Flow based Containerized Honeypot Approach for Network Traffic Analysis: An Empirical Study

Sibi Chakkaravarthy Sethuraman, *Member, IEEE,*, Tharshith Goud Jadapalli, *Member, IEEE,*
Devi Priya Vimala Sudhakaran, *Member, IEEE,* and Saraju P Mohanty, *Senior Member, IEEE*

Abstract*—The world of connected devices has been attributed to applications that relied upon multitude of devices to acquire and distribute data over extremely diverse networks. This caused a plethora of potential threats. In the field of IT security, the concept of digital baits, or honeypots, which are typically network components (computer systems, access points, or switches) launched to be interrogated, savaged, and impacted, is currently popular as it allows scientists to comprehend further on assault patterns and behavior. Combining the inherent modularity with the administration enabled by the container makes security management simple and permits dispersed deployments, resulting in a very dynamic system. This study delivers several contributions in this regard. First, it comprehends the patterns, methods, and malware types that container honeypots deal with thus examining new developments in existing honeypot research to fill gaps in knowledge about the honeypot technology. A broad range of independently initiated and jointly conducted container honeypot strategies and studies that encompass various methodologies is surveyed. Second, using numerous use cases that aid scientific research, we address and investigate a number of challenges pertaining to container honeypots, such as identification problems, honeypot security issues, and dependability issues. Furthermore, based on our extensive honeypot research, we developed VIKRANT, a containerized research honeypot which assists researchers as well as enthusiasts in generating real-time flow data for threat intelligence. The configured approach was monitored resulting in several data points that allowed relevant conclusions about the malevolent users' activities.

***Index Terms*—Cyber Security, IDS, Deception, Virtualization, Containers, Honeypots, Flow data**

I. Introduction

In the constantly evolving world of cybersecurity, threats and attacks are always adapting and becoming more sophisticated. While malware and viruses have always been the primary subjects of discussions about cybersecurity, a recent study has shown a striking finding that in 62% of assault cases, non-malware approaches involving human intervention on the computer are mostly utilized [1]. This information emphasizes how crucial it is to comprehend and address the human component since it is significant for the successful conduct of cyberattacks. Autonomous machine learning is insufficient to halt devoted attackers as former deal with pre-existing attack patterns while adversaries are improving their tradecraft to get beyond legacy security measures. Finding malicious activity in a system or network is the biggest difficulty in the modern era of internet and communication technology. Cyber warriors struggle to stop and identify complex attacks like Advanced Persistent Threat (APT) centered incursions, despite the defense-in-depth method, which deploys many layers of conventional security controls throughout the target network. The conventional method that is used to minimize activities related to invasion is an intrusion detection system (IDS) [2]. IDSs are crucial for protecting both business and personal systems [3]–[5]. They can be typically categorized into passive and active types [6]. Passive IDSs do not actively try to stop or obstruct attacker activity. Active IDSs take the initiative to communicate with the adversary in order to halt an attack [7]. The most widely used type of IDS is the firewall, which is a component of almost all corporate networks' internet security. They are normally classified as passive network defense mechanisms.

The basic objective of a firewall is to prevent unauthorized entry to or exit from a system or network [3], [4], [8]. A firewall operates as the network's administrator, reviewing access requests. The internal network, which might be seen as safe, and the external network or the Internet, which is known to be harmful, are separated by a network firewall. They accept or reject particular packets in accordance with the predefined defense policy(combination of configurable rules) [9]. The firewall is not equipped to handle new attacks, nevertheless, for which the security policy of the firewall has not yet outlined any rules. These types of systems typically perform a rule-based analysis of network activity. Only inbound communications that a firewall has already been mapped up to accept are allowed. Figure. 1 provides an overview of pertinent security methods and primitives on several layers [10]. Publishing security events and indicators of compromise (IoCs) facilitate rapid and significant inference in relation to efficient defenses against cyber attacks [11]. However, the present threat intelligence gathering solutions do not make it simple for threat detection systems to communicate and share knowledge, especially Intrusion Detection Systems (IDS) that use Machine Learning (ML) approaches [12], [13]. By generating a herd immunity against novel (potentially

This research work was supported by the Pi DataCenters (Cloud Infrastructure) and Centre of Excellence, Cyber Security, VIT-AP University

S. C. Sethuraman is with Center of Excellence, Artificial Intelligence & Robotics (AIR) and Center of Excellence, Cyber Security and School of Computer Science and Engineering, VIT-AP University, Andhra Pradesh, India, 522237, e-mail: chakkaravarthy.sibi@vitap.ac.in

Tharshith Goud Jadapalli is with Information Technology - Information Security (MSIT-IS), Carnegie Mellon University, Pittsburgh, PA, USA

Devi Priya V S is with Center of Excellence, Artificial Intelligence & Robotics (AIR) and Center of Excellence, Cyber Security and School of Computer Science and Engineering, VIT-AP University, Andhra Pradesh, India, 522237. e-mail: vsdevipriya@gmail.com

Saraju P. Mohanty is with the Department of Computer Science and Engineering, University of North Texas, Denton, TX 76207 USA (e-mail: smohanty@ieee.org).

undiscovered) assaults and malware, the coupling of proactive threat information sharing and defensive mitigation measures enables the development of resilient entities [14]. In this situation, a honeypot is useful since it collects and updates data on recent traffic thus enhancing the ability to detect malicious activities.

When a system is accessed fraudulently, systems called honeypots are used to track, monitor, and analyze behavior patterns [11], [15]–[17]. They fall within the category of actively operating network security mechanisms and are only put in place expecting to be attacked. Attractiveness to an assailant is a crucial aspect of a good honeypot, ie. attractiveness refers to the honeypot appearing to be precisely the target that the adversary is seeking. A honeypot is a key security provision designed to compromise its assets to examine illicit accessing in search of potential security flaws in running systems thus reducing the hazards [15], [18]. It can assist in addressing the shortcomings of other existing security techniques because of its distinctive design and application capabilities. Besides thwarting attempts to attack actual systems, they enable intelligence about hostile intent, competence, and strategies. Honeypots often excel in data collection, possibly providing intrusion detection patterns, packet analysis, and inter-network analysis, as well as quick screening and fine-tuning. Some GUI based honeypots, for instance, let you choose an event message to generate ignore rules that filter out normal communications. In most cases, honeypots are installed in networks where they serve as both sensors and decoys [19].

The purpose of honeypot information security [20] is to entice attackers into fictitious environments so as to:

- Discover their interests
- How do they engage in striving to fulfill their objectives
- Explore how to halt bullying.

An enterprise typically engages a few vulnerable systems or services and allows a few plot holes to be untied while establishing a honeypot approach. The honeypot entices intruders into a secure setting by exposing sensitive information. Researchers in cyber security could watch how the offenders behave while gathering crucial data including endpoints, open ports, and file systems being downloaded. Defense-focused security teams can utilize this information to enhance security measures and put new defenses in place to flee similar assaults. Furthermore, honeypots have an advantage over firewalls since they can help offset both identified and unidentified threats, providing them the capacity to receive and handle threats that were previously unseen. Over 3,312 open-source honeypot projects were available to the public on GitHub at the time this article was written. These include large-scale open-source alliances, open-source enthusiast initiatives, and commercial honeypots. There is no dispute that there are also a significant number of customized production solutions, and research honeypots documented in articles and conference papers further increase this number.

The security community is quite excited about software containerization, a fairly young technique. To construct segregated user-space environments for their programs, leading software enterprises like Microsoft, Google, and Facebook containers operate by combining all of an application's constituent parts, such as binaries, libraries, and all of their dependencies, into a single aspect entitled *a container image* [21], were using containers.

Implementing containers simplifies the software development process, boosts safety and stability, and reduces the likelihood of configuration errors in systems by segregating processes and allowing various applications to operate concurrently [22]. Additionally, containers make system administration simpler by shifting the burden of managing program dependencies from the systems engineer to the container designer. Developers may quickly develop on one host and switch to others because of containers' portability. Applications that need to be migrated from cloud servers to more compact edge endpoints can especially benefit from portability [23], [24]. Container adaptive reuse has the possibility of lowering expenses and promoting resource efficiency [25], [26]. Security measures are continually developing, and containers are a pervasive technology. Monolithic apps, which are often expensive to redesign or split into microservices, are not suitable for containers. The possibility of cross-contamination between a production process and a honeypot can be almost eliminated by employing a cloud infrastructure [27]. Any enterprise prepared to make the first ventures into the world of honeypot deployments would profit greatly from the cloud's tremendous flexibility in terms of container honeypot deployment locations.

Our Contribution: The contributions of our works have been enumerated below:

- By leveraging containerization and a centralized network tracking framework, our work comprehends genuinely feasible active security solutions for critical system infrastructures that are the focus of increasingly complex attacks.
- Information leakages that occurred when setting up containerized honeypots were carefully explored and examined. We explored the fundamental causes of these information leaks in more depth to benefit researchers.
- Highlighted how these seemingly trivial information leaks can actually pose significant security risks to the honeypot framework by exploring several use case assaults. Though not limited, we were able to pinpoint eight use cases that, if they materialized, would lead to assaults. We also detailed attack vectors and mitigation techniques for each use case.
- Effectively designed and implemented a multidimensional deception framework based on containers, entitled *VIKRANT*, that provided enthusiasts with an ample quantity of flow data for deeper analysis of invasion.

Novelty: Although research on honeypots has increased recently, our study stands out because it is the first in-depth analysis to examine containerized honeypot models that thoroughly investigated the container technology for developing it, and enumerated potential use cases based on the challenges associated thereafter. There is widespread agreement that container honeypots have the potential to improve security. But for their performance in actual-world circumstances to be supported, an empirical investigation is required. We determine container honeypots' actual performance in spotting threats in dynamic

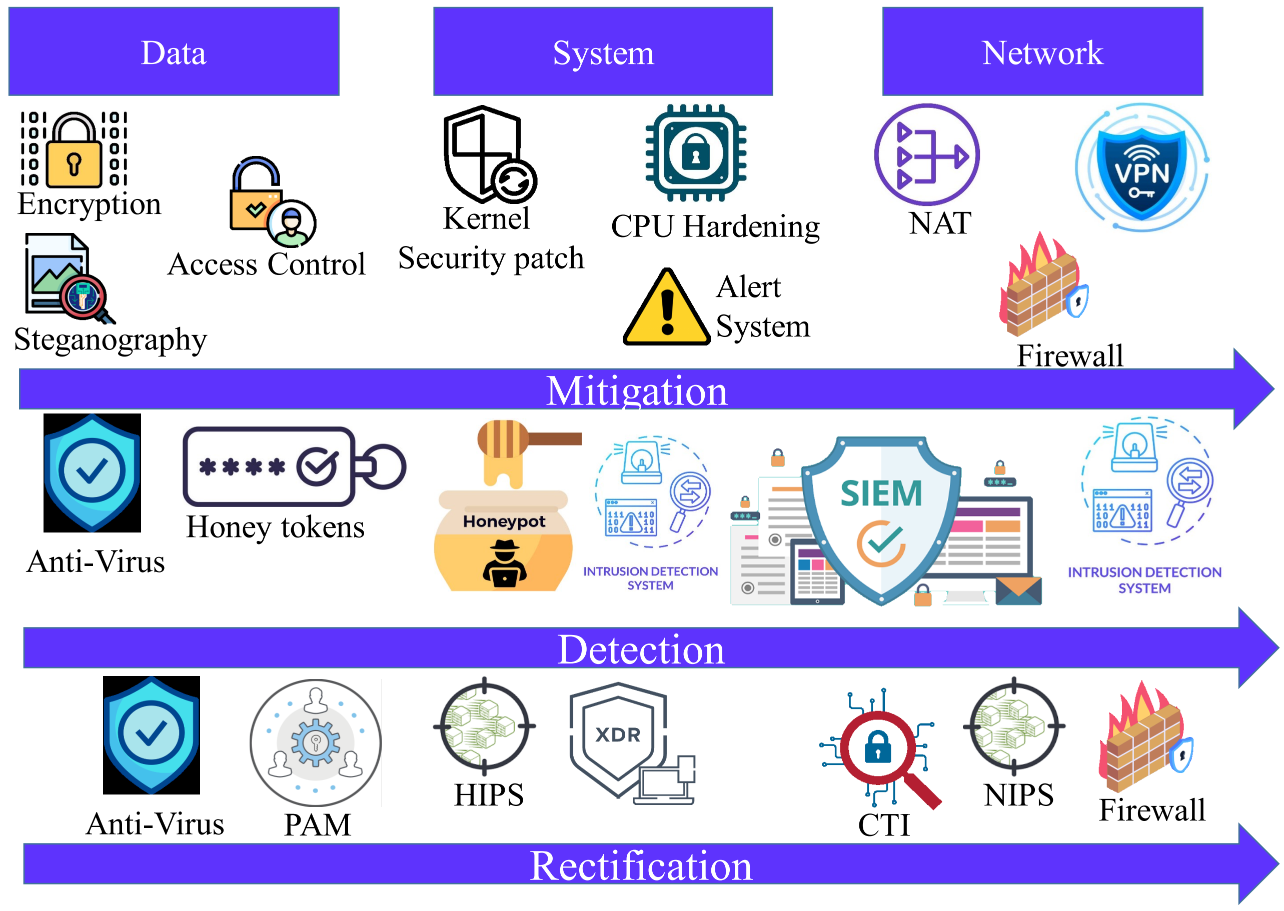


Figure 1: Breakdown of Essential Security Approaches

and complex communication contexts by putting them through tests and simulations. Our paper offers a thorough analysis of advancements in the container-based honeypot, spanning its technology and highlighting it with extensive citations from the literature. A typical honeypot survey covers a broad range of threats and attack vectors, therefore its effects may not be as significant as those of containerization. Figure. 2 represents the overall taxonomy derived from our research.

The following queries were answered by our findings:

- How could honeypots based on containers be used to protect information services against hostile intrusions?
- What benefits arise from using docker containers to deploy the honeypot?
- What factors are correlated with containers to deploy the honeypot?

The advancement of the document could be summed up as-section II offers an overview of the honeypot's key aspects, section III explains the specifics of the containerization technology utilized for the honeypot, section IV outlines the variety of threats that users of honeypot containers could experience at various points throughout the development pipeline. Section V details research findings-inspired use case investigations. Section VI describes the honeypot allied works, and section VII details the framework of VIKRANT. Section VIII discusses the results and Section IX outlines inferences drawn from our research. Section X enumerates the containerized honeypot provision for various uses, respectively. Finally, section XI concludes the work.

II. Preliminaries

A. *Workflow Anatomy of Honeypot*

All honeypots perform the same basic operations (Figure. 3). In order to entice intruders, they first disclose several services and port numbers [15]. The next step is to log at least the timeline, location, source address of the invasion (often the IP address), and payload exchanged during the connection attempt. If they are not explicitly instructed to be ignored, all connectivity initiatives should be logged and flagged so that an incident handling team may get activated. The significant advantage of an effective honeypot is thus utilized in data analysis, whether that be by performing deep packet inspection, login attempt analysis, or grouping similar inquiries into a single event. The evaluation relies on how adeptly and successfully each honeypot does this.

B. *Emulation services and levels*

Every honeypot typically replicates one or more applications, and in order to do so, they have to be configured to listen on the appropriate TCP and UDP (including ICMP)

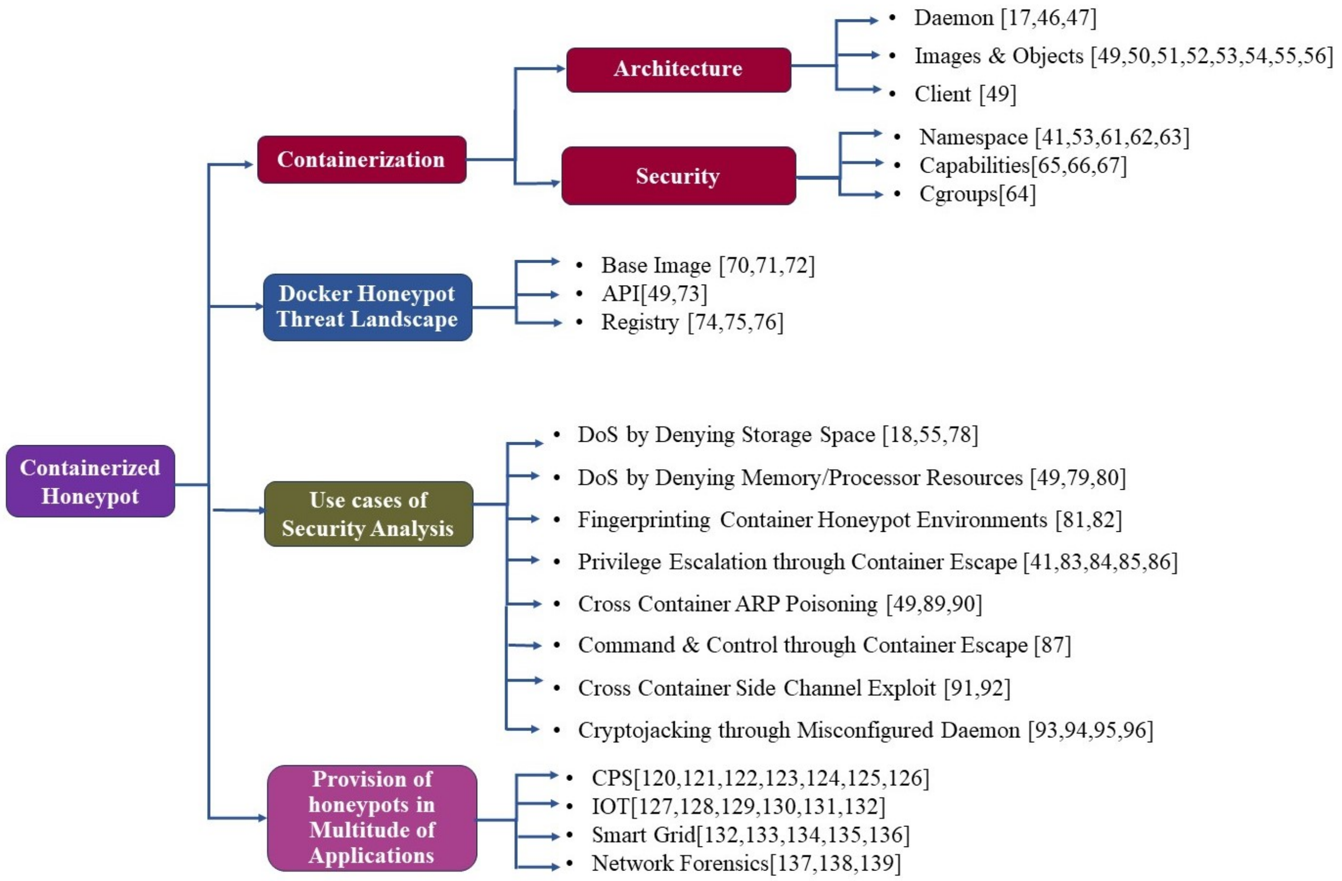

Containerized Honeypot
Containerization
Architecture
• Daemon [17,46,47]
• Images & Objects [49,50,51,52,53,54,55,56]
• Client [49]
Security
• Namespace [41,53,61,62,63]
• Capabilities[65,66,67]
• Cgroups[64]
Docker Honeypot Threat Landscape
• Base Image [70,71,72]
• API[49,73]
• Registry [74,75,76]
Use cases of Security Analysis
• DoS by Denying Storage Space [18,55,78]
• DoS by Denying Memory/Processor Resources [49,79,80]
• Fingerprinting Container Honeypot Environments [81,82]
• Privilege Escalation through Container Escape [41,83,84,85,86]
• Cross Container ARP Poisoning [49,89,90]
• Command & Control through Container Escape [87]
• Cross Container Side Channel Exploit [91,92]
• Cryptojacking through Misconfigured Daemon [93,94,95,96]
Provision of honeypots in Multitude of Applications
• CPS[120,121,122,123,124,125,126]
• IOT[127,128,129,130,131,132]
• Smart Grid[132,133,134,135,136]
• Network Forensics[137,138,139]

Figure 2: Taxonomy of the Paper

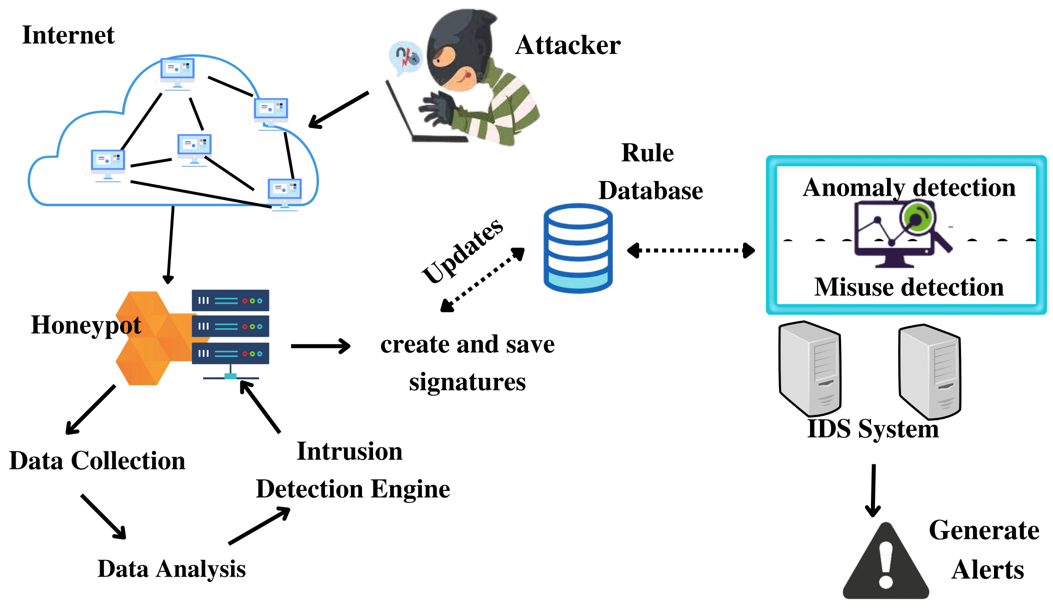


Figure 3: Workflow of Container Honeypots coalescing with IDS System

ports. Numerous honeypots replicate only a minimal number of ports. A port that has previously been bound to the host machine cannot be used by a honeypot. For instance, in order to simulate NetBIOS features, Windows-based honeypots must disable files and printer sharing on a host and SMB(Server Message Block) / CIFS(Common Internet File System). The majority of attackers do not perform network analysis and fingerprinting. They seek out a port, discover it, and swiftly attempt to learn what it is running. Network stack emulation is crucial when an attacker uses a comprehensive fingerprinting platform (like Nmap and Xprobe2) [28], which tends to happen in a low minority of incidents.

C. Logging and Alerting

Without reliable warning and tracking, a honeypot is unproductive [29]. On sensors or a centralized interface, all honeypots show connectivity incidents as flags. Criticality grades at each sensor, originating Internet address, port, or even attack fingerprint ought to be possible to set in alerts. Even though certain probes to a honeypot are more suspicious than others, all should be scrutinized. For instance, a probe coming via a more secure site can signal a more severe vulnerability. The majority of honeypots permit fine-tuning future warnings based on the existing alarms, usually to segregate actual traffic. A robust honeypot makes the process of fine-tuning faster and easier. The logs collected by a honeypot could be coupled to firewall logs, IDS alarms, and many other system logs. As a result, a holistic profile of suspicious transactions inside an organization may be constructed and is possible to set up alerts that are more appropriate with fewer false positives. A SIEM (Security Information and Event Management) can receive alarms from the honeypot or can be inspected straightforwardly in the honeypot manager's dashboard. While properly configured honeypots can act as high-priority SIEM alarms, they shouldn't produce as many logs as Sysmon (System Monitor) or Windows Activity logging. When the only alerts produced are those that indicate an assault, it is simpler to filter out false positives due to this strategy.

D. Approaches for Deployment

Sacrificial Lamb (SL), Proximity Decoys(PD), Deception Port (DP), Minefields, and Redirection Shield (RS) are five effective forms for deploying decoys [15]. A typical system that is willing to be penetrated by attackers, such as Cuckoo Sandbox [30], is regarded as a sacrificial lamb. It has no alliances with production networks. It could be an off-the-shelf PC from a store, a switch, or a gateway. In a typical implementation, the OS is loaded, a few services are enabled, and then the system is left on the network to see what actually occurs. Additional network implications would be necessary as they lack integrated traffic screening capabilities. Deception ports are honeypots that imitate numerous services on multiple ports. For instance, on port 80, SMTP is imitated, on port 20 FTP is mimicked, and so on. In simplest terms, these honeypots observe the OS they are using and then portray these services in that context. These kinds of honeypots are frequently seen in Honeyd honeypot [31], [32]. A content rich enhancement to this methodology of using honeypots is Spectre [33].

Proximity Decoys(PD) method is intended to be the most efficient and least noticeable of all for legal reasons. When the honeypot joins the same subnet as the primary servers, it becomes a part of their network and are permitted to keep an eye on activity related to the network. Additionally, if a malicious assault is discovered in the proximity of the production environment, it will be simple to either record the assault or reroute traffic. This helps to prevent the spread of viruses and worms. Research honeypot deployment using VMware and User mode Linux are two examples of these types of honeypots [34].

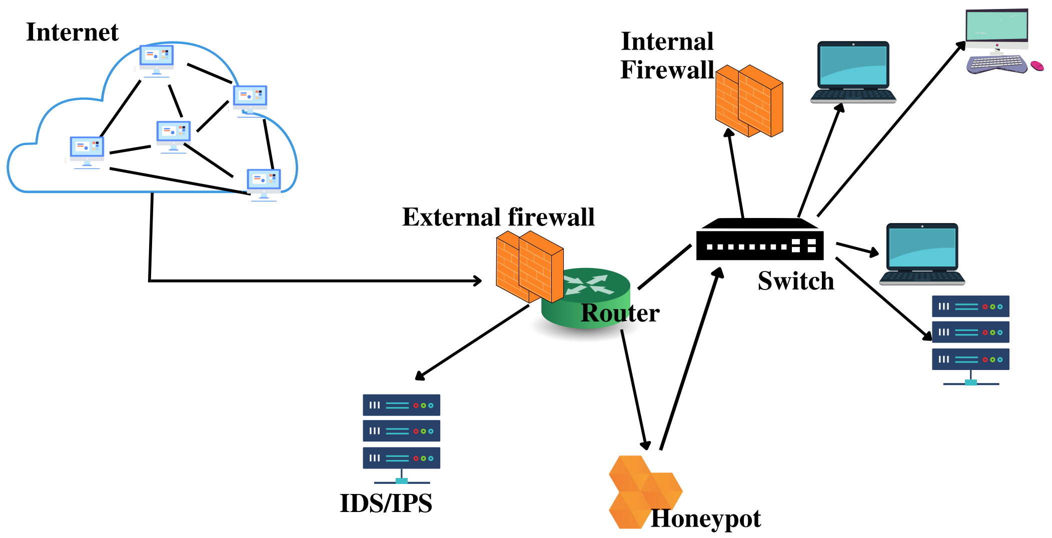
Redirection shield honeypots are essentially taking the place of production environments by redirecting ports or rerouting traffic (refer Figure. 4(a)). These measures serve as barriers, therefore will soon be the most advanced constituent of honeypots seeking commercial application. It could be expanded to offer commercial services to larger networks. Minefield deployment of honeypots is designated at the outer edge of the organization's network (refer Figure. 4(b)). They explode when they get hit (usually by an intruder). IDS and vulnerability scanners are however installed in the network to assure absolute security. Consequently, both leverage the information in the honeypots to raise alarms rather than keeping an eye on the production servers. Hence, there is only a minimal chance of false alarms being generated. These honeypots, which trap, trick, track, or tarpit intruders, essentially serve as a third line of defense. LaBrea and Honeyd, both when employed in passive mode [35], and Mantrap [36] are among examples.

To yield the most relevant (and helpful) results, a honeypot must be connected in some way to the entity that a potential attacker is interested in. This could be accomplished using a licensed web domain or even a phony company website. Furthermore, installing a honeypot inside the company's existing cloud perimeter, if feasible, can also aid in the detection of targeted assaults, however, its seclusion should be meticulously orchestrated.

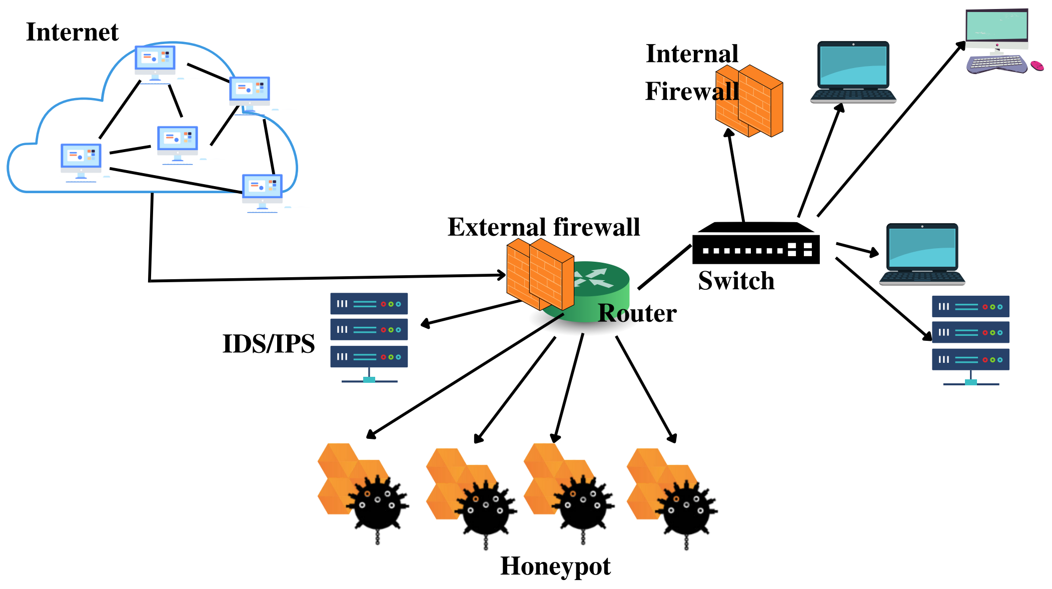
E. Challenges in Honeypot

For an expert, it is not hard to distinguish honeypots as bait devices. Low interaction honeypots rarely manage intricate services underneath since they are mimicked [37]. The system could be successfully undermined by making such an effort. Since we are dealing with the network stack in honeypots with modest levels of involvement, finding information over the network becomes critical. For instance, honeypots like Honeyd [32] frequently cut off connections when they are unable to handle them. Honeyd disconnects the connection if the *SYN* package has a related vulnerability. With this knowledge, hackers can use any tool that can help in verifying connections through a honeypot to discover the system as a honeypot [38]. The output of the program itself will disclose any dropped connections to an intrusive party.

In containerized honeypots, performance and efficiency improvements are made feasible by employing a single shared kernel, however, this also makes container breakout attainable



(a) Redirection Shields



(b) Minefield

Figure 4: Redirection Shields and Minefield Deployment Strategy of Honeypots

with a simple exploit. The most prominent challenge is the possibility that an attacker may successfully use a honeypot as a launching pad for a lateral breakthrough into the real production network. Hence required to keep it separate from all other networks. This may seem like a straightforward process, but all it takes is one abandoned system or one firewall rule modification to turn a basic task into something quite harmful [18]. The time commitment and associated expenses involved in managing a honeypot present another challenge. Inherently, configuration and renovations of the system are necessitated. Not just that, the organization's security personnel must utilize the captured behavior in order to make it useful. The organization of this system and its integration with security protocols will take a significant amount of time. The information could result in actionable intelligence, such as the ability to block an adversary's connectivity, set up rules for an intrusion prevention system, even create or update malware signatures.

By hosting a honeypot on a public cloud system, some of the aforementioned difficulties can be circumvented. Complete separation from any production network is offered by the public cloud [29], [39]. Additionally, there is no requirement for specialized technology or exclusive infrastructure connections. Snapshots can be used to restore a system to its previous configuration prior to the intrusion once a system has been compromised and the data has been gathered. By choosing the appropriate geographical regions inside the cloud system setup, a cloud computing infrastructure can be deployed anywhere in the world for a honeypot implementation. With a couple of button presses, a detector may be set in East Japan one day and can be migrated to Central India the next day. This is excellent for research and information gathering because detectable intrusions and adversaries can vary greatly relying on where the vulnerable device is housed. Some honeypot solutions, like the Thinkst (Cloud) Canary [40], have also been designed around a private cloud server. Devices called canary honeypots are placed in key areas of the client's connection. When an actual perpetrator inadvertently interacts with any of these sensors, they instantly communicate to a single cloud-based system, enabling the client to observe peripheral traffic as well as lateral displacement within the production process. When it comes to honeypots, positioning and significance are also directly correlated. The usage of honeypots has legal and regulatory implications as well. Several providers of cloud services need not appreciate the thought of letting hackers penetrate their infrastructures and depositing malware on their systems [24]. Admittedly, there is a likelihood that if the host is compromised, it may then be utilized to exploit other online targets which inturn can pollute the cloud provider's credibility.

III. Containerisation in Honeypots

Virtualization is a technique that offers an interface for external spectators to view internal operating system (OS) activities. Virtual Machines (VMs) streamline the isolation between a computer system's virtual space and the real system underneath it, enhancing security against network service breaches [41]–[43]. A monolithic operating system manages entire firmware resources in a nonvirtualized system. A new layer of software called the Virtual Machine Monitor(VMM) is present in a virtualized system that builds a Virtual Machine(VM) environment. Traditionally, honeypots have been set up as virtual computers or as hypervisor services. Criticalities brought on by the overheads added by hypervisors, especially for low-end embedded devices, necessitated further research into strategies based on lighter virtualization options, like containers. Virtualization at the OS level, like Linux Containers or Docker containers, has gained traction with the continuous infrastructure transfer to the cloud for a number of reasons. The use of the Docker Engine and containers makes Docker one of the most popular open-source implementations of this. Since operating system images are not there in containers, they consume minimal resources than conventional or hardware virtualization environments. A container image created for an application and all of its dependencies are published in a public/ private registry so that an infinite number of users can operate in the same application context [44], [45]. Due to Docker's popularity, the terms docker and container have been used interchangeably in this article.

A. *Configuration*

Honeypots must be properly set up to draw in the right victims, much like realistic traps [17], [46]. As a result, the programmers of the honeypots should deliberate about what events ought to be watched, whose transactions should be observed, when certain initiatives should be noticed, and so on. A honeypot that is improperly set up not only keeps failing to attract its victim but also leaves itself vulnerable to hijacking by adversaries [17], [47]. Additionally, it might be very challenging to interpret the data after blindly gathering a huge amount of network events. Henceforth, it is crucial that honeypots are set up to achieve certain goals.

In a typical design, the gateway should ideally be the only connection made to the sacrificial host; it shouldn't have access to the internet absolutely. A prebuilt VM image with Docker container installed ought to be used to maintain this host updated. Using various tool like Packer [48], the updating of this docker VM image can be automated. The gateway host should be configured so that it may be seen from the Internet. The sacrificial VM should ideally run on its own exclusive physical hardware to eliminate information disclosure as an outcome of CPU vulnerabilities.

B. *Docker Architecture*

The Docker daemon, which controls the formation, execution, and propagation of Docker containers, and the Docker client interact according to a client-server configuration [49](refer Figure. 5). It is possible for a Docker client to operate on the same system as the Docker daemon or to connect to a remote Docker daemon. The client communicates to the daemon using REST API, UNIX sockets, or a network interface. A series of containers can be interacted with using Docker Compose, which is another Docker client.

1) Docker daemon: Docker daemon (dockerd) handles Docker objects like images, containers, networking, and volume while observing Docker API requests. Daemons can interact with one another to administer Docker services.

2) Docker Client: Docker Client is the preferred method of communication for Docker users(docker). It articulates with the daemon to manage the formation, administration, as well as dissemination of containers. When a command like *docker run* is used, the dockerd service receives and executes this command from the client. Docker API is utilized by the docker command. Additionally, the client interacts with several daemons.

3) Docker Objects: Users build and employ images, containers, networks, volumes, plugins, and other objects whilst using Docker [49].

- Image: An image, which is a read-only template, is used to generate the Docker container. Often, an image is based on another image and then modified [50], [51]. An image is created by layering on top of an earlier one and then subsequently modified. To construct an image, write a Dockerfile with a concise structure specifying the actions required to build and execute the image. Every command in a Dockerfile adds a new layer to the image [50], [52].
- Container: A container is a runnable snapshot of an image [53], [54]. By using Docker API or CLI, a container can be created, started, stopped, moved, or destroyed. A container can attach memory to it, communicate with

multiple ports, or generate a new image based on its existing state. A container is often quite well segregated from its host system as well as other containers. Each container has its own networking stack.
- Volumes: The suitable technique for storing data produced by containers is through volumes. Bind mounts depend on the host machine's Kernel and directory structure, however, Docker manages storage volumes predominantly [54], [55]. The volume's composition is persistent outside of a specific container's lifespan.
- Network: Network drivers used by Docker provide support for networking containers. The bridge and overlay drivers are the two network drivers that Docker by default makes available. However, writing a network driver plugin to construct drivers is a laborious operation [56].

C. Upperhand of Containers in Honeypot Deployment

Being that they are frequently utilized to isolate a specific application, containers seem to be a lightweight substitute for full-machine virtualization [42], [57], [58]. The concept of microservices has made containers extremely popular. There is no requirement for a bootup period as containers use the kernel of the host OS and hence containerized application could be deployed in a few seconds. Docker is an open-source software framework for creating, deploying, and managing virtualized container technology on a shared operating system (OS) [59]. Docker has swiftly gained popularity due to the advantages that docker containers offer. Docker's primary benefits include rapidity, mobility, scalability, faster service, and density [60].

- While the programmer is in charge of maintaining the apps inside the docker container, the administrator is in charge of deploying and maintaining the server containing containers. Since all necessary dependencies are included in the programmes and are fully tested, containers can operate in any environment.
- Applications developed using Docker containers are portable that can be transported readily as a single package while maintaining the same efficiency.
- Scalability can be illustrated, for instance, a web application executing inside a Docker container and hosted by the Apache web server linked to a MySQL database backend running inside a separate Docker container. Without changing the configuration of the front end or back end, the application can scale by adding more Apache nodes to handle more web traffic.
- Since containers are so small, the time needed to build one is amazingly fast. As containers are compact, construction, debugging, and distribution may be accomplished more quickly. After being developed, containers can be pushed for testing before being used in the production environment.
- Since all the components required to operate a program are contained within the container, containers also alleviate platform compliance concerns. Then, using the same operating system as all other containers, the image can operate in a contained environment. The host operating system places restrictions on the container's ability to utilize the system's physical resources, thus one container cannot exhaust the host's resources entirely. As a result, any issues and challenges in a certain container will only affect that container.

D. Containerization for Security

Integrating security features into the steadily growing container system is one area that requires a significant amount of work. The containerization of honeypot applications has enormous potential, and several noted advantages of this strategy are outlined here [53].

- Trivializing and Customizing: The concept that containers are virtually replaceable is one of the most appealing features to use them for honeypot implementation. It is simple to delete a container completely and re-deploy it with no system disturbance if it is compromised by malware [49]. Containers allow for fine-grained administrator management and are extremely configurable. Everything, including execution and permissions, can be regulated allowing control over the resources that a certain container is allowed to access.
- Container Isolation: Three key Linux Kernel mechanisms [53]— Namespaces, CGroups, and Capabilities —achieve containerized honeypot isolation.
  1) Namespaces: A key Linux feature for resource isolation across many containers is namespaces. Entity table instances along with other constructs linked to kernel global resources can be separated using namespaces. They segregate hostnames, system files, users, services, and other components. There will be distinct mount tables and root directories for each system files namespace as a result [41], [53]. For instance, the PID namespace ensures that a job will only view activities that are in its own PID namespace. Many researchers examined the Docker container escape issue and provided a security strategy based on namespace status [61] and information leakage pathways inside containers [62], [63].
  2) CGroups: Linux features that regulate resource utilization such as including CPU running time, storage, input/output, and communications bandwidth, impose liability for resource allocation. CGroups restrict the number of resources that can be used in contrast to namespaces, which regulate which assets a container honeypot could see. Further, it prevents containers from consuming all resources and depriving other processes. ContainerDrone, a real-time defense against DoS assaults presented by Chen et al. [64], makes use of cgroup solutions.
  3) Capabilities: The divergence between roots privilege and non-roots privilege is transformed into fine-grained access control via capabilities [65]. As a result, containers typically do not need complete root access(assuming the capability exists to perform the necessary activities). There are forty-one capabilities that can be used for a range of activities [66]. For instance, *CAP_NET_BIND_SERVICE* allows a port below 1024 to be opened by the container, *CAP_KILL* avoid requesting authorization before sending signals. SELinux, as implemented in RedHat, or MAC (Mandatory Access Control)

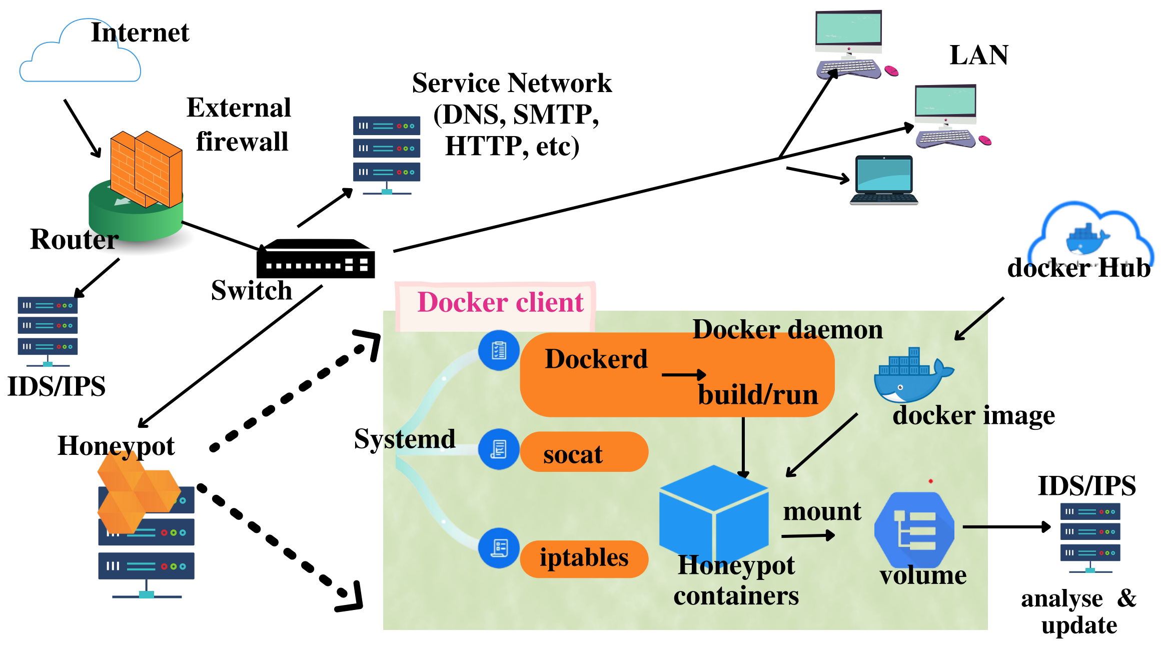


Figure 5: Comprehensive process flow of the containerized honeypot, including its underpinnings

techniques like AppArmor, used in Ubuntu are additional honeypot container security measures [67]. Conversely, it is debatable if the confinement layer is sufficient for a honeypot solution considering that the OS is shared.

- Docker and IPtables: When Docker honeypot is installed on a host that is connected to the internet, IPtables rules can be set to limit access to containers and other services that are running on the host [68]. The two custom iptable chains DOCKER-USER and DOCKER that Docker installs make sure that incoming traffic always is examined foremost. The DOCKER chain contains all of Docker's iptables regulations. This chain should not be calibrated. If required, add rules to the DOCKER-USER chain that execute prior to Docker's rules.

E. *Container Honeypots versus Conventional Honeypots*

Compared to conventional honeypots, container honeypots use a fundamentally different technology. A container honeypot leverages containerization technology and operates inside a sandbox container with its own distinctive file system, activities, and communications. This isolation adds an additional degree of protection by barring successful assaults from having an effect on the underlying system. Traditional honeypots tend to be resource-intensive, mandating memory, CPU, and storage space on a dedicated machine. Contrarily, container honeypots are compact, reliable, and efficient since they share the underlying system's kernel, thereby minimizing overhead. The speed of deployment is another significant distinction. It can take quite a while to set up a typical honeypot as the operating system must be manually programmed and a separate machine must be provisioned. However, container honeypots can be quickly set up by utilizing assembled images and platforms like Docker. A dedicated IP address, which is possibly limited in quantity, is frequently needed for standard honeypots [69]. Container honeypots, on the flip hand, could utilize exactly the same IP address as the system that hosts them, optimizing resource utilization. In addition, container honeypots use namespaces as a way to isolate their internet traffic and increase security, while traditional honeypots require a distinct network or VLAN for effective isolation.

Frequent modifications, upgrades, and inspection of the underlying OS are required for standard honeypots in terms of administration and support. In contrast, container honeypots gain access to security patches and updates of the host system, which requires minimal maintenance. Container honeypots also provide better scalability. Conventional honeypot scaling may turn out to be difficult as it is resource-intensive, necessitating the use of additional specialized systems. Container honeypots are highly efficient for large-scale implementations since they expand simply by adding new containers. Besides, they utilize resources more effectively, enabling greater distribution of computational power. In terms of security, container honeypots offer a better level of protection by containing potentially harmful behaviors in a controllable setting. They can be readily replaced in the event of a compromise, decreasing downtime as well as the likelihood of damage. Due to their effective resource usage and flexibility to operate on already existing facilities, they could be even more cost-effective. Ultimately, they are beneficial in contemporary security initiatives as they offer a sandboxed setting for analyzing attacker action, facilitating effective research of possible threats. The comparison is summarised in Table I.

IV. Docker Honeypot Threat Landscape

Since containerization technology is still relatively new compared to full virtualization, deploying containers as honeypots has generated considerable discourse. Furthermore, it

Table I: Summary of Container Honeypots v/s Conventional Honeypots

Items	Normal Honeypot	Container Honeypot
Isolation	Runs on host system	Runs in isolated container
Resource Utilization	Resource-intensive, May require dedicated IP address	Lightweight and efficient, Shares IP with host system
Speed of Deployment	Takes time to set up	Rapid deployment using containers
Scalability	Limited	Easily scalable with containers
Management/Maintenance	Requires OS maintenance	Benefits from host system updates
Scalability	Limited scalability	Easily scalable with containers
Recovery Time	Slower recovery after attack	Faster recovery with container replace
Flexibility	OS dependent	Container dependent
Security	Potentially less secure	Enhanced security with isolation

could be simple for some administrators to disregard potential setup concerns like the ones pictured following.

A. *Base Images*

The base images themselves can contain the majority of the vulnerabilities [70]. When using Docker, the base image is created from an operating system that is often modified from another well-known image that is available in Docker Hub [71], [72]. Besides the original base image, modifications—whether intentional or unintentional—create several opportunities for assaults, possibly even by the most unskilled hackers. Attackers may post images to Docker Hub containing files laced with viruses, malware, or various potentially hazardous programs. These harmful codes can be unknowingly activated by users who download these images.

B. *Docker Engine API*

For container management, the Docker CLI(Command Line) makes use of the Docker Engine API [49]. This API can also be directly accessible by remote applications on Linux systems for debugging and automation purposes like a REST API. An API is a set of rules or an interface for programming applications. Regardless of how it is implemented, it is a standard method for communicating amongst programmes of a wide range. Representational State Transfer, REST [73], is a convention that programmers can use to obtain and transfer files with other apps by sending a request for data in the form of a specific URL and retrieving information in the payload of the returning message. The Docker API may be used by an attacker to escalate their privileges and get access to the host system if they manage to access a container with elevated privileges.

C. *Docker Hub*

Although fraudulent Docker containers have now been deleted from Docker Hub, the public repository's vulnerability to misuse and abuse persists [74], [75]. A supply chain assault, in which an intruder takes control of a trustworthy image or repo and then utilizes it to spread fraudulent software to naive consumers can leverage Docker Hub. Caution should be exercised while using base images downloaded from Docker Hub. However, since many administrators feel that convenience frequently outweighs the risk, implementing sensible security precautions minimizes the risk of an attack. The Threats, Attack Vectors, and Mitigation Strategy overview for the Honeypot Container Development Pipeline are outlined in Table. II. The mapping of threats to real-world exploits is depicted in Table. III [76].

V. Usecases of security analysis of Container Honeypots

The analysis relies on the assumption that the hacker has successfully accessed the container honeypot and obtained root privilege(uid=1) command line capability. Compromised Docker Honeypots are used for various attacks like [77]. The following use cases for honeypot security assessments were deduced during our examination (aspects of potential attacker behavior) as depicted in Figure. 6.

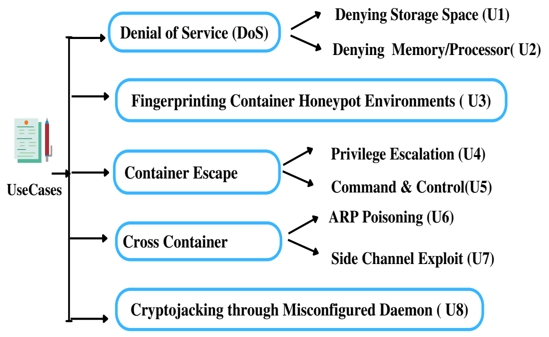

UseCases
Denial of Service (DoS)
Denying Storage Space (U1)
Denying Memory/Processor(U2)
Fingerprinting Container Honeypot Environments (U3)
Container Escape
Privilege Escalation (U4)
Command & Control(U5)
Cross Container
ARP Poisoning (U6)
Side Channel Exploit (U7)
Cryptojacking through Misconfigured Daemon (U8)

Figure 6: Usecase Observations Assembled by Research Findings

A. *Usecases*

1) DoS(Denial Of Service) by Denying Storage Space, U1: Maintaining logs of the attacker's activities is one of the functions of the honeypot [18]. In the case of the Containerized Honeypot they are typically kept in a permanent storage location that the container could navigate. As a result, the attacker may employ a security vulnerability to prevent other containers from accessing storage space, and consequently, from being executed. Another rationale is to stop Honeypot from storing the recorded actions and thereby just erase the attack's footprints [78]. The core argument of vulnerability is that there isn't much storage space available.

The placement of Docker files varies per operating system [55]. It can be found at */var/lib/docker/* for Ubuntu, Linux and fedora, at *C:\ProgramData\DockerDesktop* for windows, and at *∼/Library/Containers/com.docker.docker/Data/vms/0/* for MacOS. Volumes and bind mounts are two options provided for containers to archive logs on the host so as to persist them when the container shuts down. There is a third option for Linux users using Docker: mounts tmpfs (Figure. 7 illustrates a clear distinction).

*Lessons Learnt:*Examining the use case offers significant revelations that advance our comprehension of container security and the function of honeypots in identifying and thwarting such attacks:

Mitigation Strategies could include:

- Isolated Honeypot Setting: Establish the honeypot in an isolated region with restricted accessibility and interaction from outside. To block illicit access to the honeypot, use network segregation, network regulations, etc.

Table II: Threats, Attack Vectors, and Mitigation Strategy Overview for Honeypot Container Development Pipeline

Threat	Attack Vector	Mitigation Strategy
Image Tampering	Attacker gains unauthorized access to the image repository and makes changes to a container image without permission.	- Use access-controlled, secure image registries. - Use digital signatures to confirm the integrity of images. - Use ongoing image repository monitoring and auditing.
Container Breakout	Exploiting vulnerabilities to escape container isolation and access the host system.	- Update the host system's and the container runtime's security frequently. - Use secure container configurations best practices. - Use security tools like SELinux or AppArmor.
Malicious Payload	Using containers that include malware, ransomware, or other hazardous software pre-installed.	- Before deployment, check container images for malware. - For runtime monitoring, use container security tools. - Use least privilege concepts for determining container permissions.
Service Exploitation	Attacker takes advantage of flaws in simulated services.	- Keep pushing security updates to emulated services. - Use network segmentation to separate honeypot traffic. - Keep an eye out for any exploitation attempts or strange behavior in service logs.
Harvesting of credentials	Attacker targets authentication methods in an effort to collect login credentials.	- For emulated services, implement multi-factor authentication. - Monitor for patterns in login attempts that point to credential harvesting. - Inform users about safe login procedures
Network Inspection	Attacker performs network scans to find possible targets or weak points in a system.	- Use network segmentation to reduce the visibility of honeypots. - Use intrusion prevention systems or firewalls to find and stop scanning activity. - Maintain a check for strange patterns in network traffic.
Exfiltration Attacks	Attacker tries to steal potentially sensitive data from the honeypot environment.	- Use data loss prevention (DLP) tools to keep an eye on and stop unauthorized data transfers. - Implement encryption and access restrictions for the honeypot's sensitive data. - Keep an eye out for oddities in outgoing traffic.

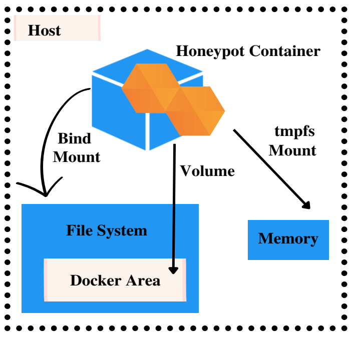


Figure 7: Storage Pathways of Bind Mount, Volume, and tmps Mount in Docker Linux Honeypots

- Snapshots of containers: When employing containerized honeypots, periodically acquire a snapshot of the container's state, particularly the logs. In the event that logs have been interfered with, snapshots can be utilized as proof.
- Certificate of Custody: Retain a thorough record of the line of possession for log files, documenting whoever retrieved, altered, or distributed the logs. This improves transparency and accountability.
- Centralized storage and immutable logs: Implement immutable logging for your containerized honeypot, using technologies like blockchain or write-once storage. Send logs to a centralized and secure logging server that resides outside the container environment. This prevents attackers from directly tampering with logs within the container.
- Establish read-only filesystems: Set containerized honeypot with a read-only filesystem. As a result, hackers are unable to change or delete log files that are kept inside the container.
- Monitoring of execution behaviour: Implement runtime security tools that keep watch on the activity of containers in real-time. Any effort to alter or delete files could lead to alarms and swift responses.
- Security profiles for containers: Use resource segregation policies and security policies for containers to limit access to sensitive data, such as logs, and enforce resource isolation.
- Replication and backup: Frequently archive container logs to a safe location off-site. Keep backup copies of logs to ensure access even in the event of a breach.

2) DoS(Denial Of Service) by Denying Memory/ Processor Resources, U2: Denial of memory and processing resources constitutes further potential danger. Both the host memory status (*/proc/meminfo*) and the processor resources (*/proc/procinfo*) are accessible for every container instance. A container has no resource restrictions by default and is allowed to use all of the given resources that the host's kernel optimizer permits, ultimately preventing the execution of other containers on the host [49]. By looking at the Htop(interactive real-time process monitoring on Linux/Unix-based platforms) output, the potential vulnerability could be observed. On Linux hosts, the kernel will throw an OOME(Out Of Memory Exception) and initiate terminating processes to release memory if it finds that there is insufficient memory to carry out crucial system functionality [79], [80]. Any operation, including Docker and other vital applications, is vulnerable to termination. If the incorrect process is terminated, could theoretically bring the whole system to a halt.

*Lessons Learnt:*The weaknesses and mitigation measures

Table III: Mapping of Threats to Real-world Exploits

Threat Impact	CVE ID	CVSS Score	Description
Image Tampering	CVE-2019-3841	6.8	When importing data into PVCs from container registries, Kubevirt/virt-cdi-importer, releases 1.4.0 till 1.5.3, reportedly suppresses TLS certification checking. The result could expose the container registry to attacks such as man-in-the-middle, resulting in allowing trusted container image files to be altered without being noticed.
Container Breakout	CVE-2014-3519	6.5	When utilising simfs, the open_by_handle_at procedure in the vzkernel version 042stab090.5 of the OpenVZ patch for the Linux kernel 2.6.32 could allow native container consumers with specific capabilities to get around a container's envisioned protective mechanism and access any file on a file system.
	CVE-2020-28914	7.1	Kata Containers before version 1.11.5 can be compromised due to incorrect file permissions. While installing a file or directory as read-only within a container employing a Kubernetes hostPath Volume, it is mounted as read-only within the container but remains editable inside the guest system. In the event of a container breakout, a malicious user could change or remove files and directories that should be read-only.
	CVE-2021-30465	8.5	Container Filesystem escape via Path Traversal could be achieved with runc previous to 1.0.0-rc95. An attacker builds several containers with particular mount configurations in order to take advantage of this vulnerability. A race condition-based symlink-exchange attack is what causes the issue.
	CVE-2021-4154	8.8	The Linux kernel's cgroup v1 parser was found to have a use-after-free bug. By making use of the fsconfig syscall argument, an insider with user privileges could elevate their privileges, resulting in a container breakout and a DoS on the system.
Malicious Payload	CVE-2021-43784	5	By simply inserting their own netlink payload that disables all namespaces, an attacker could take advantage of this vulnerability and get around the container's namespace constraints, although they would need some control over the settings of the container to do so. The people who permit untrusted images with untrusted configurations to execute on their computers are those who are most negatively impacted.
	CVE-2023-36457	8.8	A management interface for Linux server maintenance and operation is called 1Panel. When adding container repositories, an authenticated attacker could deploy an illicit payload to perform injecting commands prior to version 1.3.6. In version 1.3.6, this flaw had been resolved.
Service Exploitation	CVE-2017-2935	8.8	When handling the Flash Video container file format, Adobe Flash Player versions 24.0.0.186 and prior have an exploitable memory overflow vulnerability. A successful exploit could result in the execution of arbitrary code.
	CVE-2018-11756	9.8	If the user-supplied code in the PHP Runtime for Apache OpenWhisk is susceptible to code exploitation, an attacker could replace it within the container using a Docker action inheriting from one of the Docker tags openwhisk/action-php-v7.2:1.0.0 or openwhisk/action-php-v7.1:1.0.1 (or earlier).
	CVE-2018-11757	9.8	If a user function inside a container inherits the Docker tag openwhisk/dockerskeleton:1.3.0 (or older) and is subject to code exploitation, an attacker may be able to replace it. This is true for Docker Skeleton Runtime for Apache OpenWhisk.
	CVE-2022-39395	9.9	A Golang-based Pipeline Automation (CI/CD) framework, Vela, is based on Linux container technology. Vela administrators can change the worker's setting to explicitly void, use the server component's setting to limit access to a list of repositories that are allowed to be enabled, and disable pull requests if they are not required.
	CVE-2022-41942	7.8	A command Injection vulnerability in the gitserver service was discovered by Sourcegraph, a code intelligence platform. A lack of input validation on the endpoint's host parameter led to this vulnerability. It was possible to craft a request to be sent to the gitserver, which would run commands inside the container.
	CVE-2022-46756	6.7	Versions of Dell VxRail before 7.0.410 have a vulnerability, Container Escape. This vulnerability could be used by a local, high-privileged attacker to execute arbitrary OS commands on the operating system under the container.
	CVE-2023-41319	7.2	It is possible to upload customized integrations as a ZIP file using the Fides webserver API. Although Fides can be set up to accept the addition of customized Python code, this ZIP file must contain YAML files. It is possible to go around the sandbox and run any arbitrary code in the target system.

Table III: Mapping of Threats to Real-world Exploits (continued..)

Harvesting of Credentials	CVE-2016-8954	9.8	Given that the credentials for IBM dashDB Local are hard-coded, a remote attacker could be able to access the database or Docker container.
	CVE-2017-1000113	5.5	Passwords were kept in plain text as part of the setting of the Deploy to Container Plugin. This made those passwords accessible to people with Jenkins master local file system access or users with Extended Read access to the jobs it is used in.
	CVE-2017-12351	5.7	An authenticated, local attacker could access and send packets that are outside the boundaries of the guest shell container through the Cisco NX-OS System. By performing "Unauthorized Internal Interface Access," a hacker could take advantage of this vulnerability. For this attack to be successful, an attacker would require legitimate administrator credentials.
	CVE-2018-1223	8.8	Prior to version 0.14.0, Cloud Foundry Container Runtime (kubo-release) could publish UAA and vCenter credentials to application logs. These credentials could be used to increase privileges by a malicious user who has access to read the application logs.
	CVE-2018-15763	8.8	IaaS credentials are exposed to application logs in Pivotal Container Service editions earlier than 1.2.0 due to information exposure. These credentials may be obtained by a malicious user with access to application logs, who could then use them to carry out actions.
	CVE-2018-5543	8.8	The BIG-IP username and password are passed as command line parameters for Kubernetes 1.0.0-1.5.0, which make the container's credentials public.
	CVE-2019-10200	7.2	A bug in OpenShift Container Platform 4 allows users with access to create pods to schedule workloads on master nodes by default. The master AWS IAM role's security credentials can be retrieved by pods running on master nodes that have permission to access the host network, providing administrative access to AWS.
	CVE-2020-10750	5.5	Before version 1.18.1, when the Kafka data store was utilized, there existed a vulnerability in jaegertracing/jaeger that allowed sensitive information to be written to a log file. An attacker who has access to the container's log file can use this weakness to learn the Kafka credentials.
	CVE-2020-11075	9.9	A shell escape vulnerability in the anchore engine analyzer service during an image analysis process can be triggered in version 0.7.0 of the Anchore Engine by using a carefully constructed container image manifest that is fetched from a registry. In the event of a successful attack, commands that run in the analyzer setting can be executed with identical privileges as the user that the anchor engine operates.
	CVE-2021-3602	5.5	When creating containers utilizing chroot isolation, Buildah was found to have an information disclosure issue. Environment variables can be accessed by running processes in container builds, such as Dockerfile RUN instructions.
	CVE-2022-2403	6.5	The OpenShift Container Platform has a credential breach. Any authorized OpenShift user or service account could access the external cluster certificate's private key since it was improperly saved in the oauth-serving-cert ConfigMaps.
	CVE-2023-24827	7.5	A Software Bill of Materials (SBOM) can be created from filesystems and container images using the CLI tool and Go library syft. Versions 0.69.0 and 0.69.1of Syft contains a bug that can reveal passwords. The SYFT_ATTEST_PASSWORD environment variable contains the password, which is exposed by this bug.
Network Inspection	CVE-2020-11939	9.8	Due to the granularity of the overflow primitive and the ability to remotely manipulate the layout and contents of the heap memory of the nDPI library, this vulnerability could be exploited to fully execute remote code against any network inspection stack that is linked against nDPI and utilizes it to analyze network traffic.

associated with invasions on the accessibility of container assets are the key insights provided by this study.

Mitigation Strategies could include:

- Limits& Regulations for Resources: Establish rigid CPU and memory consumption constraints. Enforce these restrictions to stop one container from using too many resources and resulting in a denial of service.
- Allocating Resources Automatically: Implement automatic scheduling and distribution of resource rules that give priority to significant containers. While dynamically modifying allotment according to demand, make sure that honeypots have a minimum amount of committed resources.
- Response and prevention in real-time: Create autonomous responses for resource depletion circumstances. Prevent the breach from spreading to additional containers or the whole infrastructure by dynamically segregating or throttling the infected container.
- Scaling and Capacity Planning: Plan to make sure that the container environment can withstand resource exhaustion attacks without creating significant disruptions. Extend the size of the infrastructure to prepare for unexpected demand.
- Response to Incident Strategy: Create a thorough incident response strategy that specifies how to identify as well as resolve resource exhaustion threats. Establish procedures

for redistributing resources, isolating impaired containers, and restoring regular operations.

3) Fingerprinting Container Honeypot Environments, U3: There are numerous ways for an attacker to figure out what they are within by executing the commands *cat /proc/1/cgroup*, *cat /proc/1/stat*, etc. This agrees with [81] conclusions for assessing the vulnerability of containers towards fingerprinting. As mentioned by Barron et. al. [82] in their investigation of attacker habits employing the Cowrie honeypot, it is apparent that a skilled invader could recognize the honeypot as not a secure environment and disassociate. As an instance, the Cowrie honeypot's default hostname is svr04, knowing the default values of Cowrie an attacker interacting with this honeypot will recognize that they are dealing with Cowrie honeypot.

Lessons Learnt: It offers insightful information about the methods, causes, and effects of attackers trying to discover and take advantage of honeypot setups.

Mitigation Strategies could include:

- Diversity and Randomization: To add diversity, randomly select the reaction times and conduct of honeypots. By doing this, attackers are hindered from noticing recurring patterns that might be utilized as fingerprints.
- Monitoring of Traffic and Interactions: To spot anomalous transfers that can indicate fingerprinting, maintain surveillance on traffic both in and out. Examine interconnections and patterns of traffic to find prospective intruders.
- Delays and Synthetic Latency: Add fabricated lag and delays to honeypot sessions. Attackers attempting to track down fingerprints by observing response timings could be confused by this.
- Interaction with the Security Community: To the larger security community, disseminate knowledge and methods. Work together with others to create workable defenses, and keep current on new fingerprinting strategies.
- Inherent Heterogeneity: Use a variety of operating systems, network configurations, and container technologies in deployments. It becomes more difficult for attackers to identify the characteristics of the honeypot due to the heterogeneity's complexity.
- Maintenance and Updates on a Regular Basis: Maintain the most recent software updates, security patches, and patch levels in the honeypot environment. Attackers frequently depend on known vulnerabilities that can be reduced by patches.

4) Privilege Escalation through Container Escape, U4: Since containers offers OS-level virtualization, system files are shared with the core service for building and maintaining containers [41]. This promotes system efficiency, lowers the levels of redundancy among system components, and/or makes container management easier. Attackers, however, may take advantage of kernel or service management flaws to abuse the container host or escape from the container and pose a threat to other containers on the same host [83], [84]. A compromised honeypot container that already exists or the deployment of a newer malignant honeypot container image could exploit the container breakout threat. RunC, the core runtime being used to execute container functions, including Docker, containerd, and similar Linux-based container systems, has major vulnerability CVE-2019-5736(Docker containers operating under basic configurations could be exploited by an adversary to seize control of the device at the system level [85]). The typical AppArmor or Security-Enhanced Linux (SELinux) rules on certain Linux systems, notably Fedora, do not restrict the utilization of such an exploit. However, this can be avoided by using user namespaces properly, where the host root is not mirrored inside of the container's user namespace [86].

Lessons Learnt: It offers a more thorough understanding of how fraudulent users take use of vulnerabilities to escape from confined environments and get illegal access to the host.

Mitigation Strategies could include:

- Hardening the Host: Increase the host system's security by following security standard procedures, performing frequent upgrades, and reducing unused applications and services.
- Image Verification and Scanning: Before deployment, inspect container images for vulnerabilities. To maintain image integrity, use trustworthy image sources and integrate image certification and verification.
- Reduce the Attack Landscape: Uninstall all irrelevant binary files, software libraries, and scripts from containers. For a less extensive attack surface and fewer opportunities for privilege escalation, use minimalist baseline images.
- Isolation of User Namespaces: To prevent probable escalated privileges through client manipulation, employ user namespace for mapping containers user IDs to non-privileged Id in the host.
- Tracking and Auditing: Engage comprehensive tracking and observation of container responses. Track the changes or illicit access efforts that could imply privilege escalation.
- Network guidelines and accessibility control: Implement stringent network and access constraints to thwart illegitimate interactions and shifting between containers.

5) Command & Control through Container Escape, U5 : The honeypot container sends a communication to the assailant's server in search of its next instructions once it has been compromised. It will follow the instructions from the C2 (Command and Control) server of the attacker and may add new software [87]. Once the attacker fully controls the container host, they are free to run any code. A botnet—a cluster of infected computers—is often devised for this as a malicious payload that spreads laterally. Typically, hackers can do the following operations [88] via command and control(C&C):

- Data Robbery: Financial records and other private firm secrets could be collected or transmitted to an assailant's server.
- Shutdown: An attacker could take down one or more devices or possibly the entire network of a corporation.
- Reboot: Devices that are compromised could be abruptly and frequently rebooted, disrupting regular corporate operations.
- Distributed Denial of Service: DDoS assaults saturate servers or networks with internet traffic, turning them

ineffective.

An intruder who is not permitted to access the corporate network therefore can take complete control of the network.

Lessons Learnt: It deepens our comprehension of how adversaries use containerized settings to hide, persist, and take control of compromised machines.

Mitigation Strategies could include:

- Analysis of behaviour: Establish standards for typical container behaviour employing behavioural analysis tools, and look for any abnormalities that would suggest C&C activity.
- Sandboxing and Dynamic Evaluation: For dubious containers, conduct dynamic evaluation and sandboxing. Permit regular activities to continue while isolating potentially impaired containers for additional analysis.
- Runtime Security for Containers: To avoid unapproved container escape, adopt a reliable container environment that ensures isolation and accessibility rules. Control container privileges by employing solutions like Seccomp, SELinux, AppArmor, etc.
- Runtime Anomaly Detection: Establish behavioural and anomaly recognition criteria to spot shady behaviour suggestive of C&C attempts.
- Filtering of Egress Traffic: To restrict communications from containers, establish egress filtration of traffic. Verify and control the parts and procedures that containers are permitted to use.

6) Cross Container ARP Poisoning, U6: The bridged network (default) interface in the Docker configuration aids the networking of containers [49]. Virtualized layer 2 networks are created when containers within the same host are bridged while employing a simple (but popular) container network in the ensemble. This configuration is typical for Kubernetes(framework for executing and managing containers from numerous container runtimes [89]) deployments. The *NET_RAW* linux capability, which gives Pods low-level access to network interactions, is another prevalent practise. When these two aspects are coupled, a malevolent pod could utilize the ARP protocol(layer 2 protocol to translate MAC addresses to IP addresses [90]) to pretend to be another pod on the same node's IP address. As a result, other pods on the node potentially connect with the attacker's pod rather than the authentic Pod. Utilizing *Pod.spec.securityContext.capabilities* to disable the *NET_RAW* capability from pods as a corrective solution can be considered.

Lessons Learnt: It provides us better understand how attackers use ARP flaws to disrupt network communication, snoop on data being sent between containers, and thus gain unauthorised access.

Mitigation Strategies could include:

- Identifying ARP Spoofing: In the honeypot environment, deploy ARP spoofing detection tools. Use tools that can recognise aberrant ARP behaviour and send out alarms on occurrence, such as when MAC-IP mappings unexpectedly shifts.
- ARP Static Entries: To thwart attacks relying on ARP spoofing, establish static ARP entries within containers. By doing so, attackers are unable to alter ARP mappings, ensuring their accuracy.
- Network Telemetry: Utilise telemetry and network monitoring tools to monitor and record ARP traffic among containers. For the purpose of capturing and analysing ARP data between containers, use network monitoring and telemetry tools. ARP poisoning may be indicated by peculiar patterns, consequently keep monitoring out for these.

7) Cross Container Side Channel Exploit, U7: The ability to observe the CPU and memory consumption by the attacker who seized control of the container creates the opportunity for a side channel attack. Using a side-channel attack, a protected perimeter can be breached via exploiting technical weaknesses rather than directly attacking it with brute force technique [91]. For programmable units belonging to different security domains, Sprabery et.al [92] provided a hardware-software method that will mitigate cache based side-channels. This method combines the cache partitioning capabilities of the Intel CAT architecture with cutting-edge scheduling strategies and state-cleansing technologies to provide cache based side-channel free processing for Linux oriented containers and virtualization software.

Lessons Learnt: It improves our knowledge of how attackers might take advantage of minute fluctuations in usage of resources like CPU memory cache; and network connections, to steal details from nearby containers.

Mitigation Strategies could include:

- Policies for Container Scheduling: Make use of scheduling principles for containers that guarantee a physical distinction of containers from the underlying hardware, lowering the possibility of inadvertent sharing of resources.
- Memory and CPU Affinity: Affinity settings for the CPU and memory guarantee that containers are assigned to different physical cores or memory banks, thwarting cache-based side channel attacks.
- Surveillance of Shared Resources: Maintain an eye out for any unconventional behaviour involving shared resources, especially the CPU cache or RAM. Establish safeguards for identifying and preventing inappropriate sharing of assets.
- Effective Resource Isolation: To reduce the risk of exploitation of shared resources, use container orchestration technologies for allocating exclusive resources (RAM, CPU, etc.) to all containers container.

8) Cryptojacking through Misconfigured Daemon, U8: Cryptojacking describes a situation in which an assailant gains unauthorized access and abuses computational resources to mine bitcoins [93], [94]. Four successive actions typically make up a cryptojacking attack- Target system compromise, mining code execution, communication with mining pooled servers, and evade detection over extended periods of time [95]. The Honeypot Docker daemon, which by default monitors Unix socket, confronts a restful API that enables users to communicate with the daemon. The default for connecting to the Docker service remotely utilizing REST API, which enables creating, activating, and stopping containers, is TCP port 2375 or 2376. The daemon could be customized to listen to TCP socket if remote management is desired. The problem is that utilizing a TCP socket by default lacks a verification or approval mechanism. Everyone who had access to the daemon has ultimate control. As part of opportunistic attacks intended to illegally mine bitcoin, a new cryptojacking campaign has been discovered that targets susceptible Docker and Kubernetes infrastructures [96].

Lessons Learnt: It highlights the possible repercussions of improper daemon settings and the simplicity with which attackers take advantage of such errors to use CPU resources for the illegal mining of cryptocurrency covertly.

Mitigation Strategies could include:

- Auditing Runtime Integrity: Utilise technologies for runtime integrity monitoring that constantly monitor containers behaviour and parameters involved. Notifications or automatic reactions can be set off by any illegal alterations.
- AI and Machine Learning: Utilise ML and AI techniques to find cryptojacking behaviour patterns that could remain undetected by conventional methods.
- Solutions Based on Blockchain: To track and validate legal container setups and operations, consider blockchain technology. Any changes that are not authorized can be instantly identified and fixed.
- Architecture with Zero Trust: Embrace a zero-trust architecture in which every interaction is regarded as having the potential to be harmful. Apply micro-segmentation, constant verification, and rigorous access constraints.
- Integration of Threat intelligence: To keep up with trends and warning signs of cryptojacking, integrate threat intelligence channels. With the aid of this knowledge, emergent dangers can be proactively identified and countered.

The overall lessons learned from the usecase investigations are summarised in Table.IV.

VI. Related Works

A. Selection of Primary Studies

We created a combination of keywords (honeypot/honeynet/ contanerized honeypot/ docker honeypot/ research and production honeypot) in the search string and sent it to specific search engines in order to gather the collection of pertinent research. The title, keywords, and abstract were all searched against. Springer, ACM, ScienceDirect, Google Scholar, and Scopus are a few databases in addition to IEEE xplore Digital library. Iterations of forward and backward snowballing were also used to ensure that almost all papers meeting the criteria for inclusion is included. Both qualitative and quantitative research papers on honeypots are included in this extensive search spanning the years from 2013 to the present. Although research on honeypots was conducted for a very long time, the usage of docker containers for honeypots only became common after 2013. Numerous honeypots belonging to several categories, including the Database Honeypot, Web, Service, Distributed, SCADA, Honeypot for USB, High-Interaction Honeypots, etc., were uncovered. Other language articles and the extended conference edition of a report that has a journal edition were excluded. From the list of open-source honeypots returned by the search engine, we choose those that went public after 2013 when Docker container technology began to gain broad popularity.

B. Findings

After applying the inclusion and exclusion criteria, the results are narrowed down to 19 honeypots as listed in the Table V.

TCP proxy and SSH honeypot is offered by Glutton [98]. Between the attacker and the server, the SSH proxy acts as a MITM(Man In The Middle) to log everything in plain text. However, TCP proxies do not yet have logging capabilities. IPP honeypot [102] mimics a printer that is connected to the Internet and supports the Internet Printing Protocol. It incorporates techniques from several existing honeypots, including Elasticpot, Citrix Honeypot, and ADBHoneypot (for support of output plugins). It needs regular maintenance and supervision to make sure that it is established, revised, and observed appropriately. A honeypot framework called DDoSPot [100] is used to track and monitor UDP-based Distributed Denial of Service (DDoS) attacks. The honeypot servers and services are supported by the platform as pots, which are very straightforward plugins, are NTP server- that offers 3 NTP packet response modes(Client, Control, monlist) , SSDP server- responds to legitimate multicast (M-SEARCH) queries and offers a very limited and lightweight MiniUPnP emulation. It is entirely customizable, similar to NTP, DNS server- attempts to mimic actual DNS service as closely as possible by sending all requests to a legitimate DNS resolver and returning results for queries. To capture brute force attempts and the interaction with the attacker's shell, an SSH and Telnet honeypot with medium to high engagement levels was set up in Cowrie [105]. It substitutes for SSH as well as telnet in high interaction mode (as proxy) for observing adversary activities in another machine. It simulates a UNIX system in Python when in medium interaction mode (shell). The SSH-Honeypot [115] logs the client's IP address, username, and password while listening for incoming SSH connections. This honeypot has a low interaction level and excludes login attempts from hackers or malware. It was not intended for use in production and was originally built to gather basic intelligence about brute-force attacks. The gathered data is mounted to *VOLUME ["/home/honeycomb/log", "/home/honeycomb/rsa"]* directory in the docker file linked to the honeypot. In the honeypot [114], the attacker is exposed to actual vulnerable programmes, however, all communication with the machine is recorded in clear-text. To make the deployment of the honeypot simpler, Lyrebird uses Docker containers by default. Both the mitmproxy dump file and a human readable HTML report contain information about the attacker's activity.

Attackers typically utilize search engines and particularly suited search queries to discover potential targets. The Lukas Rist [113] Glastopf honeypot offers such keywords (also known as dork) in order to entice visitors in and further harvests them from queries, thus increasing its attack vector.

Table IV: Summary of the Usecase Studies

Use Case	Attack Vectors	Mitigation strategies
DoS(Denial Of Service)by Denying Storage Space	Prevent other containers from accessing the storage space of honeypot logs. Prevent Honeypot from keeping the action logs and merely erasing the attack's traces.	Isolated honeypot setting
		Snapshots of containers
		Certificate of Custody
		Establish read-only filesystems
		Monitoring of execution behaviour
		Security profiles for containers
		Replication and Backup
DoS(Denial Of Service)by Denying Memory/ Processor Resource	Denial of memory and processing resources.	Limits & Regulations for resources
		Allocating resources automatically
		Response and prevention in real-time
		Scaling and capacity planning
		Response to incident strategy
Fingerprinting Container Honeypot Environments	By carrying out certain commands, an attacker can figure out details about the targets.	Diversity and randomization
		Monitoring of traffic and interactions
		Delays and synthetic latency
		Interaction with the security community
		Inherent heterogeneity
		Maintenance and update on regular basis
Privilege Escalation through Container Escape	Utilize kernel or service management shortcomings to misuse the container host or elude the container.	Hardening the host
		Image verification and scanning
		Reduce attack landscape
		Isolation of user namespaces
		Tracking and Auditing
		Network guidelines and accessibility control
Command & Control through Container Escape	Attacker fully controls the container host and runs any code.	Analysis of behaviour
		Sandboxing and dynamic evaluation
		Runtime security for containers
		Runtime anomaly detection
		Filtering of egress traffic
CrossContainer ARP Poisoning	Bridge between containers on the same host.	Identifying ARP spoofing
		ARP static entries
		Network telemetry
Cross Container Side Channel Exploit	Protected perimeters can be cracked by taking advantage of technical vulnerabilities rather than charging it using brute force attacks.	Policies for container scheduling
		Memory and CPU affinity
		Surveillance of shared resources
		Effective resource isolation
Cryptojacking through Misconfigured Daemon	Compromise Target system. Mining code execution. Communication with Mining pooled Servers. Evade detection over extended periods of time.	Auditing runtime integrity
		AI and machine learning
		Solutions based on Blockchain
		Architecture with zero trust
		Integration of threat intelligence

A built-in PHP sandbox is used for remote file inclusion, whereas POST requests are used for local file inclusion. For centralized data collection, HPFeeds logging is enabled. Dionaea honeypot [111] tracks malware by monitoring exploitable weak points revealed by services provided to a network. To be informed when it can read from or write to a socket, dionaea uses libev(high-performance event loop/event model). For IPv4 as well as IPv6, dionaea can provide services using TCP/UDP and TLS. If necessary, it can implement rate restrictions and accounting constraints for each connection to TCP and TLS connections.

There are instances when you only need a straightforward credential-collecting honeypot. Then we could utilize Heralding honeypot [109] which supports ftp(File Transfer Protocol), telnet, ssh (Secure Socket Shell), http(Hyper Text transfer protocol), https(HTTPSecure), pop3(Post Office Protocol), pop3s (POP3Server), imap(Internet Message Access protocol), imaps(IMAP over SSL/TLS), smtp(Simple Mail Transfer Protocol), vnc(Virtual Network Computing), postgresql, and socks5. It maintains pertinent information in three files: log auth.csv, log session.csv, and log session.json. Conpot [108] is an ICS honeypot designed to gather information about the objectives and tactics of adversaries aiming to compromise industrial control systems. To broaden the honeypots' attack surface and enhance their deceiving powers, it offered the option of hosting a unique human-machine user experience. The services' response times can be purposefully decelerated to simulate the behavior of a system that is constantly under demand.

The Android Debug Bridge (ADB) [104] is a framework intended to monitor actual and emulation-based smartphones, Televisions, and Video recorders linked to a specific system. It includes a number of commands (such as adb shell, adb push, and others) that help developers push content to the device during debugging. Typically, a USB cable is employed for it, and there are numerous authentication and protection systems in place.

T-Pot [116] is an all-in-one, popular, preferably decentralized, multiarch (amd64, arm64) honey trap architecture that supports more than 20 honeypots, a wide range of visualization capabilities using the Elastic Stack, dynamic real-time threat mappings, and a variety of security features to enhance the deception scenario. T-Pot is grounded on the Debian 11 (Bullseye) Net installer and makes use of docker as well as docker-compose to accomplish its objectives of running multiple tools concurrently thus fully utilizing the host's hardware.

Table V: Learning Experience of Container Implemented Honeypots

HONEYPOT	YEAR	INTERACTION LEVEL	LANGUAGES USED	SIMULATED SERVICES	ROLE	OPENSOURCE	APPLICATION	SCALABILITY
medpot [97]	2022	LIH	Go, Dockerfile, Shell, Makefile	Emulate HL7 / FHIR	Server	yes	Medical Field	✓
glutton [98]	2021	LIH	Go, Dockerfile	SSH and TCP proxy	Server	yes	Web Application	✓
Citrix Honeypot [99]	2020	LIH	HTML, Python, Dockerfile	Detect and log CVE-2019-19781 (remote code execution vulnerability) scan and exploitation attempts.	Server	yes	Citrix ADC(Application Delivery Controller)	✓
ddospot [100]	2020	LIH	Python,Dockerfile	Tracking and monitoring UDP-based DDoS attacks.	Server	yes	General	✓
dicompot [101]	2020	LIH	Go, Dockerfile	Digital Imaging and Communications in Medicine (DICOM)	Server	yes	Medical	✓
IPP Honey [102]	2020	LIH	Python, C++, Shell, Dockerfile	Simulates printer service	Server	yes	Printer	✓
mailoney [103]	2020	LIH	Python, Dockerfile	SMTP	Server	yes	E-mail	✓
ADBHoney [104]	2019	LIH	Python, Dockerfile	Android Debug Bridge(ADB) over TCP/IP	Server	yes	Emulated and real phones/TVs/DVRs	✓
Cowrie [105]	2019	LIH-MIH	Python, Dockerfile	SSH and Telnet	Server	yes	General	✓
endlessh [106]	2019	LIH-MIH	C, Shell, Roff, Python, Makefile, Dockerfile	SSH	Server	yes	General	✓
CiscoASA [107]	2018	LIH	Javascript, Python, CSS, HTML, Dockerfile	Detect CVE-2018-0101, a DoS and remote code execution vulnerability	Server	yes	Cisco Routers	✓
Conpot [108]	2018	LIH	Python, Dockerfile	Collect intelligence about the motives and methods of adversaries targeting industrial control systems.	Server	yes	ICS/SCADA	✓
heralding [109]	2017	LIH	Python, Dockerfile	ftp, telnet, ssh, http, https, pop3, pop3s, imap, imaps, smtp, vnc, postgresql and socks5.	Server	yes	General	✓
mysql-honeypotd [110]	2017	LIH	C, Shell, Dockerfile, Makefile	Mimic MySQL Database	Server	yes	Database Storage server	✓
dionaea [111]	2016	LIH	Python, C, Cmake, Shell, Dockerfile, HTML	Detect shellcodes, support ipv6, TFTP and TLS.	Server	yes	General	✓
MongoDB-HoneyProxy [112]	2016	MIH	Javascript, Dockerfile	Proxy and log all traffic to a dummy mongodb server.	Server	yes	Database Storage server	✓
glastopf [113]	2016	LIH	Python, HTML, Hack, Shell, Dockerfile, CSS	Local file incorporation using files out of a virtualized system files, remote file integration using a built-in PHP Sandbox, and HTML insertion leveraging POST methods	Server	yes	WebApplication	✓
Lyrebird [114]	2016	HIH	Python, Dockerfile	Man-in-the-middle SSH interception	Server	yes	General	✓
ssh-honeypot [115]	2016	LIH	C, Shell, Dockerfile, Makefile	Fake sshd that logs ip addresses, usernames, and passwords.	Server	yes	General	✓

The T-Pot GitHub repository contains all of the source code and configuration files. The docker directory contains each individual Dockerfile and its config. It also provides a web interface for easy understanding.

VII. Experimental Setup

A. Preliminary Exploration

We experimented with honeypots inspired by Deutsche Telekom's work [116]. In our previous work [69] we were able to operate multiple honeypot sensors along the same connection interface without experiencing any issues, making the total mechanism relatively minimal maintenance. Many ports are exposed to inbound transmission, which makes them prone to tampering. We further comprehended data by thoroughly examining and assessing everything gathered through honeypot exploration. However, the lack of flow-related information in the gathered data hindered our attempts for further meticulous data analysis.

B. VIKRANT

Vikrant is installed on the Azure Cloud instance(region: Central India) and managed to run Debian 11 (Bullseye-Gen1 Image, VM Instance-type E2ds v4) having two virtualized CPUs(No.s 2), 16 GB of memory, and 128 GB hard drive. The system's installed containers run in host-network mode, which shares the host system's network virtualization stack and prevents the assignment of a separate IP address. The instance was made public and active from 6^{th} February to 12^{th} February, 2023. The overall architecture for VIKRANT is shown in Figure.8.

The framework is designed to deceive the invaders that they have obtained accessibility into the actual system so as to encourage themselves to remain. The docker encapsulation of the honeypot daemon allows for effective sandbox segregation and straightforward upgrading processes. The folder */data* contains all permanent information files from the honeypot collection(Adbhoney, CiscoASA, Cowrie, DDospot, Elasticpot, Log4Jpot, Mailoney, Medpot, Redishoneypot). Elasticsearch is used in conjunction with the other components of the ELK Stack, Logstash, and Kibana, and has been incorporated into our honeypot for functions such as data indexing and archiving. The visualization of data is possible by utilizing a single query and modified as needed through interaction with Kibana. A significant constraint, however, is that each visualization ought to be used with a specific indexing sequence. If one index contains data that is entirely different from another, separate visualizations over each list have to be constructed. TCPdump (network packet snooping utility) with predefined conditions had been employed to capture, filter, and examine traffic from the network, which includes TCP/IP frames traveling through the system and storing the captured information in a pcap file. Since they can capture packets from a wide range of protocols, such as TCP, UDP, ICMP, and ARP, and can be used on a variety of operating systems, such as Linux, Unix, macOS, and Windows, we can overlook the fact that it requires specialized hardware for processing network packets with incorrect checksums. To facilitate further

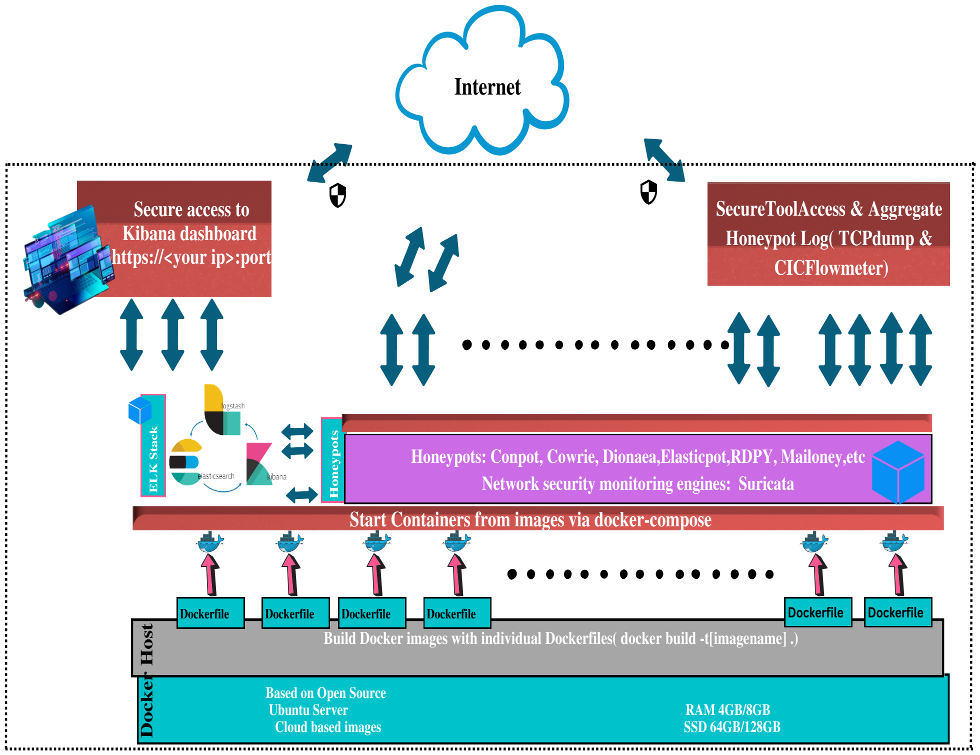

Internet
Secure access to
Kibana dashboard
https://<your ip>:port
SecureToolAccess & Aggregate
Honeypot Log(TCPdump &
CICFlowmeter)
ELK Stack
logstash
elasticsearch
kibana
Honeypots
Honeypots: Conpot, Cowrie, Dionaea,Elasticpot,RDPY, Mailoney,etc
Network security monitoring engines: Suricata
Start Containers from images via docker-compose
Dockerfile
Build Docker images with individual Dockerfiles(docker build -t[imagename] .)
Docker Host
Based on Open Source
Ubuntu Server
Cloud based images
RAM 4GB/8GB
SSD 64GB/128GB

Figure 8: Overall Architecture of VIKRANT Honeypot

analysis, CICFlowMeter has been utilized to convert pcap files to csv format.

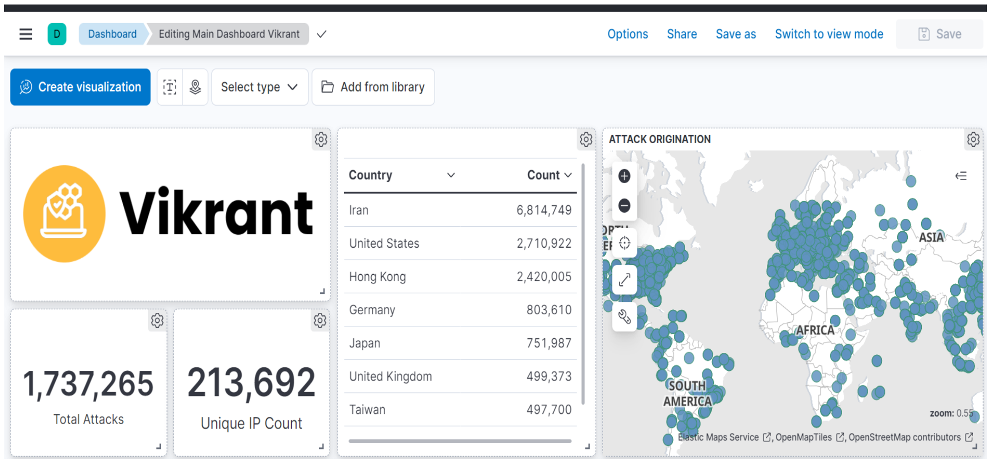
Additionally, threat intelligence is incorporated into the framework using Suricata (a comprehensive signature-based system for matching identified threats, infractions of the policy, and fraudulent activity). It has the ability to detect dubious patterns as well as discrepancies and additionally observe DNS, HTTP, and FTP traffic)

Container Threat Mitigation Techniques Adopted in Our Work: We have avoided untested or untrusted images for building containers so as to prevent the introduction of vulnerabilities and malicious code in the docker image. In order to construct containerized applications consistently, we employed multi-stage builds. Utilized custom bridge networks to limit which containers are permitted to communicate with one another and to enable automatic DNS resolution from container name to IP address. Enabled the user namespace function to offer distinct user accounts for isolated containers and limit mobility across containers. The user namespace functionality has been enabled to give unique user accounts for isolated containers and to limit movement across containers. To avoid privilege escalation, containers perpetually operate as non-root users.

VIII. Results and Discussions

Figure. 9 shows the total hits recorded by our experimental setup. For instance, to know about the highest value of bytes sent, the aggregations function at the bottom right under the Bytes Sent column can be used and pick max. Figure. 10 and Figure. 11 depict the sample flow information and the total hits gathered by *VIKRANT* respectively. The increased number of hits in ddospot honeypot (Figure. 11) illustrates that various automated tools and services supplied by third-party suppliers for distributed denial of service attacks as a service make it simple to launch DDoS attacks. Even inexperienced attackers can easily launch complex DDoS attacks by using crucial characteristics of cloud technology.

We further studied the traffic, given that the ddospot count was large, and concluded that attackers could potentially have leveraged specific attack vectors, such as SYN/ACK floods, UDP amplification, and HTTP floods. The honeypot is able to collect information on how much traffic was produced during an incident, particularly bandwidth utilization, connection rates, and metrics like packets per second. The Cowrie honeypot keeps track of all login attempts, whether they are successful or not, giving information on the usernames and passwords that attackers try to use. It is estimated that attackers would have attempted to brute force their way into the honeypot using a standard password dictionary file. It records the actions/commands executed by attackers after gaining access, illuminating the precise exploits exploited. The majority of users who connected to the honeypot used the command ’uname -a’ before exiting. This retrieves detailed system information, including the kernel version, machine architecture, and operating system type and version. Vulnerabilities that were commonly targeted were discovered by examining the commands used by attackers. The geographical origin of the attacking IP has also been revealed. The log4shell payloads sent as HTTP headers, query parameters, or POST data are captured and downloaded by Log4pot. The incredibly flexible

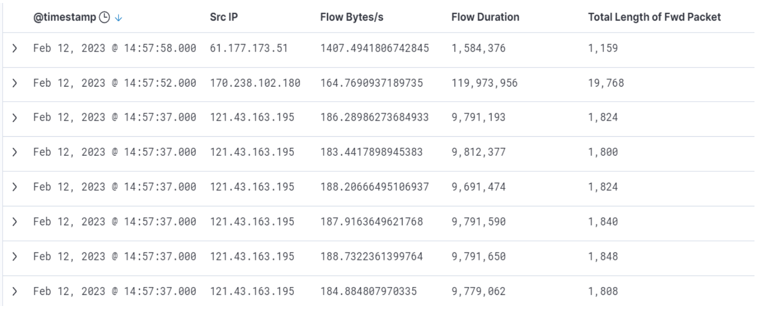


1,737,265
Total Attacks

213,692
Unique IP Count

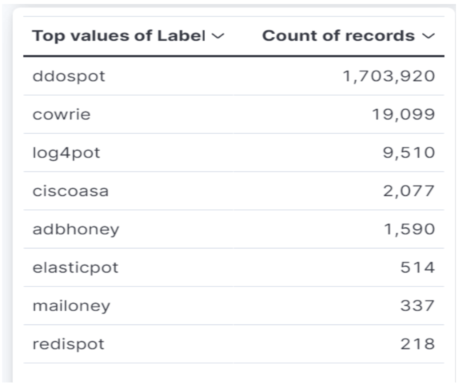
Country	Count
Iran	6,814,749
United States	2,710,922
Hong Kong	2,420,005
Germany	803,610
Japan	751,987
United Kingdom	499,373
Taiwan	497,700

Figure 9: GUI Displaying Assault Statistics and List of Top 5 Contributing Countries



@timestamp	Src IP	Flow Bytes/s	Flow Duration	Total Length of Fwd Packet
Feb 12, 2023 @ 14:57:58.000	61.177.173.51	1407.4941806742845	1,584,376	1,159
Feb 12, 2023 @ 14:57:52.000	170.238.102.180	164.7690937189735	119,973,956	19,768
Feb 12, 2023 @ 14:57:37.000	121.43.163.195	186.28986273684933	9,791,193	1,824
Feb 12, 2023 @ 14:57:37.000	121.43.163.195	183.4417898945383	9,812,377	1,800
Feb 12, 2023 @ 14:57:37.000	121.43.163.195	188.20666495106937	9,691,474	1,824
Feb 12, 2023 @ 14:57:37.000	121.43.163.195	187.9163649621768	9,791,590	1,840
Feb 12, 2023 @ 14:57:37.000	121.43.163.195	188.7322361399764	9,791,650	1,848
Feb 12, 2023 @ 14:57:37.000	121.43.163.195	184.884807970335	9,779,062	1,808

Figure 10: Sample Flow Data Information Gathered by VIKRANT



Top values of Label	Count of records
ddospot	1,703,920
cowrie	19,099
log4pot	9,510
ciscoasa	2,077
adbhoney	1,590
elasticpot	514
mailoney	337
redispot	218

Figure 11: Total Hits Received by Each Honeypot

Log4j vulnerability enables the execution of arbitrary code from both local and remote LDAP servers, among other capabilities. This vulnerability (CVE-2021-44228) has a severity rating of 10.0 [117] according to the Common Vulnerability Scoring System (CVSS 3.0).

A Cisco ASA device is simulated by the Ciscoasa honeypot. On the emulated Cisco ASA device, it can be seen that the attacker does an in-depth examination of a number of ports in an effort to find open services and potential vulnerabilities. Additionally, it monitors attempts of command injection, malicious file uploads, theft, etc. Adbhoney Honeypot is specially made to imitate Android Debug Bridge (ADB) interfaces, which are frequently targeted by attackers looking to gain unauthorized access to or control over Android devices. It gathers data on the simulated Android smartphone, which could be a foretaste to later, more focused attacks. Indication of efforts to steal confidential information from the simulated Android smartphone demonstrates possible data security vulnerabilities. A vulnerable Elasticsearch server exposed to the Internet is simulated by elasticpot. It monitors efforts to insert malicious scripts that could create client-side vulnerabilities into the Elasticsearch data that has been indexed. It could spot excessive indexing or search requests that are meant to burden the Elasticsearch server's resources and obstruct normal operation. The purpose of Mailoney is to imitate email services and protocols. It monitors phishing attempts, in which scam emails pretending to be from reliable sources (such as banks or reputable organizations) are sent to deceive recipients into disclosing personal information or clicking on links. Redis databases, a popular in-memory data structure store, have been shielded from potential threats using Redispot honeypot. Attempts to connect to the simulated Redis server using default or insecure credentials are picked up by Redispot as an effort by an intruder to get access. We make the presumption that since there is less number of hits headed to remaining honeypots, they most likely didn't receive any further activity or interactions.

Figure. 12 depicts the heat map of total forwarded packets from each Ip address. A substantial amount of forwarding packets generated by a single address could be a sign indicating that the IP address is being used in DDoS assault. It may be possible to tell whether the traffic coming from a specific IP address is coming from a trustworthy user or a hostile attacker by analyzing that IP address's traffic. As malware frequently uses an immense amount of forward packets for communication to Command and Control (C&C) servers, it may also be employed for recognizing suspected infection by malware. The source of the flow data, applications, IP addresses, processes, protocols, and end users consuming the largest amount of bandwidth provides managers with essential perspectives into the behavior and performance of their networks. Table VI includes the parameters retrieved by the honeypot that can be utilized for further security analysis. Below is a description of some of them.

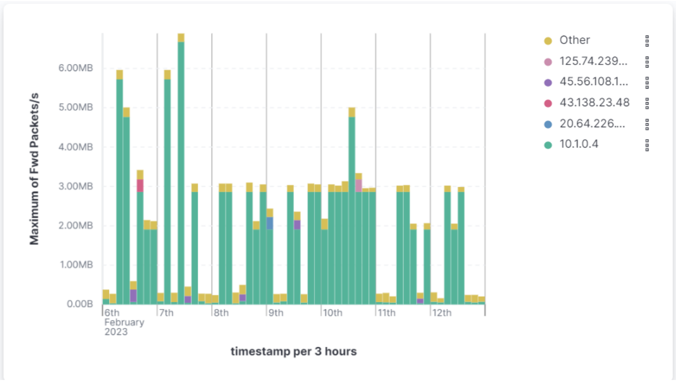


Figure 12: Statistics of Forwarded packets during the experimentation period

- *ACK Flag count*: Number of packets with ACK. ACK packet in TCP denotes that a request for communication is accepted.
- *Bwd IAT Min*: The minimum interval between two packets delivered in the opposite direction. It is a significant measure for determining network throughput and latency issues.
- *Idle Min*: Minimum time a flow was idle before becoming active.
- *PSH Flag Count*: Number of packets with *PUSH*. *PUSH* specifies that networking stacks' buffering should be overridden. Incoming data is transferred instantly when the network stack is instructed to forego buffering.
- *URG Flag Count*: Number of packets with *URG*. It might be a sign of malware activity if there are an unusually large number of TCP segments with the *URG* flag set.
- *ECE Flag Count & CWR Flag Count*: Number of Packets with *ECE & CWR* respectively. In TCP/IP networks, the *ECN* (Explicit Congestion Notification) technique is used to provide endpoints with notifications when the connection is congested. Congestion circumstances are signaled by the TCP header values *ECN-Echo (ECE)* and Congestion Window Reduced *(CWR)*. To have a fuller insight into the network behavior, the evaluation of the *ECE* flag needs to be paired with additional metrics such as round-trip duration, dropped packets, or congestion window width. The sender acknowledges receipt of the *ECE* signal and minimizes the congestion size of the window by enabling the *CWR* flag.
- *Subflow Bwd Packets*: Subflow's average number of packets in the reverse direction. Multi Path TCP (MPTCP) enables the creation and concurrent use of distinct subflows for data transmission. Each subflow may use a distinct path and possesses distinctive origin, destination, as well as port address. It is possible to utilize it to keep track of a network connection's dependability and stability. A significant decrease in the number of subflow packets, for instance, could potentially be a sign of network trouble or connection loss.
- *Fwd Act Data Pkts*: Number of packets in the forward direction with at least one byte of TCP data payload.
- *Fwd Init Win Bytes*: The number of bytes transferred in the direction of forward progress in the first window.

IX. INFERENCES

1) Prepare your security and approach: One should have a failsafe plan in place before installing a containerized honeypot in one's network since any mistakes could jeopardize the integrity of the core network. When establishing a honeypot, issues like timeline for using the honeypot, measurable goals, resources to rely on (internal or external, such as a third-party vendor), and how to use the data collected are all important considerations. The level of information provided by a honeypot is higher than that of other network monitoring technologies like intrusion detection systems (IDS). For instance, an IDS scans networks for unusual activity, notifies users, and quickly thwarts intruders. While gathering and analyzing data on attackers is a function of an IDS, it is not its primary objective.

2) Utilize the appropriate software for the task (Honeypot Software): Setting up a honeypot on your own could be challenging (and deadly if you do it imprecisely). Build honeypots using only trustworthy public container images. Otherwise, the malformed image could compromise the system and carry out lateral operations. Numerous tools like Argos [118] can come in handy and make your task easier. Before you can set up a honeypot, you must have a very thorough understanding of what it is, how it works, and how to utilize it effectively.

3) Establish a Honeypot that appears like a real system: The data generated by the honeypot can be used by security professionals, vendors, enterprises, and other organizations to identify trends in the behavior of cyberattacks and subsequently modify their solutions and defensive tactics effectively. The invaders might strike back if they learn that they were duped with a honeypot. They might manage to break into your private network and destroy it, or they might even make violent threats. For instance, an attacker who recognizes a system as a honeypot may carry out actions on the phony machine to divert the administrator's focus from the actual targeted act that is taking place on the production line. Adopt measures to avoid issues, such as choosing the right position for the honeypot, choosing the proper interaction level, and using data that seems credible.

Table VI: Parameters Retrieved from VIKRANT

Parameters		
ACK Flag Count	Bwd IAT Min	Idle Max
Active Max	Bwd IAT Std	Idle Mean
Active Mean	Bwd IAT Total	Idle Min
Active Min	Bwd Init Win Bytes	Idle Std
Active Std	Bwd PSH Flags	Packet Length Max
Average Packet Size	Bwd Packet Length Max	Packet Length Mean
Bwd Bulk Rate Avg	Bwd Packet Length Mean	Packet Length Min
Bwd Bytes/Bulk Avg	Bwd Packet Length Min	Packet Length Std
Bwd Header Length	Bwd Packet Length Std	Packet Length Variance
Bwd IAT Max	Bwd Packet/Bulk Avg	Protocol
Bwd IAT Mean	Flow IAT Max	PSH Flag Count
Bwd Packets/s	Flow IAT Mean	RST Flag Count
Bwd Segment Size Avg	Flow IAT Min	SYN Flag Count
Bwd URG Flags	Flow IAT Std	Src IP
CWR Flag Count	Flow ID	Src Port
Down/Up Ratio	Flow Packets/s	Subflow Bwd Bytes
Dst IP	Fwd Act Data Pkts	Subflow Bwd Packets
Dst Port	Fwd Bulk Rate Avg	Subflow Fwd Bytes
ECE Flag Count	Fwd Bytes/Bulk Avg	Subflow Fwd Packets
FIN Flag Count	Fwd Header Length	Timestamp
FWD Init Win Bytes	Fwd IAT Max	Total Bwd packets
Flow Bytes/s	Fwd IAT Mean	Total Fwd Packet
Flow Duration	Fwd URG Flags	Total Length of Bwd Packet
Fwd IAT Min	Fwd Packet Length Max	Total Length of Fwd Packet
Fwd IAT Std	Fwd Packet Length Mean	URG Flag Count
Fwd IAT Total	Fwd Packet Length Min	Label
Fwd PSH Flags	Fwd Packet Length Std	Fwd Seg Size Min
Fwd Packet/Bulk Avg	Fwd Packets/s	Fwd Segment Size Avg

4) Do not jeopardize privacy in order to use a honeypot: Underestimating the capabilities of intruders could lead you to unwittingly give them access to your protected network, putting your data or that of your customers in danger. Several container-related concerns should be taken into consideration when deploying honeypots, including the need to exercise caution over resource accessibility after containers have been established. It is better to use virtual computers to set up the container honeypots as they provide a secure environment and can easily be restarted after being hacked. Additionally, separate the honeypot from the primary network and Utilize false or alluring information to entice the targets.

5) Recognize the Legal Consequences: Setting up a honeypot entails making a secure target for scammers looking to enter systems without authorization. Consequently, when assailants commit crimes, the laws of the land are not taken into account, however we must. Legal problems that you could experience include:

- Privacy Issues: Administrators may face a lawsuit for breach of trust if the intruders are successful in gaining access to private information. The data owners can even take the business to court for putting their data in danger. Additionally, since various nations have their own regulations on this, there is the issue of what information can lawfully be obtained from those entangled in honeypots.
- Liability Issues: If fraudsters are able to get into the secured region of your network because of honeypots, it could be a huge concern. Afterward, they could infect any of your devices and add them to a botnet under their

control, turn the system into a transfer point for their illegal content, or inject malware or harmful programs into it. If so, the dissemination of such activities could be blamed upon honeypot administration.

- Entrapment Issues: Entrapment happens when a member of the government or another organization seduces or dupes a victim into performing an offense [119]. It's critical to take appropriate precautions to prevent the situation where it appears as though you are "inviting" intruders onto your network since that could result in these types of charges being filed.

X. Provision of honeypots in multitude of applications

A. Industrial Cyber-Physical Systems (CPS)

A crucial scientific link connecting the physical and digital worlds in industry, industrial cyber-physical systems (CPS) are a key part of the industrial information infrastructure [120]. Recent incidents like Colonial Pipeline, US [121] demonstrate how industrial CPSs have ultimately been targeted. If the attack succeeds, it will immediately impact the production chain and result in a significant loss. APTs like Stuxnet [122] (strike one specific Iranian nuclear processing facility in Natanz, Iran) and BlackEnergy [123] (during the cyberwar between Russia and Georgia, infected machines' hard drives were wiped and DDoS attacks, bank robberies, and spam distribution were launched) are still prevalent. Industrial CPS honeypots have recently become more popular because of the need for industrial security. Pure high-interaction honeypots remain inherently inadequate for CPS since they depend on either deploying yet another hardware clone of the relevant resource or some other form of virtualization. The same safety hazards are involved with deploying a clone CPS for being compromised as are with deploying the original system, as well as significant expenditures associated.

The implementation of a hybrid-interaction honeypot, where realistic CPS devices and interfaces communicate with work-flow and hardware simulations that can precisely mimic the behavior of the CPS process, becomes an alternative. A straightforward heating system built by Litchfield et.al., modelled the process and equipment involved using an early version of HoneyPhy [124], [125], and set up a real-time simulation using existing honeypot technology as the system interface.

HoneyBot using HoneyPhy was devised [126], a quantum theory honeypot framework, to look into remote attacks on networked robotic systems. Error handling is often neglected in current research on industrial CPS honeypots because the focus is typically on how to more accurately mimic the operation and interaction processes of real industrial CPS devices. Sun et.al. [18] addressed this issue by suggesting a secure fuzzy testing strategy for identifying honeypots and to differentiate between actual devices and honeypots. The usage of mutation rules, along with efficient and secure probing packet generation security principles, is advised while employing multi object fuzzy testing. These probing packets are subsequently scanned and identified.

Honeypots only gather data when a breach occurs. If there have been no attempts to access the honeypot, there is nothing to examine. Only when an attack is directed towards the honeypot network does malicious traffic get logged; otherwise, attackers will refrain from the network. Experienced hackers can typically apprehend a production system from a honeypot using fingerprinting strategies as honeypots may frequently be discriminated from legitimate workflows. Despite being cut off from the main network, some sort of connection is eventually made to them so that administrators can access the data they hold. The prevailing deliberation is that a high-interaction honeypot poses a greater risk than a low-interaction one since it seeks to lure hackers towards obtaining root privileges.

B. Honeypot in IoT

Since IoT devices are being used more often, numerous security threats including remote login are viable in IoT networks (like SSH-Secure Shell and Telnet) [127]. Allowing attackers to believe they are real systems used by people and organizations is a crucial requirement for a honeypot to produce insightful data. Due to the wide range of device types and their physical connectivity, IoT devices present hurdles in this regard.The most frequent threat to these IoT devices was discovered to be the Mirai malware. It crawls the Web for IoT devices featuring ARC processors [128]. IoT honeypots share several traits with regular application honeypots, such as the capacity to respond to incidents as they eventuate. Plug-and-play container services as in [129] RIoTPot provides considerable portability, and its hybrid interaction capabilities let users move between low- and high-engagement modes. Making sure that the honeypots display on search engines while avoiding being identifiable as a decoy system is one of the most crucial aspects of honeypot design. Because of this, owners of honeypots must keep an eye on IoT search engines like Shodan, which can identify and locate devices and honeypots online [130]. Luo et. al. IoTCandyJar honeypots [131] asserted to reproduce IoT device behaviors without posing the danger of being infiltrated. The rate of attempts on a honeypot dramatically rises during the initial weeks once it is posted on Shodan, according to Guarnizo et al. [132].

C. Honeypot in Smart Grid

Sustainable and intelligent energy management is facilitated by smart grids, which also provide the dynamic, targeted flow of information amongst grid firms and their clients [133]. Despite all the benefits, there are security risks [134] that have been carried over from the IT industry to the electrical grid, and security breaches owing to the characteristics of smart grids are evolving. AMI(Advanced Metering Infrastructure) in smartgrids is susceptible to multiple assaults since utilities and consumers (smart meters) must communicate and exchange data. A honeypot game is put forth to combat DDoS attacks in an AMI network and evaluate groups of strategies to find the best balance between assailants and genuine users [135].

Under the SCADA (Supervisory Control And Data Acquisition) HoneyNet Project [136], the Critical Infrastructure Assurance Group of Cisco Systems has modified Honeyd, one of the first attempts to create an open-source low-interaction honeypot that can replicate a number of TCP/IP services, to support SCADA protocols.

D. Honeypot in Network Forensics

Network Forensics deals specifically with the inspection of the network and its I/O traffic passing the connectivity that is suspected to be engaged in security breaches, as well as its exploration. The original data, involving texts, file sharing, emails, and browser-based records, can be recovered with the use of network forensics and reconfigured to disclose the actual transaction. There are numerous options for keeping an eye on suspect Internet traffic. Frequently involve keeping track of a substantial portion of unused IP addresses or class A. With two important exceptions: all communication to and from the honeypot itself is noteworthy, and complete capture of traffic to and from the honeypot is available locally, forensic investigation of honeypot data isn't different from a similar examination of the compromised node. The several honeypots provided numerous log files for forensic analysis. The .pcap file, which most honeypots generate, is the most prevalent type of file. Every packet that the attacker and its victim communicated is recorded in this file. The forensic could observe what interaction occurred by accessing this with Wireshark.

Reverse engineering [137], [138] is another component of forensic investigation. A fraudster would certainly transfer one or more infections upon successfully infiltrating a system. Reverse engineering entails taking a deeper look at this infection by disassembling it in an effort to comprehend its functions and goals. However time-consuming it is, this method can help the forensic examiner uncover possible dangers. To gain more significant evidence and insights, collected data by honeypots can be analyzed quantitatively using conventional network analyzers. For instance, Network flows were devised keeping invoicing and budgeting in view and reported by network devices. However, since they are a cheap and effective manner of documenting network activity, security analysts can also benefit from them for a plethora of purposes, such as denial-of-service diagnosis and tunneling surveillance [139].

E. Honeypot in Cloud

Flexibility makes the cloud system more dynamic and sophisticated, which exacerbates the erratic security issues. In this scenario, the attack surface in the dynamic cloud system can rarely be protected in real-time by the defense systems that use perpetual implementation. To evaluate the variance in framework and automatically maximize the honeynet implementation, an automated honeynet distribution technique is employed for active protection in the container based cloud. In order to evaluate and optimize the honeynet deployment process, an attack graph for the container-based cloud system is established combining the degree centrality concept to design a framework to automatically generate, deliver, and customize the honeynet deployments.

XI. Conclusion and Future Enhancement

A responsive, ever-changing threat landscape for the enclosed network is pertinent for cyber defense in order to maintain a robust security posture. The Honeypot technology gives a wonderful tool for analysing and investigating cyber attacks to both experts and researchers. They can be set up in a variety of ways to track, mislead, or hinder the actions of attackers. It is in a unique position to gather information on emerging attacks and vulnerabilities which can empower enterprises to develop better and more sophisticated ways to safeguard their networks. Additionally, these systems lessen the number of false positive notifications, one of the major flaws of anomaly-based intrusion detection techniques. Furthermore, they can adapt to safeguard against zero-day data breaches.

This descriptive analysis was carried out to look into newly emerging trends in honeypot research. As a network security measure, honeypots were studied using a variety of sources from the literature. Despite their adaptability, honeypots should be used in addition to conventional security devices and not as a substitute for firewalls, defense-in-depth strategies, or IDS/IPS. Leveraging known systematic review and reporting techniques in honeypot research, this work advances research efforts through container technology in honeypots. Literature shows excellent strides in the advancement of honeypot design to prolongate attacker sessions and capture intrusion data for analysis, scanty definitive research has been found concerning how to improve the design of honeypots to spark the enthusiasm of assailants. Most researchers' strategies involve the advantage of readily accessible honeypot datasets. Researchers should think carefully when selecting a honeypot dataset. Hence, further exploration is required to demonstrate how the honeypot dataset would improve the accuracy of models with multiple parameters. It is envisaged that the work will provide a strong foundation for intrusion detection in current and next-generation communication technologies. In our subsequent revision, we envision the deployment of honeypots on edge systems and networks so that they can be effectively utilized for exchanging information about threats to other honeypots in the network. Through this association, the honeypots' efficiency could be increased, and an expanded view of the threat environment could be obtained.

XII. Acknowledgement

This research work was supported by the Pi DataCenters (Cloud Infrastructure) and Centre of Excellence, Cyber Security, VIT-AP University. The authors would like to thank the editors and reviewers. The authors (1,2,3) would like to thank Dr. S. V. Kota Reddy, Vice Chancellor, VIT-AP University, Dr. Jagadish Chandra Mudiganti, Registrar, VIT-AP University and Dr. Hari Seetha, Director, Centre of Excellence, Artificial Intelligence & Robotics (AIR) for their support. Special thanks to the team members of the CoE, Artificial and Robotics (AIR), VIT AP University.

References

[1] crowdstrikereport, "CrowdStrike:2022 Global Threat Report," last accessed on 6 Nov 2022. [Online]. Available: https://go.crowdstrike.com/rs/281-OBQ-266/images/Report2022GTR.pdf

[2] M. Rumez, D. Grimm, R. Kriesten, and E. Sax, "An overview of automotive service-oriented architectures and implications for security countermeasures," *IEEE Access*, vol. 8, pp. 221 852–221 870, 2020.

[3] R. Panigrahi, S. Borah, A. K. Bhoi, and P. K. Mallick, "Intrusion detection systems (ids)—an overview with a generalized framework," in *Cognitive Informa. and Soft Comput.*, 2020, pp. 107–117.

[4] E. Mahdavi, A. Fanian, A. Mirzaei, and Z. Taghiyarrenani, "Itl-ids: Incremental transfer learning for intrusion detection systems," *Knowledge-Based Sys.*, vol. 253, p. 109542, 2022.

[5] Z. Yan, W. Ding, X. Yu, H. Zhu, and R. H. Deng, "Deduplication on encrypted big data in cloud," *IEEE Trans. on big data*, vol. 2, no. 2, pp. 138–150, 2016.

[6] B. A. Tama, M. Comuzzi, and K.-H. Rhee, "Tse-ids: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94 497–94 507, 2019.

[7] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "Npp: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Trans. on Big Data*, vol. 8, no. 1, pp. 14–24, 2022.

[8] K.-K. R. Choo, M. Conti, and A. Dehghantanha, "Special issue on big data applications in cyber security and threat intelligence–part 1," *IEEE Transactions on Big Data*, vol. 5, no. 3, pp. 279–281, 2019.

[9] A. Voronkov, L. H. Iwaya, L. A. Martucci, and S. Lindskog, "Systematic literature review on usability of firewall configuration," *ACM Comp.Surv. (CSUR)*, vol. 50, no. 6, pp. 1–35, 2017.

[10] D. Fraunholz, S. D. D. Antón, C. Lipps, D. Reti, D. Krohmer, F. Pohl, M. Tammen, and H. D. Schotten, "Demystifying deception technology: A survey," *ArXiv*, vol. abs/1804.06196, 2018.

[11] M. Tang, M. Alazab, and Y. Luo, "Big data for cybersecurity: Vulnerability disclosure trends and dependencies," *IEEE Trans. on Big Data*, vol. 5, no. 3, pp. 317–329, 2017.

[12] Z. Jiang, W. Wang, B. Li, and Q. Yang, "Towards efficient synchronous federated training: A survey on system optimization strategies," *IEEE Trans. on Big Data*, vol. 9, no. 2, pp. 437–454, 2022.

[13] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "Pphopcm: Privacy-preserving high-order possibilistic c-means algorithm for big data clustering with cloud computing," *IEEE Trans. on Big Data*, vol. 8, no. 1, pp. 25–34, 2017.

[14] M. Guarascio, N. Cassavia, F. S. Pisani, and G. Manco, "Boosting cyber-threat intelligence via collaborative intrusion detection," *Future Gener. Comp. Sys*, vol. 135, pp. 30–43, 2022.

[15] W. Fan, Z. Du, D. Fernández, and V. A. Villagrá, "Enabling an anatomic view to investigate honeypot systems: A survey," *IEEE Sys.J.*, vol. 12, no. 4, pp. 3906–3919, 2018.

[16] R. Kaur and M. Singh, "A survey on zero-day polymorphic worm detection techniques," *IEEE Comm. Surv. & Tutor.*, vol. 16, no. 3, pp. 1520–1549, 2014.

[17] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, "A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems," *IEEE Comm.Surv. & Tuto.*, vol. 23, no. 4, pp. 2351–2383, 2021.

[18] Y. Sun, Z. Tian, M. Li, S. Su, X. Du, and M. Guizani, "Honeypot identification in softwarized industrial cyber–physical systems," *IEEE Trans. on Industrial Inform.*, vol. 17, no. 8, pp. 5542–5551, 2020.

[19] W. Fan, Z. Du, D. Fernández, and V. A. Villagra, "Enabling an anatomic view to investigate honeypot systems: A survey," *IEEE Sys. J.*, vol. 12, no. 4, pp. 3906–3919, 2017.

[20] M. Baykara and R. Das, "A novel honeypot based security approach for real-time intrusion detection and prevention systems," *J. of Info. Secu. and Appli.*, vol. 41, pp. 103–116, 2018.

[21] M. Lin, J. Xi, W. Bai, and J. Wu, "Ant colony algorithm for multi-objective optimization of container-based microservice scheduling in cloud," *IEEE Access*, vol. 7, pp. 83 088–83 100, 2019.

[22] F. Douglis and J. Nieh, "Microservices and containers," *IEEE Inter. Compu.*, vol. 23, no. 6, pp. 5–6, 2019.

[23] K. Gai, M. Qiu, and H. Zhao, "Privacy-preserving data encryption strategy for big data in mobile cloud computing," *IEEE Trans. on Big Data*, vol. 7, no. 4, pp. 678–688, 2017.

[24] W. Dai, L. Qiu, A. Wu, and M. Qiu, "Cloud infrastructure resource allocation for big data applications," *IEEE Trans. on Big Data*, vol. 4, no. 3, pp. 313–324, 2016.

[25] N. Zhao, V. Tarasov, H. Albahar, A. Anwar, L. Rupprecht, D. Skourtis, A. K. Paul, K. Chen, and A. R. Butt, "Large-scale analysis of docker images and performance implications for container storage systems," *IEEE Trans. Parall. and Distr.Sys.*, vol. 32, no. 4, pp. 918–930, 2021.

[26] P. S. Kocher, *Microservices and containers*. Addison-Wesley Professional, 2018.

[27] C. Kelly, N. Pitropakis, A. Mylonas, S. McKeown, and W. J. Buchanan, "A comparative analysis of honeypots on different cloud platforms," *Sensors*, vol. 21, no. 7, p. 2433, 2021.

[28] W. Fan, Z. Du, M. Smith-Creasey, and D. Fernandez, "Honeydoc: an efficient honeypot architecture enabling all-round design," *IEEE J.Selec. Areas in Comm.*, vol. 37, no. 3, pp. 683–697, 2019.

[29] W. Zhang, B. Zhang, Y. Zhou, H. He, and Z. Ding, "An iot honeynet based on multiport honeypots for capturing iot attacks," *IEEE Inter. Things J.*, vol. 7, no. 5, pp. 3991–3999, 2019.

[30] cuckoo, "Automated malware analysis," last accessed on 19 Sep 2022. [Online]. Available: https://cuckoosandbox.org/

[31] L. E. S. Jaramillo, "Malware detection and mitigation techniques: lessons learned from mirai ddos attack," *J. Info. Sys. Engg. & Mgmt.*, vol. 3, no. 3, p. 19, 2018.

[32] A. Higgins, "Adaptive containerised honeypots for cyber-incident monitoring," *Integrated Masters.Comp. Engg. (MAI)*, 2018.

[33] A. Matta, G. Sucharitha, B. Greeshmanjali, M. P. Kumar, and M. N. S. Kumar, "Honeypot: A trap for attackers," *The New Adv. Soci.: Arti. Intelli. and Industr. Internet of Things Paradigm*, pp. 91–101, 2022.

[34] W. Cabral, C. Valli, L. Sikos, and S. Wakeling, "Review and analysis of cowrie artefacts and their potential to be used deceptively," in *2019 Intl. Conf. Comput. Sci. and Comput. Intelli. (CSCI)*, 2019, pp. 166–171.

[35] L. Zhang and V. Thing, "Three decades of deception techniques in active cyber defense - retrospect and outlook," *Comp.& Sec.*, vol. 106, p. 102288, 2021.

[36] E. D. Saputro, Y. Purwanto, and M. F. Ruriawan, "Medium interaction honeypot infrastructure on the internet of things," in *2020 IEEE Intl.Conf. Internet.Things and Intelli. Sys. (IoTaIS)*. IEEE, 2021, pp. 98–102.

[37] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Comm. Surv.& Tutor.*, vol. 21, no. 3, pp. 2702–2733, 2019.

[38] C. Dalamagkas, P. Sarigiannidis, D. Ioannidis, E. Iturbe, O. Nikolis, F. Ramos, E. Rios, A. Sarigiannidis, and D. Tzovaras, "A survey on honeypots, honeynets and their applications on smart grid," in *2019 IEEE Conf.on Networ. Softwarization (NetSoft)*. IEEE, 2019, pp. 93–100.

[39] S. Krishnaveni, S. Prabakaran, and S. Sivamohan, "A survey on honeypot and honeynet systems for intrusion detection in cloud environment," *J.Computat. and Theoret. Nanosci.*, vol. 15, no. 9-10, pp. 2949–2953, 2018.

[40] T. Canary, "Know. When it Matters!" last accessed on 3 Nov 2022. [Online]. Available: https://canary.tools/

[41] L. Abeni, A. Balsini, and T. Cucinotta, "Container-based real-time scheduling in the linux kernel," *ACM SIGBED Revi.*, vol. 16, no. 3, pp. 33–38, 2019.

[42] I. Mavridis and H. Karatza, "Combining containers and virtual machines to enhance isolation and extend functionality on cloud computing," *Future Gen. Comp. Sys.*, vol. 94, pp. 674–696, 2019.

[43] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5g: A survey on mec-based approaches to provisioning and flexibility," *IEEE Comm. Surv. & Tutor.*, vol. 23, no. 1, pp. 596–630, 2020.

[44] Q. Tang, F. R. Yu, R. Xie, A. Boukerche, T. Huang, and Y. Liu, "Internet of intelligence: A survey on the enabling technologies, applications, and challenges," *IEEE Comm. Surv. & Tutor.*, vol. 24, no. 3, pp. 1394–1434, 2022.

[45] Z. Xiang, S. Pandi, J. Cabrera, F. Granelli, P. Seeling, and F. H. Fitzek, "An open source testbed for virtualized communication networks," *IEEE Commun. Magazi.*, vol. 59, no. 2, pp. 77–83, 2021.

[46] A. Kyriakou and N. Sklavos, "Container-based honeypot deployment for the analysis of malicious activity," in *2018 Global Infor. Infrastruc. and N/wking.Sympo. (GIIS)*. IEEE, 2018, pp. 1–4.

[47] S. Maesschalck, V. Giotsas, B. Green, and N. Race, "Don't get stung, cover your ics in honey: How do honeypots fit within industrial control system security," *Comp. & Sec.*, p. 102598, 2021.

[48] Packer, "packer:Build Automated Machine Images," last accessed on 27 Sep 2022. [Online]. Available: https://www.packer.io/

[49] dockerdocs, "Dockerdocs," last accessed on 11 Dec 2022. [Online]. Available: https://docs.docker.com/get-started/overview/

[50] S. Nathan, R. Ghosh, T. Mukherjee, and K. Narayanan, "Comicon: A co-operative management system for docker container images," in *2017 IEEE Intl.Conf. Cloud Engg. (IC2E)*. IEEE, 2017, pp. 116–126.

[51] U. Choi and K. Lee, "Dense or sparse: Elastic spmm implementation for optimal big-data processing," *IEEE Trans. on Big Data*, vol. 9, no. 2, pp. 637–652, 2022.

[52] S.-H. Han, H.-K. Lee, S.-T. Lee, S.-J. Kim, and W.-J. Jang, “Container image access control architecture to protect applications,” *IEEE Access*, vol. 8, pp. 162 012–162 021, 2020.

[53] S. Sultan, I. Ahmad, and T. Dimitriou, “Container security: Issues, challenges, and the road ahead,” *IEEE Access*, vol. 7, pp. 52 976–52 996, 2019.

[54] J. Nickoloff and S. Kuenzli, “Running software in containers,” in *Docker in action*. Simon and Schuster, 2019.

[55] docker Volume, “Dockerdocs:Use Volumes,” last accessed on 18 Dec 2022. [Online]. Available: https://docs.docker.com/storage/volumes/

[56] docker networks, “Dockerdocs: Network Containers,” last accessed on 12 Feb 2023. [Online]. Available: https://docs.docker.com/engine/tutorials/networkingcontainers/

[57] Z. Tang, W. Jia, X. Zhou, W. Yang, and Y. You, “Representation and reinforcement learning for task scheduling in edge computing,” *IEEE Trans. on Big Data*, vol. 8, no. 3, pp. 795–808, 2020.

[58] X. Wang, C. Xu, K. Wang, F. Yan, and D. Zhao, “Memory scaling of cloud-based big data systems: A hybrid approach,” *IEEE Trans. on Big Data*, vol. 8, no. 5, pp. 1259–1272, 2020.

[59] A. M. Potdar, D. Narayan, S. Kengond, and M. M. Mulla, “Performance evaluation of docker container and virtual machine,” *Proce. Comp.Sci.*, vol. 171, pp. 1419–1428, 2020.

[60] U. Altaf, G. Jayaputera, J. Li, D. Marques, D. Meggyesy, S. Sarwar, S. Sharma, W. Voorsluys, R. Sinnott, A. Novak *et al.*, “Auto-scaling a defence application across the cloud using docker and kubernetes,” in *2018 IEEE/ACM Intl.Conf. Utili.and Cloud Compu. Compani. (UCC Companion)*. IEEE, 2018, pp. 327–334.

[61] X. Gao, Z. Gu, M. Kayaalp, D. Pendarakis, and H. Wang, “Container-leaks: Emerging security threats of information leakages in container clouds,” in *2017 47th Annual IEEE/IFIP Intl.Conf.Dependa. Sys. and N/w. (DSN)*, 2017, pp. 237–248.

[62] X. Gao, B. Steenkamer, Z. Gu, M. Kayaalp, D. Pendarakis, and H. Wang, “A study on the security implications of information leakages in container clouds,” *IEEE Trans. on Dependa. and Secu. Compu.*, vol. 18, no. 1, pp. 174–191, 2018.

[63] D. Yu, Y. Jin, Y. Zhang, and X. Zheng, “A survey on security issues in services communication of microservices-enabled fog applications,” *Concurrency and Comput.: Practi. and Experi.*, vol. 31, no. 22, p. e4436, 2019.

[64] J. Chen, Z. Feng, J.-Y. Wen, B. Liu, and L. Sha, “A container-based dos attack-resilient control framework for real-time uav systems,” in *2019 Design, Automat. & Test in Europe Conf. & Exhibition (DATE)*, 2019, pp. 1222–1227.

[65] F. Reghenzani, G. Massari, and W. Fornaciari, “The real-time linux kernel: A survey on preempt_rt,” *ACM Comp.Surv. (CSUR)*, vol. 52, no. 1, pp. 1–36, 2019.

[66] L. Training, “capabilities(7) — Linux manual page,” last accessed on 19 Jan 2023. [Online]. Available: https://man7.org/linux/man-pages/man7/capabilities.7.html

[67] A. Randal, “The ideal versus the real: Revisiting the history of virtual machines and containers,” *ACM Comp. Surv.(CSUR)*, vol. 53, no. 1, pp. 1–31, 2020.

[68] dockerdocs, “dockerdocs:Docker and iptables,” last accessed on 18 Jan 2023. [Online]. Available: https://docs.docker.com/network/iptables/

[69] V. D. Priya and S. S. Chakkaravarthy, “Containerized cloud-based honeypot deception for tracking attackers,” *Scienti. Repor.*, vol. 13, no. 1, p. 1437, 2023.

[70] K. Wist, M. Helsem, and D. Gligoroski, “Vulnerability analysis of 2500 docker hub images,” in *Advanc. in Secu., Networ., and Internet of Things*. Springer, 2021, pp. 307–327.

[71] S. Kwon and J.-H. Lee, “Divds: Docker image vulnerability diagnostic system,” *IEEE Access*, vol. 8, pp. 42 666–42 673, 2020.

[72] M. De Benedictis and A. Lioy, “Integrity verification of docker containers for a lightweight cloud environment,” *Future gen. comp. sys.*, vol. 97, pp. 236–246, 2019.

[73] V. Atlidakis, P. Godefroid, and M. Polishchuk, “Restler: Stateful rest api fuzzing,” in *2019 IEEE/ACM 41st Intl. Conf. S/w. Engg. (ICSE)*. IEEE, 2019, pp. 748–758.

[74] A. Martin, S. Raponi, T. Combe, and R. Di Pietro, “Docker ecosystem–vulnerability analysis,” *Comp.Comm.*, vol. 122, pp. 30–43, 2018.

[75] M. Sabuhi, P. Musilek, and C.-P. Bezemer, “Studying the performance risks of upgrading docker hub images: A case study of wordpress,” in *Proc.2022 ACM/SPEC on Intl.Conf. Performance Engg.*, 2022, pp. 97–104.

[76] “CVE:Common Vulnerability Exploits,” last accessed on 28 September 2023. [Online]. Available: https://cve.mitre.org/cve/

[77] CROWDSTRIKE, “CROWDSTRIKE,” last accessed on 13 Aug 2022. [Online]. Available: https://www.crowdstrike.com/blog/compromised-docker-honeypots-used-for-pro-ukrainian-dos-attack/

[78] A. Ahmed and G. Pierre, “Docker-pi: Docker container deployment in fog computing infrastructures,” *Intl. J. Cloud Comp.*, vol. 9, no. 1, pp. 6–27, 2020.

[79] R. A. Oliveira, M. M. Raga, N. Laranjeiro, and M. Vieira, “An approach for benchmarking the security of web service frameworks,” *Future Gener. Comp. Sys.*, vol. 110, pp. 833–848, 2020.

[80] D. Huang, H. Cui, S. Wen, and C. Huang, “Security analysis and threats detection techniques on docker container,” in *2019 IEEE 5th Intl.Conf. Comp. and Comm. (ICCC)*. IEEE, 2019, pp. 1214–1220.

[81] A. Kedrowitsch, D. Yao, G. Wang, and K. Cameron, “A first look: Using linux containers for deceptive honeypots,” in *Proc. 2017 Workshop on Automated Decision Making for Active Cyber Defense*, 2017, pp. 15–22.

[82] T. Barron and N. Nikiforakis, “Picky attackers: Quantifying the role of system properties on intruder behavior,” in *Proc. 33rd Annual Comp. Sec. Apps. Conf.*, 2017, pp. 387–398.

[83] L. Artem, B. Tetiana, M. Larysa, and V. Vira, “Eliminating privilage escalation to root in containers running on kubernetes,” *Scienti. and practi. cyber sec. J.*, 2020.

[84] S. Suneja, A. Kanso, and C. Isci, “Can container fusion be securely achieved?” in *Proc. 5th Intl. Workshop on Container Tech. and Container Clouds*, 2019, pp. 31–36.

[85] NIST, “NIST: NATIONAL VULNERABILITY DATABASE,” last accessed on 12 Aug 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-5736/

[86] A. Jerrbi, “Aqua Blog: Mitigating High Severity RunC Vulnerability,” last accessed on 13 Dec 2022. [Online]. Available: https://blog.aquasec.com/runc-vulnerability-cve-2019-5736

[87] J. M. Acton, “Escalation through entanglement: How the vulnerability of command-and-control systems raises the risks of an inadvertent nuclear war,” *Intl. sec.*, vol. 43, no. 1, pp. 56–99, 2018.

[88] J. C. Kirchhof, L. Malcher, and B. Rumpe, “Understanding and improving model-driven iot systems through accompanying digital twins,” in *Proc. 20th ACM SIGPLAN Intl.Conf.Genera. Progra.: Concepts and Experi.*, 2021, pp. 197–209.

[89] C. Carrión, “Kubernetes scheduling: Taxonomy, ongoing issues and challenges,” *ACM Comp. Surv. (CSUR)*, 2022.

[90] S. M. Morsy and D. Nashat, “D-arp: An efficient scheme to detect and prevent arp spoofing,” *IEEE Access*, vol. 10, pp. 49 142–49 153, 2022.

[91] Y. Wang, Q. Wang, X. Chen, D. Chen, X. Fang, M. Yin, and N. Zhang, “Containerguard: A real-time attack detection system in container-based big data platform,” *IEEE Trans. Industr.Informat.*, 2020.

[92] R. Sprabery, K. Evchenko, A. Raj, R. B. Bobba, S. Mohan, and R. Campbell, “Scheduling, isolation, and cache allocation: A side-channel defense,” in *2018 IEEE Intl.Conf. Cloud Engg. (IC2E)*. IEEE, 2018, pp. 34–40.

[93] A. Zimba, Z. Wang, M. Mulenga, and N. H. Odongo, “Crypto mining attacks in information systems: An emerging threat to cyber security,” *J. Comp. Info. Sys.*, 2018.

[94] R. R. Karn, P. Kudva, H. Huang, S. Suneja, and I. M. Elfadel, “Cryptomining detection in container clouds using system calls and explainable machine learning,” *IEEE Trans. Parall. and Distri. Sys.*, vol. 32, no. 3, pp. 674–691, 2020.

[95] K. Jayasinghe and G. Poravi, “A survey of attack instances of cryptojacking targeting cloud infrastructure,” in *Proc. 2020 2nd Asia pacific information tech. conf.*, 2020, pp. 100–107.

[96] R. Lakshmanan, “New Cryptojacking Campaign Targeting Vulnerable Docker and Kubernetes Instances,” last accessed on 14 Jan 2023. [Online]. Available: https://thehackernews.com/2022/10/new-cryptojacking-campaign-targeting.html

[97] M. Schmall, “Github:schmalle/medpot ,” last accessed on 11 Dec 2022. [Online]. Available: https://github.com/schmalle/medpot

[98] “Github:mushorg/glutton ,” last accessed on 4 Dec 2022. [Online]. Available: https://github.com/mushorg/glutton

[99] MalwareTech, “Github:MalwareTech/CitrixHoneypot ,” last accessed on 22 Dec 2022. [Online]. Available: https://github.com/MalwareTech/CitrixHoneypot

[100] aelth, “Github:aelth/ddospot ,” last accessed on 14 Dec 2022. [Online]. Available: https://github.com/aelth/ddospot

[101] M. Keri, “Github:nsmfoo/dicompot ,” last accessed on 14 Dec 2022. [Online]. Available: https://github.com/nsmfoo/dicompot

[102] V. Bontchev, “Github:bontchev/ipphoney ,” last accessed on 22 Dec 2022. [Online]. Available: https://github.com/bontchev/ipphoney

[103] phin3has, “Github:phin3has/mailoney ,” last accessed on 25 Dec 2022. [Online]. Available: https://github.com/phin3has/mailoney

[104] G. Cirlig, “Github:huuck/ADBHoney,” last accessed on 14 Dec 2022. [Online]. Available: https://github.com/huuck/ADBHoney

[105] cowrie, “Github:cowrie/cowrie ,” last accessed on 12 Dec 2022. [Online]. Available: https://github.com/cowrie/cowrie

[106] C. Wellons, “Github:skeeto/endlessh ,” last accessed on 1 Dec 2022. [Online]. Available: https://github.com/skeeto/endlessh

[107] Cymmetria, “Github:Cymmetria/ ciscoasa_honeypot ,” last accessed on 14 Dec 2022. [Online]. Available: https://github.com/Cymmetria/ciscoasa_honeypot

[108] “Github:mushorg/conpot ,” last accessed on 15 Dec 2022. [Online]. Available: https://github.com/mushorg/conpot

[109] J. Vestergaard, “Github:johnnykv/heralding ,” last accessed on 11 Dec 2022. [Online]. Available: https://github.com/johnnykv/heralding

[110] V. Kolesnykov, “Github:sjinks/mysql-honeypotd ,” last accessed on 24 Dec 2022. [Online]. Available: https://github.com/sjinks/mysql-honeypotd

[111] DinoTools, “Github:DinoTools/dionaea ,” last accessed on 2 Dec 2022. [Online]. Available: https://github.com/DinoTools/dionaea

[112] D. Katz, “Github:Plazmaz/MongoDB-HoneyProxy ,” last accessed on 22 Dec 2022. [Online]. Available: https://github.com/Plazmaz/MongoDB-HoneyProxy

[113] MushMush, “Github:mushorg/glastopf ,” last accessed on 26 Dec 2022. [Online]. Available: https://github.com/mushorg/glastopf

[114] Lyrebird, “Dockerhuib:lyrebird/honeypot-base ,” last accessed on 2 Dec 2022. [Online]. Available: https://hub.docker.com/r/lyrebird/honeypot-base/

[115] D. Roberson, “Github:droberson/ssh-honeypot ,” last accessed on 12 Dec 2022. [Online]. Available: https://github.com/droberson/ssh-honeypot

[116] tpot, “Github: telekom-security/tpotce ,” last accessed on 13 Dec 2022. [Online]. Available: https://github.com/telekom-security/tpotce

[117] “NATIONAL VULNERABILITY DATABASE,” last accessed on 28 September 2023. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-44228

[118] G. Portokalidis, A. Slowinska, and H. Bos, “Argos: an emulator for fingerprinting zero-day attacks,” in *Proc. ACM SIGOPS EUROSYS’2006*, Leuven, Belgium, April 2006.

[119] C. L. School, “Legal Information Institute LLI: Open Access to Law Since 1992,” last accessed on 12 Dec 2022. [Online]. Available: https://www.law.cornell.edu/supremecourt/text/287/435

[120] Y. Sun, Z. Tian, M. Li, S. Su, X. Du, and M. Guizani, “Honeypot identification in softwarized industrial cyber–physical systems,” *IEEE Trans. Industr. Informat.*, vol. 17, no. 8, pp. 5542–5551, 2021.

[121] S. M. Kerner, “Gartner,” 2022, last accessed on 26 Dec 2022. [Online]. Available: https://www.techtarget.com/whatis/feature/Colonial-Pipeline-hack-explained-Everything-you-need-to-know

[122] J. Tian, R. Tan, X. Guan, Z. Xu, and T. Liu, “Moving target defense approach to detecting stuxnet-like attacks,” *IEEE trans. smart grid*, vol. 11, no. 1, pp. 291–300, 2019.

[123] A. Wang, W. Chang, S. Chen, and A. Mohaisen, “A data-driven study of ddos attacks and their dynamics,” *IEEE Trans. Dependable and Sec. Compu.*, vol. 17, no. 3, pp. 648–661, 2018.

[124] S. Litchfield, D. Formby, J. Rogers, S. Meliopoulos, and R. Beyah, “Rethinking the honeypot for cyber-physical systems,” *IEEE Internet Computing*, vol. 20, no. 5, pp. 9–17, 2016.

[125] A. Vetterl and R. Clayton, “Honware: A virtual honeypot framework for capturing cpe and iot zero days,” in *2019 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 2019, pp. 1–13.

[126] C. Irvene, D. Formby, S. Litchfield, and R. Beyah, “Honeybot: A honeypot for robotic systems,” *Proc.IEEE*, vol. 106, no. 1, pp. 61–70, 2018.

[127] R. K. Shrivastava, B. Bashir, and C. Hota, “Attack detection and forensics using honeypot in iot environment,” in *Intl. Conf. Distr. Compu. and Internet Techno.* Springer, 2019, pp. 402–409.

[128] S. M. Sajjad, M. Yousaf, H. Afzal, and M. R. Mufti, “emud: enhanced manufacturer usage description for iot botnets prevention on home wifi routers,” *IEEE Access*, vol. 8, pp. 164 200–164 213, 2020.

[129] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis, “Riotpot: a modular hybrid-interaction iot/ot honeypot,” in *26th European Sympos. Resear. in Comp. Sec. (ESORICS) 2021*. Springer, 2021.

[130] A. Ziaie Tabari and X. Ou, “A multi-phased multi-faceted iot honeypot ecosystem,” in *Proc. 2020 ACM SIGSAC Conf.Comp. and Comm. Sec.*, 2020, pp. 2121–2123.

[131] T. Luo, Z. Xu, X. Jin, Y. Jia, and X. Ouyang, “Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices,” *Black Hat*, vol. 1, pp. 1–11, 2017.

[132] J. D. Guarnizo, A. Tambe, S. S. Bhunia, M. Ochoa, N. O. Tippenhauer, A. Shabtai, and Y. Elovici, “Siphon: Towards scalable high-interaction physical honeypots,” in *Proc.3rd ACM Workshop on Cyber-Physi. Sys. Secu.*, 2017, pp. 57–68.

[133] T. S. Ustun, S. M. Farooq, and S. S. Hussain, “A novel approach for mitigation of replay and masquerade attacks in smartgrids using iec 61850 standard,” *IEEE Access*, vol. 7, pp. 156 044–156 053, 2019.

[134] C. Dalamagkas, P. Sarigiannidis, D. Ioannidis, E. Iturbe, O. Nikolis, F. Ramos, E. Rios, A. Sarigiannidis, and D. Tzovaras, “A survey on honeypots, honeynets and their applications on smart grid,” in *2019 IEEE Conf. on Networ. Softwarization (NetSoft)*, 2019, pp. 93–100.

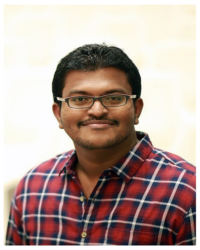
[135] K. Wang, M. Du, S. Maharjan, and Y. Sun, “Strategic honeypot game model for distributed denial of service attacks in the smart grid,” *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2474–2482, 2017.

[136] A. Belqruch and A. Maach, “Scada security using ssh honeypot,” in *Proc.2nd Intl.Conf.N/wwking., Infor. Sys.& Sec.*, 2019, pp. 1–5.

[137] M. Marchetti and D. Stabili, “Read: Reverse engineering of automotive data frames,” *IEEE Trans. Infor. Forensi. and Sec.*, vol. 14, no. 4, pp. 1083–1097, 2018.

[138] W. Choi, S. Lee, K. Joo, H. J. Jo, and D. H. Lee, “An enhanced method for reverse engineering can data payload,” *IEEE Trans. Vehicu.Techn.*, vol. 70, no. 4, pp. 3371–3381, 2021.

[139] P. Xu, C. Eckert, and A. Zarras, “Falcon: malware detection and categorization with network traffic images,” in *Intl. Conf. Artifi.Neural N/ws.* Springer, 2021, pp. 117–128.



Sibi C. Sethuraman (M’18) received his Ph.D from Anna University in the year 2018. He is an Associate Professor in the School of Computer Science and Engineering at Vellore Institute of Technology – Andhra Pradesh (VIT-AP) University. Further, he is the coordinator for Artificial Intelligence and Robotics (AIR) Center at VIT-AP. He is an active reviewer in many reputed journals of IEEE, Springer, and Elsevier. He is a recipient of DST fellowship.

Tharshith Goud Jadapalli (M’22) is currently pursuing his masters at Carnegie Mellon University (CMU), Pittsburgh, PA 15213, United States. His research interest includes Cyber Security, Network Security, Honeypots, Intrusion Detection System (IDS) etc.

Devi Priya Vimala Sudhakaran (Student Member, IEEE) is currently pursuing her Ph.D in Vellore Institute of Technology- Amaravathi (VIT-AP). Her areas of interest include Cloud Computing, Container Security, IDS, Deep Learning Approaches, Internet of Things in Consumer Electronics and Embedded Hardware.

Saraju P. Mohanty (Senior Member, IEEE) received the bachelor's degree (Honors) in electrical engineering from the Orissa University of Agriculture and Technology, Bhubaneswar, in 1995, the master's degree in Systems Science and Automation from the Indian Institute of Science, Bengaluru, in 1999, and the Ph.D. degree in Computer Science and Engineering from the University of South Florida, Tampa, in 2003. He is a Professor with the University of North Texas. His research is in "Smart Electronic Systems" which has been funded by National Science Foundations (NSF), Semiconductor Research Corporation (SRC), U.S. Air Force, IUSSTF, and Mission Innovation. He has authored 450 research articles, 5 books, and 10 granted and pending patents. His Google Scholar h-index is 55 and i10-index is 229 with 12,000 citations. He is regarded as a visionary researcher on Smart Cities technology in which his research deals with security and energy aware, and AI/ML-integrated smart components. He introduced the Secure Digital Camera (SDC) in 2004 with built-in security features designed using Hardware Assisted Security (HAS) or Security by Design (SbD) principle. He is widely credited as the designer for the first digital watermarking chip in 2004 and first the low-power digital watermarking chip in 2006. He is a recipient of 16 best paper awards, Fulbright Specialist Award in 2020, IEEE Consumer Electronics Society Outstanding Service Award in 2020, the IEEE-CS-TCVLSI Distinguished Leadership Award in 2018, and the PROSE Award for Best Textbook in Physical Sciences and Mathematics category in 2016. He has delivered 22 keynotes and served on 14 panels at various International Conferences. He has been serving on the editorial board of several peer-reviewed international transactions/journals, including IEEE Transactions on Big Data (TBD), IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), IEEE Transactions on Consumer Electronics (TCE), and ACM Journal on Emerging Technologies in Computing Systems (JETC). He has been the Editor-in-Chief (EiC) of the IEEE Consumer Electronics Magazine (MCE) during 2016-2021. He served as the Chair of Technical Committee on Very Large Scale Integration (TCVLSI), IEEE Computer Society (IEEE-CS) during 2014-2018 and on the Board of Governors of the IEEE Consumer Electronics Society during 2019-2021. He serves on the steering, organizing, and program committees of several international conferences. He is the steering committee chair/vice-chair for the IEEE International Symposium on Smart Electronic Systems (IEEE-iSES), the IEEE-CS Symposium on VLSI (ISVLSI), and the OITS International Conference on Information Technology (OCIT). He has mentored 3 post-doctoral researchers, and supervised 15 Ph.D. dissertations, 26 M.S. theses, and 20 undergraduate projects.