

Article

Microservice-Oriented Cyber Deception Platform with Containerized Honeypots and Real-Time Telemetry

Muhammad Shahzad and Muhsin Hassanu Saleh *

School of Computing, Engineering and Physical Sciences University of West of Scotland, London Campus,
London E14 2BA, UK; b01798648@studentmail.uws.ac.uk

* Correspondence: muhsin.hassanu@uws.ac.uk

Abstract

The growing reliance on cyber deception as a defensive mechanism has revealed persistent limitations in existing deception infrastructures, particularly in their ability to scale, adapt, and provide continuous observability under realistic adversarial workloads. Conventional honeypot deployments are predominantly monolithic and statically configured, which constrains their responsiveness to dynamic attack conditions and limits their applicability in contemporary distributed environments. This work presents a microservice-oriented cyber deception platform that reconceptualizes deception infrastructure as a composition of loosely coupled, independently deployable services. The platform integrates containerized honeypots, a lightweight API-driven orchestration layer, and a centralized telemetry pipeline to enable rapid instantiation, dynamic reconfiguration, and high-resolution monitoring of attacker interactions. Unlike prior approaches that treat deployment, orchestration, and monitoring as separate concerns, the proposed design explicitly unifies these components within a single, measurable system architecture. To support principled reasoning about system behaviour, the paper introduces first-order analytical models that characterize deployment latency, resource utilisation, telemetry throughput, and operational cost as functions of attacker concurrency. These models are not intended as exact predictors, but as tractable abstractions that enable interpretation of system performance and guide capacity planning. Model parameters are empirically derived and validated through controlled experimentation. Evaluation is conducted within a reproducible cyber-range environment using scripted adversarial workloads that emulate reconnaissance, authentication attempts, and sustained interactive sessions. The results indicate that containerised deployment reduces instantiation latency to approximately 1.2 s under warm-start conditions, compared to tens of seconds for virtual machine-based baselines. Resource utilisation exhibits approximately linear scaling under moderate concurrency, while the telemetry pipeline sustains ingestion rates exceeding 18,000 events per minute without observable loss. Stress testing further reveals that telemetry processing, rather than orchestration, constitutes the primary scalability bottleneck. These findings suggest that microservice-based architectures can provide a viable and extensible infrastructure substrate for cyber deception, supporting both operational deployment and integration with higher-level adaptive and learning-based defence mechanisms. The contribution of this work lies not in introducing new deception strategies, but in enabling their practical realisation through a scalable and observable system design.

Keywords: cyber deception; honeypots; microservices; containerization; scalability; observability; telemetry; API-driven orchestration; behavioral analytics; adaptive defense; cyber-range evaluation



Academic Editor: Martin Gilje Jaatun

Received: 16 February 2026

Revised: 8 May 2026

Accepted: 25 May 2026

Published: 2 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons](https://creativecommons.org/licenses/by/4.0/)

[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Cyber deception has emerged as a significant defensive approach for mitigating advanced cyber threats by misleading adversaries, diverting malicious activity, and enabling the collection of high-fidelity behavioural intelligence. In practice, such threats often manifest as multi-stage and automated attack campaigns involving reconnaissance, exploitation, and sustained interaction with target systems. Traditional deception mechanisms, including honeypots and decoy services, have been widely used to observe attacker behaviour and delay compromise of production environments [1–3]. However, the increasing scale, automation, and adaptability of modern attacks have exposed fundamental limitations in existing deception deployments, particularly in terms of scalability, observability, and operational flexibility.

Early honeypot systems were typically deployed as standalone or monolithic environments, manually configured and statically bound to specific network contexts [1,4]. While effective in controlled or small-scale settings, such approaches do not readily support rapid reconfiguration, elastic scaling, or continuous monitoring under dynamic attack conditions. Consequently, deception assets are frequently underutilised or quickly identified and bypassed by adversaries employing reconnaissance and environment validation techniques [5,6]. These limitations become more pronounced in contemporary enterprise and cloud environments, where infrastructure is inherently dynamic and attacker activity may involve large numbers of concurrent connections driven by automated tooling.

Recent research has highlighted the need for deception infrastructures that can operate effectively under realistic threat models, supporting both scalability and adaptability [6–8]. In parallel, advances in containerisation and microservice architectures have transformed the design of distributed systems by enabling modularity, rapid deployment, and fine-grained resource management [9,10]. These paradigms have been widely adopted in cloud-native applications and, to a limited extent, in security tooling, including containerised honeypots (e.g., Docker-based deployments) and modern telemetry mechanisms such as eBPF-based monitoring. However, existing approaches typically address these capabilities in isolation and do not provide a unified framework that integrates scalable deployment, orchestration, and observability within a single, analytically grounded system.

A further limitation of many deception platforms lies in their insufficient support for real-time telemetry and behavioural monitoring. Effective cyber deception requires not only the deployment of convincing decoy environments but also the ability to capture, process, and analyse large volumes of interaction data generated by attackers [3,7]. In the absence of robust telemetry pipelines, behavioural evidence may be lost, delayed, or rendered unsuitable for downstream analysis and decision-making. Although centralised logging frameworks such as the ELK (Elasticsearch–Logstash–Kibana) stack have demonstrated strong capabilities for high-throughput data ingestion and visualisation in other security domains [11], their integration into cyber deception platforms has received comparatively limited empirical evaluation, particularly under adversarial workloads.

Motivated by these gaps, this paper presents a microservice-oriented cyber deception platform designed to support scalable deployment, dynamic orchestration, and real-time telemetry collection of containerised honeypots. The proposed approach reconceptualises deception infrastructure as a composition of loosely coupled services spanning virtualisation, container management, orchestration, telemetry, and monitoring layers. This modular design enables rapid instantiation and reconfiguration of deception assets, fine-grained control over resource utilisation, and continuous visibility into attacker interactions. The platform is implemented using widely adopted virtualisation and container technologies and evaluated within a controlled cyber-range environment to ensure reproducibility and experimental rigor.

In addition to the system architecture and implementation, this paper introduces first-order analytical models to characterise key operational properties of the platform, including deployment latency, resource utilisation, telemetry throughput, and operational cost under varying attacker loads. These models are intended as tractable abstractions that support reasoning about system behaviour and capacity planning, rather than exact predictive tools. Their parameters are empirically derived and validated using controlled experiments, enabling alignment between analytical insights and observed system performance.

This work is complementary to existing research on adaptive and AI-driven cyber deception, including closed-loop frameworks that integrate detection, deception, and behavioural attribution. While such approaches focus on autonomous decision-making, they implicitly assume the availability of a scalable and observable execution substrate. The contribution of this paper lies in providing that substrate: a deployable, measurable, and extensible microservice-based infrastructure that enables the practical realisation and evaluation of higher-level adaptive defence mechanisms.

Contributions

This paper makes the following contributions to the design and evaluation of scalable cyber deception systems:

- **A microservice-oriented architecture for cyber deception platforms.**
We present a modular system architecture that reconceptualises traditional monolithic honeypot deployments as a composition of loosely coupled microservices spanning virtualisation, container management, orchestration, telemetry, and monitoring layers. This design enables independent scaling, improved fault isolation, and flexible system evolution, addressing structural limitations of tightly coupled deception systems.
- **Containerized honeypot deployment with API-driven orchestration.**
We design and implement a lightweight control plane that exposes programmatic interfaces for the lifecycle management of containerised honeypots, including deployment, reconfiguration, and teardown. This abstraction decouples deception strategy from infrastructure control, enabling integration with external decision-making components such as adaptive or learning-based controllers.
- **Integrated real-time telemetry and observability pipeline.**
We develop a centralised telemetry architecture capable of high-throughput ingestion, indexing, and visualisation of behavioural data generated by attacker interactions. The proposed pipeline provides continuous visibility into both system-level metrics and adversarial activity, enabling operational monitoring, forensic analysis, and data-driven evaluation of deception effectiveness.
- **Formal analytical models for performance and scalability.**
We develop analytical models that characterize key operational properties of the platform, including honeypot deployment latency, CPU and memory utilisation, telemetry ingestion throughput, and operational cost as a function of attacker concurrency. These models provide a principled basis for reasoning about scalability limits, capacity planning, and performance trade-offs in practical deployments.
- **Empirical validation within a controlled cyber-range environment.**
We conduct a systematic experimental evaluation within a controlled cyber-range environment using reproducible adversarial workloads. The evaluation includes a baseline comparison with virtual machine-based honeypot deployment, as well as stress testing to identify system bottlenecks. The results demonstrate significant improvements in deployment responsiveness, predictable resource scaling, and sustained telemetry ingestion under concurrent workloads.

- A deployable foundation for higher-level adaptive defence mechanisms. By focusing on scalability, observability, and measurable system behaviour, the proposed platform provides a practical foundation for higher-level adaptive and autonomous defence mechanisms. In particular, it enables integration with reinforcement learning-based controllers, behaviour-driven attribution systems, and closed-loop deception frameworks that require responsive and observable execution environments.

The remainder of this paper is organised as follows. Section 2 reviews related work on cyber deception platforms, container-based security systems, and telemetry-driven monitoring. Section 3 presents the architecture of the proposed platform. Section 4 introduces the analytical performance models. Section 5 describes the implementation and experimental setup. Section 6 reports and analyses the experimental results. Section 7 discusses limitations and practical implications, and Section 8 concludes the paper and outlines directions for future work.

2. Related Work

Cyber deception has been extensively studied from the perspectives of attacker engagement, strategic interaction, and behavioural intelligence extraction. However, comparatively less attention has been devoted to the infrastructure-level design of deception systems, particularly in terms of scalability, orchestration, and observability under realistic adversarial workloads. This distinction is critical, as the effectiveness of deception strategies is inherently constrained by the capabilities of the underlying system used to deploy and monitor deception assets. Existing research can be broadly categorised into three intersecting domains: (i) honeypot-based deception frameworks, (ii) containerisation and microservice-based system design in security contexts, and (iii) telemetry-driven monitoring and data collection infrastructures. While each of these domains has contributed to important building blocks, they are typically addressed in isolation, resulting in systems that either lack deployment flexibility, provide limited observability, or fail to scale under concurrent attack conditions. Honeypot-based systems have historically focused on interaction fidelity and behavioural data collection, often at the expense of scalability and automation. Conversely, modern containerised and cloud-native approaches enable rapid deployment but frequently abstract away operational constraints, providing limited insight into system performance under adversarial load. Similarly, telemetry frameworks offer high-throughput data ingestion and analysis capabilities, yet their integration into deception platforms remains largely ad hoc and insufficiently evaluated. As a result, there remains a gap between functional deception capabilities and operationally viable infrastructure, particularly in environments characterised by high concurrency, dynamic reconfiguration requirements, and continuous monitoring demands. This gap motivates the need for a unified architectural approach that explicitly integrates deployment, orchestration, and telemetry within a measurable and scalable system design. The following subsections examine these research directions in detail, with a focus on identifying architectural and methodological limitations that inform the design of the proposed microservice-oriented cyber deception platform.

2.1. Honeypots and Deception Frameworks

Cyber deception has traditionally been grounded in the use of honeypots and decoy systems to attract, monitor, and analyse malicious activity while isolating adversaries from production assets. Early foundational work conceptualised honeypots as controlled environments designed to lure attackers and collect behavioural data for forensic analysis and defensive intelligence generation [1,2]. High-interaction honeypots, in particular, provide realistic system environments that allow adversaries to execute commands and

interact with services, thereby enabling detailed observation of tactics, techniques, and procedures. However, this increased behavioural fidelity comes at the cost of higher deployment complexity, increased resource consumption, and elevated operational risk [3].

Subsequent research extended these concepts toward distributed and coordinated deception infrastructures. Large-scale honeynet architectures and systems such as HoneyGrid demonstrated that multiple deception assets could be deployed across network environments to improve visibility into widespread and automated attacks [4,5]. While these approaches improved coverage and data collection, they were typically built upon tightly coupled system designs, custom management scripts, and static configurations. As a consequence, their ability to scale dynamically or adapt to changing attack conditions remained limited. In particular, the reliance on manual configuration and bespoke orchestration mechanisms constrained responsiveness and introduced operational overhead, making such systems difficult to manage in dynamic environments.

More recent work has shifted the focus from passive observation toward active and adaptive cyber deception. Approaches based on game theory, signalling models, and reinforcement learning have been proposed to optimise deception strategies by dynamically selecting actions in response to observed attacker behaviour [6–8]. These methods demonstrate the potential for intelligent and context-aware deception; however, they frequently abstract away the underlying system infrastructure. In many cases, deployment, reconfiguration, and monitoring of deception assets are treated as negligible-cost operations, which does not reflect practical constraints in real-world environments.

This abstraction introduces a critical disconnect between *deception strategy design* and *system-level feasibility*. Without scalable, observable, and automatable infrastructure, the effectiveness of adaptive deception strategies may be significantly limited by deployment latency, resource constraints, and insufficient visibility into attacker interactions. As such, despite advances in behavioural modelling and decision-making, existing work provides limited guidance on how to realise these capabilities within a deployable and measurable system architecture.

These limitations highlight the need for a re-examination of cyber deception from an infrastructure-centric perspective, in which deployment, orchestration, and observability are treated as first-class design concerns rather than implicit assumptions.

2.2. Microservices, Containers, and Cyber-Ranges in Security Research

In parallel with developments in cyber deception, containerisation and microservice architectures have fundamentally transformed the design of modern distributed systems. Microservice-based approaches decompose applications into independently deployable components, enabling elasticity, fault isolation, and rapid system evolution [9,10]. Container technologies such as Docker provide lightweight virtualisation with fast startup times and consistent execution environments, making them particularly well suited for dynamic and large-scale deployments.

Within the cybersecurity domain, these technologies have been increasingly adopted across a range of applications, including malware analysis sandboxes, intrusion detection systems, and cyber-range platforms [11–13]. Cyber-ranges, in particular, leverage virtualisation and containerisation to emulate realistic enterprise environments, supporting controlled experimentation, training, and evaluation of defensive techniques. Their ability to provide isolation, repeatability, and flexible environment configuration makes them a natural foundation for evaluating adversarial scenarios under controlled conditions [14].

However, the application of these paradigms to cyber deception infrastructures remains comparatively underdeveloped. Existing work on containerised honeypots — including widely used Docker-based deployments — primarily focuses on simplifying deploy-

ment and improving reproducibility, rather than addressing broader architectural concerns such as orchestration, scalability under concurrent attack workloads, and system-wide observability. In many cases, containerisation is used as a replacement for virtual machines without a corresponding rethinking of system design, resulting in deployments that remain logically monolithic despite their container-based implementation.

Furthermore, while microservice architectures inherently support modularity and independent scaling, their adoption in cyber deception systems has been limited and insufficiently evaluated. Few studies provide systematic analysis of how microservice-based decomposition impacts deployment latency, resource utilisation, or operational overhead in adversarial environments. Even fewer integrate container orchestration, API-driven control, and telemetry pipelines into a cohesive framework that enables both operational deployment and performance analysis [15–17].

This gap is particularly significant when considering the requirements of modern deception systems, which must operate under high concurrency, adapt dynamically to evolving attack conditions, and provide continuous visibility into attacker behaviour. Without an explicit architectural integration of deployment, orchestration, and observability mechanisms, containerisation alone does not resolve the core limitations of traditional deception platforms [18].

These observations suggest that the challenge is not merely the adoption of container technologies, but the design of an integrated system architecture that leverages microservices, orchestration, and telemetry in a coordinated and measurable manner. Addressing this gap is central to enabling scalable and operationally viable cyber deception infrastructures.

2.3. Telemetry and Security Monitoring

Observability constitutes a fundamental requirement for effective cyber deception, as the utility of deception systems extends beyond merely misleading adversaries to enabling the collection, analysis, and interpretation of high-fidelity behavioural evidence. In this context, telemetry pipelines serve as the primary mechanism through which attacker interactions are captured, processed, and made available for downstream analysis, including threat hunting, behavioural modelling, and attribution.

Centralised telemetry and logging frameworks have been widely adopted in security operations, intrusion detection, and incident response workflows. Platforms such as the ELK (Elasticsearch–Logstash–Kibana) stack provide scalable ingestion, indexing, and real-time visualisation capabilities, supporting high-throughput processing of security events [19,20]. These systems have demonstrated effectiveness in handling large volumes of log data across distributed environments and have become a de facto standard for operational monitoring in many security contexts.

Prior research has leveraged telemetry-driven approaches in areas such as network intrusion detection, large-scale log analysis, and threat intelligence generation. These studies highlight the importance of scalable ingestion pipelines, efficient indexing mechanisms, and real-time analytics for processing high-volume security data streams. However, within the domain of cyber deception, the integration of telemetry frameworks remains comparatively underdeveloped and often ad hoc in nature. In many cases, logging is treated as a secondary concern, with limited consideration given to ingestion capacity, latency, data integrity, or resource overhead under adversarial workloads [21].

This lack of systematic integration introduces several practical challenges. First, high-interaction deception environments can generate substantial volumes of behavioural data, particularly under concurrent or automated attack scenarios. Without carefully designed ingestion pipelines, telemetry systems may experience saturation, leading to

delayed processing or data loss. Second, the absence of formal performance modelling makes it difficult to reason about scalability limits or to provision resources appropriately. Third, insufficient validation mechanisms may result in incomplete or inconsistent datasets, undermining the reliability of subsequent analysis [22,23].

Consequently, despite the availability of mature telemetry technologies, there remains limited empirical understanding of how such systems perform when tightly coupled with cyber deception infrastructures, particularly under sustained adversarial load. This gap is significant, as telemetry bottlenecks can directly impact the fidelity of behavioural data and, by extension, the effectiveness of deception-based defence strategies.

Table 1 illustrates that existing cyber deception platforms exhibit varying levels of interaction fidelity and deployment flexibility but generally lack integrated high-throughput telemetry, fine-grained orchestration, and principled performance modelling. Traditional honeypot and honeynet systems are constrained by rigid architectures and limited scalability, while cloud-based or service-oriented approaches often prioritise functionality over transparency and analytical rigour.

Table 1. Comparative summary of representative cyber deception infrastructures.

System	Architecture	Scalability	Telemetry	Automation	Key Limitations
Early honeypots (e.g., Honeyd)	Monolithic	Very limited	Basic logs	Manual	Static; easily fingerprinted
High-interaction honeypots	VM-based	Limited	Local detailed logs	Scripted	High overhead; poor scalability
Honeynets/HoneyGrid	Distributed	Moderate	Central log aggregation	Custom scripts	Tight coupling; slow reconfiguration
Cloud-based honeypots	Cloud VM	Good	Provider logging	Cloud APIs	Coarse control; limited transparency
Deception-as-a-Service	Service-oriented	Good	Dashboard monitoring	Vendor APIs	Limited customisation
Adaptive deception frameworks	Abstracted	Assumed	Implicit	Algorithmic	Infrastructure overhead ignored
Proposed platform	Microservice + containers	Elastic	Centralised ELK	REST API	Requires container stack

In contrast, the approach proposed in this paper explicitly treats telemetry as a first-class architectural component, integrating high-throughput data ingestion with orchestration and analytical modelling. By combining containerised deception assets, API-driven control, and a validated telemetry pipeline within a unified framework, the proposed platform enables both operational observability and systematic evaluation of system performance under adversarial conditions [24].

2.4. Interpretation and Research Gap

The comparison in Table 1 reveals a consistent structural limitation across existing cyber deception systems. Traditional honeypot and honeynet-based frameworks provide high interaction fidelity and valuable behavioural data, yet they are constrained by rigid architectures, manual configuration, and limited scalability. More recent cloud-based and service-oriented approaches improve deployment flexibility and accessibility, but often do so at the expense of transparency, fine-grained control, and analytical rigour. At the same time, adaptive deception frameworks advance the state of the art in strategy selection and attacker engagement, but typically abstract away the underlying system infrastructure, implicitly assuming that deployment, orchestration, and monitoring can be achieved without significant cost or constraint.

Taken together, these observations highlight a fundamental gap between *deception capability* and *infrastructure realisation*. While substantial progress has been made in modelling attacker behaviour and designing adaptive strategies, comparatively little work has addressed how such strategies can be deployed, scaled, and observed in practice under realistic adversarial conditions. In particular, three limitations emerge consistently across prior work: (i) the absence of modular and scalable system architectures that support dynamic reconfiguration, (ii) the lack of integrated, high-throughput telemetry pipelines

capable of sustaining concurrent attacker interactions, and (iii) the limited use of formal or empirical methods to characterise performance, scalability limits, and operational cost.

This gap is not merely architectural but methodological. Without explicit performance modelling and systematic evaluation, it remains difficult to reason about the feasibility and effectiveness of deception systems beyond small-scale or controlled deployments. As a result, many existing approaches provide limited guidance for capacity planning, performance optimisation, or integration with higher-level adaptive defence mechanisms [25].

This paper addresses these limitations by introducing a microservice-oriented cyber deception platform that explicitly integrates containerised deployment, API-driven orchestration, and centralised telemetry within a unified and analytically grounded framework. Unlike prior work that treats these components independently, the proposed approach considers them as interdependent elements of a single system, enabling both operational deployment and principled analysis of system behaviour. By focusing on scalability, observability, and measurable performance, this work provides a deployable infrastructure substrate that complements existing research on adaptive and intelligence-driven deception, and supports the practical realisation of next-generation autonomous defence systems [26,27].

3. System Architecture

This section presents the architecture of the proposed microservice-oriented cyber deception platform, which constitutes the central contribution of this work. The design is guided by the need to support scalable deployment, dynamic reconfiguration, and continuous observability of deception assets under realistic adversarial workloads. Unlike conventional deception systems, where deployment, control, and monitoring are often tightly coupled or implicitly managed, the proposed architecture explicitly separates these concerns and treats them as independent but interacting system components. At a high level, the platform adopts a layered architectural model that decomposes functionality across virtualization, containerization, orchestration, telemetry, and monitoring subsystems. This separation is not purely conceptual; it enables independent scaling, clearer responsibility boundaries, and improved fault isolation across system components. In particular, the architecture is designed to ensure that deployment operations, behavioral data collection, and system observability can evolve and scale without introducing bottlenecks or tight coupling between subsystems. A key design principle underlying the architecture is the explicit decoupling of *deception strategy* from *infrastructure management*. Deception actions, such as the deployment or reconfiguration of honeypots, are exposed through a programmatic control interface, while the underlying infrastructure handles resource allocation, container lifecycle management, and telemetry integration. This abstraction enables the platform to support both manual operation and integration with higher-level control logic, including adaptive and learning-based decision systems. Furthermore, architecture treats telemetry as a first-class component rather than a secondary logging mechanism. Behavioral data generated by attacker interactions is continuously captured, processed, and made available for analysis through a centralized pipeline, ensuring that system observability is maintained even under high-concurrency conditions. This design directly addresses limitations identified in prior work, where insufficient integration of telemetry leads to incomplete or delayed behavioral insight. The following subsections describe each architectural layer in detail, including the virtualization environment, containerized deception services, orchestration and control mechanisms, telemetry pipeline, and monitoring components. Together, these elements form a cohesive and extensible system architecture that supports both operational deployment and systematic performance evaluation.

3.1. Architectural Overview

The proposed platform is structured as a composition of loosely coupled services deployed within a controlled cyber-range environment. To provide a formal abstraction of the system, the platform can be represented as the tuple:

$$P = \{V, C, O, T, M\}, \quad (1)$$

where (V) denotes the virtualisation layer, (C) the containerisation layer, (O) the orchestration and control plane, (T) the telemetry pipeline, and (M) the monitoring and visualisation subsystem. This abstraction provides a concise representation of the system structure while emphasising the separation of concerns across functional domains.

Each component of (P) is designed with clearly defined responsibilities and interfaces, enabling independent scaling and evolution. The virtualisation layer ((V)) provides an isolated execution environment for the cyber-range, supporting controlled deployment of attackers and deception nodes. The containerisation layer ((C)) encapsulates deception assets as lightweight, reproducible services. The orchestration layer ((O)) manages the lifecycle of these services, including deployment, scaling, and reconfiguration. The telemetry pipeline ((T)) is responsible for the collection, processing, and storage of behavioural data generated by attacker interactions, while the monitoring subsystem ((M)) provides real-time visualisation and system-level observability.

From an operational perspective, the system can be interpreted as a dataflow pipeline driven by adversarial interaction. Attacker-generated traffic enters the virtualized network environment ((V)) and is directed toward containerised honeypots ((C)), where interactions are captured and logged. These interaction events are then propagated through the telemetry pipeline ((T)), where they are ingested, indexed, and made available for analysis. The monitoring layer ((M)) provides a continuous view of both system performance and attacker behaviour, while the orchestration layer ((O)) dynamically adapts the deployment of deception assets in response to observed conditions.

This layered and formally defined structure enables the system to support dynamic reconfiguration and high-concurrency operation while maintaining clear boundaries between deployment, control, and observability functions. It also provides a foundation for analytical modelling (Section 4), where system behaviour can be reasoned about in terms of interactions between these components rather than as an opaque monolithic system.

Figure 1 illustrates this architecture from a layered perspective, highlighting the flow of attacker traffic through the virtualised environment and the corresponding propagation of telemetry data across system components.

Figure 1 provides a layered representation of the proposed system architecture, illustrating both the structural decomposition of the platform and the flow of adversarial interaction through its components. Attacker-generated traffic enters the cyber-range environment and is directed through the virtualised network infrastructure, where it is routed toward containerised honeypot services deployed within the containerisation layer (C). These honeypots capture detailed interaction data, which is subsequently propagated through the telemetry pipeline (T) for ingestion, processing, and indexing. The telemetry subsystem ensures that behavioural events are collected in a structured and high-throughput manner, enabling downstream analysis without introducing significant latency or data loss. The monitoring and visualisation layer (M) provides real-time visibility into both system performance and attacker activity, exposing metrics and behavioural insights through dashboard interfaces. The orchestration and control layer (O) operates orthogonally to the data flow, managing the lifecycle of containerised deception assets, including deployment, scaling, and reconfiguration in response to observed conditions.

This separation between control logic and data processing enables dynamic adaptation without disrupting ongoing telemetry collection or system stability.

Overall, the architecture emphasises a clear separation of concerns across the virtualisation (V), containerisation (C), orchestration (O), telemetry (T), and monitoring (M) layers. This design supports scalable deployment under concurrent adversarial workloads, continuous observability of system behaviour, and modular evolution of individual subsystems without introducing tight coupling or performance bottlenecks.

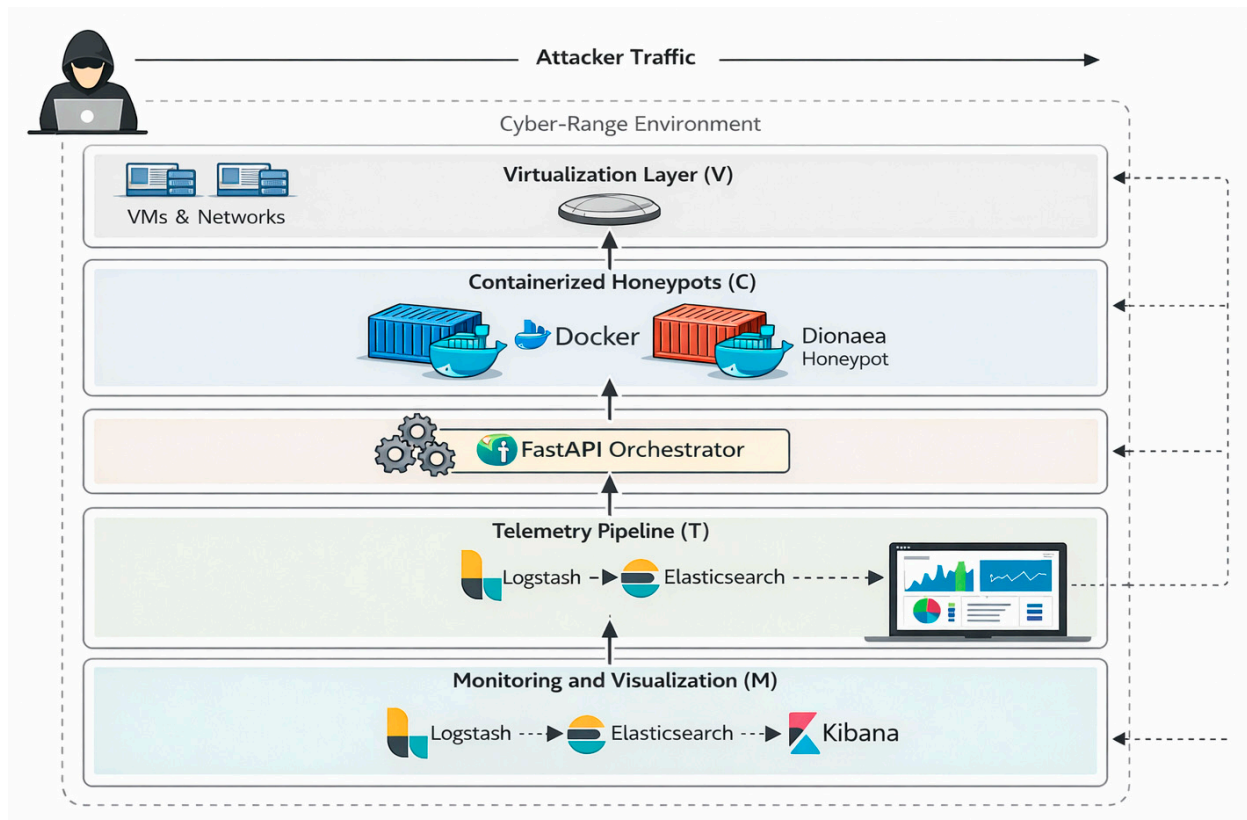


Figure 1. Layered architecture of the proposed microservice-oriented cyber deception platform.

3.2. Virtualisation and Cyber-Range Environment

The virtualisation layer ((V)) provides the foundational execution environment for the proposed cyber deception platform, enabling controlled deployment, isolation, and repeatable experimentation. The system is instantiated within a dedicated cyber-range constructed using commodity virtualisation technology, which allows adversarial activity to be safely emulated without exposure to production infrastructure. This approach ensures both experimental control and operational safety, which are essential for evaluating deception systems under realistic conditions.

The cyber-range emulates a simplified enterprise network topology comprising three primary components: (i) attacker virtual machines, which generate adversarial traffic; (ii) a management virtual machine, which hosts orchestration and telemetry services; and (iii) a dedicated deception subnet, which contains containerised honeypots. This separation reflects common enterprise network structures while maintaining clear boundaries between functional domains.

Network segmentation is employed as a core design mechanism to enforce isolation between attacker, management, and deception components. Segmented virtual networks are interconnected through controlled routing paths, enabling realistic communication flows while preventing unintended lateral movement or compromise of management ser-

vices. Ingress and egress points are explicitly defined and monitored, allowing adversarial interactions to be contained, observed, and analysed without risk to the host system or external networks.

From an experimental perspective, the cyber-range supports systematic and reproducible evaluation. Virtual machines, network configurations, and system states can be reset to predefined baselines between experimental runs, ensuring consistency across trials and reducing variance caused by residual system artefacts. This capability is critical for validating analytical models (Section 4) and for performing comparative performance evaluation under controlled attacker workloads.

While higher-performance virtualisation platforms (e.g., KVM-based environments) may offer improved scalability in production deployments, the use of commodity virtualisation in this work prioritises reproducibility, portability, and ease of replication across research environments. This design choice reflects a trade-off between experimental accessibility and raw performance, and does not constrain the architectural generality of the proposed platform.

Figure 2 illustrates the cyber-range deployment used to host the system. The environment consists of logically segmented virtual networks connecting attacker virtual machines, a management node hosting orchestration and telemetry components, and a deception subnet containing containerised honeypots. Controlled communication paths ensure realistic attacker interaction while preserving strict isolation of control and monitoring infrastructure.

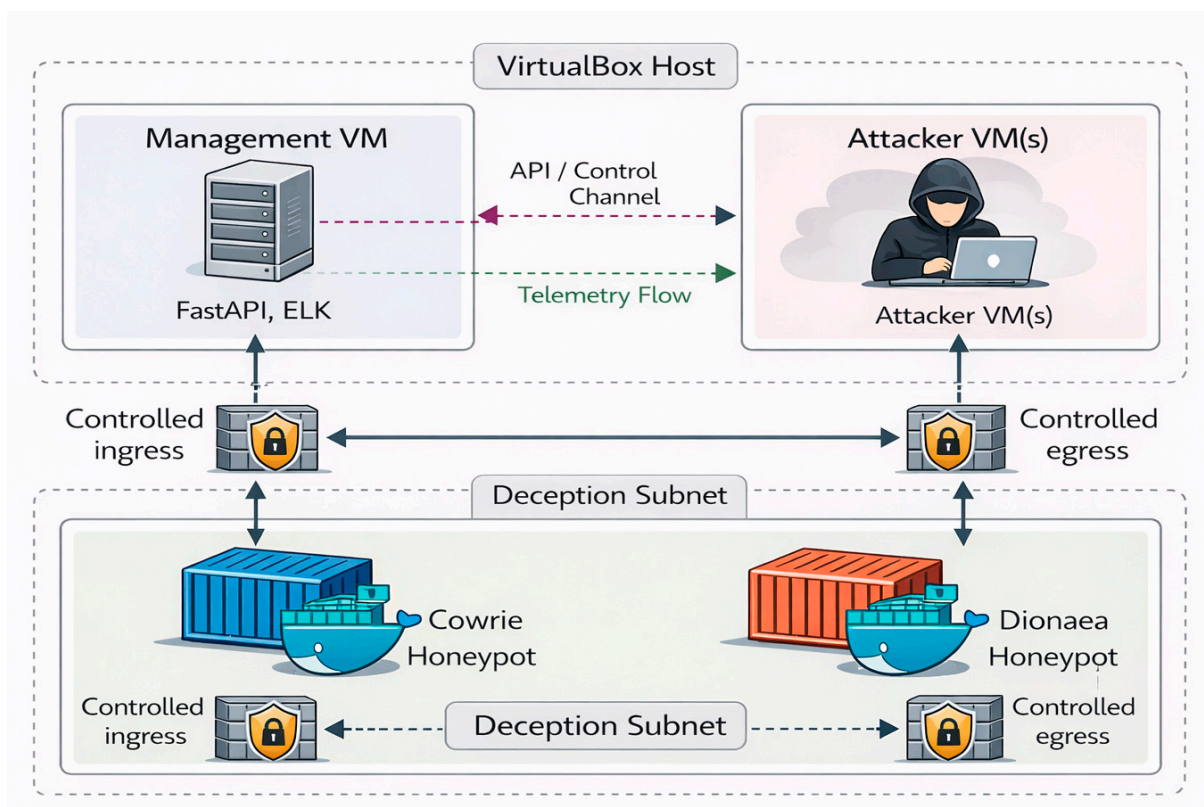


Figure 2. Cyber-range deployment and segmented virtual networking.

3.3. Containerised Deception Services

At the core of the platform, the containerisation layer ((C)) hosts the deception services responsible for interacting with adversaries and generating behavioural data. In contrast to traditional virtual machine-based honeypot deployments, the proposed system utilises

lightweight containerisation to encapsulate each deception asset as an independent and reproducible service instance.

Each containerised honeypot provides a specific deceptive capability, such as interactive shell access (e.g., SSH-based environments), network service emulation, or malware collection. High-interaction honeypots, including implementations such as Cowrie and Dionaea, are deployed within container environments to enable realistic attacker engagement while maintaining controlled execution boundaries. This approach preserves behavioural fidelity while significantly reducing the overhead associated with full-system virtualisation.

The use of containers provides several key advantages from a systems perspective. First, container instantiation incurs substantially lower latency compared to virtual machine deployment, enabling rapid scaling and dynamic reconfiguration in response to changing attack conditions. Second, containerisation supports fine-grained resource management through mechanisms such as CPU and memory constraints, allowing individual honeypot instances to be scaled, limited, or isolated according to operational requirements. Third, containers provide a consistent and portable execution environment, ensuring reproducibility across experimental runs and deployment platforms.

From an architectural standpoint, modelling each honeypot as an independent service enables heterogeneous deception deployments, in which multiple honeypot types can coexist and be managed concurrently. This heterogeneity is essential for increasing attack surface diversity and reducing the likelihood of fingerprinting by adversaries. Furthermore, the decoupling of deception services from the underlying infrastructure allows the orchestration layer (Section 3.4) to dynamically manage the lifecycle of these containers without modifying their internal behaviour.

Despite these advantages, containerisation introduces trade-offs that must be explicitly considered. In particular, while containers provide lightweight isolation, they do not offer the same level of security isolation as full virtual machines, and may expose detectable artefacts that can be exploited by sophisticated adversaries. The proposed architecture mitigates these risks through network segmentation at the virtualisation layer and controlled interaction boundaries, ensuring that compromise of individual containers does not propagate beyond the deception environment.

Overall, the containerisation layer provides a flexible and scalable mechanism for deploying deception services, enabling rapid instantiation, heterogeneous composition, and controlled resource allocation. These properties are essential for supporting dynamic cyber deception under high-concurrency adversarial workloads.

3.4. Orchestration and Control Plane

The orchestration layer ((O)) provides centralised control over the lifecycle and configuration of containerised deception assets through an API-driven control plane. This layer is responsible for coordinating all operational aspects of honeypot management, including deployment, configuration, scaling, and termination. By exposing these operations through a set of well-defined RESTful interfaces, the system enables programmatic and consistent control of deception infrastructure across varying operational contexts.

A key design objective of the orchestration layer is the explicit decoupling of *deception strategy* from *infrastructure execution*. Rather than embedding decision logic within the deployment mechanisms, the platform abstracts container lifecycle management behind a lightweight orchestration service. This abstraction allows external components—such as rule-based controllers, adaptive systems, or reinforcement learning agents—to issue high-level commands (e.g., deploy, scale, reconfigure) without requiring direct interaction with the underlying container runtime. As a result, the platform supports both manual operator-driven workflows and fully automated orchestration pipelines.

From a systems perspective, the orchestration service performs several coordinated functions. Upon receiving a deployment request, the control plane validates input parameters, selects appropriate container images based on requested honeypot types, and invokes the container runtime to instantiate new service instances. Resource constraints and network configurations are applied during this process to ensure isolation and controlled exposure. Once instantiated, containers are automatically registered with the telemetry pipeline, enabling immediate collection of interaction data and integration into monitoring workflows.

The orchestration layer also supports dynamic scaling and reconfiguration in response to observed system conditions. For example, additional honeypot instances can be deployed under increased attacker load, or existing instances can be reconfigured to expose different services or interaction profiles. Termination operations allow unused or compromised instances to be removed without disrupting the overall system.

Figure 3 illustrates the orchestration workflow implemented by the platform. External clients or control agents interact with the FastAPI-based orchestration service through RESTful endpoints, initiating lifecycle operations that propagate through the container runtime and telemetry subsystems. This workflow ensures that newly deployed deception assets are immediately observable and controllable, maintaining consistency between deployment state and monitoring visibility.

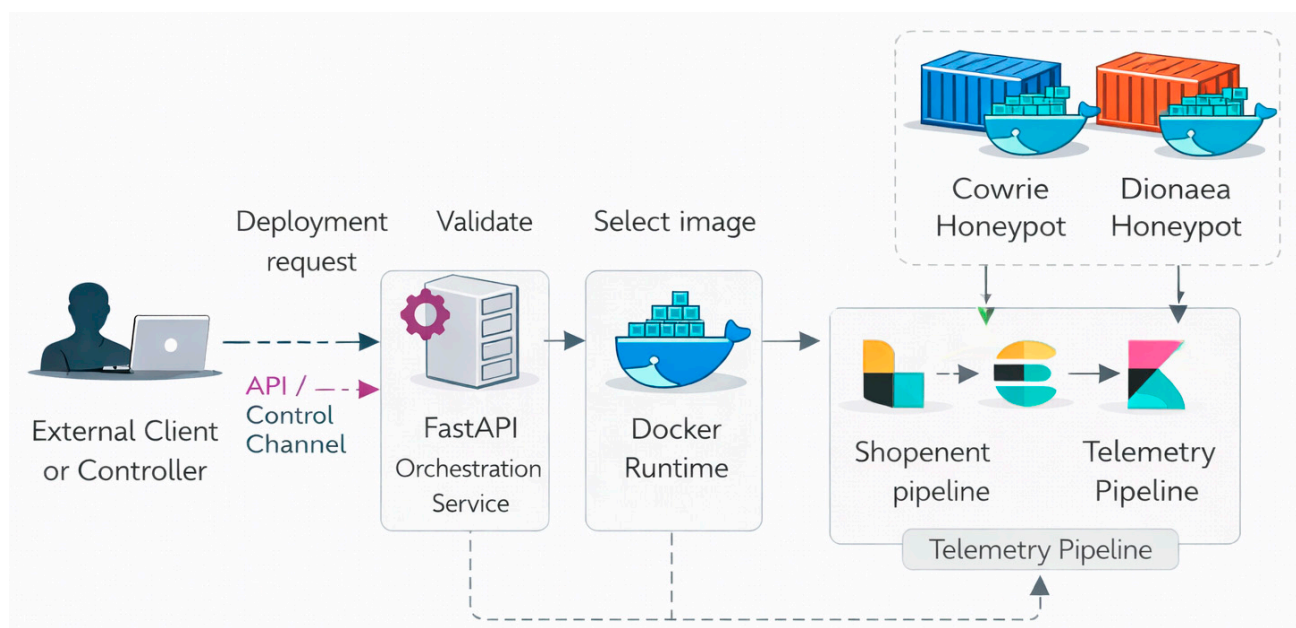


Figure 3. Fast API-based orchestration workflow for containerised honeypots.

By providing a unified and programmatic control interface, the orchestration layer enables dynamic and adaptive cyber deception while preserving system stability and observability. This capability is essential for supporting high-concurrency environments and for enabling integration with higher-level autonomous defence mechanisms.

3.5. Telemetry Collection and Data Pipeline

The telemetry layer ((T)) is responsible for the capture, transport, processing, and storage of behavioural data generated through attacker interactions with deception assets. Within the proposed architecture, telemetry is treated as a first-class system component, ensuring that all interactions with containerised honeypots are systematically recorded and made available for analysis with minimal latency.

Each containerised honeypot generates both structured and unstructured logs capturing session-level activity, including authentication attempts, executed commands, network interactions, and temporal metadata. These logs are emitted in real time and forwarded to a centralised ingestion pipeline, which is designed to support high-throughput processing under concurrent adversarial workloads.

The telemetry pipeline follows a stream-oriented architecture that decouples data producers (honeypots) from downstream consumers (storage and analytics systems). Log events are first collected by an ingestion service (e.g., Logstash), where they undergo parsing, normalisation, and enrichment before being indexed in a scalable storage backend (e.g., Elasticsearch). This design enables continuous data flow without requiring direct coupling between honeypot instances and storage systems, improving both scalability and fault tolerance.

From an operational perspective, the pipeline is designed to handle bursty and potentially adversarial data generation patterns. Under conditions of high interaction rates—such as automated scanning or brute-force attacks—log generation can increase significantly. The use of buffered ingestion and distributed indexing mechanisms ensures that data is not lost under transient load spikes, although increased ingestion latency may be observed at extreme throughput levels.

To ensure data integrity and completeness, each log event is assigned a unique identifier, enabling validation of ingestion correctness and detection of potential data loss during processing. This capability is particularly important for experimental evaluation, where accurate measurement of telemetry throughput and system performance is required.

The centralised storage of telemetry data supports multiple analytical use cases, including real-time monitoring (via the monitoring layer, (M)), retrospective forensic analysis, and integration with external analytics frameworks. By providing a unified and scalable telemetry infrastructure, the platform ensures that behavioural evidence is preserved and accessible even under sustained and concurrent attacker activity.

Despite these capabilities, the telemetry pipeline represents a potential scalability bottleneck within the overall system. As demonstrated in Section 6, ingestion throughput is ultimately constrained by processing and indexing capacity, highlighting the importance of careful resource provisioning and system tuning in high-load scenarios.

Overall, the telemetry layer enables continuous, high-fidelity observation of attacker behaviour, forming a critical foundation for both operational monitoring and analytical evaluation of cyber deception systems.

3.6. Monitoring and Observability

The monitoring layer ((M)) provides real-time visibility into both system performance and attacker activity, serving as the primary interface for observing and analysing the behaviour of the cyber deception platform. Building upon the telemetry pipeline described in Section 3.5, this layer aggregates, visualises, and contextualises data to support both operational monitoring and analytical evaluation.

From an infrastructure perspective, the monitoring subsystem continuously tracks key operational metrics, including CPU utilisation, memory consumption, container lifecycle status, and telemetry ingestion throughput. These metrics are exposed through a centralised visualisation interface (e.g., Kibana dashboards), enabling operators to assess system health, identify performance bottlenecks, and verify that deception services remain responsive under varying workload conditions. The ability to observe system-level behaviour in real time is essential for maintaining stability in high-concurrency environments and for diagnosing performance anomalies.

In addition to infrastructure metrics, the monitoring layer provides visualisation of behavioural telemetry generated by attacker interactions. This includes session timelines, command execution traces, connection patterns, and frequency distributions of attack events. By correlating these behavioural signals with system-level metrics, the platform enables a unified view of both adversarial activity and system dynamics, supporting situational awareness and exploratory analysis.

A key capability of the monitoring layer is the integration of multiple data perspectives into a single analytical interface. For example, spikes in telemetry throughput can be directly associated with specific attacker behaviours, such as automated scanning or brute-force attempts, allowing operators to interpret system load in the context of adversarial activity. This correlation is particularly valuable for validating system behaviour during experimental evaluation (Section 6) and for identifying the onset of potential bottlenecks in the telemetry pipeline.

From a methodological standpoint, the monitoring subsystem also supports the validation of analytical models by providing empirical measurements of system performance under controlled conditions. Observed metrics such as deployment latency, resource utilisation, and ingestion rates can be compared against model predictions, enabling systematic evaluation of model accuracy and system scalability.

Overall, the monitoring layer transforms raw telemetry data into actionable insight, enabling continuous observability of both infrastructure and attacker behaviour. This capability is central to the platform's design, ensuring that cyber deception is not only deployable at scale but also measurable, interpretable, and amenable to systematic analysis.

3.7. Design Rationale and Architectural Benefits

The microservice-oriented design of the proposed platform addresses several structural limitations identified in prior cyber deception systems, particularly with respect to scalability, observability, and operational flexibility. By decomposing the system into loosely coupled components, the architecture enables independent scaling of deception services, orchestration mechanisms, and telemetry pipelines. This separation reduces resource contention and allows system bottlenecks—such as telemetry ingestion—to be isolated and addressed without requiring redesign of the entire platform.

A central design principle is the explicit decoupling of infrastructure management from deception strategy. By exposing lifecycle operations through a programmatic control interface, the platform enables external components to implement adaptive or policy-driven behaviour without direct interaction with underlying system resources. This abstraction supports both manual operation and automated control, and facilitates integration with higher-level frameworks, including reinforcement learning-based controllers and behaviour-driven decision systems.

The integration of a centralised telemetry pipeline with real-time monitoring further distinguishes the proposed architecture from traditional approaches. Rather than treating logging as a secondary function, the platform incorporates observability as a core design element, enabling continuous visibility into both system performance and adversarial activity. This capability is essential for validating system behaviour, identifying performance bottlenecks, and supporting data-driven analysis of attacker interactions.

From an architectural perspective, the use of containerised services introduces flexibility in deployment and composition, allowing heterogeneous honeypot instances to be instantiated and managed dynamically. However, this flexibility is accompanied by trade-offs, including increased system complexity and the need for careful coordination between orchestration and telemetry components. The proposed design addresses these

challenges through clearly defined interfaces and layered separation of concerns, ensuring that system complexity remains manageable while preserving extensibility.

Overall, the architecture provides a scalable and observable infrastructure substrate that bridges the gap between theoretical deception strategies and their practical realisation. By enabling dynamic deployment, high-fidelity data collection, and systematic performance analysis, the platform supports both experimental research and operational cyber defence, and establishes a foundation for future work on adaptive and autonomous deception systems.

4. Analytical Performance and Scalability Modelling

To complement the architectural design and empirical evaluation of the proposed cyber deception platform, this section introduces a set of analytical models that characterise key performance and scalability properties of the system. The purpose of these models is not to provide exact predictive accuracy, but to offer tractable abstractions that enable principled reasoning about system behaviour under varying attacker workloads and deployment conditions. In contrast to prior work, where infrastructure performance is often treated implicitly or abstracted away, this work explicitly models the operational characteristics of the underlying system. This approach enables systematic analysis of how architectural design choices—such as containerised deployment, API-driven orchestration, and centralised telemetry—impact scalability, responsiveness, and resource consumption in practice. The analysis focuses on four core dimensions that capture the primary constraints governing system behaviour: (i) honeypot deployment latency, (ii) resource utilisation, (iii) telemetry throughput, and (iv) operational cost. These dimensions are selected to reflect both system-level performance (e.g., latency and resource usage) and operational feasibility (e.g., throughput limits and cost efficiency) in high-concurrency adversarial environments.

Each model is formulated as a first-order approximation that captures dominant system behaviours while abstracting away lower-level implementation details. Where appropriate, simplifying assumptions are introduced to ensure analytical tractability; these assumptions are explicitly stated and later evaluated against empirical observations in Section 6. This modelling approach enables alignment between theoretical analysis and experimental validation, allowing discrepancies to be interpreted as indicators of system bottlenecks or unmodelled effects. Figure 4 provides an overview of the analytical framework. The models describe honeypot deployment latency as a function of parallel instantiation, resource utilisation as a function of attacker concurrency, telemetry ingestion as a bounded-rate process relative to pipeline capacity, and operational cost as a function of active deception instances. Together, these models provide a structured basis for analysing scalability limits, identifying performance bottlenecks, and supporting informed deployment and capacity planning decisions.

4.1. Honeypot Deployment Latency Model

Rapid deployment of deception assets is essential for timely attacker redirection and engagement. Let T_d denote the deployment latency of a single containerised honeypot instance. This latency can be decomposed into the following components:

$$T_d = T_{\text{pull}} + T_{\text{init}} + T_{\text{net}}, \quad (2)$$

where T_{pull} represents the time required to retrieve the container image, T_{init} denotes container initialisation and configuration time, and T_{net} captures network setup and service exposure overhead.

When deploying n honeypot instances in parallel, the effective deployment latency $T_d^{(n)}$ is influenced by shared resource contention. Assuming bounded orchestration overhead and sufficient parallelism, the deployment latency can be approximated as:

$$T_d^{(n)} \approx T_d + \rho(n), \quad (3)$$

where $\rho(n)$ models additional overhead introduced by concurrent container scheduling and resource allocation. For moderate values of n , $\rho(n)$ remains small, enabling near-linear scaling of deployment operations.

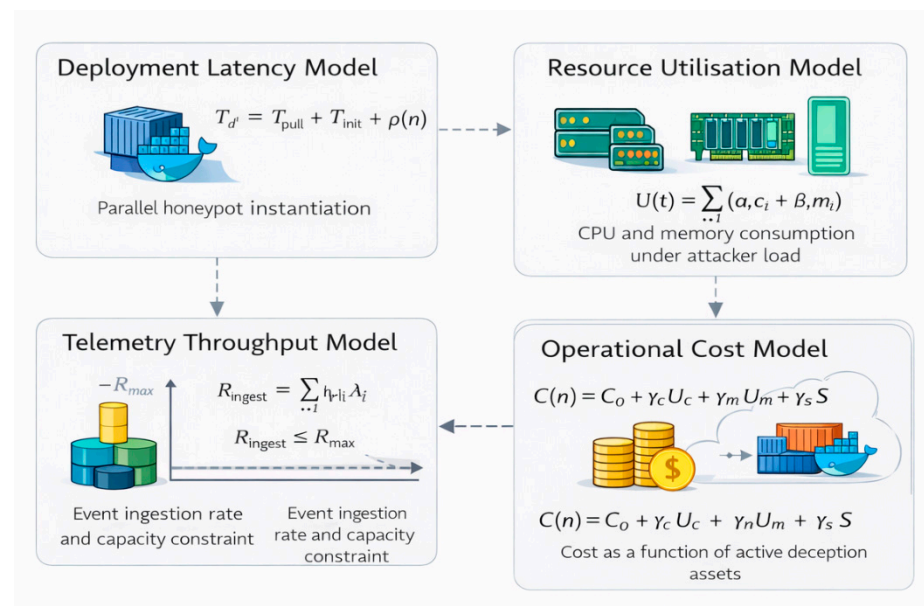


Figure 4. Analytical models for deployment latency, resource utilisation, telemetry throughput, and operational cost.

4.2. Resource Utilisation Model

System resource utilisation is modelled to assess scalability under concurrent attacker workloads. Let $U_c(t)$ and $U_m(t)$ denote CPU and memory utilisation at time t , respectively. These quantities can be expressed as:

$$U_c(t) = \sum_{i=1}^{N(t)} \alpha_i c_i, U_m(t) = \sum_{i=1}^{N(t)} \beta_i m_i, \quad (4)$$

where $N(t)$ is the number of active attacker sessions at time t , c_i and m_i represent the per-session CPU and memory demands, and α_i , β_i capture workload-dependent scaling factors.

This formulation reflects the fact that resource utilisation increases approximately linearly with the number of concurrent sessions, subject to system-level constraints. The model provides a basis for identifying saturation points beyond which additional attacker load may degrade system performance.

4.3. Telemetry Throughput Model

Telemetry collection is a critical requirement for behavioural analysis and attribution. Let R_{ingest} denote the telemetry ingestion rate (events per unit time). This rate can be expressed as:

$$R_{\text{ingest}} = \sum_{i=1}^{N(t)} \lambda_i, \quad (5)$$

where λ_i is the event generation rate associated with attacker session i . The telemetry pipeline has a finite maximum capacity $R_{\max}$, determined by ingestion, indexing, and storage capabilities.

To ensure lossless telemetry capture, the following condition must hold:

$$R_{\text{ingest}} \leq R_{\max}. \quad (6)$$

This constraint enables identification of safe operating regions for the platform and informs decisions regarding horizontal scaling of telemetry services under high attacker concurrency.

4.4. Operational Cost Model

Operational cost is modelled to capture the trade-off between deception effectiveness and resource expenditure. Let $C(n)$ denote the cost of operating n active honeypot instances over a given time horizon. The cost function can be expressed as:

$$C(n) = C_0 + \sum_{i=1}^n (\gamma_c c_i + \gamma_m m_i + \gamma_s), \quad (7)$$

where C_0 represents fixed infrastructure overhead, γ_c and γ_m are cost coefficients for CPU and memory consumption, respectively, and γ_s captures storage and telemetry processing costs.

This formulation enables analysis of cost scalability and supports optimisation of deployment strategies under resource and budget constraints.

4.5. Model Implications

Collectively, the analytical models provide insight into how deployment latency, resource utilisation, telemetry throughput, and cost scale with attacker concurrency and deception intensity. The models highlight the importance of modularity and parallelism in maintaining low deployment latency, as well as the need for sufficient telemetry capacity to preserve behavioural evidence under load.

These analytical results guide system configuration and inform the experimental evaluation presented in subsequent sections. In particular, the models enable validation of empirical results against theoretical expectations, strengthening confidence in the platform's scalability and operational viability.

5. Implementation

This section describes the concrete implementation of the proposed microservice-oriented cyber deception platform. The implementation is designed to reflect realistic deployment conditions while preserving experimental control, reproducibility, and operational safety. Widely adopted open source technologies are used throughout to facilitate replication and integration into existing security infrastructures.

5.1. Platform Implementation

The platform is deployed within a controlled cyber-range environment hosted on a dedicated physical server running a Linux-based operating system. Virtualisation is provided using VirtualBox, which is used to instantiate multiple virtual machines corresponding to different functional roles, including attacker nodes, a management node, and a deception subnet. This virtualisation layer enables the emulation of an enterprise-like topology while preserving strict isolation from production networks.

Containerised deception services are implemented using Docker. Each honeypot instance is packaged as a self-contained container image, encapsulating its operating

environment, dependencies, and configuration. High-interaction honeypots are used to maximise behavioural fidelity, allowing attackers to perform realistic reconnaissance, authentication attempts, and post-compromise actions. The use of containers enables rapid instantiation and termination of honeypots, supporting the low deployment latency and elastic scaling assumed in the analytical models.

The orchestration layer is implemented as a lightweight microservice using FastAPI. This service exposes RESTful endpoints that provide programmatic control over the lifecycle of honeypot containers, including deployment, configuration, scaling, and teardown operations. Incoming API requests are validated and mapped to container runtime operations, allowing both human operators and automated controllers to trigger deception actions. This design decouples strategy selection from infrastructure management and provides a uniform interface for integration with higher-level adaptive or AI-driven decision components.

5.2. Telemetry and Monitoring Infrastructure

Telemetry collection and monitoring are implemented using a centralised logging and analytics pipeline based on the ELK stack. Each containerised honeypot generates detailed interaction logs, including session identifiers, command histories, network connection metadata, and timing information. Log forwarding agents within the deception subnet ship these events to a dedicated ingestion service, where they are parsed, normalised, and forwarded for indexing.

Elasticsearch is used as the primary storage and indexing backend for telemetry data. It supports efficient querying, aggregation, and correlation across large volumes of events generated under sustained attacker activity. Kibana dashboards provide real-time visualisation of both behavioural telemetry and system health metrics, including CPU utilisation, memory consumption, container status, and telemetry ingestion rates. These dashboards are used during experiments to verify that the platform operates within expected performance bounds and that no telemetry loss occurs under the evaluated workloads.

The telemetry pipeline is configured to support high-throughput ingestion with low end-to-end latency, ensuring that attack interactions are captured and made available for analysis in near real time. This implementation directly supports the telemetry throughput and capacity constraints modelled analytically in Section 4 and underpins subsequent experimental evaluation of ingestion performance.

5.3. Security and Isolation Considerations

Given the use of high-interaction honeypots, security and isolation are critical design concerns. Multiple layers of containment are employed to mitigate the risk of attackers leveraging deception assets to compromise the underlying infrastructure or external systems.

At the coarse-grained level, virtual machines provide isolation between attacker hosts, the management node, and the deception subnet. Attacker traffic is restricted to the deception subnet and is prevented from directly accessing management services or the host operating system. Within the deception subnet, each honeypot runs as an isolated container with its own namespace and resource limits, enforced by the container runtime.

Network-level controls are applied to restrict outbound connectivity from honeypot containers, preventing them from being used as stepping stones for further attacks. Only strictly necessary communication channels for telemetry forwarding and orchestration are permitted. Resource limits for CPU and memory are configured per container to reduce the risk of resource exhaustion attacks. Together, these measures ensure that the platform can support realistic attacker interaction while maintaining the safety and stability of the host environment.

5.4. Reproducibility and Deployment Portability

The platform is implemented with reproducibility and portability as explicit design goals. Virtual machine configurations, network topologies, and container specifications are defined using declarative configuration files and version-controlled artefacts. This enables the entire environment to be reinstantiated from a clean baseline, supporting repeatable experiments and consistent performance evaluation.

Snapshotting capabilities within the virtualisation layer are used to restore the cyber-range to a known initial state between experimental runs, eliminating residual state that could bias results. Container images are built from scripted definitions, ensuring that honeypot behaviour is consistent across deployments and over time.

The reliance on widely adopted open source technologies—VirtualBox for virtualisation, Docker for containerisation, FastAPI for orchestration, and ELK for telemetry and monitoring—facilitates replication of the platform by other researchers and practitioners. Furthermore, the modular microservice architecture allows individual components to be replaced or extended without redesigning the entire system. For example, alternative honeypot images, orchestration frameworks, or telemetry backends can be integrated with minimal changes to the surrounding infrastructure.

6. Experimental Evaluation

This section presents an empirical evaluation of the proposed microservice-oriented cyber deception platform. The evaluation focuses on deployment latency, resource scalability, and telemetry throughput under varying attacker workloads. Experimental results are analysed in relation to the analytical models introduced in Section 4 to assess consistency between theoretical expectations and observed system behaviour.

6.1. Workloads and Methodology

Experiments are conducted within the controlled cyber-range environment described in Sections 3 and 5. Attacker workloads are generated using scripted adversarial activities designed to emulate common reconnaissance and exploitation behaviours, including port scanning, authentication probing, and sustained interaction with deception services. These workloads generate realistic interaction patterns while ensuring repeatability across experimental runs.

To evaluate scalability, experiments are executed under three attacker concurrency levels: 1, 5, and 20 concurrent attacker sessions. Each configuration is repeated multiple times, and all metrics are aggregated across runs to reduce variance introduced by transient system effects. Prior to each run, the cyber-range is reset to a known baseline state to eliminate the residual state and ensure consistent initial conditions.

Performance metrics are collected continuously during each experiment. Deployment latency is measured as the elapsed time between submission of a honeypot deployment request and the point at which the honeypot becomes reachable by attacker traffic. Resource utilisation metrics include host CPU and memory consumption. Telemetry performance is assessed by measuring the rate at which attacker-generated events are ingested and indexed by the telemetry pipeline.

6.2. Deployment Performance

Deployment latency is a critical metric for deception platforms, as delayed activation of honeypots can reduce the effectiveness of attacker redirection and engagement. Figure 5 illustrates the measured deployment latency as a function of the number of concurrently deployed honeypot instances.

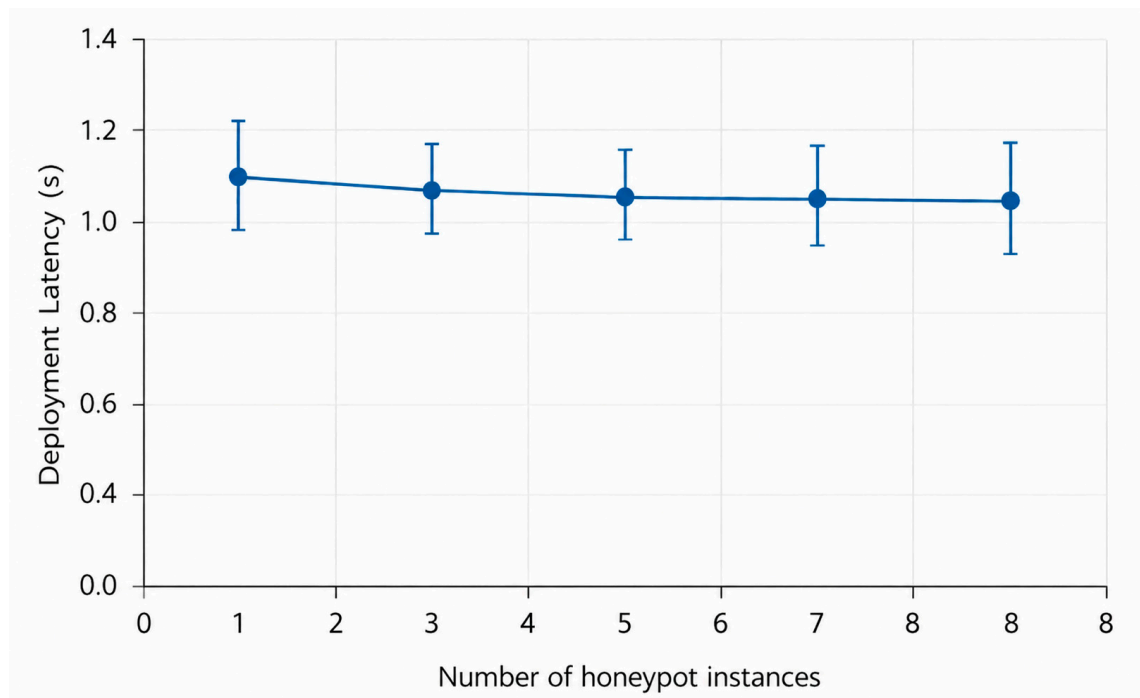


Figure 5. Deployment latency scaling with parallel honeypot instances.

Figure 5 shows the mean deployment latency observed when deploying increasing numbers of honeypot containers in parallel. Deployment time remains within a narrow range, even as concurrency increases, indicating that orchestration overhead grows modestly and does not dominate overall latency.

Empirically, single-instance deployment completes in approximately 1.2 s, while parallel deployment of multiple instances incurs only incremental additional delay. This behaviour is consistent with the analytical latency model presented in Section 4.1, in which the overhead term $\rho(n)$ remains small relative to container initialisation and network setup costs. The results confirm that the platform supports rapid elastic deployment of deception assets, enabling near real-time response to be detected threats.

6.3. Resource Scalability

To assess resource scalability, CPU and memory utilisation are measured under increasing attacker concurrency. Figure 6 presents the mean CPU and memory consumption observed for 1, 5, and 20 concurrent attacker sessions.

Figure 6 illustrates how CPU and memory utilisation scale as the number of concurrent attacker sessions increases. Both metrics exhibit approximately linear growth with attacker concurrency, remaining within acceptable bounds for the evaluated workloads.

At low concurrency, resource utilisation remains minimal, reflecting the lightweight nature of containerised honeypot instances. As attacker concurrency increases, CPU and memory usage grow predictably, without abrupt saturation or instability. This trend aligns with the resource utilisation model described in Section 4.2, where utilisation scales with the number of active sessions $N(t)$. The absence of nonlinear growth or sudden spikes indicates that the platform maintains stable performance under moderate to high load.

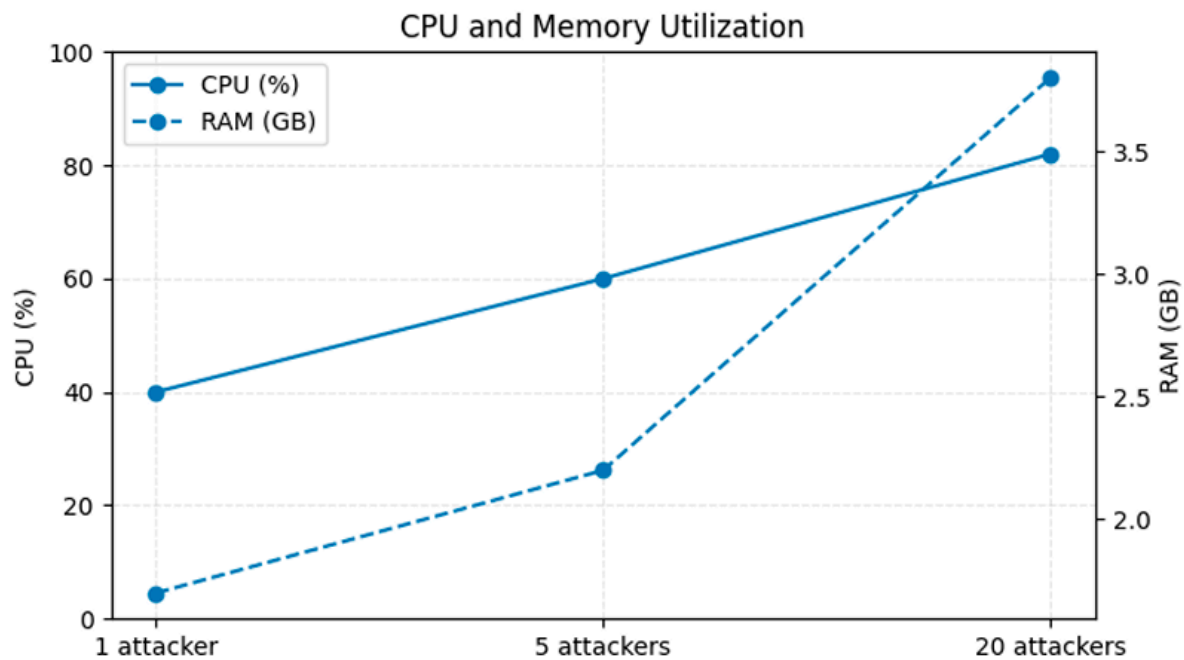


Figure 6. CPU and memory utilisation under increasing attacker concurrency.

6.4. Telemetry Throughput and Integrity

Effective deception requires not only engaging attackers but also capturing high-fidelity behavioural telemetry. Figure 7 shows the telemetry ingestion rate as a function of attacker concurrency.

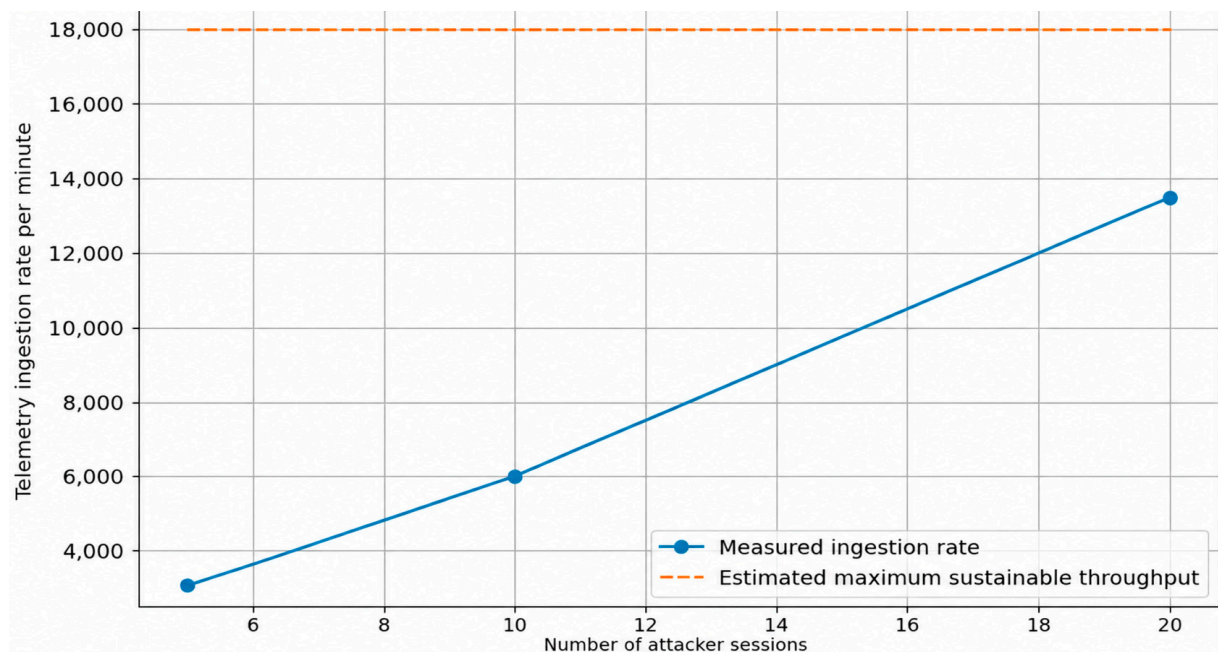


Figure 7. Telemetry ingestion rate relative to estimated maximum throughput.

Figure 7 plots the mean telemetry ingestion rate under increasing attacker concurrency. The ingestion rate increases proportionally with attacker activity and approaches approximately 18,000 events per minute at the highest evaluated load, remaining below the estimated maximum sustainable throughput.

No telemetry loss was observed during the experiments, indicating that the ingestion pipeline maintains sufficient capacity under the evaluated conditions. These results corrob-

orate the throughput constraint $R_{\text{ingest}} \leq R_{\text{max}}$ introduced in Section 4.3 and demonstrate that the telemetry infrastructure supports sustained high-interaction deception without compromising data integrity.

6.5. Engineering Comparison and Practical Implications

From an engineering perspective, the experimental results highlight several practical advantages of the proposed platform. Compared to traditional virtual machine-based honeypot deployments, which often require tens of seconds or minutes for instantiation, the containerised microservice approach achieves substantially lower deployment latency. This responsiveness is critical for integration with adaptive or automated deception controllers that rely on timely infrastructure reconfiguration.

Furthermore, the predictable scaling behaviour observed in resource utilisation and telemetry throughput enables informed capacity planning. Operators can estimate the maximum number of concurrent attacker sessions supported by a given deployment and provision resources accordingly. The integration of centralised telemetry and monitoring provides real-time observability, reducing operational uncertainty and enabling rapid diagnosis of performance bottlenecks.

7. Discussion

The experimental results demonstrate that the proposed microservice-oriented cyber deception platform achieves a favourable balance between responsiveness, scalability, and observability. Rather than optimising a single performance metric, the platform is designed to support adaptive deception strategies that require timely infrastructure reconfiguration, sustained attacker engagement, and reliable behavioural evidence collection. The discussion below interprets the empirical findings in the context of these objectives and situates them within the broader cyber deception literature.

A central finding of the evaluation is the consistently low deployment latency achieved by the containerised orchestration approach. Mean deployment times of approximately 1.2 s, with minimal variance across repeated runs, indicate that deception assets can be instantiated rapidly enough to support near-real-time attacker redirection. This capability addresses a common limitation of traditional virtual machine-based honeypot deployments, where instantiation delays can undermine the effectiveness of dynamic deception strategies. The observed latency behaviour aligns closely with the analytical deployment model introduced in Section 4, suggesting that orchestration overhead remains bounded and predictable even under parallel deployment scenarios.

Resource scalability results further support the suitability of the platform for sustained operation under increasing attacker concurrency. CPU and memory utilisation scale approximately linearly with the number of concurrent attacker sessions, without evidence of abrupt saturation or instability. This predictable scaling behaviour simplifies capacity planning and reduces operational risk, enabling defenders to provision resources based on expected attack volumes. Importantly, the absence of nonlinear degradation suggests that container isolation and lightweight orchestration introduce minimal overhead relative to the interaction workload itself.

Telemetry throughput and integrity represent another critical dimension of deception effectiveness. The ability to ingest and index up to approximately 18,000 events per minute without telemetry loss demonstrates that the platform can support high-interaction deception at scale. Centralised observability, implemented through the telemetry pipeline, enables real-time monitoring of both attacker behaviour and system health. The modest resource overhead introduced by monitoring confirms that observability can be achieved

without significantly compromising performance, reinforcing the practicality of the design for operational environments.

Beyond raw performance metrics, the results highlight the importance of infrastructure design in enabling higher-level deception and analysis objectives. Stable deployment latency, predictable resource scaling, and reliable telemetry ingestion together form a necessary substrate for adaptive and autonomous deception strategies. Without such guarantees, higher-level decision-making mechanisms—such as reinforcement learning-based controllers or probabilistic attribution engines—may be constrained by infrastructure delays or incomplete data. The results therefore support the argument that scalable infrastructure is a foundational requirement for intelligence-driven cyber deception.

From a broader perspective, the findings contribute to ongoing discussions in the cyber deception literature regarding the trade-offs between realism, scalability, and operational safety. High-interaction honeypots are often criticised for their resource demands and deployment complexity. The results presented here demonstrate that, when combined with containerisation and microservice architectures, high-interaction deception can be deployed elastically and observed comprehensively, mitigating many of these concerns. This observation challenges the assumption that scalable deception must rely on low-interaction or heavily abstracted decoys.

Overall, the discussion underscores that effective cyber deception cannot be treated solely as a problem of strategy selection or attacker modelling. Instead, it requires a tightly integrated infrastructure that supports rapid reconfiguration, sustained interaction, and high-fidelity data collection. The proposed platform demonstrates that such integration is achievable using contemporary virtualisation, containerisation, and telemetry technologies, providing a practical foundation for next-generation autonomous defence systems.

8. Limitations

While the proposed cyber deception platform demonstrates strong performance and scalability within the evaluated experimental setting, several limitations must be acknowledged.

First, the experimental evaluation is conducted within a controlled cyber-range environment. Although this setting enables repeatable experimentation and precise measurement, it cannot fully capture the complexity and variability of production enterprise networks. Real-world deployments may exhibit higher background noise, heterogeneous legitimate traffic, and more diverse attacker behaviours, which could influence observed performance metrics. Nevertheless, the relative trends observed in deployment latency, resource scaling, and telemetry throughput are expected to generalise, as they are driven primarily by architectural and orchestration design choices.

Second, attacker workloads used in the experiments are scripted and reproducible by design. While these workloads emulate common reconnaissance and interaction patterns, they may not fully represent highly adaptive or stealthy adversaries that dynamically alter behaviour in response to perceived deception. The platform is designed to support such scenarios, but their systematic evaluation remains an open area for future work.

Third, the analytical models introduced in Section 4 intentionally abstract certain system dynamics to maintain tractability. For example, resource utilisation and telemetry throughput are modelled as aggregate functions of attacker concurrency, without explicitly capturing fine-grained scheduling effects or transient spikes. While the empirical results demonstrate strong alignment with the models under tested conditions, more detailed modelling may be required for extreme-scale deployments or highly bursty workloads.

Fourth, the current implementation focuses on infrastructure-level performance and observability rather than end-to-end adversarial outcomes. While the platform enables rich behavioural telemetry capture, the present evaluation does not directly quantify improve-

ments in attacker deception effectiveness or intelligence extraction beyond infrastructure metrics. Such analyses depend on higher-level deception strategies and analytical pipelines that build upon the platform rather than the platform itself.

Finally, the telemetry and monitoring infrastructure relies on the ELK stack, which introduces its own scalability characteristics and resource overheads. Although the observed overhead is modest in the evaluated environment, large-scale deployments may require additional tuning, sharding, or alternative telemetry backends to maintain performance guarantees.

These limitations do not detract from the core contributions of this work but instead delineate the scope within which the results should be interpreted. They also highlight clear directions for future research and system extension.

9. Conclusions and Future Work

This paper presented a microservice-oriented cyber deception platform designed to support scalable, observable, and rapidly reconfigurable deployment of high-interaction deception assets. By integrating virtualisation, containerisation, API-driven orchestration, and centralised telemetry, the platform addresses key limitations of traditional monolithic deception systems, including high deployment latency, limited scalability, and insufficient observability. The proposed architecture provides a practical infrastructure substrate for adaptive and autonomous deception strategies.

Analytical models were introduced to characterise deployment latency, resource utilisation, telemetry throughput, and operational cost as functions of attacker concurrency and deception asset scale. These models provide a principled basis for capacity planning and performance reasoning. Empirical evaluation within a controlled cyber-range environment demonstrated close alignment between analytical expectations and observed behaviour. The results showed consistently low deployment latency, predictable resource scaling under increasing attacker load, and reliable high-throughput telemetry ingestion without data loss. Together, these findings validate the feasibility of deploying high-interaction deception infrastructures that remain responsive and observable under realistic adversarial workloads.

Beyond infrastructure performance, the platform establishes a foundation for intelligence-driven cyber defence. By enabling rapid deployment of believable deception assets and reliable collection of high-fidelity behavioural telemetry, the system supports downstream analysis tasks such as attacker behaviour modelling, technique attribution, and adaptive deception control. In this sense, the platform complements existing research on deception strategies and learning-based controllers by providing the operational capabilities required to realise such approaches in practice.

Several directions for future work emerge from this study. First, integration with adaptive decision-making mechanisms, including reinforcement learning and partially observable Markov decision process (POMDP) models, would enable closed-loop optimisation of deception strategies based on observed attacker behaviour. Second, the platform can be extended to distributed and cloud-native environments, supporting orchestration of deception assets across heterogeneous infrastructure and geographic regions. Third, the realism of deception environments may be enhanced through automated content generation and dynamic service emulation, increasing resilience against sophisticated adversaries capable of identifying static decoys. Furthermore, experimental evaluation by incorporating more fine-grained attacker workload characterisation, including session duration distributions, interaction frequencies, action sequencing patterns, and event generation rates will improve reproducibility and enable more detailed behavioural analysis of adversarial interactions within the proposed cyber deception platform. Finally, tighter integration

with security operations platforms and incident response workflows could operationalise behavioural insights for real-time defence, threat hunting, and decision support.

In summary, this work demonstrates that scalable and observable cyber deception is achievable using contemporary microservice architectures and telemetry technologies. By focusing on infrastructure design, analytical modelling, and empirical validation, the paper contributes a deployable and extensible foundation for next-generation autonomous cyber defence systems.

Author Contributions: M.H.S.: Conceptualization, Supervision, Validation, Methodology, Writing—review and editing, Formal analysis, Investigation, Visualization, Writing—original draft preparation. M.S.: Conceptualization, Methodology, Implementation, Data curation, Formal analysis, Investigation, Visualization, Writing—original draft preparation. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are generated within a controlled cyber-range environment as described in the paper. Any supporting datasets or experimental configurations are available from the corresponding author upon reasonable request.

Acknowledgments: The authors would like to acknowledge the support of the School of Computing, Engineering and Physical Sciences at the University of the West of Scotland for providing the infrastructure necessary to conduct this research. The Grammarly tool was used to assist in proofreading and improving the clarity of the manuscript. No generative AI tools were used for experimental design, data collection, analysis, or interpretation of results. All content has been reviewed and approved by the authors.

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

References

1. Provos, N.; Holz, T. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*; Addison-Wesley: Boston, MA, USA, 2007.
2. Nawrocki, M.; Wählich, M.; Schmidt, T.C.; Schönwälder, M. A survey on honeypot software and data analysis. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1790–1824.
3. Niclas, I.; Duplys, P.; Sisejkovic, D.; Menth, M. A survey of contemporary open-source honeypots, frameworks, and tools. *J. Netw. Comput. Appl.* **2023**, *220*, 103737. [CrossRef]
4. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP), Funchal, Portugal, 22–24 January 2018.
5. Moustafa, N. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustain. Cities Soc.* **2020**, *8*, 165660–165678.
6. Kim, J.; Kim, J.; Kim, H.; Shim, M.; Choi, E. CNN-based network intrusion detection against denial-of-service attacks. *Electronics* **2020**, *9*, 916.
7. Yuan, X.; He, P.; Zhu, Q.; Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2805–2824. [CrossRef] [PubMed]
8. Creech, G.; Hu, J. A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns. *IEEE Trans. Comput.* **2014**, *63*, 807–819.
9. Ramana, G.V. Containerization Technologies: ECR and Docker for Microservices Architecture. Available online: https://www.researchgate.net/profile/Venkata-Gudelli/publication/389590950_Containerization_Technologies_ECR_and_Docker_for_Microservices_Architecture/links/67c8888fe62c604a0dd5257c/Containerization-Technologies-ECR-and-Docker-for-Microservices-Architecture.pdf (accessed on 24 May 2026).
10. Saboor, A.; Hassan, M.F.; Akbar, R.; Shah, S.N.M.; Hassan, F.; Magsi, S.A.; Siddiqui, M.A. Containerized microservices orchestration and provisioning in cloud computing: A conceptual framework and future perspectives. *Appl. Sci.* **2022**, *12*, 5793. [CrossRef]

11. Pawlick, J.; Colbert, E.; Zhu, Q. A game-theoretic taxonomy and survey of defensive deception for cybersecurity and privacy. *ACM Comput. Surv.* **2019**, *52*, 82. [CrossRef]
12. Pawlick, J.; Zhu, Q. Deception by design: Evidence-based signaling games for network defense. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), Denver, CO, USA, 12–16 October 2015.
13. Pedro, B.-L.; Pérez, M.G.; Nespoli, P. Cyber Deception: Taxonomy, State of the Art, Frameworks, Trends, and Open Challenges. *IEEE Commun. Surv. Tutor.* **2025**, *28*, 1520–1556. [CrossRef]
14. Arif, W.A.; Sukarno, P. Taxonomy and survey of collaborative intrusion detection system using federated learning. *ACM Comput. Surv.* **2024**, *57*, 1–36. [CrossRef]
15. Husák, M.; Čermák, M.; Jirsík, T.; Čeleda, P. Survey of attack projection, prediction, and forecasting in cyber security. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 640–660.
16. Scarfone, K.; Mell, P. *Guide to Intrusion Detection and Prevention Systems (IDPS)*; NIST Special Publication: Gaithersburg, MD, USA, 2007.
17. Min, Y.; Zhang, J. Data Anomaly Detection in the Internet of Things: A review of Current Trends and Research Challenges. Available online: <https://pdfs.semanticscholar.org/0b1f/80af1d09691e799bb221892d158c36830993.pdf> (accessed on 24 May 2026).
18. Benjamin, M.A.; Samtani, S.; Zhu, H.; Chen, H.; Nunamaker, J.F., Jr. Improving threat mitigation through a cybersecurity risk management framework: A computational design science approach. *J. Manag. Inf. Syst.* **2024**, *41*, 236–265. [CrossRef]
19. Javed, A.; Moustafa, N.; Khurshid, H.; Debie, E.; Haider, W.; Wahab, A. A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions. *Electronics* **2020**, *9*, 1177. [CrossRef]
20. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
21. Kaelbling, L.P.; Littman, M.L.; Cassandra, A.R. Planning and acting in partially observable stochastic domains. *Artif. Intell.* **1998**, *101*, 99–134. [CrossRef]
22. Alpcan, T.; Başar, T. *Network Security: A Decision and Game-Theoretic Approach*; Cambridge University Press: Cambridge, UK, 2010.
23. Hutchins, E.; Cloppert, M.; Amin, R. Intelligence-driven computer network defense informed by analysis of adversary campaigns. *Lead. Issues Inf. Warf. Secur. Res.* **2011**, *1*, 80.
24. Wagner, G.; Dübendorfer, T. Automated incident response using reinforcement learning. In Proceedings of the ACM Workshop on Artificial Intelligence and Security (AISec), Virtual Event, 13 November 2020.
25. MITRE Corporation, MITRE ATT&CK Framework. Available online: <https://attack.mitre.org> (accessed on 24 May 2026).
26. Michael, Z.; Gottwalt, F.; Chang, E.; Dillon, T. Provenance-based intrusion detection systems: A survey. *ACM Comput. Surv.* **2022**, *55*, 1–36. [CrossRef]
27. Johannes, M.L. An Adaptive Multi-Layered HoneyNet Architecture for Threat Behavior Analysis via Deep Learning. *arXiv* **2025**, arXiv:2512.07827.

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