International Science Journal of Engineering & Agriculture
2025; 4(1): 1-13
https://isg-journal.com/isjea
doi: 10.46299/j.isjea.20250401.01
ISSN: 2720-6319

INTERNATIONAL SCIENCE GROUP

Orchestrating honeypot deployment in lightweight container platforms to improve security

Yurii Tulashvili
Department of Computer Science, Lutsk National Technical University, Lutsk, Ukraine
ORCID 0000-0002-0780-9529

Viktor Kosheliuk
Department of Computer Science, Lutsk National Technical University, Lutsk, Ukraine
ORCID 0000-0002-4136-5087

To cite this article:
Tulashvili Yurii, Kosheliuk Viktor. Orchestrating honeypot deployment in lightweight container platforms to improve security. International Science Journal of Engineering & Agriculture. Vol. 4, No. 1, 2025, pp. 1-13. doi: 10.46299/j.isjea.20250401.01.
Received: 01 10, 2025; **Accepted**: 01 31, 2025; **Published**: 02 01, 2025

Abstract: A significant evolution has occurred in the architectural and infrastructural domains of web applications over the past several years. Monolithic systems are gradually being superseded by microservices-based architectures, which are now considered the de facto standard for web application development owing to their inherent portability, scalability, and ease of deployment. Concurrently, the prevalence of this architecture has rendered it susceptible to specialized cyberattacks. While honeypots have proven effective in the past for gathering real-world attack data and uncovering attacker methods, their growing popularity has made them a specific target for cyberattacks. Traditional honeypots lack the flexibility of microservices architecture. Honeypots have proven effective in gathering authentic attack data and analyzing attacker tactics. The core idea that honey traps help identify malicious packets with minimal effort to remove incorrect alerts is preserved. In addition to identifying and documenting specific attack methods used by intruders, this system helps thwart attacks by creating realistic simulations of the actual systems and applications within the network. This effectively slows down and confuses attackers by making it difficult for them to gain access to real devices. This paper presents a groundbreaking approach to honeypot management within cybersecurity, utilizing virtual clusters and a microservice architecture to significantly improve the effectiveness of threat detection. To conduct our research, we initially surveyed the internet to pinpoint container and container management systems operating on standard ports that might be susceptible to attacks. The monitoring of the instrumented approach generated a massive dataset, enabling researchers to make significant inferences about the behavior and goals of malevolent users. We advocate for the implementation of honeypots on lightweight distribution orchestration tools installed on Ubuntu servers, situated behind a meticulously crafted gateway and operating on standard port configurations. In light of the scan outcomes, we recommend the deployment of honeypot orchestration on streamlined distributions. To better protect your systems based on our scan results, we recommend implementing honeypot orchestration for easier deployment and management. By deploying honeypots on lightweight operating systems, you can optimize resource usage and improve performance while maintaining essential capabilities. These capabilities include monitoring attack patterns on vulnerable systems and analyzing the security measures implemented by those responsible for managing exposed systems.

Keywords: honeypot, microservices, orchestration, security, virtual cluster

1. Introduction

A paradigm shift has occurred in the architectural and infrastructural landscape of web applications over the past several years. The trend in web application development is shifting away from monolithic systems and towards microservices-based architectures. This popularity is driven by the advantages of microservices, such as portability, scalability, and ease of deployment. However, as microservices become more prevalent, they also become a target for new cyberattacks specifically designed to exploit this architecture [1]. Honeypots have proven to be valuable tools for collecting real-world attack data and uncovering attacker methods. Security professionals can learn a lot about how attackers work and the types of attacks they use by setting up honeypots. Traditional honeypot designs haven't utilized microservices architectures effectively. This presents an opportunity to develop honeypots with new capabilities due to the inherent characteristics of microservices [2].

Effective control of cyber threats requires a thorough investigation of their nature. Since cybersecurity risks pose a serious threat, we need to investigate them thoroughly. It's crucial to uncover our current manufacturing shortcomings. This will help us understand how adversaries might take advantage of them, especially the information they'd prioritize. To effectively defend something, you need to be aware of its internal state. This highlights the importance of security observability, which allows monitoring and understanding of your system's activity [3]. To truly see what's going on inside your API/microservices system and protect it from attacks, you need to have good observability. Honeypots directly address these security concerns by enabling the observation of systems through logs, statistics, and distributed traces.

The trend in large web application development has shifted towards microservices architecture, which emerged from Service-Oriented Architecture (SOA) [4]. Microservices are built on the idea of modularity. They are broken down into smaller, self-contained pieces that communicate with each other without relying heavily on each other. This modular approach offers advantages like easier scaling, faster deployment cycles, and improved security through separation of concerns. While microservices offer advantages, their decentralized structure makes it fundamentally difficult to secure these environments. As the use of microservices grows, attackers are increasingly motivated to develop new ways to exploit them.

This includes malicious actors pushing infected Docker images to public repositories and developing malware specifically designed to compromise containers and create backdoors for further attacks [5]. The way microservices architectures are built creates security vulnerabilities that traditional monolithic applications don't have. This leads to different hacking strategies being employed to target each type of application. Containerized environments pose unique security challenges compared to traditional monolithic systems. Because of these differences, intrusion detection and prevention systems designed for monolithic applications may not be as effective in containerized settings. These limitations stem from variations in both the methods used to detect intrusions and the types of vulnerabilities targeted by the systems. This is why understanding the methods adopted by attackers when infiltrating microservices-based web applications, through real-world data, is crucial for properly identifying attack patterns and designing effective security solutions.

Security researchers use honeypots to collect valuable real-world attack data. These decoy systems, pioneered by projects like Honeynet, mimic real systems to attract attackers. By analyzing attacker behavior within the honeypot, researchers can identify the latest attack patterns. Honeytraps act as magnets for attackers. By studying how they interact with the decoy system, we can learn about their tactics and the types of malicious packets they use. This knowledge helps us improve our defenses against real attacks [6]. Instead of just spotting and reporting on specific attack methods or tools used by hackers, this system also helps prevent attacks from reaching real devices. It does this by mimicking the real systems and programs running on the network.

Honeypots provide security teams with a wider view of potential threats and the ability to defend against attacks that bypass firewalls. Many organizations around the world use them as an extra layer

of protection against both internal and external security risks. Imagine a cybercriminal looking for treasure. In cybersecurity, a honeytrap is a fake system that lures attackers in and allows security experts to learn about their methods and stop them from harming real systems. It uses methods that attackers might employ to trick people, but instead of stealing information, it gathers intelligence about the attackers themselves and how they operate. Penetration testers often scan for open ports on a system. These ports might appear vulnerable, but a penetration test is a controlled way to assess those vulnerabilities.

Unlike other security measures that focus on directly stopping intrusions, honey-potting takes a different approach. It aims to strengthen a company's ability to detect intrusions and respond to them efficiently. This allows them to effectively handle and minimize the impact of attacks. Accurate detection of anomalous and malicious traffic is essential for network security. By identifying these threats, security personnel can take action to analyze and restrict them, safeguarding the network. Researchers have developed several machine learning (ML) methods to identify and block fraudulent traffic on networks [7]. These methods use carefully chosen data points to categorize harmful traffic flows.

Designed to lure attackers away from real systems, decoy systems generate alerts whenever someone interacts with them. This interaction is likely a probing attempt, scan, or attack. To understand how attackers operate, honeypots record and analyze all system activity. This information has been crucial for developing stronger security defenses in both universities and businesses.

2. Object and subject of research

In cloud environments, secure container orchestration is essential. Orchestration systems must not only manage container lifecycles but also implement robust security measures to mitigate threats and vulnerabilities. Container orchestration platforms provide automated management, including the intelligent allocation of containers to the most suitable hosts within a cluster. Additionally, they ensure high availability by automatically restarting containers whenever they encounter issues such as crashes or unresponsive behavior.

By providing valuable insights into attacker behavior, honeypots allow information security teams to proactively defend against sophisticated attacks that traditional security measures, such as firewalls, may miss. This has led to widespread adoption of honeypots by organizations worldwide as a crucial element of their overall security strategy. A cyber honey trap, akin to a decoy, is a computer system intentionally designed to attract and capture cyberattacks. By mimicking legitimate targets and employing infiltration techniques, it lures intruders into a controlled environment, allowing security professionals to gather valuable intelligence on attackers, their methods, and their objectives. The primary goal is to enhance a company's ability to detect intrusions and respond to security incidents effectively. This involves identifying anomalous and malicious network traffic, enabling security teams to analyze and block harmful traffic flows within the communication network. While many container platforms rely on external tools for security monitoring and risk mitigation, this research delves into the core functionalities of containerization, highlighting the advantages of systems like Kubernetes.

This research explores the use of lightweight container platforms and their orchestration capabilities to enhance security through the streamlined deployment and management of honeypot systems. The focus lies in the formulation of robust security strategies specifically designed for containerized architectures operating within cloud environments. The main objectives of the study include investigating the effectiveness of current practices for using container security honeypots; identification and analysis of risk factors for honeypot integration within lightweight container platforms; develop effective container orchestration strategies with honeypot deployment. Consequently, this study aims to introduce an experimental methodology for the deployment of honeypots within the context of lightweight container platforms with the goal of bolstering system security.

3. Target of research

Container orchestration automates the deployment, scaling, and management of containers, enabling efficient and reliable application delivery. Container orchestration platforms, such as Kubernetes, provide automated mechanisms for deploying, scaling, and managing containerized applications within and across machine clusters or cloud infrastructures. These services incorporate functionalities such as load balancing, service discovery, health monitoring, and automatic scaling to guarantee robust and efficient application performance.

Just as with any other software, container images can harbor vulnerabilities. Therefore, fundamental cybersecurity practices such as generating an SBOM, identifying embedded secrets, and classifying all image layers continue to be of paramount importance. The complexity arises from the dynamic nature of containerized environments, characterized by a high container density and frequent updates, often occurring multiple times weekly due to the adoption of DevOps principles and agile development methodologies.

Frequent updates, while essential for innovation and bug fixes, inherently increase the risk of introducing security vulnerabilities, particularly in complex environments hosting thousands of containers. This is compounded by the inherent security challenges associated with managing the container runtime itself. Traditional security tools were not built for the dynamic nature of containerized environments, hindering the establishment of a secure baseline. These legacy tools frequently lack the visibility to inspect the inner workings of containers, leaving cybersecurity teams to grapple with application security issues that traditional firewalls cannot address.

Securing access to container orchestration platforms like Kubernetes is crucial. Implementing strong access control measures, including allowlisting, is essential to prevent threats from overly permissive accounts, network attacks, and the potential for unauthorized movement within the cluster, mirroring the security principles of established IT environments. Securing Kubernetes deployments requires a holistic approach that addresses a wide array of security concerns. This includes measures to protect the underlying infrastructure, container images, network traffic, and application data, ensuring the integrity and confidentiality of the entire system.

4. Literature analysis

Honeypots can be specifically targeted by attackers, which allows for the investigation of security weaknesses or the testing defenses against those weaknesses. Attacking honeypots provides valuable information for cybersecurity professionals, but it doesn't necessarily pose a direct threat since the data is often not real [8, 9]. Traditionally, intrusion detection systems (IDS), intrusion prevention systems (IPS), and firewalls functioned as independent security tools. Honeypots, on the other hand, are seen as elements within a broader surveillance system. Their specific placement depends on the type of security requirements.

The information gathered by a honeypot depends on two key factors [10]:

- Interaction Level: Honeypots come in three types: low, medium, and high interaction. Each allows attackers varying degrees of access to the system, influencing the data captured.
- Realism: How closely the honeypot mimics a real system impacts its effectiveness. A more realistic decoy attracts attackers and provides richer data.

Low-interaction honeypots act as decoys by exposing only a limited set of functionalities that appear real to attackers, instead of offering a complete system. This restricted environment allows the honeypot to gather information about the attacker's attempts, but since they can't delve deep into the system, the information collected is limited. High-interaction honeypots are attacker magnets. They are real systems configured to appear as real targets with vulnerabilities, allowing security researchers to study attacker behavior. Although high-interaction honeypots deliver richer attacker data, their deployment and maintenance are significantly more expensive. This is because they carry a higher

risk of being misused in real attacks, such as botnets. This trade-off should be considered when choosing the right honeypot for your needs.

Firewalls and honeypots serve different purposes in network security. While firewalls act as a gatekeeper, honeypots take a more investigative approach [11]:

- Firewalls are positioned at the network edge, like a security checkpoint. They control incoming and outgoing traffic based on predefined rules, blocking unauthorized access by filtering ports and content. However, they don't deeply analyze the traffic itself.
- Honeypots, on the other hand, are deliberately vulnerable systems designed to attract attackers. By studying how attackers interact with the honeypot, security teams can gain valuable insights into their tactics and techniques.

Security systems like IDS and vulnerability scanners watch for signs of hacking attempts, malware, or other threats by analyzing communication patterns. A common challenge with IDSs is managing false alarms. Signature-based systems rely on predefined patterns to identify threats. This can lead to them missing new or evolving attacks (false negatives). On the other hand, anomaly-based systems monitor for unusual activity. While this approach can catch novel threats, it can also mistake normal network behavior for an attack (false positives). When used with IDSs, honeypots can dramatically improve the accuracy of intrusion detection alerts. By incorporating honeypots, IDSs become more effective in filtering out false positives.

In a microservices architecture, an application is built from a collection of small, self-contained services. These services communicate with each other to deliver the full functionality of the application. To achieve specific functionalities, like managing users, processing payments, or sending emails, individual microservices are created [12, 13]. These self-contained services communicate with each other using network interfaces, such as remote procedure calls (RPC) or APIs. Monolithic applications are stuck with one database, but microservices are not. Each service can utilize the most suitable database for its data model and workload [14]. With a loosely coupled system design, individual services can be scaled, deployed, managed, and updated without impacting other parts of the system. Instead of sharing the kernel directly, containers use the host machine's kernel and its features like namespaces. This allows them to isolate their processes from each other while also controlling how much CPU and memory each container can use. This approach allows for independent deployments and scaling of each microservice, thanks to their custom environments.

Traditional honeypot methods are becoming less effective as containerization and microservices gain popularity among engineers and technology professionals. These new approaches to building applications complicate the use of honeypots in the traditional manner. Traditional operating system-level virtualization is being overshadowed by containerization [15, 16]. With containerization, applications are bundled with their necessary components (like libraries) into isolated units called containers. These containers act like self-contained mini-environments, ensuring consistent execution regardless of the underlying system. Because containers utilize the same underlying operating system and components as the host machine, they have a smaller footprint and require fewer resources compared to virtual machines [17]. Microservices, small and independent building blocks, are deployed using container technology. This approach creates programs that are more adaptable and can be easily expanded upon. With this approach, we can modify or scale a function without having to develop and release a whole new version each time.

These days, Kubernetes (K8s) is the preferred tool for managing containerized applications that need to be always available, handle changing demands, and be able to bounce back from failures. Kubernetes has become the preferred platform for managing large and intricate deployments of containerized applications [18]. It offers a powerful and user-friendly way to orchestrate these applications, ensuring they run efficiently and interact seamlessly. Using the container platform, you can build clusters that are reliable, adaptable, and resilient to disruptions, ensuring your applications run smoothly. As containerization, a technology that packages applications into lightweight, portable units, has gained traction, Kubernetes has become a prominent tool for managing containerized applications [19, 20]. This has led to its widespread use in diverse application areas like Fog, Edge,

and IoT computing. MicroK8s (mK8s), K3s and minikube (MK) are all lightweight versions of Kubernetes designed to simplify setting up and running Kubernetes clusters. They achieve this by streamlining and customizing core Kubernetes components. This technology focuses on making cluster setup, operation, and upkeep easier, allowing for deployments on devices with limited resources [21, 22]. However, current performance evaluations primarily address scenarios where the devices are either inactive or under heavy workload. For container orchestration to work effectively in environments with limited resources, lightweight distributions were developed to optimize resource usage.

MicroK8s. Created by Canonical, mK8s is a simplified version of Kubernetes designed for easier use on both public and private clouds. mK8s is a lightweight and fully functional Kubernetes distribution that is ideal for resource-constrained environments, particularly those focused on the Internet of Things. By default, mK8s automatically turns on all the essential parts of Kubernetes to get your cluster up and running. You can easily enable other helpful tools, like DNS, ingress, or the metrics-server, with just one simple command. Achieving high availability, where both the control plane and datastore are replicated across multiple nodes, can be accomplished with a few configuration commands. MK8s, a Kubernetes distribution, can be installed using snap, a package manager from Canonical that isolates applications in a sandbox environment. To ensure optimal performance, it's recommended to allocate at least 4GB of memory and 20GB of storage (preferably SSD) for running mK8s.

K3s, a lightweight Kubernetes offering from Rancher, is designed for simplicity and efficiency. This fully-compliant Kubernetes distribution comes with all the essential components pre-installed, making it easy to set up a highly available and fault-tolerant cluster on your nodes. K3s is ideal for running applications on low-resource environments. Deployment is achieved through a single, lightweight binary that incorporates all necessary dependencies. Like mK8s, Rancher opts for a different data storage solution: SQLite3. This, along with removing unnecessary components, keeps the overall footprint minimal. To reduce memory usage, K3s takes a different approach to organizing the control plane. Instead of having separate services for each component, K3s combines everything into a single process on both the master (server) and worker (agent) nodes. K3s offers a streamlined installation process through a shell script. This script allows for flexible deployment, enabling the application to function as either a server or an agent node. Additionally, scaling the cluster for high availability is simplified. New worker nodes can be seamlessly integrated with just a few commands. To run this application, you will need a computer with at least 1 vCPU and 512MB of memory [23].

Minikube makes it easy to run a Kubernetes cluster directly on your personal computer, whether you're using macOS, Linux, or Windows. Designed specifically for developers, it aims to be the go-to tool for building and testing applications on Kubernetes locally. Minikube also strives to support as many Kubernetes features as possible, ensuring a development experience that closely reflects a real-world Kubernetes environment. Minikube offers a local, single-node Kubernetes environment for developers. It includes all the core Kubernetes services like API server, controller manager, and scheduler, allowing you to test and learn container orchestration without needing a complex setup. Minikube simplifies managing your Kubernetes cluster with built-in commands. While Minikube is an excellent tool for learning and development, it's important to remember that it's a single-node cluster. This makes it unsuitable for production environments. For production use cases, a multi-node Kubernetes cluster is necessary.

Researchers actively promote honeypots as a method for gathering threat intelligence. These systems reveal the tools, tactics, and procedures used by threat actors, providing valuable information for defenders. The following experiment is flexible and can be adapted to various research objectives. The widespread adoption of microservices and Kubernetes has outpaced current security research efforts. To address this gap, there is a critical need for new research focused on identifying threats and vulnerabilities in these systems. While microservices-based honeypots are a relatively new concept, ongoing research aims to develop innovative honeypots that can adapt to the ever-changing technological landscape.

HoneyKube is a medium-fidelity honeypot built by C. Gupta [24] for Google Kubernetes Engine. Their discussion focused on the security risks associated with microservice architectures. Microservices are a popular design approach used in container-based systems. Gupta's research examined both adversarial activity and vulnerabilities within Kubernetes, focusing on exposed services under attack. Instead of focusing on technical details, this project (HoneyKube) aimed to understand the attacker's overall goals. Using Google Kubernetes Engine (GKE)'s security features, it created a controlled environment to observe how attackers might target a microservices architecture. To understand the threats faced by unhardened container orchestration deployments, we created a honeypot environment that simulated vulnerabilities attackers might exploit. This honeypot captured data on attacker behavior within the simulated environment.

Inspired by the Software-as-a-Service (SaaS) model, J.H. Jafarian and A. Niakanlahiji proposed a concept called "Honeypot-as-a-Service" (HaaS) [25]. This approach aims to make honeypots more accessible to small and medium-sized enterprises (SMEs) by delivering them as a service, eliminating the need for in-house setup and maintenance. This can significantly improve the cybersecurity posture of SMEs by providing them with an easy-to-use tool for threat detection and analysis. The designers prioritized creating a honeypot that was not only easy to integrate and adapt (scalable and flexible plug-and-play service) but also highly convincing to attackers. It needed to appear identical to real production servers to effectively lure attackers. The generated honeypots underwent rigorous evaluation by security professionals.

In a study by Kelly et al., researchers investigated how malicious activity on different cloud platforms changes as companies adopt remote work models [26]. They hypothesized that the increase in cloud use would be accompanied by a rise in cyberattacks. Both this study and others have observed malicious activities targeting poorly deployed container orchestration services. To investigate this further, the researchers deployed a series of pre-packaged honeypots in the form of Docker containers on three major cloud hosting providers: AWS, Azure, and GKE. The honeypots deployed in the study are low to medium-interaction honeypots, which are essentially Docker containers configured to mimic real services and collect data about attacker interactions. One of the key objectives of their research was to analyze the impact of cloud provider and geographical location on container attack methods. The researchers sought to determine if container vulnerabilities are exploited differently based on the deployment environment (cloud provider and region).

5. Research methods

This research leverages virtual clusters and containers to create isolated and scalable environments for experimentation. Virtual clusters provide a way to create isolated Kubernetes environments within a single physical cluster, allowing for improved resource utilization and multi-tenancy. We support the creation of multiple virtual clusters. Each cluster operates in a self-contained environment, completely isolated from other virtual clusters and the host cluster itself. Virtual clusters provide isolated spaces where sensitive elements can be securely managed. This allows for controlled exposure, ensuring only authorized entities can access them, unlike their potential equivalents on the host cluster which might be more vulnerable. This strategy allows us to safely expose components such as Kubernetes infrastructure and maintenance tools to attackers, ensuring the honeypot remains operational to gather valuable information.

We deployed honeypots on several popular lightweight container platforms (LCPs), including microk8s, K3s, and minikube. High-interaction decoys closely mimic real production environments, offering valuable insights into attacker behavior. However, they require more resources to maintain and necessitate stricter security measures for the testing environment. Considering operating cost efficiency, we analyze the image processing of the following Docker containers:

- ADBHoney [27]. ADB (Android Debug Bridge) is a tool that lets you communicate with Android devices like phones, TVs, and DVRs, whether they're physical devices you plugged in or running as simulations on your computer. The tool offers a suite of commands, such as

adb shell and adb push, to aid developers in debugging applications and transferring data (content) to connected devices. This process usually involves a USB cable and incorporates robust authentication and security features. This project develops a simple honeypot specifically targeting port 5555. It aims to attract and analyze malware automatically, without requiring extensive user interaction. This method helps us identify the malicious software attackers distribute to vulnerable systems with open port 5555.

- DDosPot [28]. DDosPot is a honeypot used to track and monitor DDoS attacks sent over multiple devices (Multicast UDP). This platform uses UDP (User Datagram Protocol) to create decoy servers that act like honeypots. These honeypots help monitor and track DDoS attacks. The system allows easy addition of these honeypot servers, called "pots," using simple plugins: DNS server, NTP server, SSDP server, CHAREN server and Random/mock UDP server. Communication applications utilize ports 19, 123, 1900, and 53 to function.
- ssh honeypot (cowrie). The cowrie lure is a high-interaction honeypot designed to mimic Telnet and SSH services. It operates by logging brute-force attacks targeting these protocols. In simpler terms, it acts as a decoy login system that records any hacking attempts made against SSH and Telnet, allowing security personnel to track and respond to potential security threats. Cowrie deceives attackers by presenting a fake file system and simulated terminal service, which compels them to interact with the honeypot instead of a real system. Cowrie honeypots do not aim to understand attacker motivations. Instead, they record attacker interactions with decoy services (like SSH or Telnet) to analyze their techniques and tools. This allows defenders to learn about attacker behavior and improve security measures [29].

Figure 1 shows a diagram of the placement and interaction of various honeypots in the microservice architecture of the LCP virtual cluster.

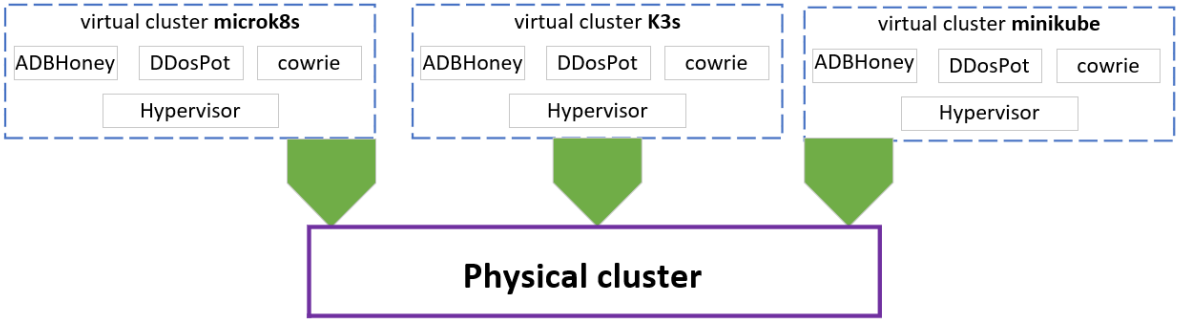


Figure 1. Model of the placement of various honeypots in the microservice architecture.

We will analyze IaaS attacker activity patterns in various geographic regions. To ensure a wider reach, we opted for Amazon Web Services (AWS) as our cloud provider. This platform grants us the control to scatter decoy servers across diverse geographic locations, along with assigning them specific IP address ranges. Out of the available AWS services for hosting, Amazon EKS was selected to orchestrate the Docker containers. Amazon EKS provides a dedicated Kubernetes control plane for each cluster, ensuring isolation between clusters and AWS accounts.

Amazon Managed Service for Prometheus offers a built-in, agentless collector that automatically collects Prometheus metrics from your Amazon EKS workloads. This frees you from the hassle of managing monitoring agents yourself. EKS metrics are automatically discovered, eliminating the need to install additional agents on your system. The service operates across multiple Availability Zones (AZs) to increase reliability and security [30]. Figure 2 illustrates the architecture of Prometheus interaction with AWS services.

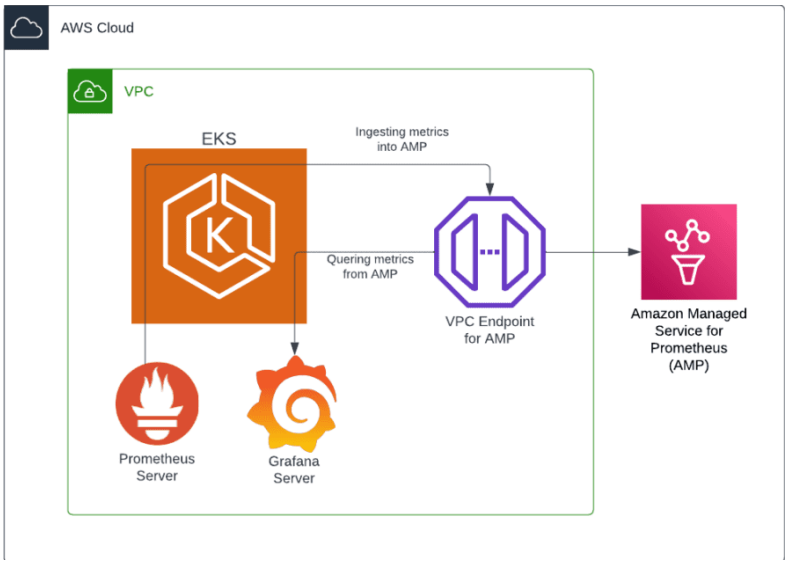


Figure 2: Amazon Managed Service for Prometheus.

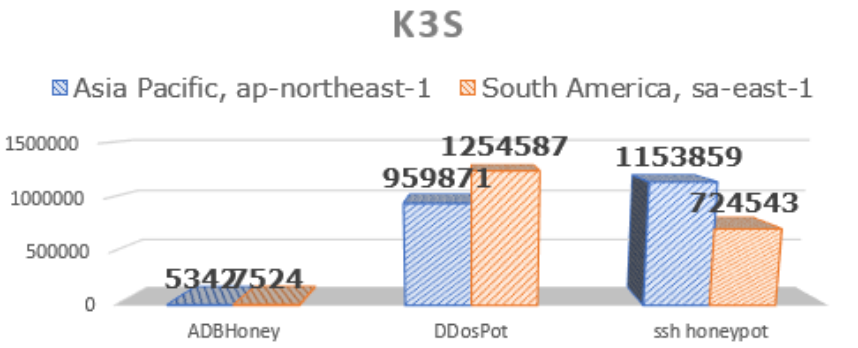
Amazon Managed Service for Prometheus uses Elastic Network Interfaces (ENIs) to collect metrics. It creates a separate ENI for every subnet you specify during scraper provisioning. The collected metrics are securely sent (remote write) to your Amazon Managed Service for Prometheus workspace within your Virtual Private Cloud (VPC) using a VPC endpoint. This keeps the data on your private network. Amazon Managed Grafana automatically uses PrivateLink VPC endpoints to access metrics directly from Amazon Managed Prometheus, ensuring secure communication within your VPC. With this system, the information we collect remains safe within our network and never goes out onto the public internet. Our highly available collector offers a secure and reliable way to monitor your systems, eliminating the need for manual agent provisioning.

6. Research results

During the experiment, each of the virtual clusters (microk8s, K3s, minikube) consists of three nodes with different honeypots (ADBHoney, DDosPot, ssh honeypot) hosted on different location (region 1: Asia Pacific, Tokyo, ap-northeast-1; region 2: South America, Sao Paulo, sa-east-1).

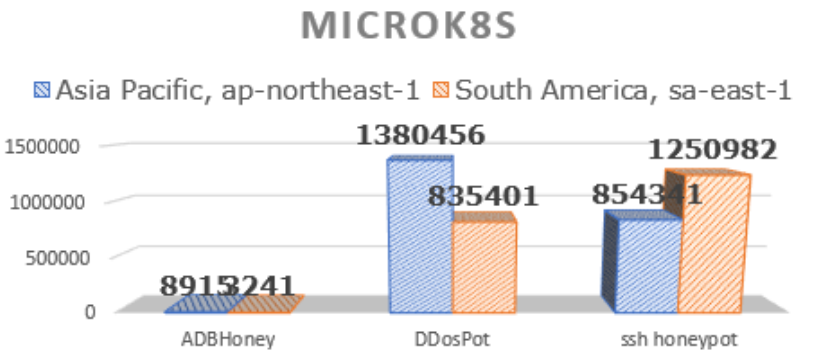
We established a controlled environment to evaluate resource consumption, guaranteeing reproducible, understandable, and consistent results. The virtual cluster consisted of three virtual machines, each running Ubuntu 22.04 and equipped with 2 virtual processors, 4 GB of memory, and a fast 50 GB solid-state drive. This environment uses a single on-premises physical host to run all virtual machines. KVM acts as the hypervisor, providing virtualization capabilities, while container handles container runtime. The host machine is equipped with a t3.2xlarge 8vCPU, 64GB memory and a fast SSD. Instead of isolating the containers on their own network, they share the network directly with the host system. This means they don't get their own IP address and use the system's address instead. In total, the system was operational 24 hours a day for 10 days, from 02th October 2024 to 12th October 2024.

Once you've set up cloud honeypot instances in two separate regions, access the front-end interface by opening a web browser and going to the following address: https://<external IP address>:64297. Then, log in using the credentials you created during the honeypot setup process. The overall number of attacks on both regions is shown in Figure 3. These attacks began after the IP address was publicly exposed on the internet. If exposing an IP address leads to immediate attacks, it suggests the presence of bots or automated scans continuously targeting cloud IP ranges. Before enabling accessibility features for operations, it is vital to secure our cyber assets. This point is further emphasized by the attack statistics gathered through our experiments, illustrated in Figures 3-5.



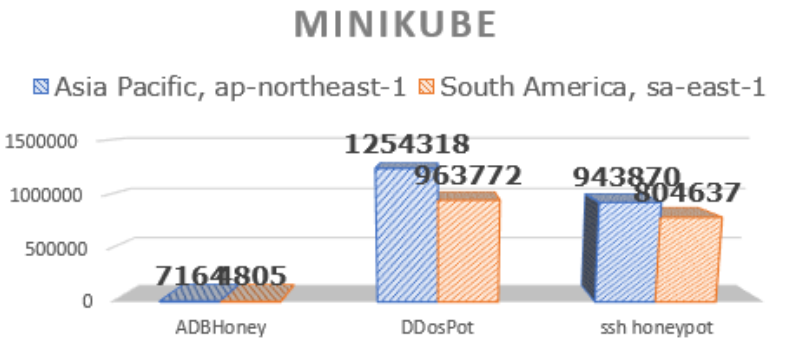
	ADBHoney	DDosPot	ssh honeypot
Asia Pacific, ap-northeast-1	5342	959871	1153859
South America, sa-east-1	7524	1254587	724543

Figure 3. Attack statistics for Honeypot instances in lightweight container platforms K3s.



	ADBHoney	DDosPot	ssh honeypot
Asia Pacific, ap-northeast-1	8915	1380456	854341
South America, sa-east-1	3241	835401	1250982

Figure 4. Attack statistics for Honeypot instances in lightweight container platforms microk8s.



	ADBHoney	DDosPot	ssh honeypot
Asia Pacific, ap-northeast-1	7164	1254318	943870
South America, sa-east-1	4805	963772	804637

Figure 5. Attack statistics for Honeypot instances in lightweight container platforms minikube.

Honeypots serve as decoys, attracting and analyzing attempted attacks. They can't predict entirely unknown threats, but they effectively detect them. Early detection can prevent attackers from reaching your critical systems. Instead of focusing on detecting attacks, let's discuss proactive security measures that can protect your infrastructure. This includes implementing strong firewall rules, creating complex passwords, and utilizing encryption, digital signatures, and authentication technologies.

7. Conclusions

Our research introduced a cutting-edge honeypot that leverages a microservices architecture for increased flexibility and scalability. The use of virtual clusters in honeypots and security research has the potential to be highly beneficial. This is a developing field with exciting possibilities for both future advancements and industry applications. This research explores the security risks of container orchestration systems publicly accessible on the internet. We first measured the prevalence of such exposed systems through an empirical internet-wide study. Following this, we constructed a low-interaction honeypot to observe real-world attacks and gain insights into the attackers' tactics, techniques, and procedures (TTPs). We are investigating the development of a deception technology that utilizes profiling of lightweight container orchestration platforms deployed on virtual clusters.

The first step was identifying the most appropriate technologies for deploying honeypots. We evaluated different options and ultimately chose virtual machines and containers as the main solutions. To gain better control and scalability for my honeypots, we opted for lightweight container orchestration with Kubernetes. To improve scalability and geographic reach, we implemented lightweight Kubernetes distributions that support multi-node deployments in different locations. This strategic decision led to efficient resource allocation and seamless honeypot deployment in diverse settings.

By deploying honeypots in various locations, we can collect a richer set of attack data. This variety helps us identify attack patterns that might be unique to certain regions. These insights are crucial for constantly developing better cybersecurity strategies. The text emphasizes the importance of innovative solutions in strengthening digital ecosystems against emerging threats. This is achieved by offering a nuanced understanding of malicious activities across different environments. While this thesis paves the way for the use of distributed honeypots, there is significant room for further investigation and development:

- safeguard communication between honeypot networks: design secure communication methods for connecting distributed honeypot clusters; prioritize data privacy (confidentiality), accuracy (integrity), and constant accessibility (availability) during communication; this will block unauthorized access and protect against data leaks.
- the power of distributed honeypots: unlock the potential of vast honeypot data by training machine learning models to identify and neutralize cyber threats in real-time.
- automate data filtering for honeypots: leverage machine learning or rule-based systems to identify non-sensitive information captured by honeypots. This allows for selective storage and analysis, focusing on valuable insights while respecting user privacy.

The research emphasizes the importance of container security to ensure the secure management and deployment of containerization and orchestration capabilities, thereby enhancing overall system reliability and reducing risks. This streamlined visualization process simplifies resource utilization monitoring while maintaining accuracy. Containerized applications within the cloud system benefit from load balancing and fault tolerance through the implementation of in-built secure protocols. The dominance of microservices makes strong security measures essential. We believe our contribution will lead to a significant improvement in the security landscape for microservices applications. We believe our research and the collected data will be instrumental in developing new security solutions for microservices architectures.

References:

1) Pahl, Claus; Jamshidi, Pooyan; Zimmermann, Olaf (2020). Microservices and Containers. Software Engineering 2020. DOI: 10.18420/SE2020_34.

2) Liu, G. et al. (2020). Microservices: Architecture, container, and challenges. in 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C). 629–635. https://doi.org/10.1109/QRS- C51114.2020.00107.

3) Wan, X., Guan, X., Wang, T., Bai, G. & Choi, B.-Y. (2018). Application deployment using microservice and docker containers: Framework and optimization. J. Netw. Comput. Appl. 119, 97–109. https://doi.org/10.1016/j.jnca. 2018. 07. 003.

4) Batchu, R. K. & Seetha, H. (2021). A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning. Comput. Netw. 200, 108498. https:// doi. org/ 10. 1016/j. comnet. 2021. 108498.

5) Halvorsen, J., Waite, J. & Hahn, A. (2019). Evaluating the observability of network security monitoring strategies with tomato. IEEE Access 7, 108304–108315. https:// doi. org/ 10. 1109/ ACCESS. 2019. 29334 15.

6) Zhi Li,Weijie Liu, Hongbo Chen, XiaoFengWang, Xiaojing Liao, Luyi Xing, Mingming Zha, Hai Jin, and Deqing Zou. (2022). Robbery on DevOps: Understanding and Mitigating Illicit Cryptomining on Continuous Integration Service Platforms. In 2022 IEEE Symposium on Security and Privacy (SP). IEEE, San Francisco, CA, USA, 2397–2412. https://doi.org/10.1109/SP46214.2022. 9833803

7) Shafiq, M., Tian, Z., Sun, Y., Du, X. & Guizani, M. (2020). Selection of effective machine learning algorithm and Bot–IoT attacks traffic identification for internet of things in smart city. Future Gener. Comput. Syst. 107, 433–442. https://doi. org/10.1016/j.future.2020.02. 017.

8) Franco, J., Aris, A., Canberk, B. & Uluagac, A. S. (2021). A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems. IEEE Commun. Surv. Tutorials 23, 2351–2383. https:// doi. org/ 10. 1109/ COMST. 2021. 3106669

9) Y. Sun, Z. Tian, M. Li, S. Su, X. Du and M. Guizani. (2021). Honeypot Identification in Softwarized Industrial Cyber–Physical Systems. *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5542-5551, Aug. 2021, https://doi.org/ 10.1109/TII.2020.3044576

10) The Honeynet Project: Spam Honeypot with Intelligent Virtual Analyzer. Available at: https://www.honeynet.org/

11) Windows Container Malware Targets Kubernetes Clusters. Available at: https://threatpost.com/windows-containers-malware-targets-kubernetes/166692/

12) Rashid, S. M., Haq, A., Hasan, S. T., Furhad, M. H., Ahmed, M., & Ullah, A. B. (2022). Faking smart industry: Exploring cyber-threat landscape deploying cloud-based honeypot. Wireless Networks, 1-15. Advance online publication. https://doi.org/10.1007/s11276-022-03057-y

13) Jay Chen. (2020) Attacker's Tactics and Techniques in Unsecured Docker Daemons Revealed. Available at: https://unit42.paloaltonetworks.com/attackers-tactics-andtechniques-in-unsecured-docker-daemons-revealed/

14) Docker. Run the Docker daemon as a non-root user (Rootless mode). Available at: https://docs. docker.com/engine/security/rootless/#known-limitations

15) Kubernetes. Good practices for Kubernetes Secrets. Available at: https:// kubernetes.io/docs/concepts/ security/secrets-good-practices/ Section: docs.

16) Kubernetes. Pods. Available at: https://kubernetes.io/docs/concepts/workloads/pods/

17) Kubernetes. Production-Grade Container Orchestration. Available at: https://kubernetes.io/

18) Andrew Martin and Michael Hausenblas. (2021). Hacking Kubernetes: threat-driven analysis and defense. O'Reilly Media, Sebastopol, CA. 300. ISBN 9781492081739.

19) Niels Provos and Thorsten Holz. (2007). Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley Professional PTG, Boston, Massachusetts. 440. ISBN 0321336321.

20) Akond Rahman, Shazibul Islam Shamim, Dibyendu Brinto Bose, and Rahul Pandita. (2023) Security Misconfigurations in OpenSource Kubernetes Manifests: An Empirical Study. ACM Transactions on Software Engineering and Methodology TBD, 37. https://doi.org/10.1145/ 3579639 Publisher: ACM New York, NY.

21) Ferreira, A.P., Sinnott, R. (2019). A performance evaluation of containers running on managed kubernetes services. In: 2019 IEEE International Conference on Cloud Computing

Technology and Science (CloudCom), pp. 199-208. IEEE. https://doi.org/10.1109/cloudcom. 2019.00038

22) Goethals, T., Turck, F.D., Volckaert, B. (2019). FLEDGE: Kubernetes compatible container orchestration on low-resource edge devices. In: Internet of Vehicles. Technologies and Services Toward Smart Cities, pp. 174-189. Springer International Publishing. https://doi.org/ 10.1007/978-3-030-38651-1 16

23) Kristiani, E., Yang, C.T., Huang, C.Y., Wang, Y.T., Ko, P.C. (2020) The implementation of a cloud-edge computing architecture using OpenStack and kubernetes for air quality monitoring application pp. 1-23. https://doi.org/10.1007/s11036-020-01620-5

24) C. Gupta. (2021). HoneyKube: designing a honeypot using microservices-based architecture. Ph.D. Dissertation. University of Twente. Available at: http://essay.utwente.nl/ 88323/

25) Jafarian, Jafar Haadi & Niakanlahiji, Amirreza. (2020). Delivering Honeypots as a Service. DOI: 10.24251/HICSS.2020.227. Available at: http://hdl.handle.net/10125/63966

26) Christopher Kelly, Nikolaos Pitropakis, Alexios Mylonas, Sean McKeown, and William J. Buchanan. (2021). A Comparative Analysis of Honeypots on Different Cloud Platforms. Sensors 21, 7 , 2433. https://doi.org/10.3390/ s21072433

27) Github : huuck / adbhoney. Available at: https://github.com/huuck/ADBHoney

28) DDosPot. Available at: https://github.com/aelth/ddospot

29) Cowrie Project. Available at: https://github.com/cowrie/cowrie

30) Inc. Amazon Web Services. Security in Amazon EKS - Amazon EKS. Available at: https://docs.aws.amazon. com/eks/latest/userguide/security.html