] "Network Policies," *Kubernetes*. https://kubernetes.io/docs/concepts/services-networking/network-policies/

[35] x]cube LABS, "Integrating Containers with Security Tools," *[x]cube LABS*, Apr. 16, 2024. https://www.xcubelabs.com/blog/integrating-containers-with-security-tools-like-selinux-and-apparmor/#:~:text=SELinux%20and%20AppArmor%20are%20Linux (accessed Jul. 18, 2024).

[36] NetApp, "7 Container Security Best Practices You Must Know," *Spot.io*. https://spot.io/resources/container-security/7-container-security-best-practices-you-must-know/#:~:text=7.- (accessed Jul. 18, 2024).

[37] Checkpoint, "Network Segmentation vs Micro-Segmentation," *Check Point Software*. https://www.checkpoint.com/cyber-hub/network-security/network-segmentation-vs-micro-segmentation/#:~:text=Instead%20of%20breaking%20a%20network (accessed Jul. 18, 2024).

[38] N. Purva, "Securing Kubernetes Secrets with HashiCorp Vault," *InfraCloud*, Jun. 15, 2022. https://www.infracloud.io/blogs/kubernetes-secrets-hashicorp-vault/ (accessed Jul. 18, 2024).

[39] H. Virdó, "What Is Immutable Infrastructure? | DigitalOcean," *www.digitalocean.com*, Sep. 26, 2017. https://www.digitalocean.com/community/tutorials/what-is-immutable-infrastructure

[40] G. Alvarenga, "Runtime Application Self-Protection (RASP) – CrowdStrike," *crowdstrike.com*, Jul. 07, 2023. https://www.crowdstrike.com/cybersecurity-101/cloud-security/runtime-application-self-protection-rasp/

[41] L. Rosencrance, "What is a Sandbox? Definition from SearchSecurity," *SearchSecurity*, Sep. 2021. https://www.techtarget.com/searchsecurity/definition/sandbox

[42] A. Madhavapeddy and D. J. Scott, "Unikernels: Rise of the Virtual Library Operating System - ACM Queue," *queue.acm.org*, 2014. https://queue.acm.org/detail.cfm?id=2566628

[43] RedHat, "Twistlock, now Prisma Cloud by Palo Alto Networks, on OpenShift," *Red Hat Customer Portal*, Jun. 14, 2024. https://access.redhat.com/solutions/3335421 (accessed Jul. 18, 2024).

[44] K. Lee, J. Kim, I.-H. Kwon, H. Park, and C.-H. Hong, "Impact of Secure Container Runtimes on File I/O Performance in Edge Computing," *Applied Sciences*, vol. 13, no. 24, p. 13329, Jan. 2023, doi: https://doi.org/10.3390/app132413329.

[45] P. Wang and D. Wood, "ECONOMIC COSTS AND IMPACTS OF BUSINESS DATA BREACHES," *Issues in Information Systems*, vol. 20, no. 2, pp. 162–171, 2019, Available: https://iacis.org/iis/2019/2_iis_2019_162-171.pdf

[46] A. Irei, "What is Incident Response? Definition from WhatIs.com," *SearchSecurity*, Mar. 2023. https://www.techtarget.com/searchsecurity/definition/incident-response

[47] "Monitoring and handling errors in production," *Shopify*. https://shopify.dev/docs/apps/build/functions/monitoring-and-errors (accessed Jul. 18, 2024).

[48] Shopify, "What Is Product Security? How to Improve Product Security (2024) - Shopify," *Shopify*, Jun. 28, 2024. https://www.shopify.com/blog/product-security (accessed Jul. 18, 2024).

[49] N. Kumar, "Vulnerability Scanning with Clair and Trivy: Ensuring Secure Containers," *Medium*, Feb. 25, 2023. https://medium.com/techbeatly/vulnerability-scanning-with-clair-and-trivy-ensuring-secure-containers-db75c27524c2 (accessed Jul. 18, 2024).

[50] J. Burke, "8 challenges every security operations center faces," *SearchSecurity*, 2020. https://www.techtarget.com/searchsecurity/tip/8-challenges-every-security-operations-center-faces

[51] DevSecOps, "CI/CD and Build Security | TryHackMe THM | Write-Up | Walkthrough," *Medium*, Feb. 27, 2024. https://medium.com/@DevSec0ps/ci-cd-and-build-security-tryhackme-thm-write-up-walkthrough-c672b7762cf9 (accessed Jul. 18, 2024).

[52] H. Kang, G. Liu, Q. Wang, L. Meng, and J. Liu, "Theory and Application of Zero Trust Security: A Brief Survey," *Entropy*, vol. 25, no. 12, p. 1595, Dec. 2023, doi: https://doi.org/10.3390/e25121595.