**International Science Journal of Engineering & Agriculture**
2025; 4(3): 15-26
https://isg-journal.com/isjea
doi: 10.46299/j.isjea.20250403.02
ISSN: 2720-6319

INTERNATIONAL SCIENCE GROUP

# Підвищення безпеки контейнерів за допомогою розгортання honeypot

**Ігор Андрущак**
Луцький національний технічний університет, Луцьк, Україна
ORCID 0000-0002-8751-4420

**Віктор Кошелюк**
Луцький національний технічний університет, Луцьк, Україна
ORCID 0000-0002-4136-5087

**Денис Ясашний**
Луцький національний технічний університет, Луцьк, Україна
ORCID 0009-0000-2409-1646

**Анотація:** Трансформація бізнесу в цифрову еру вимагає переходу на мікросервісну архітектуру, що передбачає створення додатків у вигляді незалежних сервісів. У цьому контексті контейнеризація набуває все більшого значення, а Kubernetes (K8s), завдяки своїй ефективності в управлінні контейнерами, стає критично важливим інструментом. Він забезпечує простоту розгортання, масштабування та управління додатками, що дозволяє компаніям прискорити процес цифрової трансформації. Незважаючи на свої переваги, Kubernetes не є повністю захищеною від загроз безпеки. Поряд із традиційними засобами захисту, важливим є вивчення тактики зловмисників. Honeypots стали невід'ємною частиною арсеналу засобів захисту від кіберзагроз. Розробники невпинно працюють над створенням нових видів приманок, щоб ефективно протистояти хакерам, які прагнуть проникнути в звичайні та промислові мережі. Honeypot служать для збору даних про хакерів, маскуючись під слабке місце в системі. Тому, дослідження ролі honeypots у підвищенні рівня безпеки інфраструктури є досить важливим. Ефективність систем-приманок у моніторингу кібератак та спроб вторгнення є доведеною, але їхнє розгортання в масштабах, достатніх для фіксації значної кількості інцидентів, викликає труднощі. В умовах зростання кіберзагроз, насамперед спрямованих на критичну інфраструктуру, назріла необхідність у розробці більш ефективних методів збору інформації про підозрілий трафік. Метою роботи є дослідження ефективності використання систем-приманок для виявлення вторгнень у контейнерних хмарних середовищах, зокрема в контексті Kubernetes. У цьому дослідженні ми зосереджуємось на розгортанні та використанні систем-приманок з метою підвищення безпеки середовища Kubernetes. У публікації стверджується, що ContainerSSH є найбільш вдалим рішенням для створення honeypot у середовищі Kubernetes, оскільки він поєднує в собі зрілість, простоту використання та широку сумісність. Підкреслюється також, що використання honeypot дозволяє отримати важливі дані про загрози та особливості захисту хмарних систем.

**Ключові слова:** honeypot, kubernetes, безпека, кластер.

## 1. Вступ

Хмарні обчислення та розподілені системи стали ключовими факторами трансформації IT-ландшафту. Їх популярність зумовлена можливістю отримувати ресурси за запитом, масштабованістю та зниженням витрат на управління інфраструктурою. Еволюція в напрямку мікросервісної архітектури відкрила нову еру розробки програмного забезпечення, дозволяючи створювати додатки з високим рівнем масштабованості та відмовостійкості. Розбиття програми на незалежні мікросервіси, кожен з яких виконує свою функцію, є ключовим фактором успіху в досягненні цих цілей. Зазначені обставини зумовили появу

контейнерів як оптимального рішення. Контейнери забезпечують ізоляцію мікросервісів та їх залежностей, створюючи незалежний блок, який може бути легко перенесений та запущений на будь-якій платформі, що сприяє узгодженості між середовищами розробки, тестування та впровадження [1].

Зі зростанням популярності контейнерів виникла потреба в ефективних інструментах для їх управління та масштабування. Саме тому з'явилися платформи оркестровки, такі як Kubernetes, які завдяки своїй гнучкості, масштабованості та здатності прискорювати розгортання, відіграють вирішальну роль у розвитку контейнерних технологій. У міру того, як Kubernetes стає стандартом де-факто, безпека його інфраструктури стає критично важливою. Зважаючи на те, що архітектура Kubernetes постійно ускладнюється, а кіберзагрози стають все більш витонченими, захист розгортань Kubernetes перетворюється на складне завдання для фахівців. Ефективне забезпечення безпеки контейнерних програм в Kubernetes потребує не тільки теоретичних знань, але й практичних навичок з ідентифікації та вирішення проблем безпеки, врахування можливих компромісів та застосування відповідних заходів захисту [2]. Для забезпечення надійності та стійкості систем оркестрування контейнерів необхідно досліджувати можливі вектори атак на Kubernetes та розробляти ефективні рішення для захисту.

Ефективним методом виявлення нових, ще невідомих кібератак є використання honeypot. Вони служать для відволікання зловмисників від критично важливих компонентів мережі, забезпечують раннє оповіщення про нові атаки та тенденції їх розвитку, а також надають можливість детального вивчення дій зловмисників під час та після їх контакту з honeypot [3]. Завдяки використання honeypot можна відвернути зловмисників від більш цінних систем у мережі, отримати своєчасне сповіщення про нові атаки та тренди їх здійснення, а також провести глибокий аналіз дій зловмисників під час та після їх взаємодії з honeypot.

Цінність honeypot полягає в тому, що він дозволяє отримати інформацію про зловжисників. Це досягається шляхом аналізу даних, які проходять через honeypot, таких як натискання клавіш, навіть якщо мережа захищена шифруванням. Мережеві системи захисту від вторгнень, покладаючись на сигнатури відомих атак, часто не здатні виявити нові, невідомі на момент їх встановлення, компрометації [4]. На відміну від них, honeypots мають можливість ідентифікувати навіть ті вразливості, які ще не були досліджені.

Системи-приманки (honeypots) можуть ідентифікувати компрометацію, навіть якщо зловжисник використовує невідомі раніше інструменти. Це можливо завдяки аналізу трафіку, який генерує honeypot. У хмарних системах такі приманки відіграють ключову роль в аналізі загроз, допомагаючи зрозуміти мотиви, цілі та поведінку зловмисників через зібрані та оброблені дані [5].

Незважаючи на наявність різноманітних рішень, що передбачають використання приманок, вони мають суттєвий недолік, який полягає в необхідності їх автономного розгортання та моніторингу. Кожне з цих рішень являє собою окремий сценарій або набір програмних компонентів, що потребують ручного адміністрування [6]. Сучасні технології віртуалізації та оркестрування контейнерів надають широкі можливості для розвитку honeypot-розробок, проте більшість з них залишаються невикористаними.

## 2. Об’єкт і предмет дослідження

Відсутність належного моніторингу контейнерів традиційними інструментами безпеки призводить до того, що важко встановити безпечні межі та, як наслідок, ускладнює забезпечення їх захисту. Застарілі інструменти мають обмеження у перевірці внутрішніх контейнерів, що створює проблеми для команд з кібербезпеки, оскільки вони повинні забезпечити захист додатків, які не підпадають під дію традиційних брандмауерів.

Безпека платформ оркестровки контейнерів, таких як Kubernetes, значною мірою залежить від ефективного контролю доступу. При проектуванні ІТ-інфрастуктури необхідно вжити

заходів для запобігання зловживанням з обліковими записами з розширеними привілеями, мережевим атакам та несанкціонованому переміщенню в межах системи. Для цього варто використовувати перевірені методи, як-от списки дозволених ресурсів, що вже добре зарекомендували себе в традиційних IT-середовищах. Окрім того, критично важливо забезпечити надійний зв'язок між компонентами Kubernetes кластера, особливо коли він використовується кількома програмами одночасно.

Об’єкт дослідження є підвищення захисту K8s, що охоплює спектр тем, спрямованих на забезпечення безпеки та цілісності контейнерних програм та інфраструктури, на якій вони працюють. Зосередження на технологіях та методах підвищення рівня захисту K8s може допомогти створити комплексний підхід до захисту середовищ Kubernetes, захищаючи як інфраструктуру, так і запущені програми.

В умовах зростаючої кількості кібератак, обманні стратегії та глибоке розуміння дій супротивника є ключовими елементами успішного захисту. Не дивно, що сьогодні ці питання є пріоритетними для ефективного функціонування IT-інфрастуктури організацій та державних органів. Ефективність механізмів обману полягає в їх несподіваності. Якщо противник завжди може передбачити час, місце або конфігурацію їх застосування, вони не досягнуть своєї мети. Наприклад, макет, який постійно розташований в одному і тому ж місці і не змінює свого зовнішнього вигляду, не буде ефективним засобом введення в оману. Застосування honeypot, використання фальшивих хостів, що вводяться в мережу для залучення зловмисників, є усталеною формою механізму обману в мережевому захисті. Предметом дослідження нашої роботи є ефективність розгортання honeypot для підвищення безпеки K8s.

## 3. Мета і задачі дослідження

Дослідження було спрямоване на поглиблене вивчення архітектури K8s з особливим акцентом на безпеку. Kubernetes розроблено з урахуванням вимог портативності, що дозволяє легко переносити його між різними хмарними середовищами (наприклад, між кількома відкритими хмарами). Складність конфігурації кластера робить його безпеку критично вразливою. Інженери повинні володіти повним спектром знань про можливі атаки та вразливості, що можуть виникнути через особливості системи. Тому ми пропонуємо використання honeypot для підвищення рівня безпеки K8s.

Ми зосередимось на розгортанні honeypot та комплексному аналізі необхідних процедур та інструментів, що сприятимуть підвищенню безпеки кластерів Kubernetes. Результатом роботи стане надання цінної інформації та рекомендацій для ефективного захисту середовища Kubernetes. У зв'язку з експоненціальним зростанням технологічного прогресу, глибоке розуміння проблем безпеки, що з цим пов'язані, є першочерговим завданням. Метою даної статті є внести вклад у дослідження безпеки Kubernetes, розглядаючи наступні питання:

- виявити та проаналізувати різні проблеми безпеки в архітектурі контейнерів K8s;
- запропонувати ефективні рішення для захисту архітектури Kubernetes з використанням honeypot;
- дослідити можливості розгортання honeypot для підвищення безпеки функціонування архітектури Kubernetes.

## 4. Аналіз літератури

У ході дослідження автори [7] здійснили глибокий аналіз можливостей систем-пасток у виявленні кібератак та довели їхню практичну користь. Отримані результати також свідчать про те, що інтеграція систем-пасток з методами машинного навчання відкриває нові можливості для прогнозування кібератак та розробки ефективних моделей для виявлення підозрілих профілів.

У рамках дослідження [8] було розглянуто ключові питання безпеки технології Інтернету речей (IoT). За допомогою підкріплювального навчання (RL) створено механізм-пастку для ідентифікації таких загроз, як DDoS-атаки та атаки «Людина посередині». Дослідження продемонстрували, що приманка, створена на основі алгоритмів машинного навчання з підкріпленням, характеризується високою точністю виявлення атак (до 99,96%) та перевершує за цим показником існуючі аналоги.

Дослідники Akond Rahman та ін. [9] розробили спеціальний інструмент, який допомагає знаходити помилки в конфігураціях Kubernetes. Вони застосували цей інструмент до своїх проектів, розміщених у відкритих сховищах, щоб з'ясувати, які саме проблеми безпеки найчастіше зустрічаються в маніфестах Kubernetes.

В роботі Islam Shamim [10] наведено створення кластеру для вивчення потенційних експлойтів. Дослідження, проведене автором, було спрямоване на виявлення оптимальних методів безпеки для кластерів Kubernetes. Результатом дослідження став перелік рекомендованих заходів безпеки для кластерів Kubernetes. Дослідник вивчив поширені помилки конфігурації та зібрав думки практиків щодо найкращих підходів до забезпечення безпеки.

Важливим аспектом дослідження безпеки Kubernetes є оцінка ефективності та порівняння різних підходів до захисту. В роботі [11] дослідники провели оцінку різних засобів сканування образів контейнерів, щоб з'ясувати, які з них найкраще підходять для виявлення вразливостей. Поряд з тим було здійснено порівняння продуктивності та безпеки різних середовищ виконання контейнерів для Kubernetes.

У дослідженні [12] було зазначено, що за допомогою системи приманки ми можемо захистити IT інфраструктру від розподілених атак на відмову в обслуговуванні. Вони проаналізували це за допомогою моделювання. Також в роботі було продемонстровано деякі розбіжності в системі виявлення та запобігання вторгненням для виявлення сигнатур атак, які використовують зловмисники, і прийшло до висновку, що ця система не є надійною та генерує помилкові спрацьовування та помилкові негативи.

Застосування програм-пасток [13] виявилося успішним інструментом для дослідження зловживань та зміцнення безпеки. Дослідження підтвердило, що цей метод є надійним для аналізу дій зловмисників та їхніх моделей поведінки. Експеримент з програмами-пастками продемонстрував їхню ефективність у розкритті тактик зловмисників та підвищенні рівня захисту інформаційних систем. Їхня робота з виявлення аномалій в мережі включала в себе аналіз існуючих методів, після чого вони запропонували власний метод, заснований на принципі ентропії.

В роботі [14] дослідники запропонували використовувати руткіт для приховування honeypot на рівні ядра, що забезпечить невидимість системи-пастки для зловмисників. За результатами нещодавнього дослідження [15], ChatGPT, попри наявні недоліки, розглядається як потенційно надійна система-пастка для захисту від кіберзагроз у майбутньому. Це є інноваційним рішенням у сфері honeypot. Результати мета-аналізу літератури з honeypot-систем [16] свідчать про те, що використання систем-приманок є оптимальним рішенням для виявлення сучасних методів атак та захисту архітектури IT інфраструктури.

Безпека Kubernetes є критично важливою, і дослідження, представлені в поданій публікації, закладають основу для подальшого прогресу в цій сфері.

## 5. Методи дослідження

В практичних сценаріях розробки та впровадження часто виникає необхідність використання одноузлового кластерного середовища. Незважаючи на те, що воно не відповідає вимогам для production-середовищ, воно є ефективним інструментом для локальної розробки та тестування. Для швидкого та легкого розгортання Kubernetes локально чудово підійде Minikube. Цей інструмент, створений розробниками Kubernetes, дозволяє створити

одноузловий кластер з більшістю ключових функцій, хоча слід враховувати певні обмеження, наприклад, у сфері мережі та вибору сховища. Більш реалістична модель кластера може бути створена шляхом застосування різних методів його налаштування в конфігурації з багатьма вузлами. Під час розгортання Kubernetes для оптимізації цього процесу часто застосовуються спеціальні інструменти, які допомагають спростити та автоматизувати створення й налаштування кластера:

- kubeadm забезпечує модульність встановлення Kubernetes, дозволяючи окремо розгортати компоненти керуючої площини та робочих вузлів. Конфігурація на основі файлів YAML забезпечує гнучкість та контроль над процесом встановлення;
- kubespray спрощує процес розгортання Kubernetes, автоматизуючи його за допомогою Ansible. Він підходить для розгортання в різних середовищах, таких як публічні хмари та фізичні сервери;
- kops характеризує комплексну систему для розгортання Kubernetes, яка забезпечує більше можливостей, ніж більшість інших інструментів. Крім безпосереднього встановлення кластера, kops також відповідає за створення та підтримку інфраструктури, на якій він працює. Незважаючи на підтримку різних хмарних провайдерів, kops може бути менш гнучким у порівнянні з kubeadm або kubespray.

Центральним елементом рівня керування Kubernetes виступає сервер API. Він функціонує як HTTP API, забезпечуючи можливість обміну даними та командами між користувачами, внутрішніми компонентами кластера Kubernetes та зовнішніми системами. API Kubernetes надає можливість здійснювати запити до об’єктів API Kubernetes та керувати їхнім поточним станом. Значна частина операцій реалізується за допомогою інтерфейсу командного рядка kubectl або інших інструментів командного рядка, зокрема kubeadm, які взаємодіють з API.

Сервіс kubeadm надає інструменти для контролю вузлів та оркестрування сервісів у динамічному середовищі. Головний вузол кластера Kubernetes виконує ключову роль в оркестрації, забезпечуючи створення, додавання, видалення та моніторинг як вузлів, так і сервісів. Kubeadm надає інструменти для провізіонування мінімально життєздатного Kubernetes кластера. Його функціонал також включає бутстрап головного вузла та подальшу інтеграцію робочих вузлів через механізм токенів, що встановлює двосторонні відносини довіри між ними.

Разом з тим, існує можливість безпосереднього доступу до API шляхом використання REST-викликів. Kubernetes реалізує підтримку кількох версій API, що розрізняються шляхами доступу, з метою спрощення операцій з видалення полів та реструктуризації представлень ресурсів. Версіонування API здійснюється на рівні кінцевої точки, а не на рівні окремих ресурсів або атрибутів. Це забезпечує консистентне відображення системних ресурсів та їхньої поведінки, а також надає можливість контролювати доступ до API з визначеним життєвим циклом, включаючи експериментальні реалізації.

З метою оптимізації процесу розробки та забезпечення масштабованості API, платформа Kubernetes впроваджує концепцію груп API, що надає можливість їхнього увімкнення та вимкнення. Ідентифікація API-ресурсів здійснюється на основі їхньої належності до певної групи API, типу, простору імен та назви. Архітектура API-сервера передбачає прозору обробку перетворень між різними версіями інтерфейсу. Слід зазначити, що кожна версія API є представленням єдиного джерела збережених даних, що забезпечує можливість обслуговування цих даних через множинні API.

Для створення production кластера часто використовують kubeadm. Kubespray є надбудовою над kubeadm, яка дозволяє розгортати кластер одразу на багатьох серверах. Це досягається завдяки використанню Ansible, що автоматизує повторювані дії на різних машинах. Kubespray використовує потужну можливість Ansible щодо організації хостів у групи. Це означає, що кілька серверів можна об’єднати під однією назвою, яка відображає їхню функціональну роль у Kubernetes-кластері. Наприклад, одна група може включати головні вузли, інша – робочі вузли, а третя – вузли з іншими специфічними завданнями. Після

ідентифікації інвентарю, конфігурація кластера розгортання здійснюється шляхом модифікації стандартних значень попередньо визначених параметрів. Використовуючи відповідні перемикачі та оператори, є можливість задати версію Kubernetes, необхідний мережевий плагін, час виконання контейнера, контролер вхідних з'єднань та додатки для зберігання даних. У результаті виконання Ansible playbook Kubespray, система автоматично перевірить вузли на наявність необхідних передумов та інсталює відсутні компоненти перед розгортанням кластера Kubernetes. По завершенні виконання сценарію адміністраторам буде надано можливість входу до головних вузлів та доступ до повнофункціонального кластера.

Однією з ключових переваг використання Kubespray є його підтримка керування життєвим циклом, яка забезпечує стабільність та працездатність розгорнутого кластера. Поряд з тим до ключових переваг Kubespray відносять можливість неруйнівного виконання таких операцій: додавання/видалення вузлів (робочих, головних), керування відбивачами маршрутів, оновлення/ротація сертифікатів API-сервера, а також оновлення та розгортання кластерів. Найбільш економічно ефективним рішенням є впровадження автоматичного оновлення кластера, яке охоплює не тільки оновлення версії Kubernetes, але й усіх його передумов та супутніх компонентів, зокрема etcd, середовище виконання контейнера та мережеві плагіни. Автоматизація процесів додавання та видалення робочих вузлів досягається завдяки системі моніторингу, яка здійснює динамічне масштабування кластера шляхом розгортання нових вузлів при перевищенні встановленого порогу навантаження, що виключає необхідність ручного втручання.

Сервіс kOps є інструментом автоматизованого розгортання кластерів Kubernetes. Він надає комплексну підтримку та використовує можливості хмарних провайдерів. При ініціалізації кластера Kubernetes адміністратор визначає роль для кожної групи екземплярів. Головні вузли об'єднуються в групу з роллю «Master». Робочі вузли, у свою чергу, входять до групи екземплярів з роллю «Nodes». Ці ролі є важливим фактором при визначенні порядку оновлення кластера. Однією з корисних можливостей kOps є можливість попереднього аналізу змін, які можуть бути внесені до вашого кластера Kubernetes внаслідок певної операції. Коли команда kOps виконується без параметра --yes, замість негайного застосування змін, вона демонструє їх попередній перегляд. Це дозволяє ретельно оцінити майбутні зміни та переконатися в їх коректності перед оновленням стану кластера.

Оновлення кластера відбувається поетапно. Спочатку обробляються головні групи екземплярів. Для кожної з них виконується наступна послідовність дій: вибирається головний вузол, він готується до виведення з експлуатації (злив), його робота завершується, і замість нього створюється новий головний вузол, який одразу запускається з цільовою версією Kubernetes. На цьому новому вузлі розгортаються всі сервіси, необхідні для його функціонування як головного вузла. Після їхньої готовності, цей вузол стає доступним. Цей процес повторюється для кожної головної групи екземплярів. Після успішного оновлення всіх головних вузлів, починається оновлення робочих груп. Для кожної робочої групи вибирається один або кілька вузлів, які спочатку від'єднуються від кластера. Потім створюються нові робочі вузли з оновленою версією Kubernetes, але без запущених на них робочих навантажень на початковому етапі. Зливаючи та завершуючи роботу вибраного головного вузла поточної версії, створюється новий головний вузол з потрібною версією Kubernetes. Після розгортання та готовності всіх специфічних для головного вузла модулів, цей вузол стає доступним. Потім, обираючи наступну групу головних екземплярів (за наявності), процес оновлення повторюється. Процес оновлення включає два етапи: спочатку оновлюються основні групи екземплярів, а потім – робочі. Оновлення робочих груп починається з вибору конкретної групи, після чого оновлюються її вузли. Процедура оновлення робочого вузла передбачає, на відміну від оновлення головного вузла, попереднє відключення вузла, призначеного для оновлення. Після цього ініціюється створення нового робочого вузла, який відповідає потрібній версії Kubernetes.

Готовий кластер високої доступності відображено на рисунку 1. За допомогою балансувальника навантаження клієнт буде перенаправлений на доступний сервер. Цей механізм забезпечує рівномірний розподіл навантаження між робочими вузлами, кінцевими користувачами та зовнішніми джерелами трафіку. Для оптимальної роботи з кластером рекомендується використовувати не менше трьох головних вузлів. Оптимізація архітектури може бути досягнута шляхом відокремлення кластера бази даних від панелі керування.

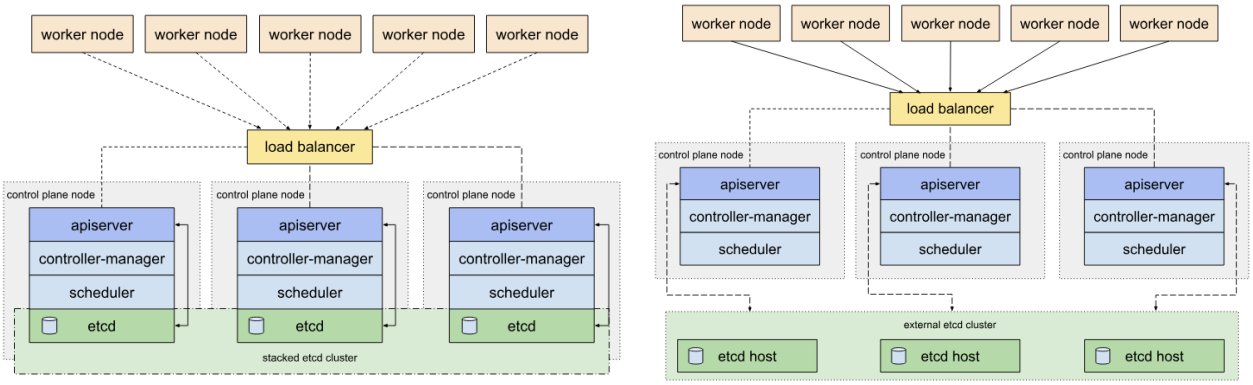


**Рисунок 1.** Топології високодоступних (HA) кластерів Kubernetes [17].

Перед ініціалізацією головного вузла Kubernetes слід звернути увагу на наступні аспекти. Система Kubernetes призначає кожному pod'у унікальну IP-адресу з доступного пулу. Отже, необхідно переконатися, що IP-адреси, присвоєні вузлам, є різними, щоб уникнути можливих конфліктів. Забезпечення успішної ініціалізації системи полягає у тому, щоб задати різні пули IP-адрес для вузлів та подів, що дозволить уникнути конфліктів та забезпечить стабільну роботу.

У цьому дослідженні розглядається рішення для хмарного розгортання, яке використовує контейнеризовану платформу Kubernetes. Рішення базується на аналізі та адаптації існуючих практик, що успішно застосовуються в традиційних корпоративних розгортаннях, для потреб хмарного середовища. Запропонований дизайн, за умови його теоретичного розгляду, може бути використаний не тільки в Kubernetes, але й в інших хмарних системах.

Оскільки технології контейнеризації стають все більш поширеними, а хмарна архітектура набуває популярності, традиційний підхід до використання honeypot та honeynets втрачає свою ефективність та продуктивність. Використання хмарної DMZ на віртуалізованому обладнанні дозволяє компанії відмовитися від фізичної демілітаризованої зони, з'єднаної з її організаційною мережею. Демілітаризована зона, відома також як «мережа периметра» або «екранована підмережа», виконує роль захисного шару між локальною мережею організації та зовнішнім світом, включаючи глобальну мережу Інтернет.

Для реалізації цього рішення було досліджено функціонал ContainerSSH з метою визначення залежностей між контейнерами. Загалом, будь-який honeypot з інтерактивними можливостями може бути використаний. Проте, у виробничому середовищі, ключовим фактором при виборі honeypot є визначення того, які системи потребують першочергового захисту.

Без розгортання приманок можна виміряти лише частину показників функціонування IT-інфраструктури, для інших потрібен кластер K8s. Honeypot необхідно налаштувати належним чином, щоб його можна було розгорнути в K8s. Ефективність роботи honeypots можна перевірити шляхом збору та аналізу журналів, які необхідно почати збирати та зберігати відразу після їх встановлення та введення в експлуатацію. З метою вимірювання використання ресурсів в умовах навантаження, клієнту слід передбачити механізм генерації трафіку. Високорівнева архітектура даної системи представлена на рисунку 2.

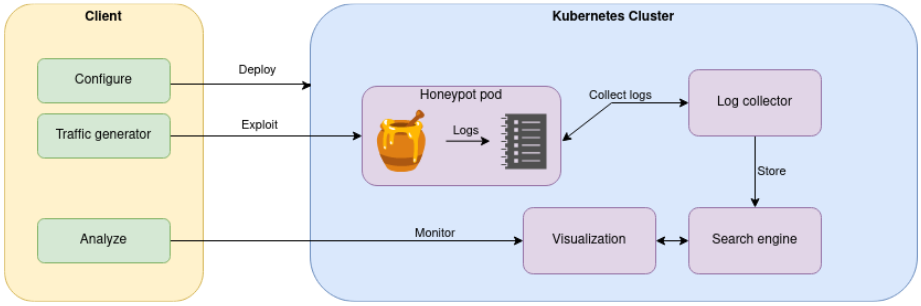


**Рисунок 2.** Архітектура функціонування honeypots в Kubernetes.

Концепція honeypot передбачає встановлення системи, навмисно сконфігурованої з недостатнім рівнем захисту операційної системи або наявністю ряду вразливостей, що спрощує несанкціоноване отримання доступу до її ресурсів. Ресурс типу «медова пастка» не має легітимного застосування для звичайних користувачів. Він розгортається з єдиною метою – фіксація дій кіберзлочинців. З цього випливає, що будь-який звичайний користувач, який взаємодіє з honeypot, з дуже високою ймовірністю (99%) є зловмисником.

Основна частина пасток для зловмисників розгортається в межах мережевих екранів і виступає інструментом для нагляду та фіксації дій порушників. Принцип їхньої роботи є протилежним до брандмауерів: якщо брандмауер пропускає вихідні з'єднання та відсікає значну частину вхідних на основі заданих інструкцій, то honeypot приймає будь-який вхідний трафік, але блокує будь-які вихідні спроби. Слід наголосити на тому, що з метою забезпечення прихованості honeypot, його реєстрація в будь-яких серверах імен або інших подібних системах є неприпустимою. Це є принциповим моментом, оскільки лише за умови правильного налаштування мережевої інфраструктури можна з високою вірогідністю ідентифікувати кожен пакет, адресований до honeypot, як потенційну атаку. Некоректна конфігурація неминуче спричинить експоненційне зростання кількості хибних позитивних спрацювань, що може мати наслідком відмову в обслуговуванні. Важливість даних, здобутих за посередництвом Honeypot, може змінюватись в залежності від його призначення.

Honeypot втрачає свою привабливість для зловмисника, якщо той знає про його існування. Те ж саме стосується і сканера портів – знання про його присутність дозволяє зловмиснику пройти повз систему захисту, залишившись непоміченим. Розв'язання цієї складної проблеми можливе шляхом застосування технології виявлення аномалій, що передбачає контроль за кожним пакетом в системі. Разом з тим, це неминуче спричинить накопичення великої кількості журнальної інформації про всі операції з пакетами. Зловмисник, володіючи знаннями про систему, може використовувати стеганографічні методи для приховування шкідливого трафіку, імітуючи безпечні пакети даних.

Збір даних про всі підозрілі дії в системі викликає обґрунтовані побоювання щодо можливого розголошення конфіденційної інформації. Безпечне зберігання цих журналів є важливим кроком до вирішення цієї проблеми, але навіть сам факт їх існування може становити загрозу для прав користувачів на недоторканність приватного життя. Зловмисник, який проникає в систему, не має жодних законних підстав для того, щоб його дії не фіксувалися. Для підсилення цього факту можна використовувати повідомлення для всіх користувачів, яке попереджає про те, що їхня діяльність в системі може бути відстежена.

## 6. Результати досліджень

У ході дослідження планується відстежити максимальну кількість атак. Це буде досягнуто шляхом розгортання екземпляру приманки ContainerSSH в кластері Kubernetes. Дані та

журнали якої будуть збиратися в базі даних. Для аналізу атак використовуються засоби візуалізації даних. З метою забезпечення безпеки, системи-пастки (honeypots) повинні функціонувати у віртуальному середовищі, щоб запобігти завданню шкоди основним системам.

Для розгортання кластеру Kubernetes ми використовували робочу станцію на базі Intel Xeon E5-2696 V4, що забезпечує частоту процесору на рівні 2.20 GHz - 3.70 GHz з кількістю ядер/потоків 22/44. Робоча станцію функціонувала на операційній системі Ubuntu Server 22.04 LTS з використанням 64Gb RAM та жорсткого диску 1Tb для постійного зберігання даних. Основні параметри розгортання приманки ContainerSSH в кластері відображено за допомогою Kubernetes Dashboard на рисунку 3.

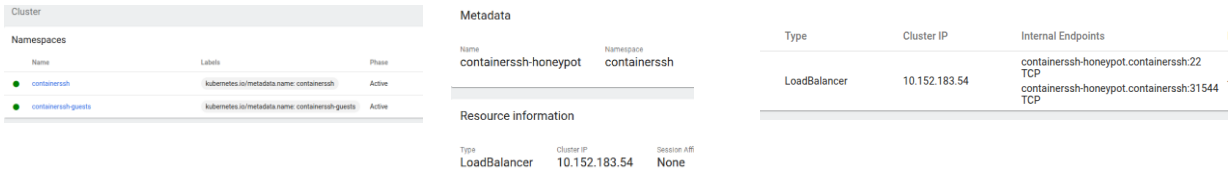


**Рисунок 3.** Параметри розгортання ContainerSSH в Kubernetes Dashboard.

Моніторинг та логування аномального трафіку відбувалось з використанням ELK Stack. ELK Stack забезпечує миттєвий пошук та аналіз даних, а також візуалізує результати, не змінюючи початкову архітектуру системи. Коректне функціонування стеку забезпечується завдяки використанню різноманітних програм Beats, які являють собою набір малоресурсних функцій, що виконують роль первинних збирачів журналів. Ці агенти розгортаються на різних серверах існуючої інфраструктури, виконуючи функції збору журналів або метрик.

Сервіс Filebeat зчитує логи, запускаючи один або кілька входів для моніторингу заданих джерел. Filebeat ініціює збирач (harvester) для кожного виявленого журналу, який послідовно зчитує нові події та передає їх до Libbeat. Libbeat агрегує події, пересилає агреговані дані на сконфігурований вихід Filebeat та зберігає стан збирача (offset), забезпечуючи можливість відновлення після помилок та усуваючи дублювання подій. На рисунку 4 продемонстровано вікно Filebeat з виявлення аномального трафіку ContainerSSH.

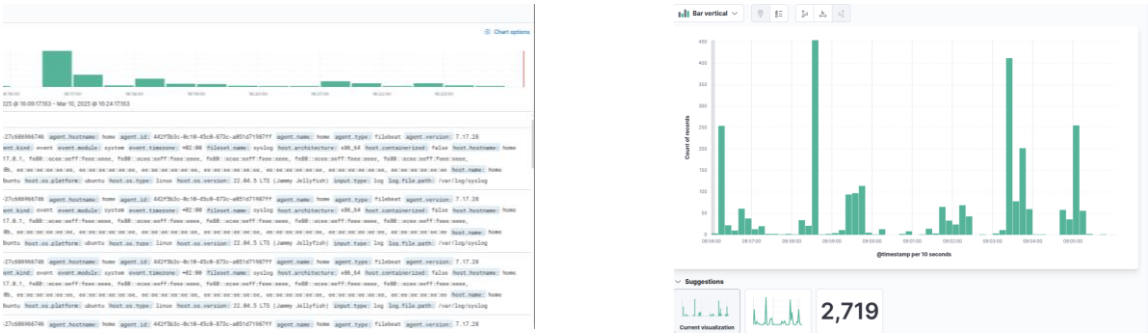


**Рисунок 4.** Інструментарій Filebeat з виявлення аномального трафіку ContainerSSH.

Використання сервісу Metricbeat дозволило контролювати необхідні показники функціонування операційної системи та серверних служб при взаємодії кластеру Kubernetes, приманки ContainerSSH та ELK Stack. Metricbeat збирає та передає метрики та статистику до вказаних джерел, таких як Logstash або Elasticsearch, забезпечуючи моніторинг серверів через збір даних з систем та серверних служб (HAProxy, Apache, MySQL, PostgreSQL, Nginx, MongoDB, Redis, System, Zookeeper).

## 7. Висновки

Основна функція honeypots полягає не в запобіганні атакам, а в зборі інформації про те, як хакери намагаються обійти систему захисту. Це допомагає нам краще зрозуміти їхні методи та підготуватися до реальних загроз. Дослідження шкідливого трафіку, такого як IP-адреси, імена користувачів, паролі та команди, дозволяє отримати краще розуміння тактики та цілей зловмисників. Інформація, зібрана з пасток, яка вважається зловживанням, є цінним ресурсом для організації, що сприяє вдосконаленню інших аспектів безпеки. Аналіз тестових даних, отриманих за допомогою honeypots, показує, що вони здатні збирати різноманітну інформацію, яку потім можна використовувати для різних цілей.

Вивчаючи записи сеансів командного рядка, ми можемо з'ясувати, які дані є найбільш привабливими для хакерів та які вразливості вони намагаються використати. Ефективним способом протидії є зміна конфігурації та переміщення файлів, що значно ускладнить зловмисникам досягнення їхніх злочинних цілей. Зібравши IP-адреси, ми можемо ідентифікувати зловмисників та заблокувати їх доступ до мережі за допомогою правил брандмауера. Додатково, аналізуючи використані імена користувачів та паролі, ми можемо зіставити їх з обліковими даними нашої організації для виявлення слабких паролів та їх подальшої заміни.

Кожен медовий горщик має свої переваги та недоліки. Наприклад, sshesame ефективний у використанні ресурсів, але має обмежені можливості налаштування. Для деяких користувачів це може бути і позитивно, але досвідченішим користувачам явно бракуватиме більшого. До того ж, sshesame має серйозний недолік у вигляді поганої документації, через що процес пошуку та усунення несправностей може стати справжнім випробуванням

За роки свого існування cowrie зарекомендувала себе як надійна технологія обману, що постійно розвивається. Маючи найкращі показники підтримки та ресурсоефективності серед аналогічних систем, cowrie все ж має слабке місце – реєстрація даних сеансу командного рядка. Якщо зловмисники отримають доступ до вашого SSH-сеансу та зможуть виконувати команди, і якщо вони успішно змінять файлову систему вашого honeypot, вам потрібно буде перезапустити honeypot, щоб повернути його до початкового стану.

Кожен раз при з'єднанні з ContainerSSH запускається новий контейнер, що вирішує згадану проблему. Проте, варто враховувати, що це може призвести до більшого використання ресурсів, ніж у інших інструментів. Результати дослідження підтверджують, що використання приманки ContainerSSH в K8s є ефективним способом захисту існуючих сховищ.

Для ефективного захисту від нових загроз, слід відмовитися від фрагментарного підходу та перейти до використання комплексних систем з інтегрованими пастками-приманками (honeypot), що дозволять автоматично виявляти та реагувати на них. Ефективність використання honeypot у виявленні загроз та реагуванні на атаки є беззаперечною. Інтеграція з сучасними K8s-фреймворками для аналізу даних в режимі реального часу та поширення інформації про загрози між компонентами безпеки, такими як брандмауери, IDS/IPS, антивіруси та системи оповіщення, дозволяє створити систему безпеки, яка постійно адаптується та здатна виявляти та нейтралізувати нові атаки в процесі їх виконання.

Результати дослідження свідчать про те, що організації можуть успішно виявляти зловмисні дії за допомогою honeypot-рішень, інтегрованих в архітектуру K8s. Це дослідження дало ствердну відповідь на питання про те, чи підвищують honeypots безпеку K8s-інфраструктури. Зважаючи на зростаючу складність атак та шкідливого програмного забезпечення, більш ефективні адаптивні конструкції honeypot можуть відіграти вирішальну роль у забезпеченні безпеки майбутніх розгортань K8s. Дана робота зосереджена на оцінці технології ContainerSSH. Проте, розвиток ринку може призвести до появи нових K8s-орієнтованих технологій, що відкриє перспективи для майбутніх досліджень у цій сфері.

**Список літератури:**

1) Priya, Devi & Chakkaravarthy, Sibi. (2023). Containerized cloud-based honeypot deception for tracking attackers. Scientific Reports. 13. 1437. 10.1038/s41598-023-28613-0.

2) Trần, Minh-Ngọc & Vu, Dinh-Dai & Kim, Younghan. (2022). A Survey of Autoscaling in Kubernetes. 263-265. 10.1109/ICUFN55119.2022.9829572.

3) Kovalenko, V.V. & Bukasov, Maksym. (2024). Scheduling Methods and Models for Kubernetes Orchestrator. Visnyk of Vinnytsia Politechnical Institute. 175. 86-94. 10.31649/1997-9266-2024-175-4-86-94.

4) Jadhav, Yogesh & Sable, Arjun & Suresh, Maithri & Hanawal, Manjesh. (2023). Securing Containers: Honeypots for Analysing Container Attacks. 225-227. 10.1109/COMSNETS56262.2023.10041276.

5) Viktor Kosheliuk, Yurii Tulashvili. Implementing Honeypots for Detecting Cyber Threats with AWS using the ELK. International Journal of Computing, 23(4), 618-624. https://doi.org/10.47839/ijc.23.4.3761

6) Reddy, Dinesh. (2023). Security in Kubernetes: A Comprehensive Review of Best Practices. International Journal of Science and Research (IJSR). 12. 10.21275/SR24304111526.

7) Amal, M. R., & Venkadesh, P. (2022). Review of cyber attack detection: Honeypot system. Webology, 19(1), 5497-5514.

8) Pashaei, A., Akbari, M. E., Lighvan, M. Z., & Charmin, A. (2022). Early Intrusion Detection System using honeypot for industrial control networks. Results in Engineering, 16, 100576.

9) A. Rahman, S.I. Shamim, D.B. Bose, and R. Pandita. Security Misconfigurations in Open-Source Kubernetes Manifests: An Empirical Study. ACM Transactions on Software Engineering and Methodology, Volume 32, Issue 4 Article No.: 99, pp. 1 – 36 https://doi.org/10.1145/357963

10) Md Shazibul Islam Shamim, Farzana Ahamed Bhuiyan, Akond Rahman. Xi commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices. 2020 IEEE Secure Development (SecDev). pp. 58-64

11) Shay Berkovich, Jeffrey Kam, and Glenn Wurster. 2020. UBCIS: ultimate benchmark for container image scanning. In Proceedings of the 13th USENIX Conference on Cyber Security Experimentation and Test (CSET'20). USENIX Association, USA, Article 10, 10.

12) Wang, X., Guo, N., Gao, F., & Feng, J. (2019). Distributed denial of service attack defence simulation based on honeynet technology. Journal of Ambient Intelligence and Humanized Computing, 1-16.

13) Shukla, A. S., & Maurya, R. (2018). Entropy-based anomaly detection in a network. Wireless Personal Communications, 99(4), 1487-1501.

14) Mohammadzad, M., & Karimpour, J. (2023). Using rootkits hiding techniques to conceal honeypot functionality. Journal of Network and Computer Applications, 103606.

15) McKee, F., & Noever, D. (2023). Chatbots in a Honeypot World. arXiv preprint arXiv:2301.03771.

16) Ikuomenisan, Gbenga & Morgan, Yasser. (2022). Meta-Review of Recent and Landmark Honeypot Research and Surveys. Journal of Information Security. 13. 181-209. 10.4236/jis.2022.134011.

17) Options for Highly Available Topology. Available at: https://kubernetes.io/docs/setup/productionenvironment/tools/kubeadm/ha-topology/

# Improving container security with honeypot deployment

**Igor Andrushchak**
Department of Software Engineering, Lutsk National Technical University, Lutsk, Ukraine
ORCID 0000-0002-8751-4420

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

**Denys Yasashnyi**
Department of Computer Science, Lutsk National Technical University, Lutsk, Ukraine
ORCID 0009-0000-2409-1646

**Abstract:** The transformation of business in the digital era requires a transition to a microservices architecture, which involves creating applications as independent services. In this context, containerization is gaining increasing importance, and Kubernetes (K8s), due to its efficiency in container management, is becoming a critical tool. It provides ease of deployment, scaling, and management of applications, which allows companies to accelerate the process of digital transformation. Despite its advantages, Kubernetes is not completely protected from security threats. Along with traditional means of protection, it is important to study the tactics of attackers. Honeypots have become an integral part of the arsenal of means of protection against cyber threats. Developers are constantly working on creating new types of decoys to effectively resist hackers seeking to penetrate conventional and industrial networks. Honeypots are used to collect data about hackers by disguising themselves as a weak point in the system. Therefore, research into the role of honeypots in improving infrastructure security is quite important. The effectiveness of decoy systems in monitoring cyberattacks and intrusion attempts is proven, but their deployment on a scale sufficient to capture a significant number of incidents is difficult. In the context of increasing cyber threats, primarily aimed at critical infrastructure, there is a need to develop more effective methods for collecting information about suspicious traffic. The purpose of the work is to study the effectiveness of using decoy systems to detect intrusions in container cloud environments, in particular in the context of Kubernetes. In this study, we focus on the deployment and use of decoy systems to improve the security of the Kubernetes environment. The publication claims that ContainerSSH is the most successful solution for creating a honeypot in the Kubernetes environment, as it combines maturity, ease of use and broad compatibility. It is also emphasized that using a honeypot allows you to obtain important data about threats and features of cloud system protection.

**Keywords:** honeypot, kubernetes, security, cluster.