МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи магістра

на тему: «Проект підсистеми моніторингу подій безпеки SOC в Cloud-Native середовищах на базі безкоштовних відкритих рішень»

Виконавв студент 2-го курсу, групи КБм-22-1
спеціальності 125 «Кібербезпека»
_____ М.В. Гончаров
Керівник, завідувач кафедри комп'ютерної інженерії
та кібербезпеки, кандидат технічних наук, доцент
_____ А.А. Єфіменко
Рецензент, завідувач кафедри комп'ютерних наук,
доктор філософії (PhD)
_____ М.С. Граф

Житомир – 2023

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
_____ Андрій Єфіменко
13 грудня 2023 р.

ЗАВДАННЯ

на випускну кваліфікаційну роботу

Студент: *Гончаров Михайло Валерійович*

Тема роботи: *Проект підсистеми моніторингу подій безпеки SOC в Cloud-Native середовищах на базі безкоштовних відкритих рішень.*

Затверджена Наказом університету від «13» жовтня 2023 р. № 572/с

Термін здачі студентом закінченої роботи _____ 2023 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження):

Об'єктом дослідження є процес проектування підсистеми моніторингу подій безпеки SOC з використанням рішень IBM QRadar та аналіз її функціонування шляхом проведення тестування на проникнення. Предметом дослідження є технологія проектування підсистем моніторингу подій безпеки SOC з використанням рішень IBM QRadar.

Обладнання та програмне забезпечення:

1. Інструменти віртуалізації мережі GNS3 та Oracle VM Virtualbox;
2. ОС Ubuntu Server;
3. SIEM-система ELK + фреймворк ElastAlert 2;
4. SOAR-система TheHive Project (TheHive + Cortex);
5. Умисно вразливе cloud-native середовище Kubernetes Goat
6. Інструмент безпеки виконання в cloud-native середовищах Falco.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Воротніков В.В.	13.10.2023	13.10.2023
2	Воротніков В.В.	13.10.2023	13.10.2023
3	Воротніков В.В.	13.10.2023	13.10.2023

Студент _____ М.В. Гончаров

Керівник _____ А.А. Єфіменко

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Видача завдання	01.03.2022	виконано
2	Розробка плану роботи	02.04.2022	виконано
3	Розробка розгорнутого плану	08.04.2022	виконано
4	Подача чернетки керівнику для перевірки	10.04.2022	виконано
5	Доопрацювання роботи та перевірка керівником	12.04.2022	виконано
6	Затвердження графічних матеріалів у керівника	19.04.2022	виконано
7	Проведення попереднього захисту	20.06.2022	виконано
8	Отримання відгуку керівника	20.06.2022	виконано
9	Допуск завідувача кафедри	20.06.2022	виконано
10	Подання роботи на рецензування	20.06.2022	виконано

Студент _____ М.В. Гончаров

Керівник _____ А.А. Єфіменко

АНОТАЦІЯ

Завдання кваліфікаційної роботи магістра полягає у вивченні та обґрунтуванні вибору інструментів, програмо-апаратних рішень, та методів виявлення загроз в cloud-native середовищах, а також використання отриманих знань для розробки підсистеми моніторингу подій безпеки, що комбінує в собі безкоштовні відкриті рішення та становить базу для SOC підприємств малих та середніх масштабів.

Кваліфікаційна робота магістра на тему «Проект підсистеми моніторингу подій безпеки SOC в Cloud-Native середовищах на базі безкоштовних відкритих рішень» складається з інженерного, логічного, аналітичного, та теоретичного комплексів та пояснювальної записки.

Пояснювальна записка до випускної кваліфікаційної роботи містить в собі 73 сторінки, 7 таблиць та 55 рисунків. Список використаних джерел містить 21 найменування і займає 2 сторінки.

КЛЮЧОВІ СЛОВА: SOC, МОНІТОРИНГ, РЕАГУВАННЯ, ІНЦИДЕНТ БЕЗПЕКИ, SIEM, SOAR, CLOUD-NATIVE, КОНТЕЙНЕР, ПОД, ВУЗОЛ, КЛАСТЕР.

ANNOTATION

The task of the master's thesis is to study and justify the choice of tools, software and hardware solutions, and methods for detecting threats in cloud-native environments, as well as to use the knowledge gained to develop a security event monitoring subsystem that combines free open-source solutions and forms the basis for the SOC of small and medium-sized enterprises.

The master's thesis on the topic "Design of a subsystem for monitoring security events of SOC in Cloud-Native environments based on free open-source solutions" consists of engineering, logical, analytical, and theoretical complexes and an explanatory note.

The explanatory note to the qualification work contains 73 pages, 7 tables and 55 figures. The list of used sources contains 21 names and occupies 2 pages.

KEYWORDS: SOC, MONITORING, RESPONSE, SECURITY INCIDENT, SIEM, SOAR, CLOUD-NATIVE, CONTAINDER, POD, NODE, CLUSTER.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс програмування додатків (Application Programming Interface)

ELK Stack – SIEM-система, що складається з компонентів Elasticsearch, Logstash та Kibana.

FOSS – безкоштовне відкрите програмне забезпечення (Free Open-Source Software)

IOC – індикатор компрометації (Indicator of Compromise)

SOC – операційний центр безпеки (SOC)

SIEM – система керування інформацією та подіями безпеки (Security Information and Event Management)

SOAR – система оркестрації, автоматизації та реагування (Security Orchestration, Automatization and Response).

K8s – відкрита система для автоматизованого розгортання, масштабування та управління контейнерами (Kubernetes).

ВСТУП

Актуальність теми. Сучасний технологічний ландшафт надто динамічний, і перехід до cloud-native середовищ та контейнеризації відкриває нові можливості для розвитку бізнесу, проте при цьому стає набагато складніше забезпечити ефективний рівень кібербезпеки. Зростаюча складність сучасних інфраструктур та збільшення кількості загроз стають викликом для забезпечення безпеки в cloud-native середовищах. Обрана тема виходить за межі звичайних стратегій безпеки, пропонуючи інноваційний підхід до створення власного системного рішення, об'єднуючи безкоштовні відкриті інструменти та ефективно інтегруючи їх для виявлення та відповіді на загрози в cloud-native середовищах.

Крім того, динаміка атак в цифровому просторі швидко змінюється, і важливість миттєвого виявлення та реагування на інциденти надзвичайно висока. Застосування системи моніторингу подій безпеки (SOC) в cloud-native середовищах не тільки вирішує це завдання, але й стає ключовим елементом стратегії кібербезпеки, дозволяючи швидше виявляти аномалії та атаки, а також вживати необхідні заходи для їх запобігання. У контексті використання безкоштовних відкритих рішень ця тема стає особливо актуальною, оскільки дозволяє підвищити доступність розширеним організаціям до ефективних засобів моніторингу та реагування на інциденти, навіть при обмеженому бюджеті для кібербезпеки. Таким чином можна сказати, що **тема роботи** направлена на проектування підсистеми моніторингу подій безпеки SOC в cloud-native середовищах на базі безкоштовних відкритих рішень **є актуальною**.

Мета і завдання роботи. Метою кваліфікаційної роботи є розробка та впровадження ефективної підсистеми моніторингу подій безпеки в cloud-native середовищах на базі безкоштовних відкритих рішень. Робота націлена на створення комплексного підходу до забезпечення безпеки в таких середовищах, де використовуються контейнери та інші cloud-native технології.

Досягнення поставленої мети передбачає розв’язання наступних завдань:

- Проведення аналізу технологічного ландшафту cloud-native;
- Визначення SOC, як важливого елементу безпеки cloud-native;

– Аналіз ринку безкоштовних відкритих рішень SOC;

– Вибір конкретних рішень для реалізації підсистеми моніторингу;

– Проектування та налаштування віртуального середовища;

– Визначення моделі загроз, що відповідатиме середовищу;

– З’єднання всіх компонентів середовище в єдину підсистему моніторингу, виконання перевірки функціонування системи та надання рекомендацій що до її покращення.

Об’єктом дослідження є процес проектування та побудови підсистеми моніторингу SOC для cloud-native середовищ на базі безкоштовних відкритих рішень.

Предметом дослідження є технології SOC для моніторингу cloud-native середовищ, зокрема SIEM-система ELK Stack, SOAR-система TheHive + Cortex та інструмент безпеки виконання в cloud-native середовищах Falco.

Методи дослідження. В кваліфікаційній роботі було використано наступні методи: літературний аналіз та аналіз відкритих даних, експериментальне моделювання, практичний експеримент, аудит безпеки, тестування на проникнення, апробація результатів.

Практичне значення роботи полягає у наступному:

– спроектована підсистема моніторингу SOC надає розширені можливості виявлення та реагування на інциденти безпеки в cloud-native середовищах, при цьому базуючись на безкоштовних відкритих рішеннях;

– розроблені пропозиції використання спроектованої підсистеми моніторингу у роботі аналітиків безпеки SOC в компаніях та організаціях від малих до середніх масштабів, які розробляють хмарні додатки та працюють в cloud-native середовищах.

Апробація результатів:

– 1. Єфіменко А.А., Гончаров М.В. The study of the possibilities of using SOC based on free and open source software // Наукова стаття NGSec 2022: International Conference on Next Generation Cybersecurity Systems and Applications, 26-27 квітня 2023 р. — Київ: Національний авіаційний університет, 2023.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ

1.1 Обґрунтування актуальності теми роботи

В сучасному цифровому ландшафті, обґрунтування актуальності теми роботи вимагає розгляду ключового концепту — cloud-native технологій. Cloud-native підходить для розробки та експлуатації програмного забезпечення, яке максимально використовує переваги хмарних сервісів та сприяє швидкій та ефективній роботі додатків. Цей підхід орієнтований на контейнеризацію, оркестрацію та автоматизацію процесів розгортання та масштабування, забезпечуючи гнучкість та реагування на зміни в обсязі навантаження.

Зростання популярності cloud-native технологій обумовлене необхідністю підприємств адаптуватися до швидко змінюючогося середовища бізнесу та технологій. Організації переходять до хмарних сервісів та контейнеризації для поліпшення ефективності, прискорення виходу на ринок та зниження витрат на IT. Забезпечення безпеки в cloud-native середовищах стає пріоритетним завданням, оскільки традиційні підходи до безпеки стають неефективними в умовах зростаючої комплексності та динамічності інфраструктури.

Із зростанням популярності cloud-native технологій, загрози для безпеки інфраструктури також збільшуються. Кіберзлочинці швидко адаптуються до новітніх технологій, використовуючи різноманітні вразливості та атаки. Розробка системи моніторингу та реагування на інциденти в cloud-native середовищах стає необхідністю для виявлення та запобігання сучасним кіберзагрозам.

Згідно зі звітом Snyk, платформи безпеки для розробників для захисту користувацького коду, залежностей з відкритим вихідним кодом, контейнерів та хмарної інфраструктури, станом на 2023 рік [12] все частіше підіймається питання безпеки в cloud-native. Подана статистика (рис. 1.1) демонструє що, близько 99% всіх опитаних компаній визнають безпеку важливим аспектом cloud-native середовищ.

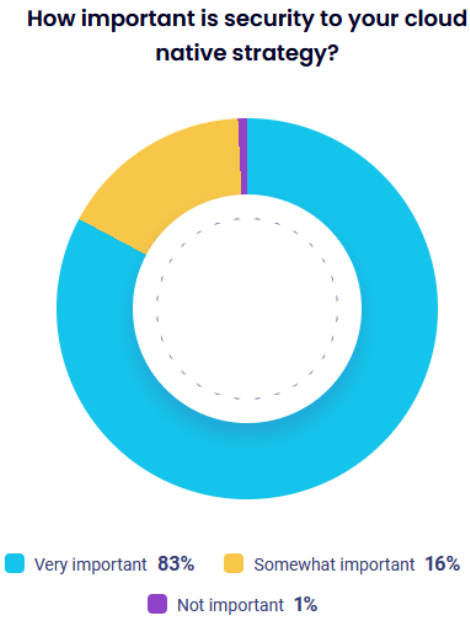


Рисунок 1.1 – Кругова діаграма, що показує важливість безпеки в cloud-native середовищ

Стрімке впровадження хмарних технологій, безсумнівно, змінило систему безпеки багатьох компаній. Опитування Snyk показало, що після переходу на хмарні технології занепокоєння щодо стану безпеки в організаціях зросло майже в 4 рази (рис. 1.2).

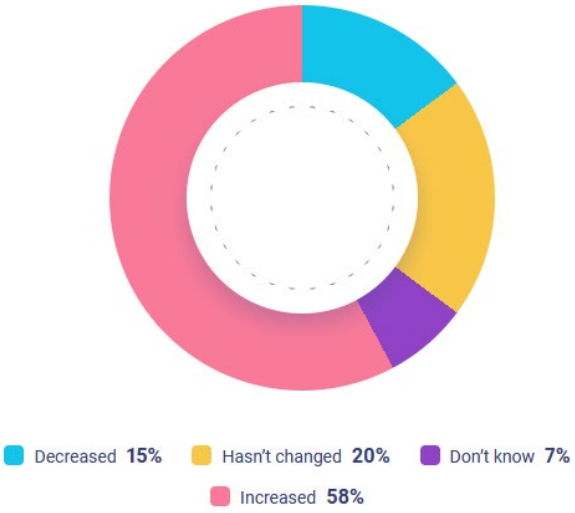


Рисунок 1.2 – Кругова діаграма, що показує занепокоєність питанням безпеки cloud-native середовищ

CrowdStrike, один з глобальних лідерів в питаннях кібербезпеки, в своєму звіті за 2023 рік [13] зазначила, що кількість кібератак з використанням хмарних

технологій стрімко зросла з 2021 по 2022 рік. Кількість спостережуваних випадків використання хмарних технологій зросла на 95%, а кількість випадків за участю зловмисників, націлених на хмарні середовища, зросла майже втричі, тим самим склавши 288% у порівнянні з попереднім роком.

Також серед 2 з 5 необхідних кроків (рис. 3) для захисту cloud-native середовищ, CrowdStrike були виділені кроки по розширенню моніторингу таких середовищ та реакції на події та інциденти безпеки в режимі реального часу.

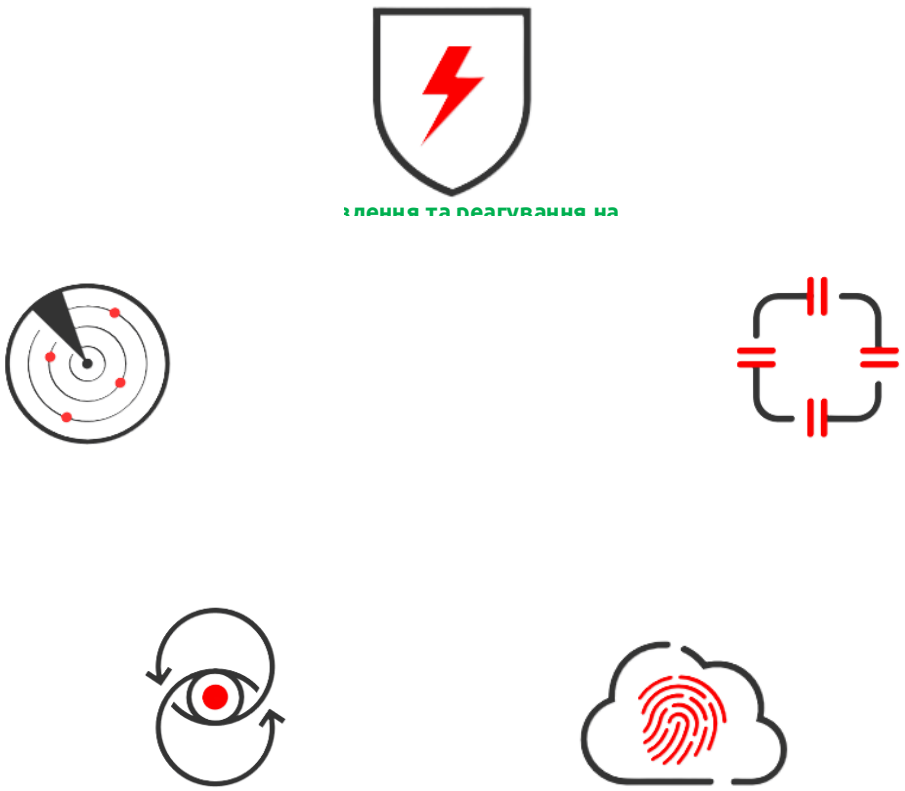


Рисунок 1.3 – 5 кроків до захисту Cloud-Native середовищ

Саме ці кроки і забезпечує підсистема моніторингу подій безпеки SOC, однак наразі далеко не всі IT-компаній можуть собі її дозволити. Навіть в сучасну еру постійних кіберзагроз роль SOC в деяких компаніях виконує не кваліфікований для цього персонал, з відсутніми належними інструментами та технологіями і з не існуючими алгоритмами процесів виявлення та реагування на інциденти безпеки. В більшості випадків така ситуація обумовлена браком коштів. Системи моніторингу

подій безпеки, які наразі існують на ринку, мають складне ліцензіювання та вимагають щорічних значних витрат, які не можуть собі дозволити компанії малих та середніх масштабів. Але це не заводить ситуацію в глухий кут. Ринок безкоштовних відкритих рішень достатньо розвинутий та пропонує рішення, здатні задовільнити потреби таких компаній.

Проектування та побудова такого SOC дійсно займе чимало часу та людських ресурсів, але натомість відкриє фінансовий простір для найму кваліфікованих аналітиків безпеки та фахівців здатних вибудувати необхідні процеси виявлення та реагування та підтримувати їх у належному стані.

1.2 Організація безпеки cloud-native

Проектування та побудова будь-якої стратегії безпеки в Cloud-native середовищах базується на основі концепції безпеки Defense in Depth. Defense in Depth – це концепція, що використовується в інформаційній безпеці, при якій в IT-системі розміщується декілька рівнів контролю безпеки. Її мета – забезпечити надмірність на випадок відмови засобів контролю безпеки або використання вразливості, яка може охоплювати аспекти кадрової, процедурної, технічної та фізичної безпеки протягом усього життєвого циклу системи.

Інтерпретація цієї концепції в cloud-native середовищах описується через 4-шарову модель 4 C's (рис. 1.4), яку буде розглянуто пошарово. [14]

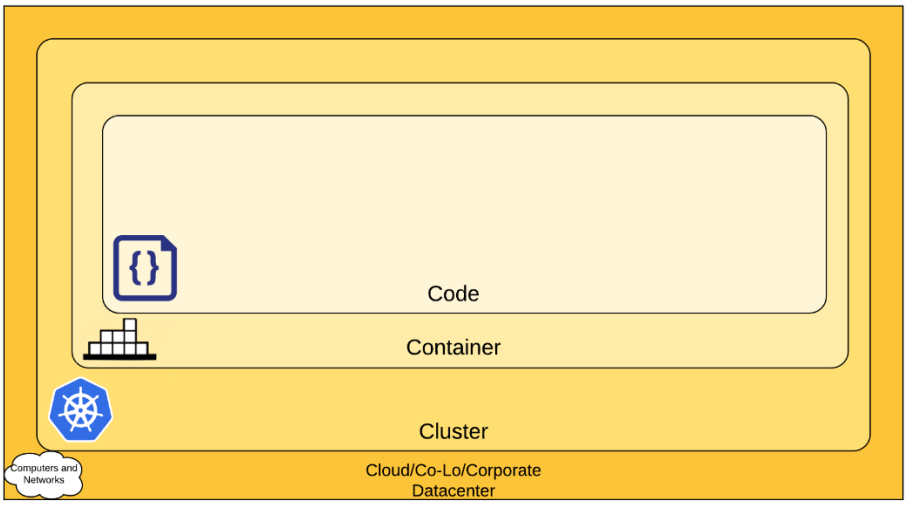


Рисунок 1.4 – Графічне представлення концепції 4 C’s

1.2.1 Cloud (Хмара)

Хмарний рівень складається з інфраструктури, яка забезпечує роботу хмарних ресурсів. По великому рахунку саме цей виступає фундаментом cloud-native (рис. 1.2.1). Коли сервер налаштовується у постачальника хмарних послуг (CSP), провайдер відповідає за більшу частину безпеки інфраструктури. Однак замовник несе відповідальність за конфігурацію послуг, захист даних і нагляд за безпекою.

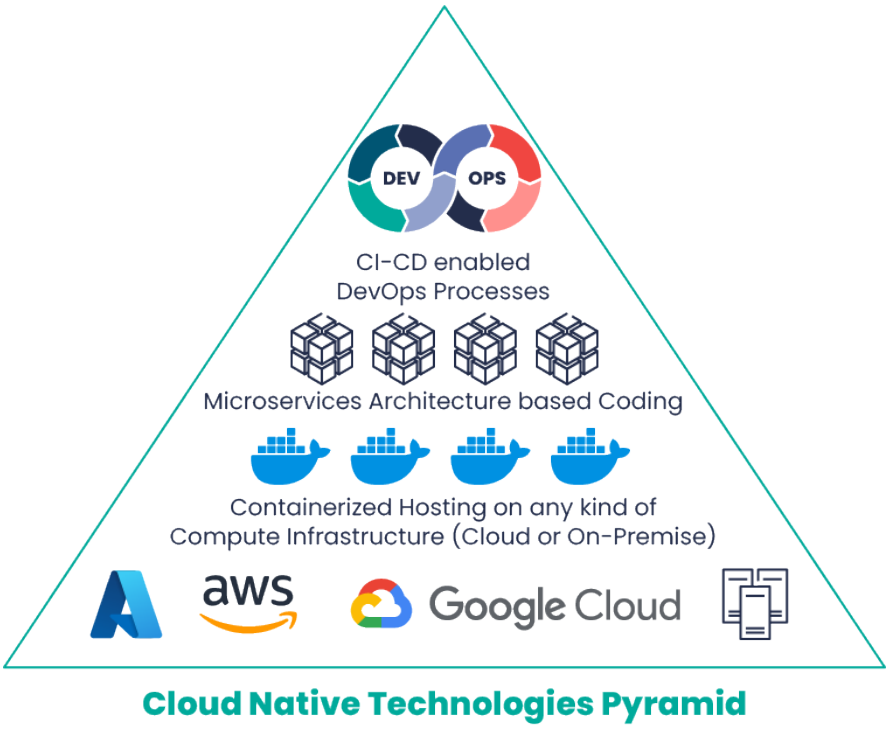


Рисунок 1.2.1 – Піраміда Cloud-native

Типові проблеми безпеки, що впливають на хмарний рівень, включають неправильні конфігурації та автоматизовані атаки. Зловмисники можуть використовувати неправильні конфігурації, що виникають через помилки або недбалість, наприклад, незмінні налаштування за замовчуванням або слабкий захист доступу до консолі адміністрування. Зловмисники також можуть використовувати автоматизацію для пошуку вразливостей і швидкого запуску атак.

Організації можуть уникнути цих проблем, дотримуючись рекомендацій свого хмарного провайдера та регулярно проводячи аудит, щоб переконатися, що

все налаштовано належним чином, перш ніж розгортати хмарні сервіси у виробничих умовах та виводити їх в Інтернет.

Щодо загальних рекомендацій по підвищенню безпеки кластеру Kubernetes на рівні хмари можна виділити наступні:

Таблиця 1.2.1 – Рекомендації для захисту кластеру в хмарі

Сфера занепокоєння	Рекомендація
Мережевий доступ до API-сервера (Площина управління)	Весь доступ до площини керування Kubernetes із мережі Інтернет не дозволений, контролюється мережевими списками контролю доступу, обмеженими переліками IP-адрес, необхідних для адміністрування кластера.
Мережевий доступ до вузлів	Вузли слід налаштувати так, щоб вони приймали з'єднання (через списки контролю доступу до мережі) лише з площини управління на вказаних портах, а також приймали з'єднання для сервісів у Kubernetes типу NodePort та LoadBalancer. Якщо можливо, ці вузли не слід повністю виставляти на загальний огляд в Інтернеті.
Доступ Kubernetes до API CSP	Кожен хмарний провайдер повинен надавати різний набір дозволів на площину управління та вузли Kubernetes. Найкраще надавати кластеру доступ до хмарного провайдера за принципом найменших привілеїв для ресурсів, які йому потрібно адмініструвати.
Доступ до etcd	Доступ до etcd (сховища даних Kubernetes) повинен бути обмежений лише площиною управління. Залежно від вашої конфігурації, слід спробувати використовувати etcd через TLS.
Шифрування etcd	Коли це можливо, рекомендується шифрувати всі сховища у стані спокою, а оскільки etcd зберігає стан всього кластера (включно з Secrets), його диск особливо слід шифрувати у стані спокою.

1.2.2 Cluster (Кластер)

Шар кластера складається з компонентів Kubernetes, що утворюють робочі вузли та площину управління (рис. 1.2.2). Саме на цьому рівні захищаються робочі навантаження Kubernetes. Компоненти Kubernetes використовують зашифрований зв'язок, вимагаючи сертифікати TLS для автентифікації один з одним.

Найважливішим компонентом для захисту є kube-api-сервер, який є основним інтерфейсом Kubernetes. За замовчуванням доступ до цього сервера можна

отримати лише через HTTPS, хоча також може використовуватись сторонній постачальник ідентифікаційних даних для додаткового захисту за допомогою автентифікації. Зазвичай організації використовують спеціальні правила контролю доступу на основі ролей (RBAC) для авторизації на API-сервері, тому можливо адмініструвати кластер і його робочі навантаження, не потребуючи доступу до Secure Shell.

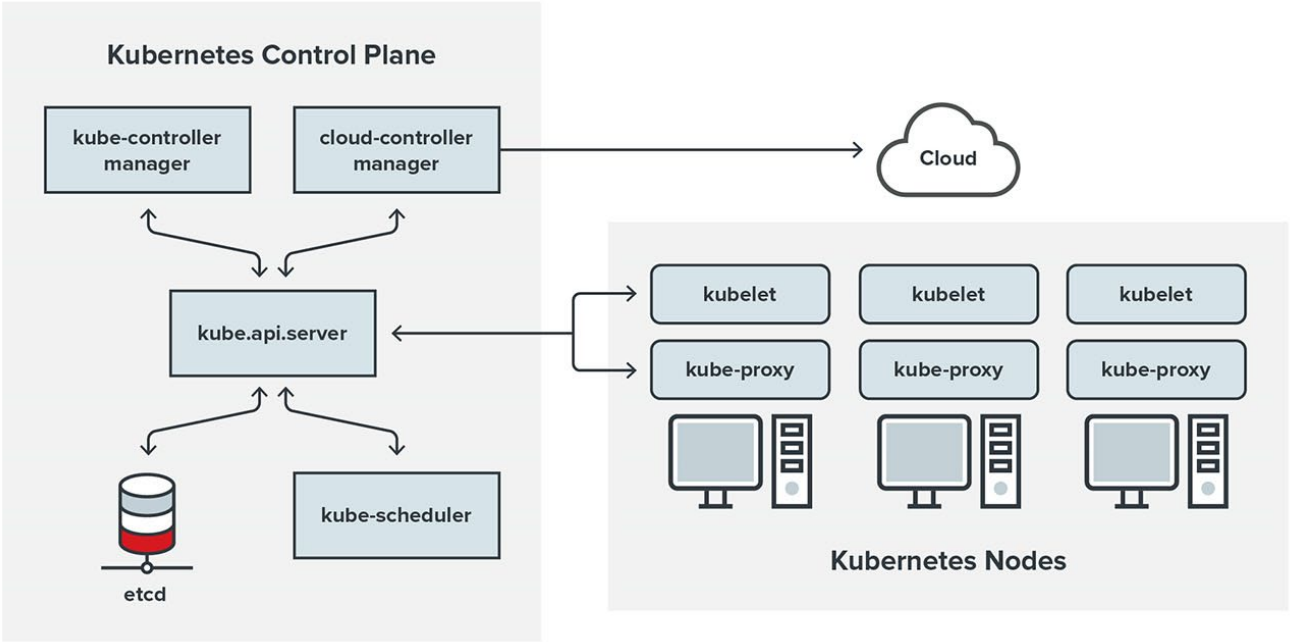


Рисунок 1.2.2 – Архітектура кластеру Kubernetes

Таблиця 1.2.2 – Рекомендації для захисту робочого навантаження в кластері

1.2.3 Container (Контейнер)

Шар контейнерів складається з образів контейнерів, які можуть містити вразливості, що можуть бути виявлені під час сканування. Організації зазвичай не звертають уваги на такі проблеми, як безпека образів, використання невідомих джерел та слабкі конфігурації привілеїв. Важливо регулярно оновлювати контейнери, щоб мінімізувати вразливість через відомі вразливості. Також слід сканувати та перевіряти всі програми, що працюють у контейнерах.

Будь-який образ повинен бути створений відомим джерелом або отриманий з надійного реєстру. Інструмент підписання образів, такий як Docker Content Trust

(DCT), може допомогти переконатися, що вміст контейнера надходить з надійних джерел.

Контейнери не повинні запускатись з привілейованим обліковим записом користувача, щоб вони не мали привілейованого доступу до хост-машини.

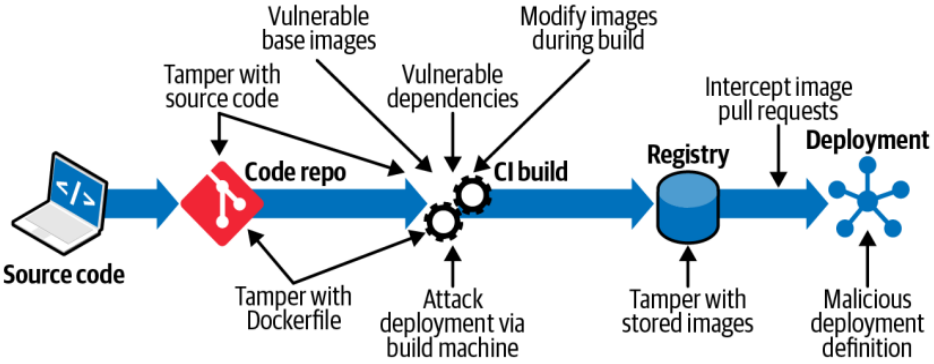


Рисунок 1.2.3 – Поверхня атаки на образ контейнеру

Таблиця 1.2.2 – Рекомендації для захисту контейнеру

Сфера занепокоєння	Рекомендація
Сканування контейнеру на вразливості та застарілі залежності	На етапі створення образу слід просканувати контейнери на наявність відомих вразливостей та застарілих залежностей пакунків.
Цифрове підпис образів контейнерів	Підпис образів контейнерів, для підтримки системи довіри до вмісту контейнерів.
Виключення привілейованих користувачів	Створення всередині контейнерів користувачів з найменшим рівнем привілеїв операційної системи, необхідним для виконання завдань контейнера.
Використання рушія контейнеризації з більшою ізольованістю	Вибір рушія контейнеризації, що забезпечує кращу ізольованість ресурсів одне від одного

1.2.4 Code (Код)

Шар коду, також відомий як прикладний рівень, забезпечує найвищий рівень контролю безпеки. На цьому рівні можна обмежити відкриті кінцеві точки, порти та служби для управління ризиками безпеки. Зв'язок між внутрішніми та зовнішніми службами має бути захищений за допомогою шифрування TLS.

Типові проблеми безпеки на рівні коду включають незахищений код, недостатню оцінку ризиків і вразливості в залежному сторонньому програмному забезпеченні. Для перевірки коду на ці проблеми, слід використовувати інструмент статичного аналізу коду (SCA) для швидкого виявлення незахищеного коду та забезпечення безпечного кодування. Регулярне сканування і тестування додатків забезпечить стійкість до таких атак, як міжсайтова підробка запитів (CSRF) і міжсайтовий скриптинг (XSS).

Більшість хмарних додатків складаються зі сторонніх бібліотек або залежностей, які часто залишаються поза увагою статичного аналізу. Для виявлення вразливих залежностей можна скористатися інструментом аналізу складу програмного забезпечення, таким як OWASP Dependency-Check.

Питання безпеки коду зазвичай інтегрується в інструменти CI/CD (безперервної інтеграції та безперервного розгортання) та вбудовується в наявні pipeline з розгортання додатку, що і здійснюють всі необхідні перевірки та в результаті генерують відповідний звіт (рис. 1.2.4)

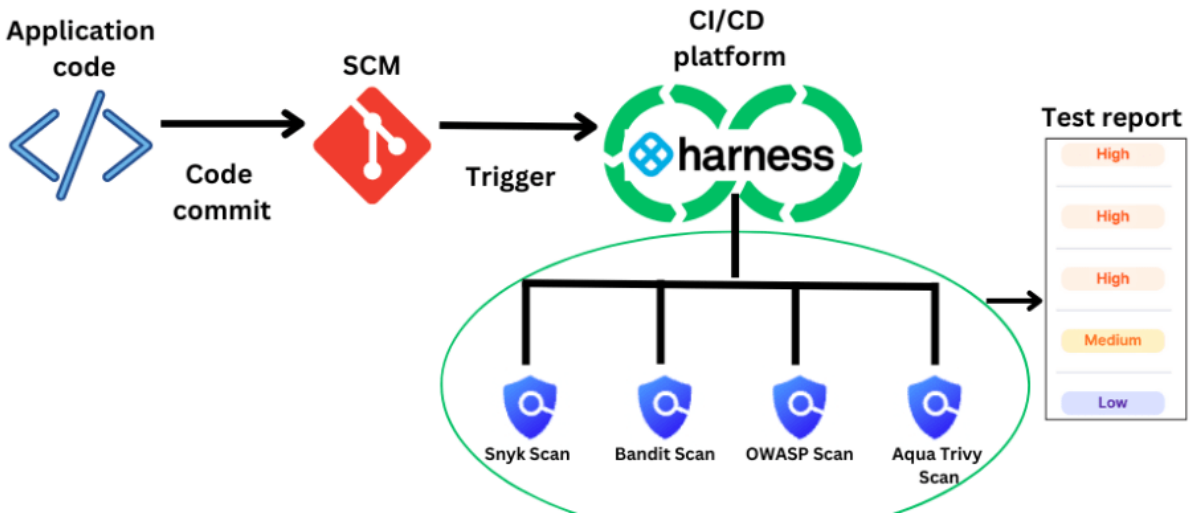


Рисунок 1.2.4 – Приклад інтеграції автоматизованого процесу перевірки коду в CI/CD

1.3 Роль, місце та функції сучасного SOC в cloud-native середовищах

З розвитком технологій завжди з’являються нові способи атаки на мережі, компрометації користувачів, їх робочих станцій та даних, що на них знаходяться.

Природно, що зі сторони фахівців кібербезпеки так само розвиваються і методи та технології їх захисту та запобігання зловмисній активності. Говорячи про SOC сьогодення, можна сказати, що він переживає свою шосту ітерацію розвитку [15].

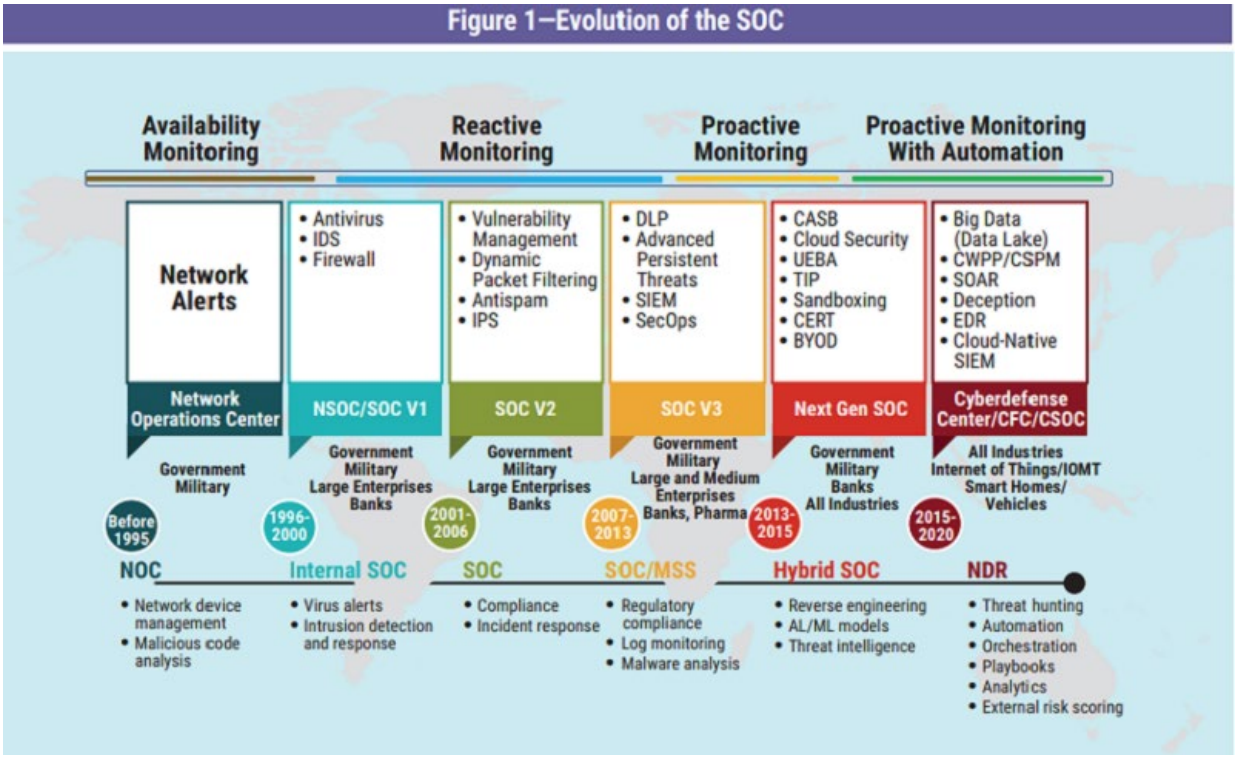


Рисунок 1.2.5 – Еволюція SOC

Сучасний SOC якісно вирізняється на фоні попередніх ітерацій наступними особливостями:

- здатністю обробляти великі об’єм структурованих та неструктурованих даних (Big Data / Data Lake);
- підтримкою рішень CWPP для розширення видимості хмарних ресурсів, в першу чергу для захисту хмарних робочих навантажень.
- підтримкою платформ CSPM, що впроваджують безперервні, автоматизовані процеси безпеки та відповідності в першу чергу для захисту інфраструктури, де розгортаються робочі навантаження.
- функціями з оркестрації та автоматизації робочих процесів (SOAR);
- використанням інфраструктури «пасток», які виступатимуть пріоритетною ціллю зловмисника та нададуть змогу проактивно реагувати на кібератаки.
- додатковою видимістю загроз на кінцевих точках за рахунок використання рішень EDR;

– наявністю SIEM-системою з підтримкою моніторингу середовищ cloud-native.

Наявність в SOC інструментів по роботі з великими об’ємами даними, EDR / XDR та SOAR є далеко не новиною для сучасних команд безпеки. На противагу їм технології моніторингу та реагування на інциденти cloud-native є відносно новими та не чітко вписаними в процеси функціонування SOC багатьох компаній.

В ідеалі сучасний SOC повинен включати різнопланових фахівців безпеки, в тому числі інженерів DevSecOps, що займаються безпекою в cloud-native. Однак більшість SOC працюють в звичайному режимі, але з використанням інструментів DevSecOps. В разі виникнення подій та інцидентів безпеки в середовищі cloud-native, такі SOC тісно співпрацюють з командою розробників та DevOps над реагуванням та коригуючими мірами.

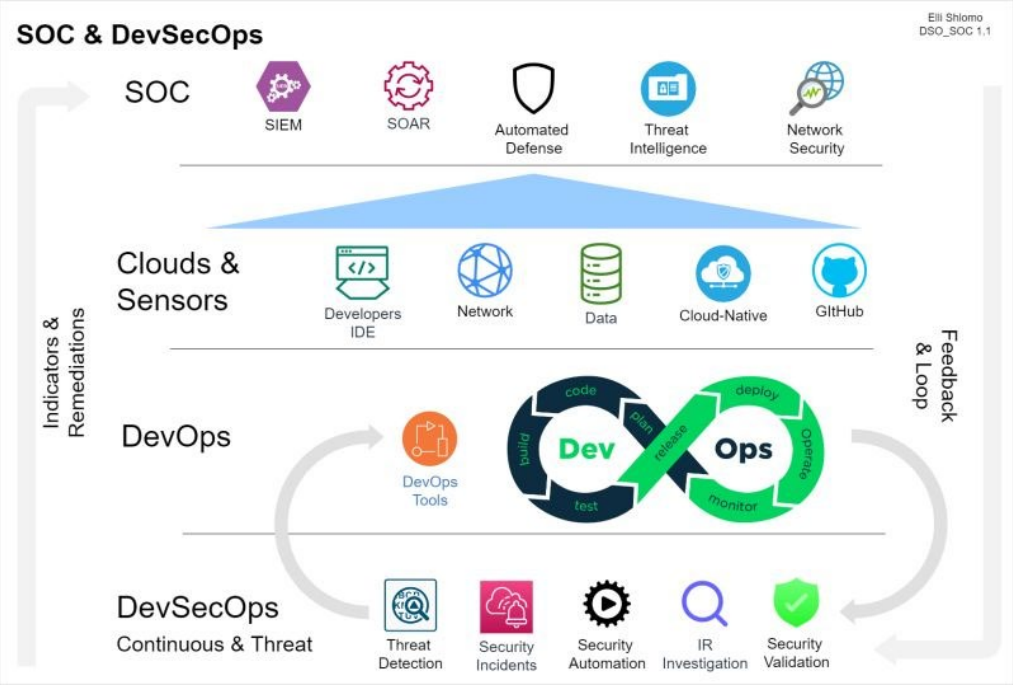


Рисунок 1.3.2 – Місце SOC та DevSecOps в cloud-native

Саме з розрахунком на такі команди SOC, в наступному підрозділі буде проаналізовано ринок безкоштовних відкритих рішень та буде обрано інструменти, що дозволять належним чином виявляти та реагувати на інциденти в cloud-native.

1.4 Аналіз сучасного ринку FOSS та обґрунтування вибору SOC рішень

Слід відразу зазначити, що безкоштовні відкриті рішення розробляються командами на обмежені ресурси, а іноді навіть на волонтерських засадах, тож далеко

не всі рішення в сфері кібербезпеки можуть конкурувати з масштабними та гарно зарекомендувавшими себе рішеннями великих компаній. Однак це не означає, що серед них немає насправді гарних та інноваційних продуктів. Враховуючи специфіку обраного середовища проведено аналіз SIEM, SOAR та CWPP систем, здатних до моніторингу та взаємодії з ним.

1.4.1 SIEM

Управління інформацією та подіями безпеки є фундаментальною системою в сучасній кібербезпеці. Інші інструменти безпеки представляють собою інформаційні потоки, які SIEM може обробляти і витягувати з них цінну інформацію. Не всі SIEM мають однакові можливості; вибір SIEM, який відповідає потребам вашої організації, може становити різницю між запобіганням і пропуском критичного інциденту безпеки.

Організації можуть використовувати інструменти SIEM з відкритим вихідним кодом, щоб зменшити витрати на ліцензування програмного забезпечення та оцінити певні можливості перед тим, як збільшити інвестиції в продукт.

Нижче наведений аналіз 5 найвище оцінених сучасних FOSS SIEM-рішень:

Таблиця 4.1 – Рейтинг FOSS SIEM-рішень

SIEM	Опції з розгортання	Головні риси	Обмеження
ELK Stack Колекція з трьох продуктів з відкритим вихідним кодом: Elasticsearch, Logstash та Kibana. Ці три інструменти можна використовувати для візуалізації та аналізу подій	Віртуальні середовища, фізичне обладнання, приватна хмара, приватна зона в публічній хмарі або публічна хмара (наприклад, Google, Azure, AWS)	- Журналювання та аналіз журналів - Обробка, фільтрація, кореляція та покращення даних журналу, які він збирає - Індексування та зберігання даних часових рядів	- Аналіз журналів загального призначення - Не розроблений як SIEM-система

Продовження таблиці 4.1 – Рейтинг FOSS SIEM-рішень

Apache Metron Відносно новий гравець в індустрії. Фреймворк безпеки, який об'єднує кілька проектів з відкритим вихідним кодом в єдину платформу.	Наразі працює з трьома сховищами даних: HBase, HDFS та Elastic Search	- Фреймворк, що підключається, для додавання нових кастомних парсерів для нових джерел даних - Зберігає збагачені телеметричні дані - Алгоритми виявлення аномалій та машинного навчання, які можна застосовувати в режимі реального часу	- Можна встановити лише на обмежену кількість середовищ та операційних систем - Інтерфейс знаходиться на ранній стадії розробки і не підтримує автентифікацію
SIEMonster Заснований на технології з відкритим вихідним кодом. Доступний безкоштовно та як платне рішення (преміум-версія та багатокористувацька версія MSSP).	У хмарі з використанням контейнерів Docker, а також на віртуальних машинах і "голому металі" (Mac, Ubuntu, CentOS і Debian).	- Фреймворк для обробки розвідданих про загрози - ELK Stack використовується для зберігання, збору, обробки та візуалізації	- Безкоштовна версія не пропонує аналітику поведінки користувачів, машинне навчання, функції HoneyNet та Threat Kill з повної версії продукту - Відсутня онлайн-документація
Prelude Об'єднує різні інші інструменти з відкритим вихідним кодом. Це версія з відкритим вихідним кодом однойменного комерційного інструменту.	Linux, OpenBSD, FreeBSD, NetBSD, Sun/Solaris, MacOSX, Tru64 та інші системи на базі UNIX.	- Кореляція, фільтрація та оповіщення - Можливості аналізу та візуалізації	- Призначений для досліджень, оцінки та тестування в дуже невеликих середовищах - За словами розробників, продуктивність Prelude з відкритим вихідним кодом значно нижча, ніж у комерційного продукту Prelude SIEM

Закінчення таблиці 4.1 – Рейтинг FOSS SIEM-рішень

OSSIM Платформа SIEM, що включає збір, нормалізацію та кореляцію подій	Локальні фізичні та віртуальні середовища	- Виявлення активів - Оцінка вразливостей - Кореляція подій SIEM - Виявлення вторгнень -Поведінковий моніторинг	- Масштабні проблеми з продуктивністю - Дуже обмежене управління журналами - Може бути розгорнута лише для одного сервера - Немає інтеграції з рішеннями UEBA - Обмежений моніторинг додатків та баз даних - Обмежена база даних графіків, що дозволяє лише частково аналізувати власних користувачів - Відсутня підтримка та інтеграція з інструментами DAM, CASB, DAP та DLP

Враховуючи проведений порівняльний аналіз, а також обсяг спільноти та наявність документації по рішенню, SIEM-системою для підсистеми моніторингу подій безпеки було обрано ELK Stack [16]. ELK Stack спочатку не вважалась повноцінною SIEM-системою, так як не включала в себе функцій безпеки, однак сучасні версії системи містять окремий модуль «Security», що дозволяє писати власні правила виявлення, виявляти та обробляти інциденти безпеки. ELK Stack також була доволі високо оцінено в цьогорічному Gartner Magic Quadrant SIEM-систем (рис. 1.4.1). ELK Stack буде виступати основною точкою збору та кореляції інформації з мережевих пристроїв, робочих станцій та безпосередньо самого cloud-native середовища, за рахунок своєї широкої бази інтеграцій зі сторонніми продуктами.

Рисунок 1.4.1 – Gartner Magic Quadrant SIEM 2023

1.4.2 SOAR

Система оркестрування, автоматизації та реагування на загрози безпеці (SOAR) – це набір сумісних програм, які дозволяють організації збирати дані про загрози безпеці та реагувати на події безпеки з мінімальною участю людини або взагалі без неї. Метою використання платформи SOAR є підвищення ефективності операцій з фізичної та цифрової безпеки:

1. TheHive Project. TheHive Project – це масштабна платформа реагування на інциденти безпеки, тісно інтегрована з MISP (платформою обміну інформацією про шкідливе програмне забезпечення), покликана полегшити життя SOC, CSIRT, CERT і будь-яким фахівцям з інформаційної безпеки, які мають справу з інцидентами безпеки, що потребують швидкого розслідування та реагування.[17]

2. Shuffle – це SOAR з відкритим вихідним кодом. Він має на меті надати всі можливості, необхідні для передачі даних по всьому підприємству, за допомогою додатків, що працюють за принципом "підключи і працюй", роблячи автоматизацію доступною для кожного.

3. Walkoff – гнучка, проста у використанні система автоматизації, що дозволяє користувачам інтегрувати свої можливості та пристрої, щоб позбутися повторюваних, нудних завдань, які сповільнюють роботу.

Серед описаних вище рішень TheHive Project є найбільш повною версією SOAR, так окрім можливостей автоматизації та оркестрації також містить платформу для керування інцидентами. Саме рішення розбито на 2 частини, які тісно взаємодіють між собою.

Перше рішення, TheHive, і є тією платформу для керування інцидентами. В ньому аналітики здатні брати в роботу так звані алерти, або спрацювання систем безпеки. Якщо алерт дійсно являє собою серйозну загрозу безпеці, та не є хибним спрацюванням, аналітик здатен підвищити алерт до статусу кейсу (інциденту) та провести необхідно розслідування та документацію його кроків.

Друга ж частина, Cortex, допомагає аналітику автоматизувати ті чи інші задачі при розслідуванні та реагуванні на інцидент. Наприклад аналітики можуть доповнювати наявні індикатори компрометації (IP адреси, хеші файлів, URL посилання) виділені з алерту за допомогою аналізаторів (analyzers) Cortex. Після розслідування, аналітики матимуть змогу відреагувати на тей чи інший інцидент за допомогою відповідачів (responders) Cortex, наприклад заблокувавши IP-адресу зловмисника на мережевому екрані.

Для автоматизації запланованих процесів, таких як сканування мережі на вразливості, потрібно використовувати стороні інструменти, як вже згаданий вище Shuffle. Цю інтеграцію не розглянуто в рамках даної кваліфікаційної роботи.

1.5 Постановка завдання на розробку

Метою кваліфікаційної роботи є проектування та побудова підсистеми моніторингу подій безпеки SOC для cloud-native середовищ на базі безкоштовних відкрити рішень. Проект являтиме собою віртуальне програмно-апаратне середовище в якому будуть зсимульована мережа розробки хмарних застосунків, а також мережа SOC на базі обраних рішень.

Побудова підсистеми складатиметься з наступних кроків:

– розробка проекту віртуальної комп’ютерної мережі засобами мережевого емулятора GNS3;

– проектування фізичної та логічної топології мережі;

– налаштування мережевого обладнання та перевірка зв’язності в мережі.

– моделювання загроз cloud-native середовища для подальшого налаштування механізмів безпеки підсистеми моніторингу;

– розробка та налаштування підсистеми моніторингу подій безпеки на базі SIEM-системи ELK Stack та SOAR-системи TheHive + Cortex;

– розробка та налаштування кластеру, що імітуватиме production середовище.

– заведення журналів з мережевого обладнання та інструментів безпеки та моніторингу cloud-native середовища в підсистему моніторингу подій безпеки.

– реалізація декількох сценаріїв атаки на середовище;

– оцінка ефективності та якості виявлення атак застосованими правилами, а також формулювання рекомендацій що до розширення можливостей з виявлення та реагування SOC у разі тих чи інших інцидентів в cloud-native середовищах.

Висновки до 1-го розділу:

Проаналізовано актуальні звіти та статистику, сформовані відомими компаніями в сфері кібербезпеки про актуальність питання безпеки cloud-native, а саме наявності підсистеми моніторингу, яка дозволить підтримувати цю безпеку на належному рівні.

В результаті аналізу специфіки cloud-native середовищ було сформоване уявлення про принцип роботи, взаємозв’язки та сильні та слабкі сторони їх складових.

Визначено особливості сучасного SOC, його роль у роботі та підтримці процесів cloud-native.

Проведено порівняльну характеристику безкоштовних відкритих рішень, на яких базуватиметься проектована підсистема моніторингу подій безпеки.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА НАЛАШТУВАННЯ ПІДСИСТЕМИ МОНІТОРИНГУ ПОДІЙ БЕЗПЕКИ SOC

2.1 Проектування та налаштування мережевої інфраструктури

Корпоративна мережа – це логічно відокремлена група комп’ютерів, маршрутизаторів та інших частин IT-інфраструктури, які функціонують поза традиційними межами Інтернету. Її часто називають Інтранет. Термін Інтранет описує мережу, яка, на відміну від Інтернету, призначена для доступу лише для певної групи людей.

Основною метою створення корпоративних мереж є забезпечення безпеки та надійності інформаційної системи управління, системи обробки даних, передачі даних та зв'язку – у співробітників може не виникати потреби використовувати програмне забезпечення з обмеженими даними, коли вони знаходяться за межами робочого місця. Крім того, розміщення програм, які стосуються лише членів компанії, виключно всередині корпоративних мереж, може бути набагато менш складним. Корпоративні мережі використовують різні типи пристроїв LAN та WAN, а також широко доступні протоколи та стандарти зв’язку для забезпечення таких функцій.

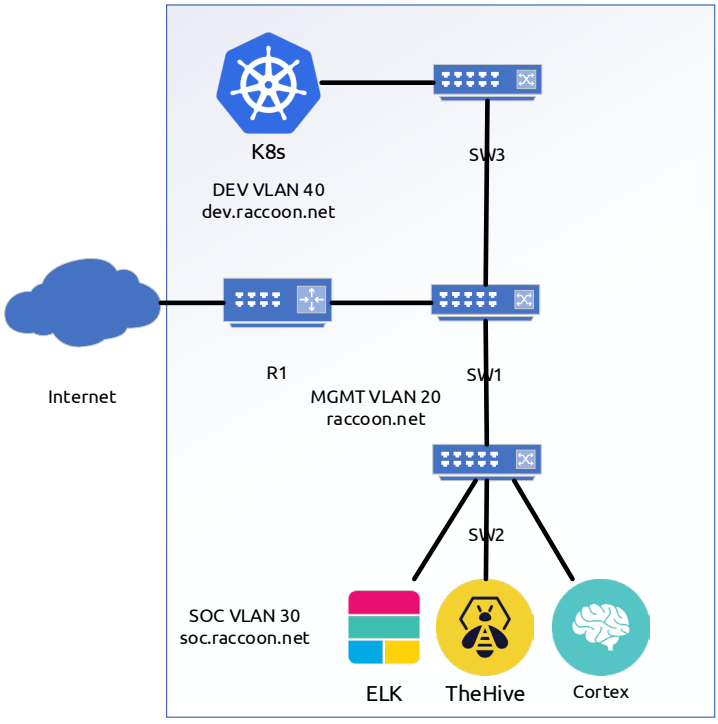


Рисунок 2.1 – Діаграма топології корпоративної мережної інфраструктури

Платформою розгортання SOC обрані сервери під ОС Ubuntu Server. Кожному з представлених рішень виділено окремий сервер.

Для емуляції мережної інфраструктури будуть використовуватися домашній маршрутизатор (Mercusys AC12G), ПК (Windows 11 x64 Pro, 8 ГБ ОЗП). Cloud-native середовище представлено локальним Kuberentes кластером на сервері під ОС Ubuntu Server. Розгортання кластеру виконано за допомогою minikube під рушієм контейнеризації containerd.

Таблиця 2.1.1 – Параметри інтерфейсів пристроїв

Пристрій	Інтерфейс	Підключення до пристрою	Підключення до інтерфейсу
Маршрутизатор R1 (adventerprisek9-ms.155-2.T)	E0/0	Internet	br0
	E0/1	Комутатор SW1	E0/0
Комутатор SW1 (adventerprisek9-15.2d)	E0/0	Маршрутизатор R1	E0/1
	E0/1	Комутатор SW1	E0/0
	E0/2	Комутатор SW2	E0/0
Комутатор SW2 (adventerprisek9-15.2d)	E0/0	Комутатор SW1	E0/1
	E0/1	Сервер ELK	E0
	E0/2	Сервер TheHive	E0
	E0/3	Сервер Cortex	E0
Комутатор SW3 (adventerprisek9-15.2d)	E0/0	Комутатор SW1	E0/2
	E0/1	Сервер K8s	E0
Сервер ELK	E0	Комутатор SW2	E0/1
Сервер TheHive	E0		E0/2
Сервер Cortex	E0		E0/3
Сервер K8s	E0	Комутатор SW3	E0/1

Таблиця 2.1.2 – Параметри IP-адресації мережі

Мережа/ Пристрій	Інтерфейс/Мережний адаптер/Шлюз	IP-адреса	Маска	/xx
VLAN 20 raccoon.net	–	192.168.44.0	255.255.255.192	/26
VLAN 30 soc.raccoon.net	–	192.168.44.64	255.255.255.192	/26
VLAN 30 dev.raccoon.net	–	192.168.44.128	255.255.255.192	/26
Маршрутизатор R1	Інтерфейс E0/0	DHCP		
	Інтерфейс E0/1.20	192.168.44.62	255.255.255.192	/26
	Інтерфейс E0/1.30	192.168.44.126	255.255.255.192	/26

Закінчення таблиці 2.1.2 – Параметри IP-адресації мережі

	Інтерфейс E0/1.40	192.168.44.190	255.255.255.192	/26
Комутатор SW1	Інтерфейс Vlan 20	192.168.44.11	255.255.255.192	/26
	Шлюз за замовчуванням	192.168.44.62	–	–
Комутатор SW2	Інтерфейс Vlan 20	192.168.44.12	255.255.255.192	/26
	Шлюз за замовчуванням	192.168.44.62	–	–
Комутатор SW3	Інтерфейс Vlan 20	192.168.44.13	255.255.255.192	/26
	Шлюз за замовчуванням	192.168.44.62	–	–
Сервер ELK	Мережний адаптер	192.168.44.65	255.255.255.192	/26
	Шлюз за замовчуванням	192.168.44.126	–	–
Сервер TheHive	Мережний адаптер	192.168.44.66	255.255.255.192	/26
	Шлюз за замовчуванням	192.168.4.126	–	–
Сервер Cortex	Мережний адаптер	192.168.44.67	255.255.255.192	/26
	Шлюз за замовчуванням	192.168.4.126	–	–
Сервер K8s	Мережний адаптер	192.168.44.129	255.255.255.192	/26
	Шлюз за замовчуванням	192.168.4.190	–	–

Сценарії налаштування параметрів IP-адресації, статичної маршрутизації, протоколу віддаленого керування SSH та протоколу мережевого часу NTP на маршрутизаторі R1:

R1:

```
Router>en
Router#conf t
Router(config)#hostname R1
R1(config)#username admin privilege 15 algorithm-type sha256 12#$qwER
R1(config)#int e0/0
R1(config-if)#ip address dhcp
R1(config-if)#no shut
R1(config-if)#int e0/1
R1(config-if)#no shut
R1(config-if)#int e0/1.20
R1(config-if)#encapsulation dot1Q 20
R1(config-if)#ip address 192.168.44.62 255.255.255.192
```

```
R1(config-if)#no shut
R1(config-if)#int e0/1.30
R1(config-if)#encapsulation dot1Q 30
R1(config-if)#ip address 192.168.44.126 255.255.255.192
R1(config-if)#no shut
R1(config-if)#int e0/1.40
R1(config-if)#encapsulation dot1Q 40
R1(config-if)#ip address 192.168.44.190 255.255.255.192
R1(config-if)#no shut
R1(config-if)#exit
R1(config)#ip route 0.0.0.0 0.0.0.0 192.168.1.1
R1(config)#ip domain-name raccoon.net
R1(config)#ip domain-list raccoon.net
R1(config)#ip domain-list soc.raccoon.net
R1(config)#ip domain-list dev.raccoon.net
R1(config)#ip host r1.raccoon.net 192.168.44.62
R1(config)#ip host sw1.raccoon.net 192.168.44.11
R1(config)#ip host sw2.raccoon.net 192.168.44.12
R1(config)#ip host sw3.raccoon.net 192.168.44.13
R1(config)#ip host elk.soc.raccoon.net 192.168.44.65
R1(config)#ip host thehive.soc.raccoon.net 192.168.44.66
R1(config)#ip host cortex.soc.raccoon.net 192.168.44.67
R1(config)#ip domain lookup
R1(config)#ip dns server
R1(config)#ip name-server 192.168.1.1 8.8.8.8
R1(config)#ip ssh version 2
R1(config)#crypto key generate rsa general-key modulus 2048
R1(config)#line vty 0 4
R1(config-line)#logging synchronous
R1(config-line)#login local
R1(config-line)#exec-timeout 20
R1(config-line)#transport input ssh
R1(config-line)#exit
R1(config)#ntp server 0.ua.pool.ntp.org
R1(config)#clock timezone EET +2
R1(config)#clock summer-time EEST recurring
R1(config)#do write
```

```
R1#show ip int br | i up
Ethernet0/0                192.168.1.103   YES DHCP   up                    up
Ethernet0/1                unassigned      YES NVRAM  up                    up
Ethernet0/1.20             192.168.44.62   YES NVRAM  up                    up
Ethernet0/1.30             192.168.44.126  YES NVRAM  up                    up
Ethernet0/1.40             192.168.44.190  YES NVRAM  up                    up
```

Рисунок 2.1.2 – Результат виконання команди ***show ip interface brief*** на маршрутизаторі **R1**

```
R1#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override

Gateway of last resort is 192.168.1.1 to network 0.0.0.0

S*    0.0.0.0/0 [1/0] via 192.168.1.1
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.1.0/24 is directly connected, Ethernet0/0
L        192.168.1.103/32 is directly connected, Ethernet0/0
      192.168.44.0/24 is variably subnetted, 6 subnets, 2 masks
C        192.168.44.0/26 is directly connected, Ethernet0/1.20
L        192.168.44.62/32 is directly connected, Ethernet0/1.20
C        192.168.44.64/26 is directly connected, Ethernet0/1.30
L        192.168.44.126/32 is directly connected, Ethernet0/1.30
C        192.168.44.128/26 is directly connected, Ethernet0/1.40
L        192.168.44.190/32 is directly connected, Ethernet0/1.40
```

Рисунок 2.1.3 – Результат виконання команди ***show ip route*** на маршрутизаторі **R1**

```
R1#show ip ssh
SSH Enabled - version 2.0
Authentication methods:publickey,keyboard-interactive,password
Authentication Publickey Algorithms:x509v3-ssh-rsa,ssh-rsa
Hostkey Algorithms:x509v3-ssh-rsa,ssh-rsa
Encryption Algorithms:aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc
MAC Algorithms:hmac-sha1,hmac-sha1-96
Authentication timeout: 120 secs; Authentication retries: 3
Minimum expected Diffie Hellman key size : 1024 bits
IOS Keys in SECSH format(ssh-rsa, base64 encoded): R1.raccoon.net
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDB2T/4q4hwkI2QQ9pPb+CCnRQkd228nm5fhfslK3fC
EY2idApU/bOZYAmd/AEIRRcRLjtb6eca3erCax2YKgpCIJqCRL5GxEUVgLwWJJ8/13A2eN3Hhj7nb+7+
Dr/4yoJVeeffL8gAH4gJofysXTxT8n7MI1D031lbZXC5bSpmskyGNLKg/vOSLJTeL8H6Rl/XHjNZKWn2
gvha5iGB72vD4XTEj1N2lqdizmNXC2xHiH2Hmq8wYUYt8iacr+UOWr2LgCxDgCz7bGtHicfFURj/zklA
qC0aj4DUuMDjuAIevL7/GZvq/xJ6yvA2nDy2DaH0jBcsD9t1NE8er0plJN+P
```

Рисунок 2.1.4 – Результат виконання команди ***show ip ssh*** на маршрутизаторі **R1**

```
R1#show ntp asso
R1#show ntp associations

  address         ref clock       st   when   poll reach  delay  offset   disp
 ~127.127.1.1     .LOCL.           4      3     16   377  0.000   0.000  1.204
*~194.54.80.29    78.26.180.80     2     54     64   377  4.988  -2.486  1.967
 * sys.peer, # selected, + candidate, - outlyer, x falseticker, ~ configured
R1#show clock
*22:23:47.657 EET Sat Dec 23 2023
```

Рисунок 2.1.5 – Результат виконання команд ***show ntp associations*** та ***show clock detail*** на маршрутизаторі **R1**

Сценарії налаштування параметрів IP-адресації, протоколу VTP, підмереж VLAN та протоколу мережевого часу NTP на комутаторах:

SW1:

```
Switch>en
```

```
Switch#conf t
Switch(config)#hostname SW1
SW1(config)#username admin privilege 15 algorithm-type sha256 secret
12#$qwER
SW1(config)#no ip routing
SW1(config)#int vlan 20
SW1(config-if)#ip address 192.168.44.11 255.255.255.192
SW1(config-if)#exit
SW1(config)#ip default-gateway 192.168.44.62
SW1(config)#ntp
SW1(config)#ip domain-name raccoon.net
SW1(config)#ip name-server 192.168.44.62
SW1(config)#vtp domain raccoon.net
SW1(config)#vtp password 12#$qwER
SW1(config)#vtp mode master
SW1(config)#ip name-server 192.168.44.62
SW1(config)#vlan 20
SW1(config-vlan)#name MGMT-VLAN-20
SW1(config-vlan)#vlan 30
SW1(config-vlan)#name SOC-VLAN-30
SW1(config-vlan)#vlan 40
SW1(config-vlan)#name MGMT-VLAN-40
SW1(config-vlan)#exit
SW1(config)#int range e0/0-2
SW1(config-if-range)#switchchport trunk encapsulation dot1q
SW1(config-if-range)#switchchport mode trunk
SW1(config-if-range)#switchchport trunk allowed vlan 20,30,40
SW1(config-if-range)#switchchport nonegotiate
SW1(config-if-range)#exit
SW1(config)#ntp server 192.168.44.62
SW1(config)#clock timezone EET +2
SW1(config)#clock summer-time EEST recurring
SW1(config)#do write
```

SW2:

```
Switch>en
Switch#conf t
Switch(config)#hostname SW2
SW2(config)#username admin privilege 15 algorithm-type sha256 secret
12#$qwER
SW2(config)#no ip routing
SW2(config)#int vlan 20
SW2(config-if)#ip address 192.168.44.12 255.255.255.192
SW2(config-if)#exit
SW2(config)#ip default-gateway 192.168.44.62
SW2(config)#ntp
SW2(config)#ip domain-name raccoon.net
SW2(config)#ip name-server 192.168.44.62
SW2(config)#vtp domain raccoon.net
SW2(config)#vtp password 12#$qwER
SW2(config)#vtp mode client
SW2(config)#ip name-server 192.168.44.62
SW2(config)#int e0/1
SW2(config-if)#switchchport trunk encapsulation dot1q
SW2(config-if)#switchchport mode trunk
SW2(config-if)#switchchport trunk allowed vlan 20,30,40
SW2(config-if)#switchchport nonegotiate
SW2(config-if)#exit
SW2(config)#ntp server 192.168.44.62
SW2(config)#clock timezone EET +2
SW2(config)#clock summer-time EEST recurring
SW2(config)#do write
```

SW3:

```
Switch>en
Switch#conf t
Switch(config)#hostname SW3
SW3(config)#username admin privilege 15 algorithm-type sha256 secret
12#$qwER
SW3(config)#no ip routing
SW3(config)#int vlan 20
SW3(config-if)#ip address 192.168.44.12 255.255.255.192
SW3(config-if)#exit
SW3(config)#ip default-gateway 192.168.44.62
SW3(config)#ntp
SW3(config)#ip domain-name raccoon.net
SW3(config)#ip name-server 192.168.44.62
SW3(config)#vtp domain raccoon.net
SW3(config)#vtp password 12#$qwER
SW3(config)#vtp mode client
SW3(config)#ip name-server 192.168.44.62
SW3(config)#int e0/1
SW3(config-if)#swithchport trunk encapsulation dot1q
SW3(config-if)#swithchport mode trunk
SW3(config-if)#swithchport trunk allowed vlan 20,30,40
SW3(config-if)#swithchport nonegotiate
SW3(config-if)#exit
SW3(config)#ntp server 192.168.44.62
SW3(config)#clock timezone EET +2
SW3(config)#clock summer-time EEST recurring
SW3(config)#do write
```

```
SW1#show ip int br | i Vlan20
Vlan20                 192.168.44.11   YES NVRAM  up                    up
SW1#show ntp as
SW1#show ntp associations

  address         ref clock       st   when   poll reach  delay  offset   disp
  192.168.44.13   .INIT.          16      3     64     0  0.000   0.000 15937.
*~192.168.44.62   194.54.80.29     3    123     64    76  0.000   0.000  4.746
 * sys.peer, # selected, + candidate, - outlyer, x falseticker, ~ configured
SW1#show clock detail
22:43:55.921 EET Sat Dec 23 2023
Time source is NTP
Summer time starts 02:00:00 EET Sun Mar 10 2024
Summer time ends 02:00:00 EEST Sun Nov 3 2024
```

Рисунок 2.1.6 – Результат виконання команд ***show ip interface brief***, ***show ntp associations*** та ***show clock detail*** на комутаторі **SW1**

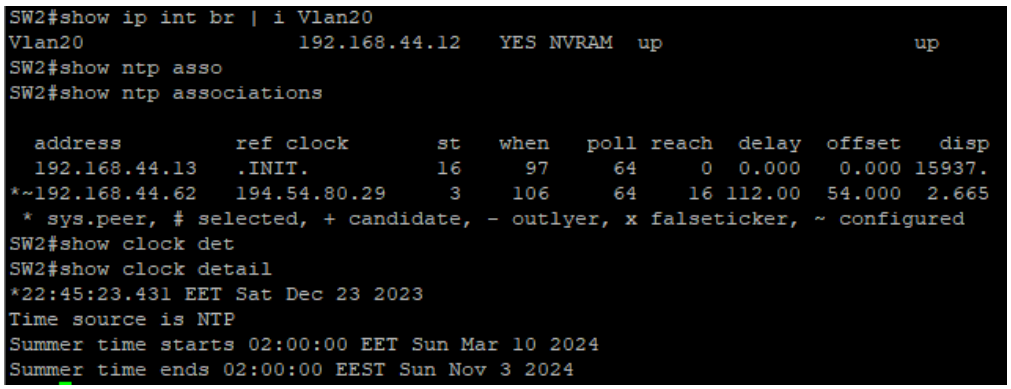


Рисунок 2.1.7 – Результат виконання команд ***show ip interface brief***, ***show ntp associations*** та ***show clock detail*** на комутаторі **SW2**

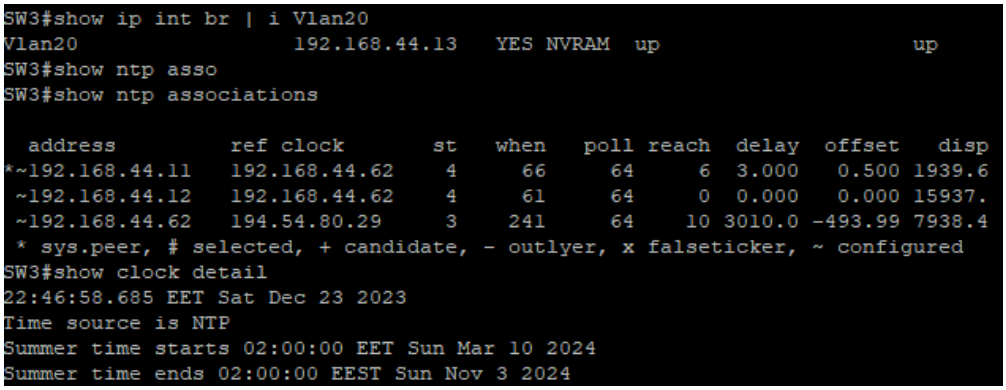


Рисунок 2.1.8 – Результат виконання команд ***show ip interface brief***, ***show ntp associations*** та ***show clock detail*** на комутаторі **SW3**

Мережеві налаштування на серверах Ubuntu Server виконувались за допомогою утиліти netplan шляхом модифікації конфігураційних файлів. Відредагувавши їх необхідним чином буде відредаговано файли самих сервісів SOC.

Налаштування параметрів IP-адресації серверу ELK:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: false
      addresses: [192.168.44.65/26]
      nameservers:
        addresses: [192.168.44.126, 8.8.8.8]
      routes:
        - to: default
          via: 192.168.44.126
```

Налаштування параметрів elasticsearch серверу ELK (***/etc/elasticsearch/elasticsearch.yml***):

```
cluster.name: soc
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch
network.host: 192.168.44.65
http.port: 9200
discovery.type: single-node
xpack.security.enabled: true
xpack.security.authc.api_key.enabled: true
xpack:
  security:
    authc:
      realms:
        native:
          native1:
            order: 0
script.allowed_types: inline, stored
```

Налаштування параметрів elasticsearch серверу ELK (***/etc/kibana/kibana.yml***):

```
server.port: 5601
server.host: "192.168.44.65"
elasticsearch.hosts: ["http://192.168.44.65:9200"]
elasticsearch.username: "kibana_system"
elasticsearch.password: "123!@#"
logging:
  appenders:
    file:
      type: file
      fileName: /var/log/kibana/kibana.log
      layout:
        type: json
xpack.encryptedSavedObjects.encryptionKey:
'01234567890123456789012345678901'
```

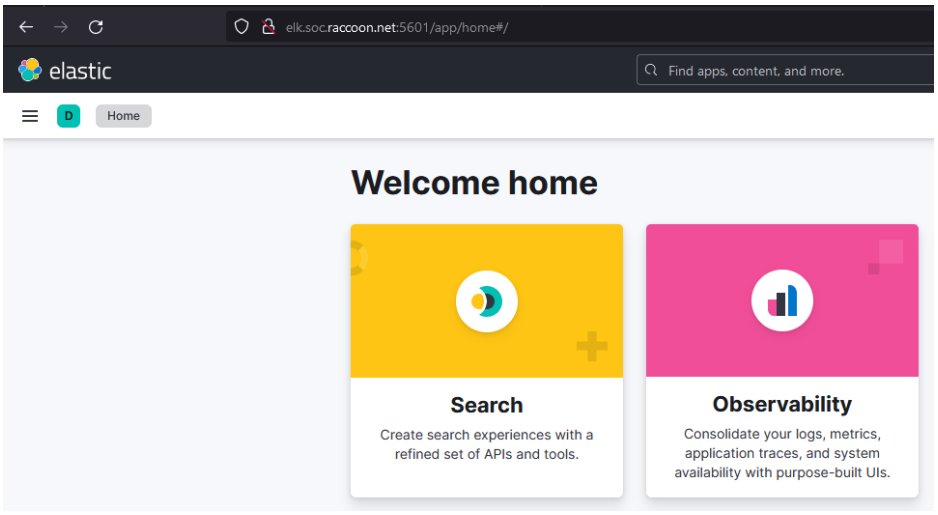


Рисунок 2.1.9 – Перевірка доступності веб-інтерфейсу ELK Stack

Доступ до ELK наявний, тож можемо підключити TheHive, так як він використовує систему elasticsearch для зберігання власних даних.

Налаштування параметрів IP-адресації серверу TheHive:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: false
      addresses: [192.168.44.66/26]
      nameservers:
        addresses: [192.168.44.126, 8.8.8.8]
      routes:
        - to: default
          via: 192.168.44.126
```

Налаштування параметрів thehive серверу TheHive (***/etc/thehive/application.conf***):

```
http.address = 192.168.44.66
http.port = 9000
include "/etc/thehive/secret.conf"
db.janusgraph {
  storage {
    backend = cql
    hostname = ["192.168.44.66"]
    # Cassandra authentication (if configured)
    # username = "thehive"
    # password = "password"
    cql {
      cluster-name = soc
      keyspace = thehive
    }
  }
  index.search {
    backend = elasticsearch
    hostname = ["192.168.44.65"]
    index-name = thehive
    elasticsearch {
      http {
        auth {
          type = basic
          basic {
            username = elastic
            password = "123!@#"
          }
        }
      }
    }
  }
```

```
    }
    storage {
      provider = localfs
      localfs.location = /opt/thp/thehive/files
    }
    play.http.parser.maxDiskBuffer = 1GB
    play.http.parser.maxMemoryBuffer = 10M
    application.baseUrl = "http://192.168.44.66:9000"
    play.http.context = "/"
    scalligraph.modules += org.thp.thehive.connector.cortex.Cortex-
Module
```

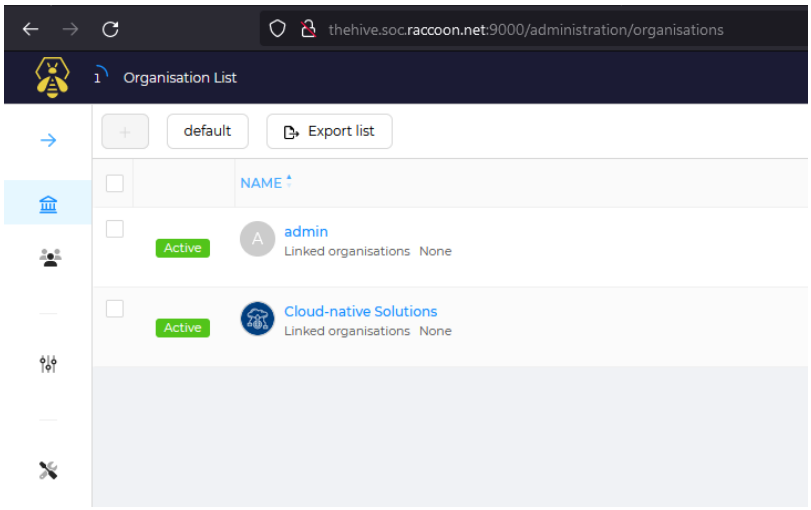


Рисунок 2.1.10 – Перевірка доступності веб-інтерфейсу TheHive

TheHive також доступний, що дозволяє перейти до налаштування Cortex.

Налаштування параметрів IP-адресації серверу Cortex:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: false
      addresses: [192.168.44.66/26]
      nameservers:
        addresses: [192.168.44.126, 8.8.8.8]
      routes:
        - to: default
          via: 192.168.44.126
```

Налаштування параметрів thehive серверу TheHive (***/etc/thehive/application.conf***):

```
    play.http.secret.key="FZNFXs37fC7QHcA9IoSFu83TS0wxH5Hc9jNfDud0JgmqwtdgQoMKX9ytcRlUa8d0"
```

```
  index = cortex
  uri = "http://192.168.44.67:9200"
  user = "elastic"
  password = "123!@#"
cache.job = 10 minutes
auth {
        provider = [local]
        ad {
        }
        ldap {
        }
    oauth2 {
    }
    sso {
    }
}
analyzer {
  urls = [
    "https://download.thehive-project.org/analyzers.json"
  ]
  fork-join-executor {
    parallelism-min = 2
    parallelism-factor = 2.0
    parallelism-max = 4
  }
}
responder {
  urls = [
    "https://download.thehive-project.org/responders.json"
  ]
  fork-join-executor {
    parallelism-min = 2
    parallelism-factor = 2.0
    parallelism-max = 4
  }
}
```

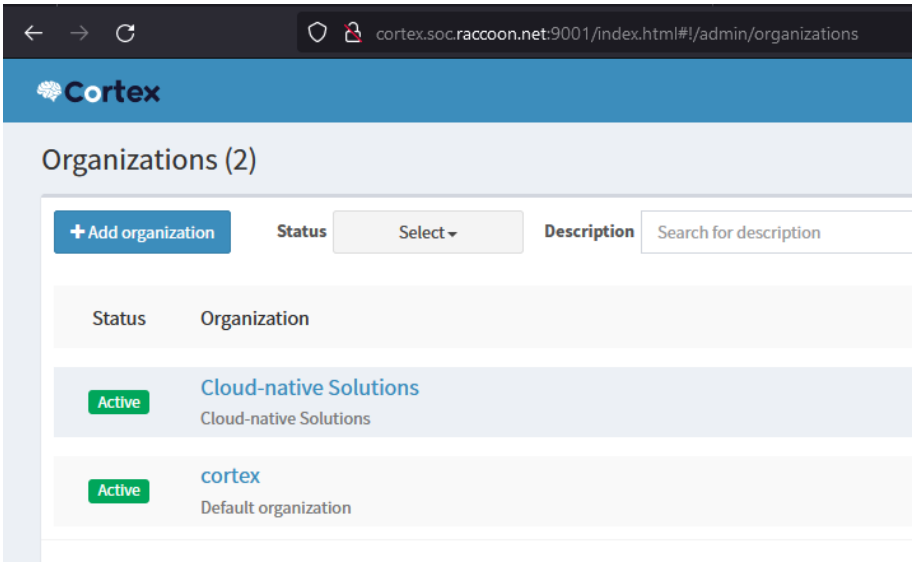


Рисунок 2.1.11 – Перевірка доступності веб-інтерфейсу TheHive

В останню чергу проведемо налаштування серверу K8s.

Налаштування параметрів IP-адресації серверу K8s:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: false
      addresses: [192.168.44.129/26]
      nameservers:
        addresses: [192.168.44.190, 8.8.8.8]
      routes:
        - to: default
          via: 192.168.44.190
```

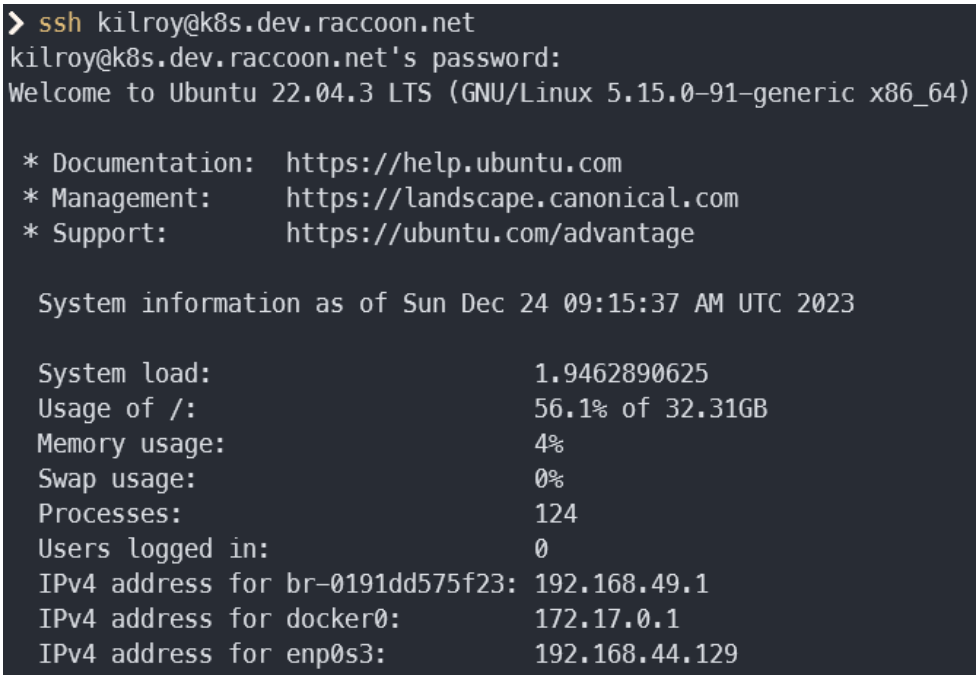


Рисунок 2.1.12 – Перевірка доступності K8s по SSH

2.2 Визначення моделі загроз cloud-native середовища

Станом на 2023 рік в сфері безпеки існує безліч так званих фреймворків для опису атак та підходів зловмисників, що дозволяє полегшити SOC написання правил виявлення цих самих атак, а розробникам – шляхом пом'якшення або усунення можливості ці атаки проводити.

Трьом найбільш впізнаваними та популярними фреймворками є матриця MITRE ATT&CK, Cyber Kill Chain та Dimond Model, однак саме матриця MITRE

ATT&CK є індустріальним стандартом, якщо ця індустрія відноситься до корпоративного сегменту, мобільного сегменту або сегменту критично важливої інфраструктури.

Матриця ATT&CK складається з тактик, технік і процедур (рис 2.2.1), інакше відомих як TTP. У 12 стовпчиках, або тактиках, зліва направо, наведено ще один варіант кроків, які зловмисник зазвичай виконує під час атаки на організацію.

Для реалізації однієї і тієї ж тактики може бути використано кілька технік, і залежно від основної мети зловмисника, не всі 12 тактик можуть бути використані. Сукупність технік, що використовуються під час атаки, відома як профіль поведінки - процедура, якої дотримувався зловмисник для досягнення своєї кінцевої мети, атакуючи систему.

Сучасна матриця ATT&CK є 14 ітерацією вдосконаленою ітерацією, однак при цьому вона містить в собі не всі техніки, які зловмисник може використовувати при атаці на cloud-native середовище. Повертаючись до 4 C's безпеки цих середовищ, матриця ATT&CK v14 покриває лише питання, що стосуються хмари та контейнерів. Питання коду можна більшою мірою віднести до практик безпечної розробки, а ось TTP кластеру поки ще не описані.

В квітні 2020 компанія Microsoft випустила свою матрицю Threat Matrix for Kubernetes [18]. Ця матриця, використовуючи знайомий підхід матриці MITRE ATT&CK, класифікувала та описала TTP для середовища Kubernetes, враховуючи його особливості.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Impact
Using cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access Kubernetes API server	Access cloud resources	Images from a private registry	Data destruction
Compromised image In registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account	Collecting data from pod	Resource hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Container service account	Network mapping	Cluster internal networking		Denial of service
Application vulnerability	Application exploit (RCE)	Malicious admission controller	Access cloud resources	Connect from proxy server	Application credentials in configuration files	Exposed sensitive interfaces	Application credentials in configuration files		
Exposed sensitive interfaces	SSH server running inside container	Container service account			Access managed identity credentials	Instance Metadata API	Writable hostPath mount		
	Sidecar injection	Static pods			Malicious admission controller		CoreDNS poisoning		
							ARP poisoning and IP spoofing		

Рисунок 2.2.1 – Матриця Threat Matrix for Kubernetes

Додатково, для розширеного розуміння векторів атаки на контейнери всередині кластеру Kubernetes було проведено моделювання загроз. Модель загроз було побудовано за методологією STRIDE, яка класифікує всі загрози в наступні категорії:

Таблиця 2.2.1 – Модель загроз STRIDE

Загроза	Властивість безпеки, якій загрожують
Spoofing (Підміна)	Автентифікація
Tampering (Незаконна зміна)	Цілісність
Repudiation (Заперечення)	Аудит
Information Disclosure (Розкриття даних)	Конфіденційність
Denial of Service (Відмова в обслуговуванні)	Доступність
Elevation of Privilege (Підвищення повноважень)	Авторизація

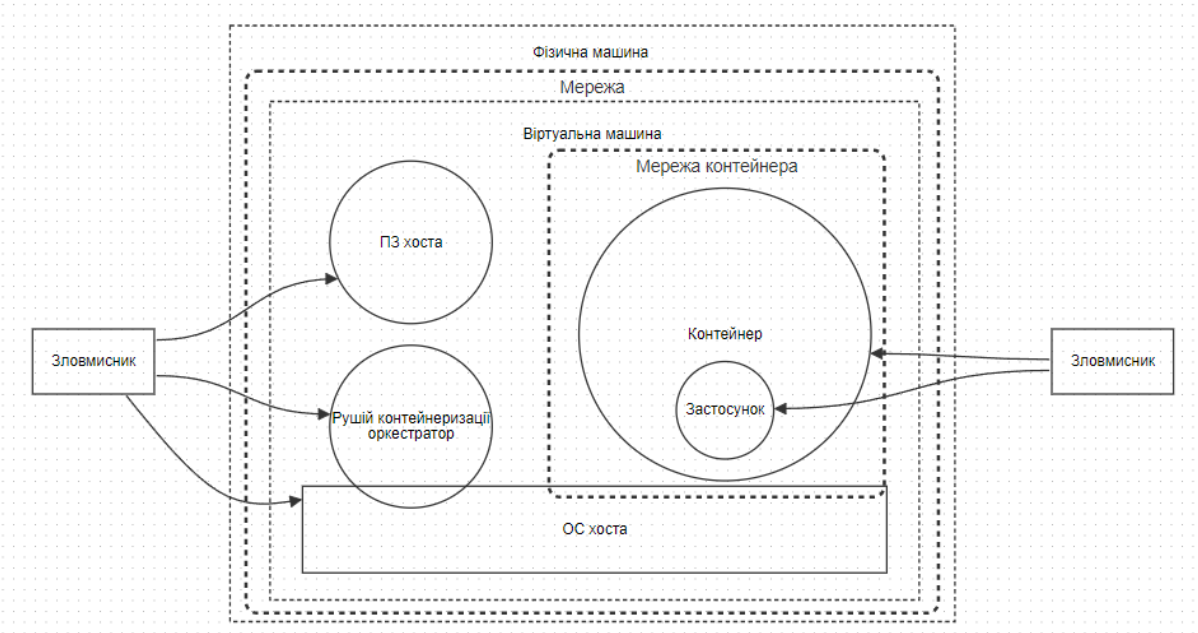


Рисунок 2.2.2 – Модель загроз контейнеризованого середовища

Моделювання проводилось з використанням продукту OWASP Threat Dragon. На наступних скріншотах будуть показані компоненти контейнеризованого середовища, загрози, які їм може становити зловмисник та коригуючі міри, що можуть повністю або частково запобігти реалізації цих загроз. Також слід зауважити, що загрози таких компонентів контейнеризованого середовища, як фізична та віртуальні машини, не розглядались у розрізі сконструйованої моделі загроз.

ПЗ хоста (Process)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
8	Використання вразливостей коду	Elevation of privilege	High	Mitigated	9	Зловмисник може використати вразливості коду, що наявні у встановленому на хості ПЗ та пакунках, тим самим підвищивши собі повноваження або повністю зкомпрометувавши контейнеризоване середовище.	Запровадження механізмів скаування вразливостей ПЗ, фікс-циклів та перевірок на відповідність використовуючи інструменти типу OpenSCAP

Рисунок 2.2.3 – Вектори атаки на ПЗ хоста

Рушій контейнеризації оркестратор (Process)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
13	Використання вразливостей коду	Elevation of privilege	Medium	Mitigated	6	Зловмисник може використати вразливості коду, що наявні у самій ОС.	Регулярне встановлення останніх оновлень ОС.

Рисунок 2.2.4 – Вектори атаки на рушій контейнеризації/оркестратор

Застосунок (Data Flow)

Number	Title	Type	Priority	Status	Score	Description	Mitigations
18	Використання вразливостей коду	Elevation of privilege	High	Mitigated	9	Зловмисник може використати вразливості коду, контейнеризованого застосунку, тим самим отримавши доступ до контейнеру, з можливістю ескалацію до рівню хоста.	1. Запровадження процесів QA в цикл робзробки 2. Дотримання методології DevSecOps 3. Перекриття можливості використання вразливостей веб-застосунку, описаних в OWASP Top 10
19	Перехоплення чутливих даних	Information disclosure	Medium	Mitigated	7	Майже будь-які контейнери для повноцінного функціонування потребують облікові дані, токени чи паролі. При некоретній передачі цих даних контейнеру, зловмисник може отрмати до них доступ.	Будь-які чутливі дані бажано зберігати в пам'яті ОС, тобто в тимчасовому каталозі. Доступ до цього каталогу з контейнеру буде здійснюватись шляхом його монтування. Також це дозволить змінювати необхідні чутливі дані в процесі роботи контейнеру, без необхідності його перезавантаження.

Рисунок 2.2.5 – Вектори атаки на контейнеризований застосунок

Контейнер (Process)

Ізольоване середовище роботи додатку

Number	Title	Type	Priority	Status	Score	Description	Mitigations
20	Зкомпрометований образ контейнеру	Tampering	High	Mitigated	8	Отримавши доступ до процесу створення контейнерного образу зловмисник може змінити файл Dockerfile, або підмінити легітимний образ на власний шкідливий.	1. Використання сканерів контейнерних образів, наприклад Trivy, Clair чи Anchore. 2. Проведення сканування на етапах розробки та CI/CD, а також всередині реєстру контейнерів. 3. Впровадження механізму цифрового підпису для образів.
21	Небезпечне налаштування образу контейнеру	Elevation of privilege	High	Mitigated	10	Через некоректне використання директив в Dockerfile, запуск контейнеру від імені користувача root або з прапорцем --privileged, зкомпрометувавши контейнер, зловмисник може швидко набути найвищих повноважень та захопити все контейнеризоване середовище.	1. Створення образів контейнеру без використанням демону Docker 2. Гранулярний підхід до призначення можливостей всередині контейнера, на противагу використання прапорція --privileged 3. Створення Dockerfile згідно безпечних практик
22	"Втеча" з контейнеру	Elevation of privilege	Medium	Mitigated	7	Інколи в сучасних контейнерах знаходять вразливості, що дозволять зловмиснику швидко "вибратись" із контейнеру на рівень хоста.	1. Впровадження механізму Seccomp, яка обемжує список системних викликів, які може виконувати процес. 2. Застосування LSM-модуля AppArmor, що довзоляє створювати окремі профілі для виконуваних файлів, що дозволяють обмежувати їх можливості та доступ до файлів. 3. Застосування LSM-модуля SELinux, що довзоляє створювати окремі домени, які у свою чергу обмежують можливості процесу маніпулювати з файлами та взаємодіяти з іншими процесами.

Рисунок 2.2.6 – Вектори атаки на контейнер

Таким чином, обравши матрицю Threat Matrix for Kubernetes та змоделювавши поверхню атаки на контейнер всередині Kubernetes можемо приступити до обрання необхідного, специфічного інструменту для середовища cloud-native, що покриє питання виявлення TTP та векторів атаки. Впевнившись в тому, що інструмент працює належним чином, його буде інтегровано в наявні рішення SOC.

2.3 Налаштування та інтеграція рішень між собою

Після короткого аналізу відкритих джерел було обрано рішення Falco від компанії Sysdiag [19]. Це рішення виявляє підозрілі дії над кластером та контейнерами, забезпечує велику кількість вже написаних правил та правил від спільноти, підтримує написання власних правил, а також має можливість відсилати свої спрацювання за допомогою sidecar-поду Falcosidekick, в тому числі в ELK.

Та це все одно не дасть змоги аналітики реагувати належним чином, через обране рішення TheHive. Таке питання здатен вирішити фреймворк ElastAlert 2

[20]. Він працює на основі власних правил та здатен аналізувати індекси elasticsearch, в які і будуть потрапляти спрацювання Falco. Цей фреймворк містить велику кількість алертів, що підтримують різні продукти, серед яких TheHive.

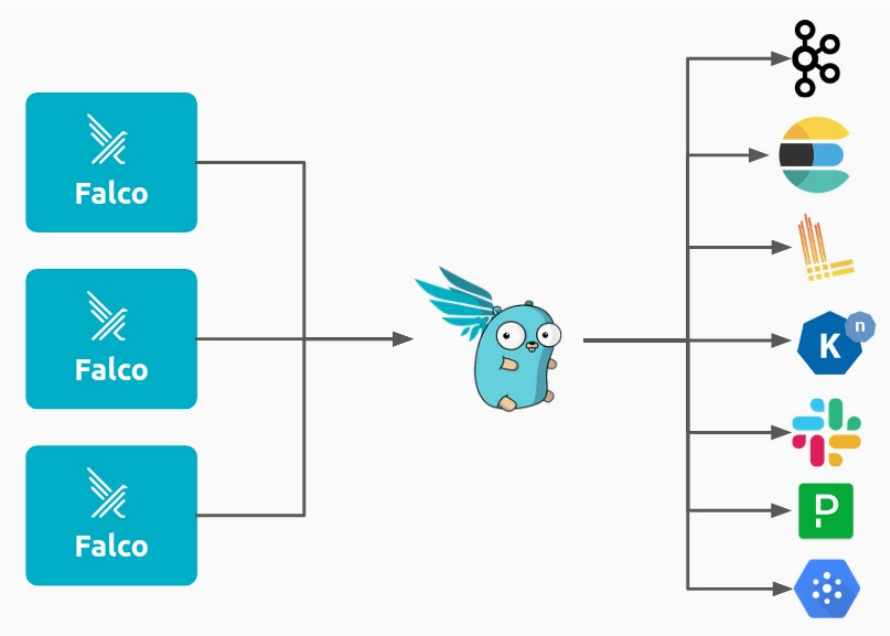


Рисунок 2.3.1 – Спрощена схема роботи Falco та Falcosidekick

Для розширення області видимості аналітиків з безпеки в SIEM-систему ELK Stack також будуть надсилатись журнали з мережевих пристроїв та метрики кластеру Kubernetes.

Як вже було сказано в першому розділі, ELK складається з 3 компонентів: Elasticsearch, Logstash та Kibana. В межах кваліфікаційної роботи не використовувався Logstash. Хоч він і дозволяє робити оптимізаційні трансформації над даними та може працювати з будь-якими журналами незалежно від пристрою, що їх відправив, він також потребує власних немалих обчислювальних ресурсів, кількість яких обмежена у зв’язку з експериментальністю спроектованого середовища.

Замість Logstash буде використовуватись ще одне з безкоштовних відкритих рішень Elastic Beats. Beats – це швидкі та малоресурсні агенти, які дозволяють збирати дані з великої кількості різних пристроїв та відправляти їх для обробки в Logstash, або як у нашому випадку – напряму в Elasticsearch. В межах середовища зокрема використовуватимуться filebeat для збирання журналів Syslog та потоків

трафіку протоколу NetFlow з мережевих пристроїв Cisco, а також metricbeat – для збирання метрик та подій кластеру Kubernetes. Додатково в кластері буде встановлено повноцінний агент Elastic Agent, як альтернативу metricbeat.

Обравши необхідні для виявлення загроз в cloud-native середовищах інструменти, було створено принципову схему роботи (рис. 2.3.2) такої підсистеми моніторингу.

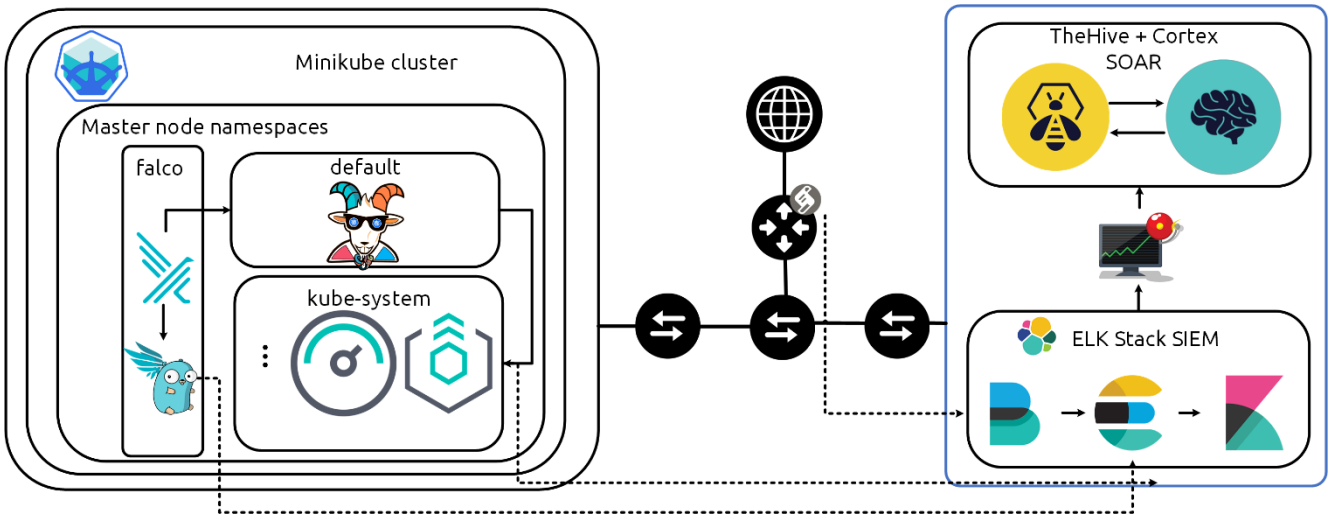


Рисунок 2.3.2 – Принципова схема підсистеми моніторингу подій безпеки

Покроковий принцип роботи такої системи можна описати наступним чином:

1) Falco виявляє спробу компрометації в просторі імен default, де запущено Kubernetes Goat.

2) Дані про виявлену подію передаються на sidecar-под Falcosidekick, де нормалізуються та відправляються до elasticsearch.

3) На сервері ELK встановлено ElastAlert 2 та написано правило на виявлення спрацювань falco, що періодично відпрацьовує та відсилає алерти до SOAR-системи TheHive.

4) Аналітики інцидентів відпрацьовують алерт, та приймають рішення про підвищення його до статусу інциденту, при цьому використовуючи аналізатори та відповідачі Cortex.

5) Додатково в SIEM-системі ELK ввімкнені свої правила для виявлення підозрілих дій в Kubernetes на основі подій та метрик, надісланих з подів metricbeat та elasticagent з простору імен kube-system.

Проведемо налаштування filebeat на сервері ELK Stack, який буде отримувати журнали протоколів Syslog та NetFlow від маршрутизатора R1.

Сценарії налаштування Syslog та NetFlow на маршрутизаторі R1:

R1:

```
R1(config)#service timestamps log datetime
R1(config)#logging host 192.168.44.65 transport udp port 9002
R1(config)#logging trap 5
R1(config)#login on-failure log every 1
R1(config)#login on-success log every 1
R1(config)# ip flow-export source Ethernet0/1.30
R1(config)# ip flow-export version 9
R1(config)# ip flow-export template options export-stats
R1(config)# ip flow-export template options timeout-rate 120
R1(config)# ip flow-export template options refresh-rate 25
R1(config)# ip flow-export template timeout-rate 90
R1(config)# ip flow-export template refresh-rate 15
R1(config)# ip flow-export interface-names
R1(config)# ip flow-export destination 192.168.44.65 2055
R1(config)#int e0/0
R1(config-if)#ip flow ingress
R1(config-if)#ip flow egress
R1(config-if)#exit
R1(config)#do write
```

Файл налаштувань filebeat на сервері ELK (***/etc/filebeat/filebeat.yml***):

```
filebeat.inputs:
- type: filestream
  id: my-filestream-id
  enabled: false
  paths:
    - /var/log/*.log
filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false
setup.template.settings:
  index.number_of_shards: 1
setup.kibana:
  host: "192.168.44.65:5601"
output.elasticsearch:
  hosts: ["192.168.44.65:9200"]
  username: "elastic"
  password: "123!@#"
processors:
  - add_host_metadata:
      when.not.contains.tags: forwarded
  - add_cloud_metadata: ~
  - add_docker_metadata: ~
  - add_kubernetes_metadata: ~
```

Файл налаштувань модуля cisco на сервері ELK (***/etc/filebeat/modules.d/cisco.yml***):

```
- module: cisco
```

```
    asa:
      enabled: false
    ftd:
      enabled: false
    ios:
      enabled: true
      var.input: syslog
      var.syslog_host: 192.168.44.65
      var.syslog_port: 9002
    nexus:
      enabled: false
    meraki:
      enabled: false
    umbrella:
      enabled: false
    amp:
      enabled: false
```

Файл налаштувань модуля netflow на сервері ELK (*/etc/filebeat/modules.d/netflow.yml*):

```
- module: netflow
  log:
    enabled: true
    var:
      netflow_host: 192.168.44.65
      netflow_port: 2055
```

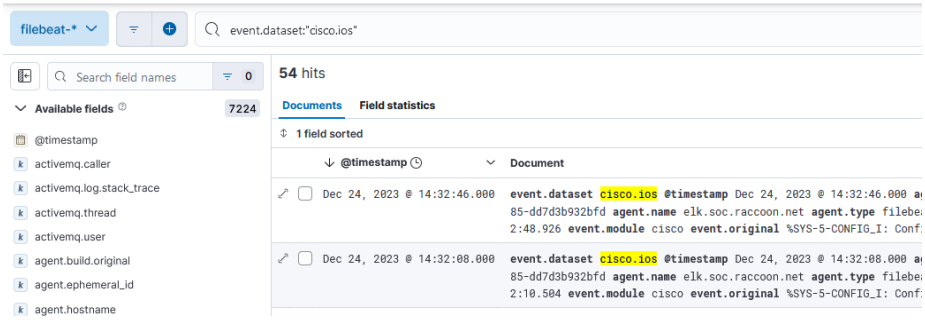


Рисунок 2.3.3 – Приклад отриманих журналів Syslog

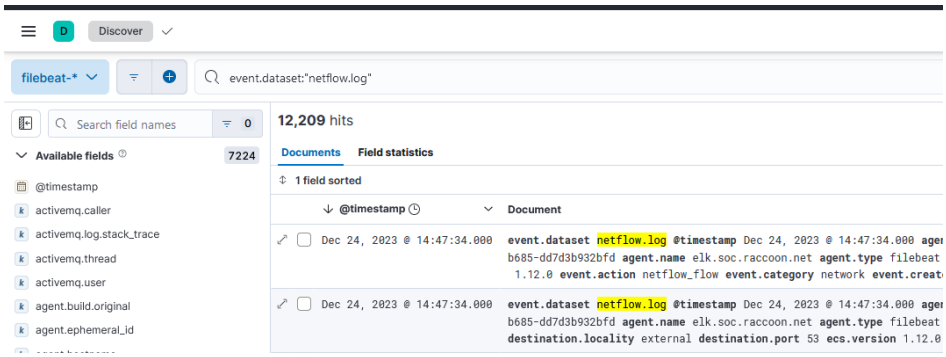


Рисунок 2.3.4 – Приклад отриманих журналів NetFlow

Перший крок по заведенню та інтеграції журналів в підсистему моніторингу можна вважати завершеним.

Перейдемо до збирання подій та метрик із середовища Kubernetes. Для цього необхідно розгорнути elasticagent та metricbeat в наявному кластері (що також потребує наявності в кластері поду kube-state-metrics), а також налаштувати falcosidekick для пересилання алертів в ELK.

Налаштування, які необхідно змінити у manifest-файлі metricbeat (***/home/kilroy/es/metricbeat-kubernetes.yml***):

```
output.elasticsearch:
      hosts: ['http://192.168.44.65:9200']
      username: 'elastic'
      password: '123!@#'
```

Налаштування, які необхідно змінити у manifest-файлі elasticagent (***/home/kilroy/es/elastic-agent-standalone-kubernetes.yml***):

```
outputs:
      default:
        type: elasticsearch
        hosts:
          - 'http://192.168.44.65:9200'
        username: 'elastic'
        password: '123!@#'
```

Запуск сервісів kube-state-metrics та falco відбувався через Helm – менеджер пакунків Kubernetes. Налаштування falcosidekick відбувалось шляхом модифікації змінних його Helm-чарту.

Команда helm, використана для запуску kube-state-metrics в кластері:

```
helm install -n kube-system kube-state-metrics prometheus-community/kube-state-metrics
```

Команда helm, використана для запуску falco в кластері в окремому просторі імен:

```
helm install --namespace falco --create-namespace falco falcosecurity/falco --set driver.kind=modern-bpf \
--set tty=true \
--set falcosidekick.enabled=true \
```

```
    --set falcosidekick.webui.enabled=true \
    --set falcosidekick.webui.user='falco:123!@#' \
    --set falcosidekick.config.elasticsearch.host-
port='http://192.168.44.65:9200' \
    --set falcosidekick.config.elasticsearch.username='elastic' \
    --set falcosidekick.config.elasticsearch.password='123!@#'
```

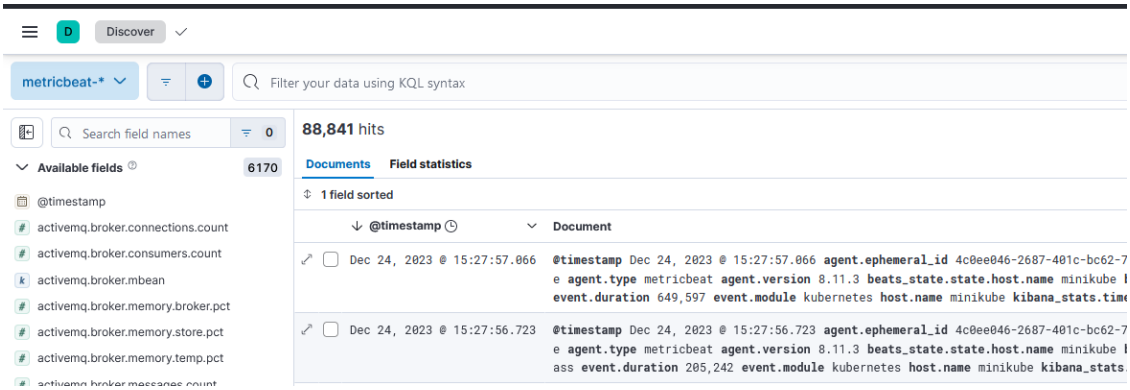


Рисунок 2.3.5 – Приклад отриманих метрик від metricbeat

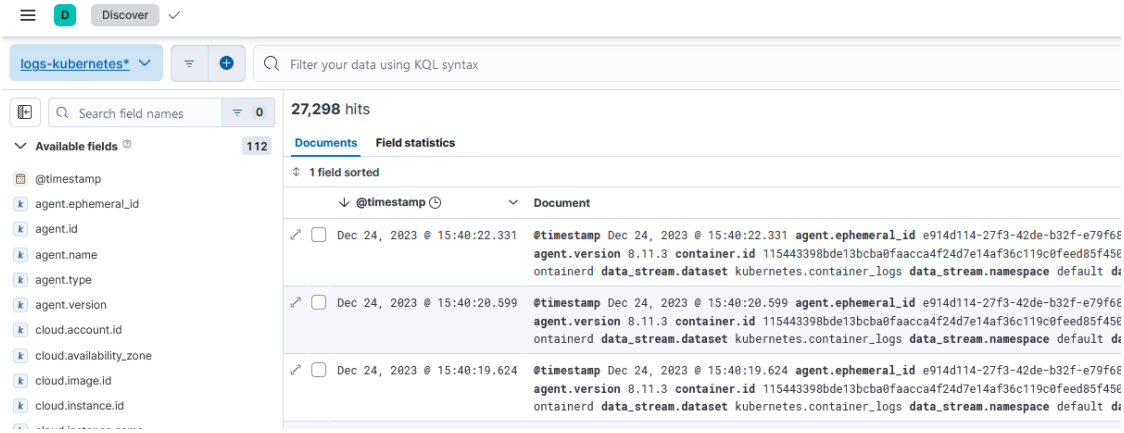


Рисунок 2.3.6 – Приклад отриманих подій від elasticagent

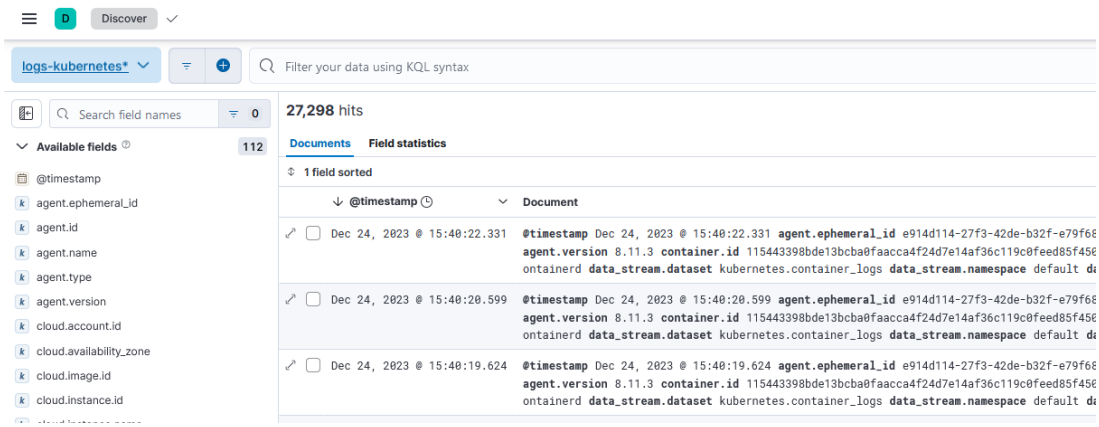


Рисунок 2.3.7 – Приклад отриманих метрик від elasticagent

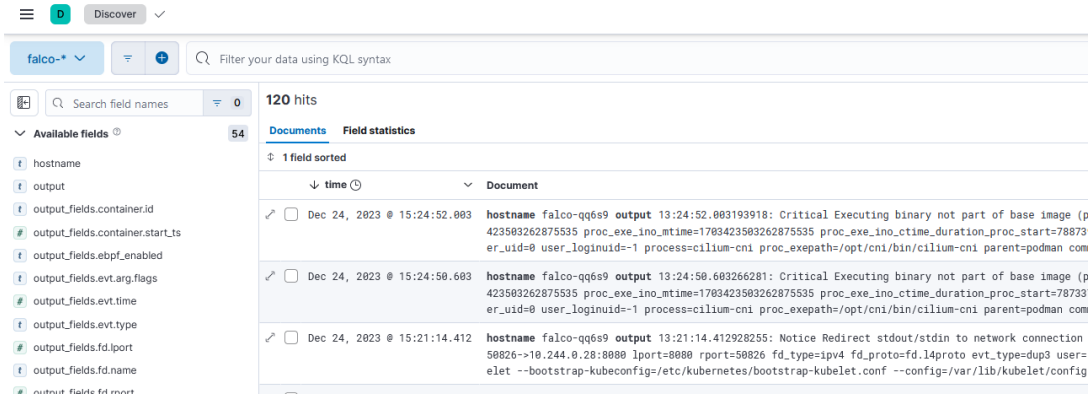


Рисунок 2.3.8 – Приклад отриманих спрацювань від falcosidekick

Перейдемо до останього кроку – інтеграції TheHive та Cortex між собою. Для цього створимо користувача TheHive в Cortex, який взаємодіятиме з Cortex через власний API-токен. Після цього зможемо додати Cortex в систему TheHive.

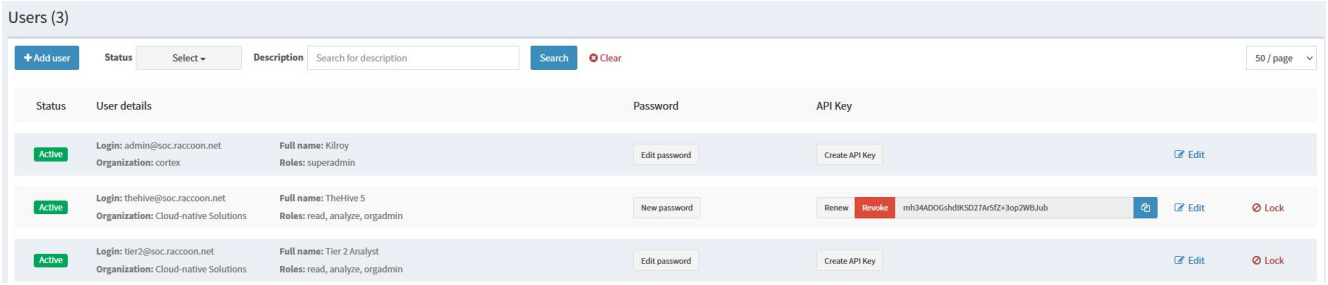


Рисунок 2.3.9 – Користувачі Cortex

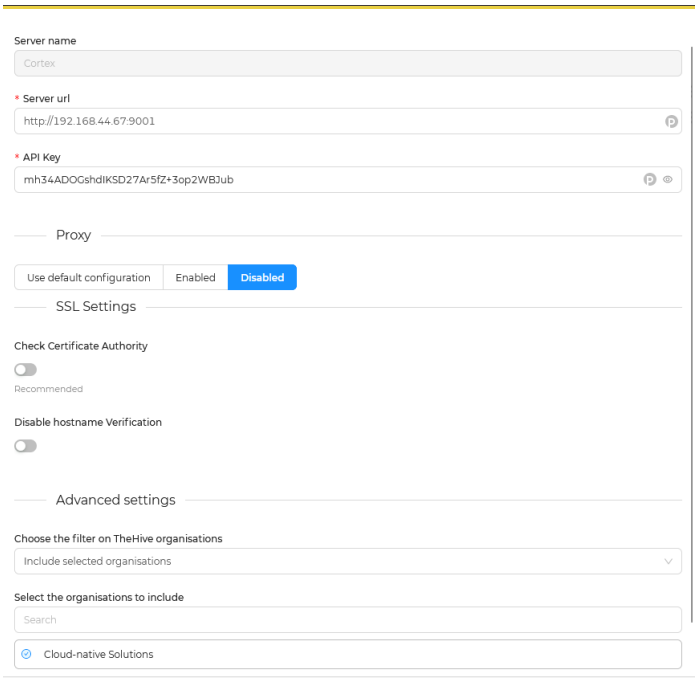


Рисунок 2.3.10 – Налаштування підключення Cortex в системі TheHive

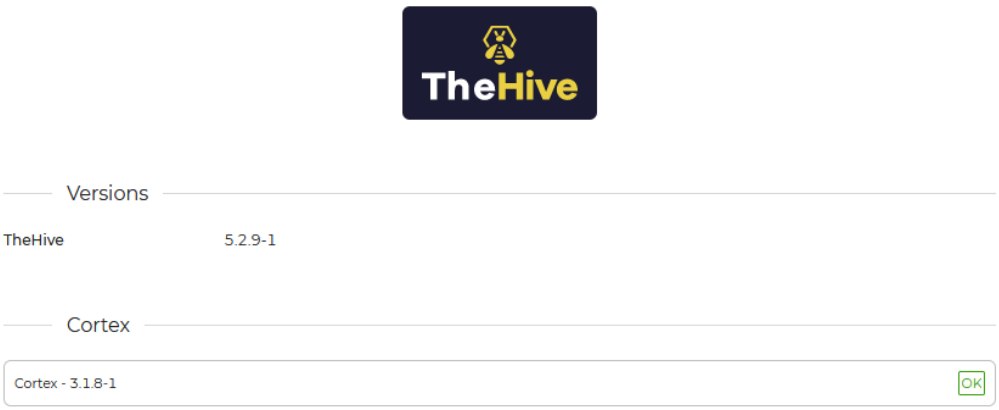


Рисунок 2.3.11 – Демонстрація успішної інтеграції систем

На цьому питання налаштування та інтеграції підсистеми моніторингу в cloud-native середовище можна вважати вирішеним. В наступному розділі буде створено та протестовано правила безпеки, що дозволять виявляти порушників в середовищі.

Висновки до 2-го розділу:

В даному розділі виконано проектування віртуального мережевого середовища, проведено налаштування мережевого обладнання, серверів підсистеми моніторингу та серверу розробки, яке виступає елементом cloud-native.

На основі спроектованого середовища визначено його модель загроз та обрано фреймворки безпеки, що належним чином описують його поверхню та вектори атаки.

Останнім кроком було проведено інтеграцію рішень безпеки між собою в одну підсистему моніторингу подій безпеки та з’єднання цієї підсистеми з середовищем cloud-native.

РОЗДІЛ 3 ПЕРЕВІРКА ФУНКЦІОНУВАННЯ ТА ЕФЕКТИВНОСТІ ВИЯВЛЕННЯ ЗАГРОЗ ПІДСИСТЕМОЮ

3.1 Створення правил безпеки,

Falco зі стандартним набором правил вже перекриває більшість тактик та технік MITRE ATT&CK [21], що сильно спрощує роботу аналітиків по написанню контенту. ELK Stack в свою чергу пропонує набір правил по Kubernetes доступних через відповідну інтеграцію (рис. 3.1.1), але ці правила не діють без відповідних налаштувань kube-apiserver та elasticagent.

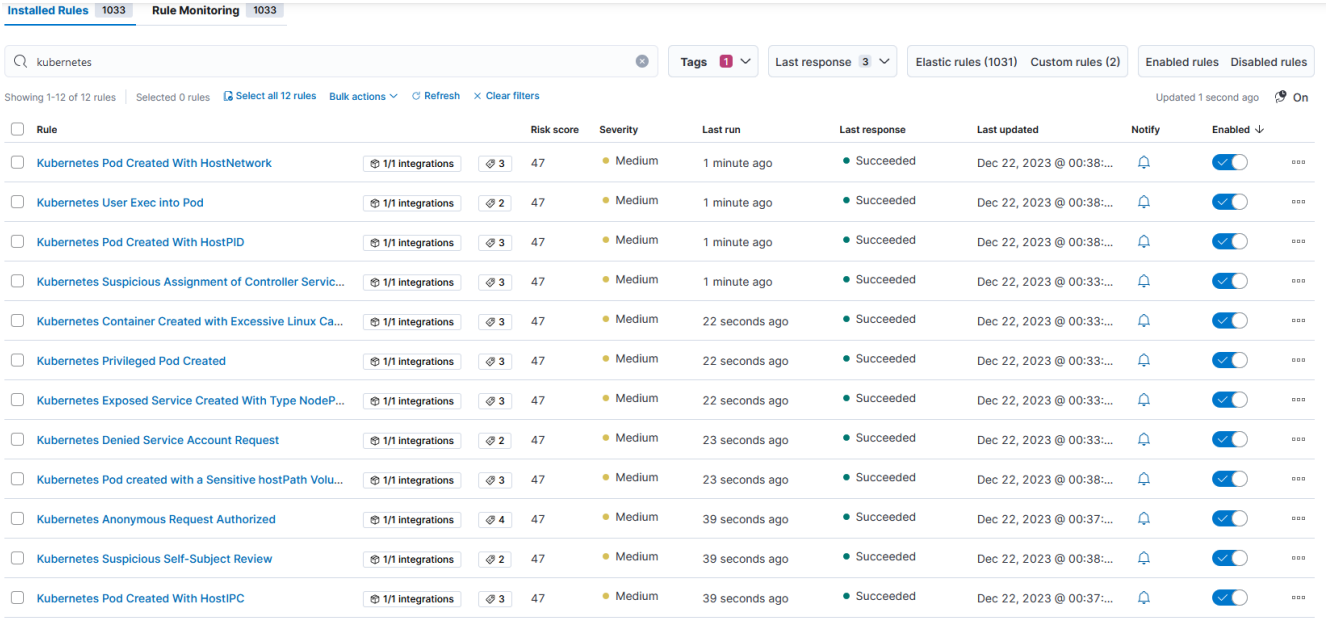


Рисунок 3.1.1 – Набір готових правил для середовища Kubernetes

Таким для чином підсистемі моніторингу залишається лише виявляти та відправляти спрацювання цих правил до системи керування інцидентами безпеки TheHive. Як вже було сказано в попередньому розділі, це легко реалізується через безкоштовний інструмент, написаний на мові програмування Python, під назвою ElastAlert.

Реалізуємо правила ElastAlert, що буде працювати на основі алертів Falco та експортуватимуть ці алерти до системи TheHive.

Файл конфігурації для запуску правил ***/opt/elastalert/config-thehive.yaml***:

```
rules_folder: /opt/elastalert/rules/falco # Розташування директорії з правилами
run_every:
  minutes: 5 # Як часто будуть запускатись правила з директорії
buffer_time:
  minutes: 15 # Останніх 15 хв, за які всі події будуть перевірені на співпадання з
правилами
es_host: 192.168.44.65 # IP-адреса серверу elasticsearch
es_port: 9200 # Порт серверу elasticsearch
es_username: elastic # Ім'я для авторизації на сервері elasticsearch
es_password: '123!@#' # Пароль для авторизації на сервері elasticsearch
writeback_index: elastalert_status # Індекс, в який записувати алерти в elasticsearch
writeback_alias: elastalert_alerts # Псевдонім попереднього індексу
```

Файл конфігурації для запуску правил ***/opt/elastalert/config-thehive.yaml***:

```
name: Falco Notice Rule # Ім'я правила
type: any # Тип правила
fields:
  - "priority" # Поля, по яким вібудватиметься фільтрація
filter:
- query:
    query_string:
      query: "priority: Notice" # Фільтр, для визначення критичності алерту
index: 'falco-*' # Індекс, в якому буде здійснено пошук
is_enabled: true # Правило ввімкнене
realert:
  minutes: 0 # Отримувати кожен алерт
timestamp_field: 'time' # Поле журанлу elasticsearch, в якому зазначений час
timestamp_type: iso # Форматування поля часу
use_strftime_index: false # Не форматувати індекс

alert: hivealerter # Тип експорту - до TheHive
hive_connection:
  hive_host: http://192.168.44.66 # URL TheHive
  hive_port: 9000 # Порт TheHive
  hive_apikey: sGPUw9LB/ZXYnza4TJNfJjRUNzot9NnW # API-токен для взаємодії з API
TheHive

hive_alert_config:
  title: 'Falco: {0} - {1}' # Загаловок алерту, сформований з шаблоних полів
  title_args: [ priority, rule ]
  type: '{}' # Тип алерту, сформований з шаблоних полів
  type_args: [ evt.type ]
  source: '{0} - {1}' # Джерело алерту, сформоване з шаблоних полів
  source_args: [ k8s.pod.name, source ]
  description: '{0}: \\n{1}' # Опис алерту, сформований з шаблоних полів
  description_args: [ rule, output ]
  severity: 1 # Критичність алерту
  tags: ["tags"] # Теги, напряму перенесені з поля tags алерту Falco
  tlp: 3 # Ступінь чутливості інформації всередині алерту
  status: 'New' # Ступінь чутливості інформації всередині алерту
```

Аналогічні правила були створені для 3 рівнів критичності відповідно до рівнів, які виставляє Falco.

```
kilroy@elk:~$ elastalert --verbose --config /opt/elastalert/config-thehive.yaml &
[2] 3008
[1]   Terminated              elastalert --verbose --config /opt/elastalert/config-thehive.yaml
kilroy@elk:~$ INFO:elastalert:3 rules loaded
INFO:elastalert:Starting up
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 299.993921 seconds
INFO:elastalert:Queried rule Falco Warning Rule from 2023-12-24 20:05 UTC to 2023-12-24 20:08 UTC: 0 / 0 hits
INFO:elastalert:Ran Falco Warning Rule from 2023-12-24 20:05 UTC to 2023-12-24 20:08 UTC: 0 query hits (0 already seen), 0 matches,
0 alerts sent
INFO:elastalert:Falco Warning Rule range 210
INFO:elastalert:Queried rule Falco Notice Rule from 2023-12-24 20:04 UTC to 2023-12-24 20:08 UTC: 0 / 0 hits
INFO:elastalert:Ran Falco Notice Rule from 2023-12-24 20:04 UTC to 2023-12-24 20:08 UTC: 0 query hits (0 already seen), 0 matches, 0
 alerts sent
INFO:elastalert:Falco Notice Rule range 229
INFO:elastalert:Queried rule Falco Critical Rule from 2023-12-24 20:05 UTC to 2023-12-24 20:08 UTC: 0 / 0 hits
INFO:elastalert:Ran Falco Critical Rule from 2023-12-24 20:05 UTC to 2023-12-24 20:08 UTC: 0 query hits (0 already seen), 0 matches,
 0 alerts sent
```

Рисунок 3.1.2 – Запуск написаних правил

Як можна бачити (рис. 3.1.2), правила відпрацювали, але не було виявлено жодних спрацювань Falco. Для повноцінної перевірки правил проведемо декілька атак на Kubernetes, використавши для цього спеціальне вразливе середовище Kubernetes Goat.

3.2 Імітація зловмисної активності

Kubernetes Goat – це інтерактивний навчальний майданчик з безпеки Kubernetes. Він містить навмисно вразливі сценарії, щоб продемонструвати типові помилки конфігурації, реальні вразливості та проблеми безпеки в кластерах, контейнерах та хмарних середовищах Kubernetes.

Також Kubernetes Goat використовується для перевірки виявлення підозрілих подій наявними системами безпеки, що ми саме і будемо робити.

Скачавши офіційний репозиторій та розгорнувши середовище Kubernetes Goat, проведемо декілька атак на нього.

Перша з проведених атак буде експлуатувати факт того, що більшість CI/CD та конвеєрних систем використовують основний хост Docker для створення контейнерів у конвеєрі за допомогою так званого DIND (docker-in-docker) з UNIX-сокетом. У цьому сценарії ми спробуємо скористатися цією помилковою конфігурацією і отримати доступ до хост-системи, вийшовши з контейнера.

Перейшовши на сайт **http://k8s.dev.raccoon.net:1231**, зможемо побачити простий додаток (рис. 3.2.1), що використовується для перевірки статусу серверів за допомогою команди ***ping***.

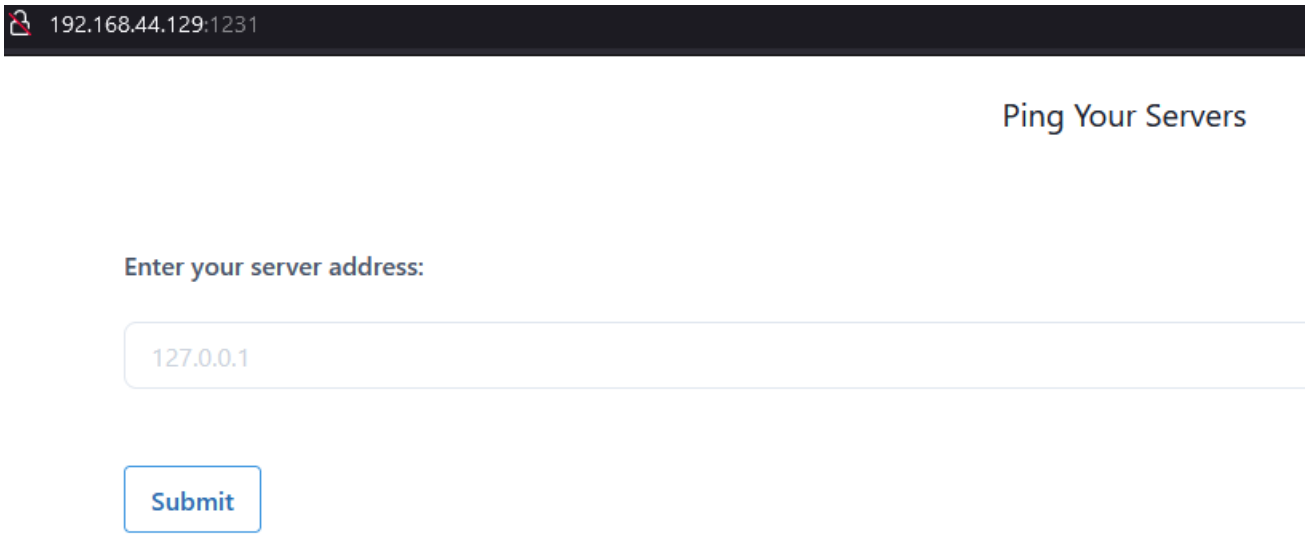


Рисунок 3.2.1 – Головне вікно додатку

Наступним кроком буде перевірка додатку на типову вразливість ін'єкції Unix команд. Спробуємо виконати декілька команд, та проаналізуємо їх вивід:

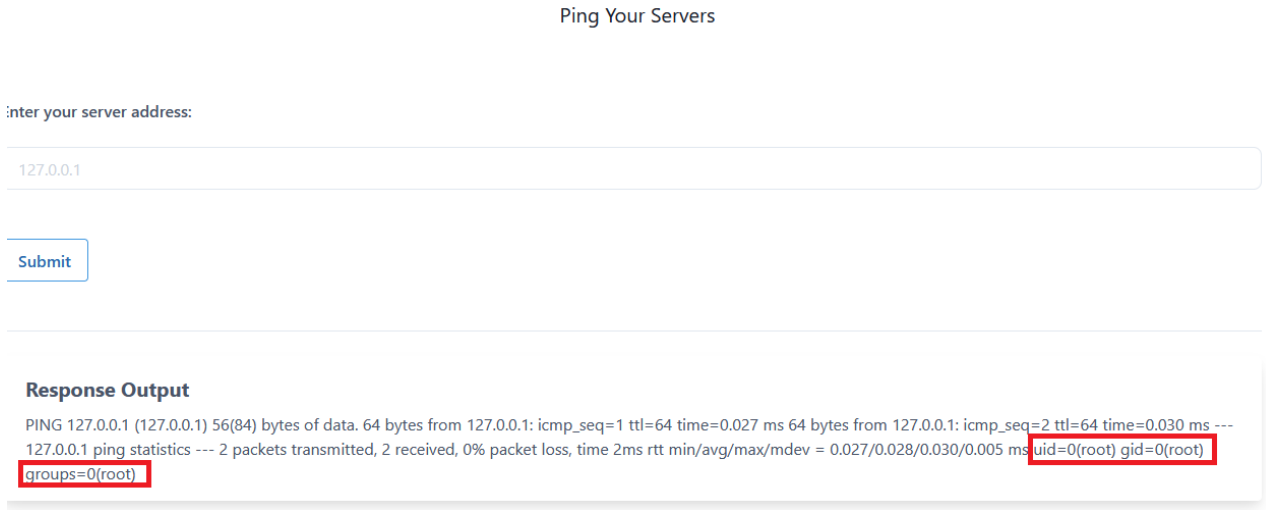


Рисунок 3.2.2 – Ін’єкція команди для перевірки поточного користувача

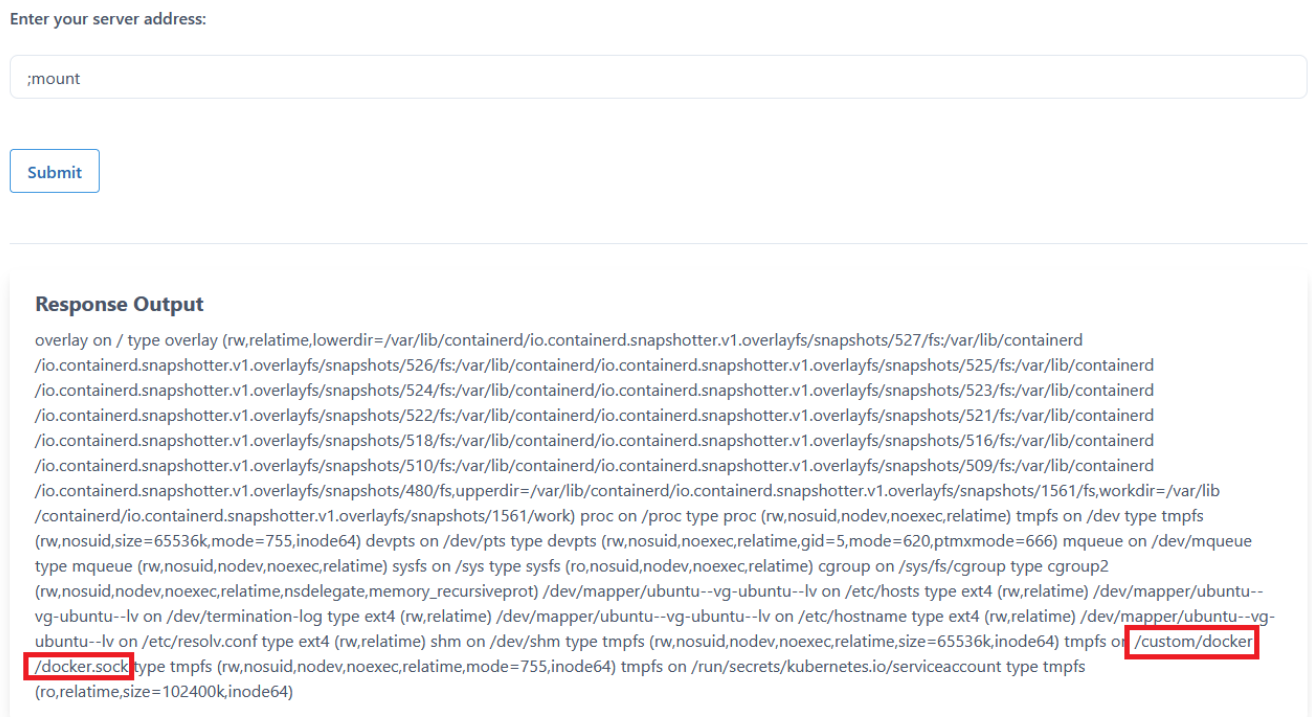


Рисунок 3.2.3 – Ін’єкція команди для перевірки змонтованих директорій та файлів

Виконавши ці дії, ми отримали інформацію, що відповідний контейнер містить змонтований docker.sock, що в теорії повинно дозволити звертатися до нього з середини контейнеру, за умови необхідного виконуваного файлу docker.

З’ясуємо версію ОС після чого завантажимо відповідний виконуваний файл.

Рисунок 3.2.4 – Ін’єкція команди для перевірки версії ОС

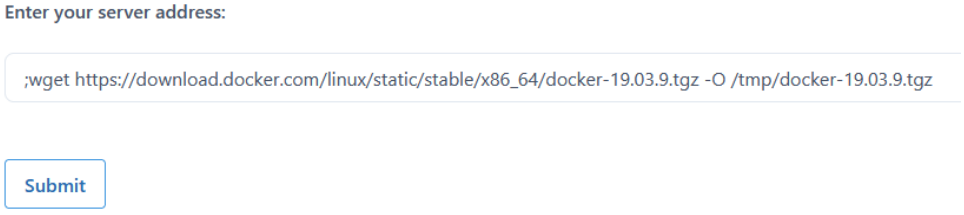


Рисунок 3.2.5 – Завантаження необхідного виконуваного файлу docker

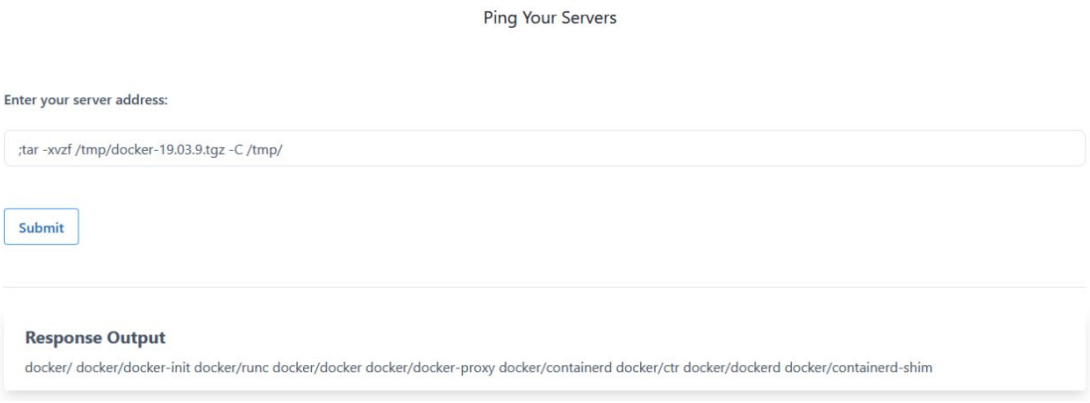


Рисунок 3.2.6 – Розпакування файлу

Отримавши необхідний виконуваний файл, спробуємо вичитати дані зі змонтованого docker.sock.

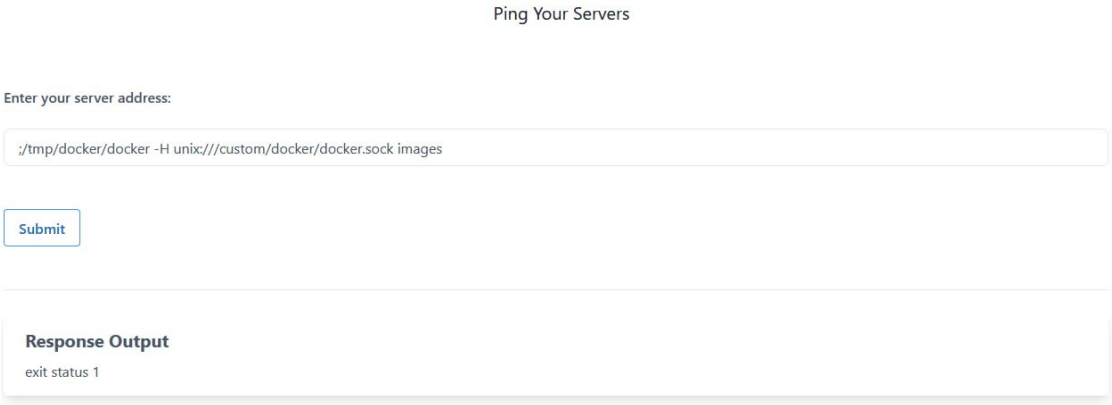


Рисунок 3.2.7 – Спроба вичитки docker.sock

Спроба виявилась невдалою, адже наявність на машині, на якій запущено контейнер з додатком, файлу docker.sock не гарантує, що рушієм контейнеризації дійсно виступає Docker. В даному випадку це дійсно так, адже кластер працює з використанням рушію containerd. Але не зважаючи на це наявні системи безпеки вже повинні були виявити деякі з в виконаних дій підозрілими.

Проведемо ще один сценарій, однак він вже буде пов’язаний не з атакою, а легітимною активністю адміністратора кластера. Скажімо, що адміністратор забув облікові дані одного з записів контейнеру в поді та не може їх відновити. Для цього він заходить на самий контейнер та проводить пошук записаного паролю в чутливих файлах ***/etc/passwd*** та ***/etc/shadow***.

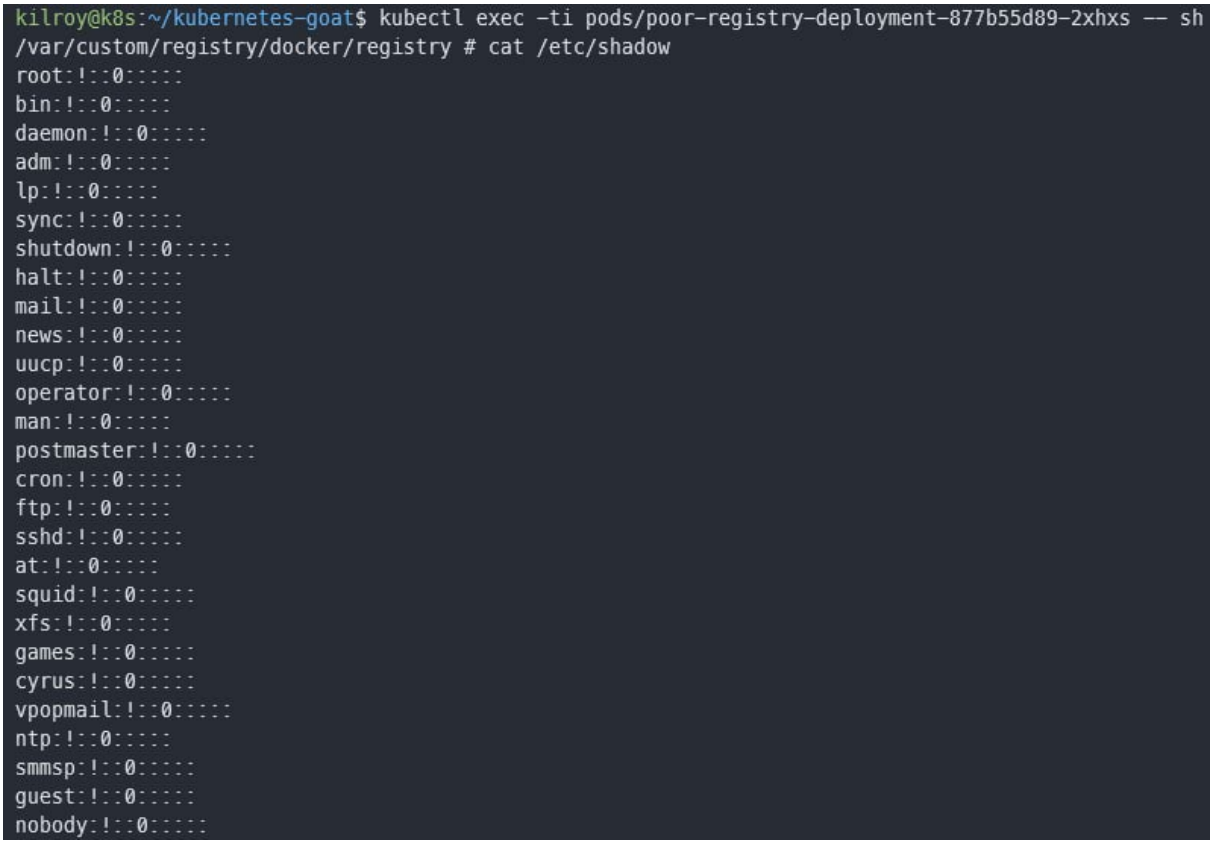


Рисунок 3.2.7 – Дії адміністратора пов’язані з читанням чутливих файлів

Таке читання є доволі не типовим та більше притаманне зловмисникам, адже файл ***/etc/shadow*** до того всього ще й захищений і може читатись лише користувачем root. Наявні механізми безпеки зобов'язані виявляти ці підозрілі дії, після чого аналітикам необхідно з'ясувати легітимність такої активності та за необхідності провести розслідування.

Наступним кроком буде перевірка відпрацювання описаних правил та налаштованих механізмів безпеки середовища.

3.3 Оцінка ефективності виявлення атак та рекомендації що до покращення

Після створення правил ElastAlert 2 та імітації зловмисної активності в cloud-native середовищі, за умови правильного налаштування, аналітики безпеки вже повинні були б спостерігати спрацювання. Перевіримо це, відкривши вікно алертів TheHive

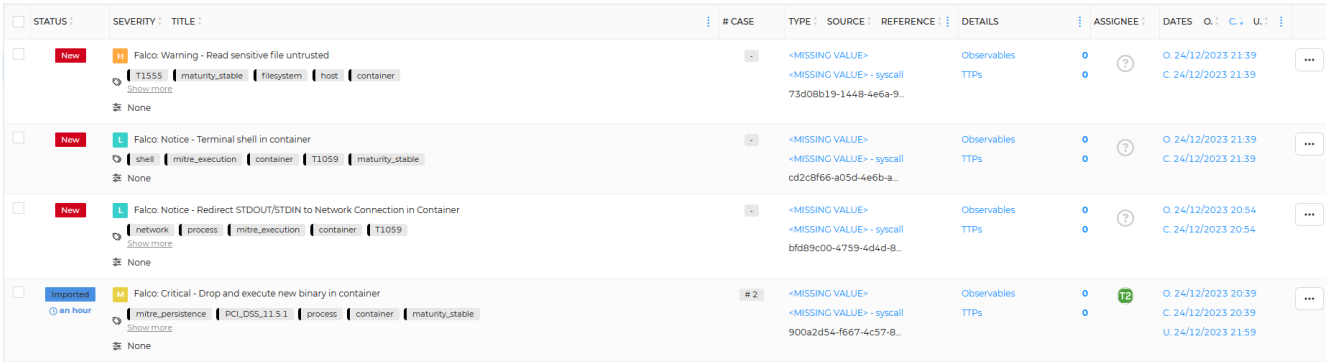


Рисунок 3.1.1 – Створені алерти TheHive

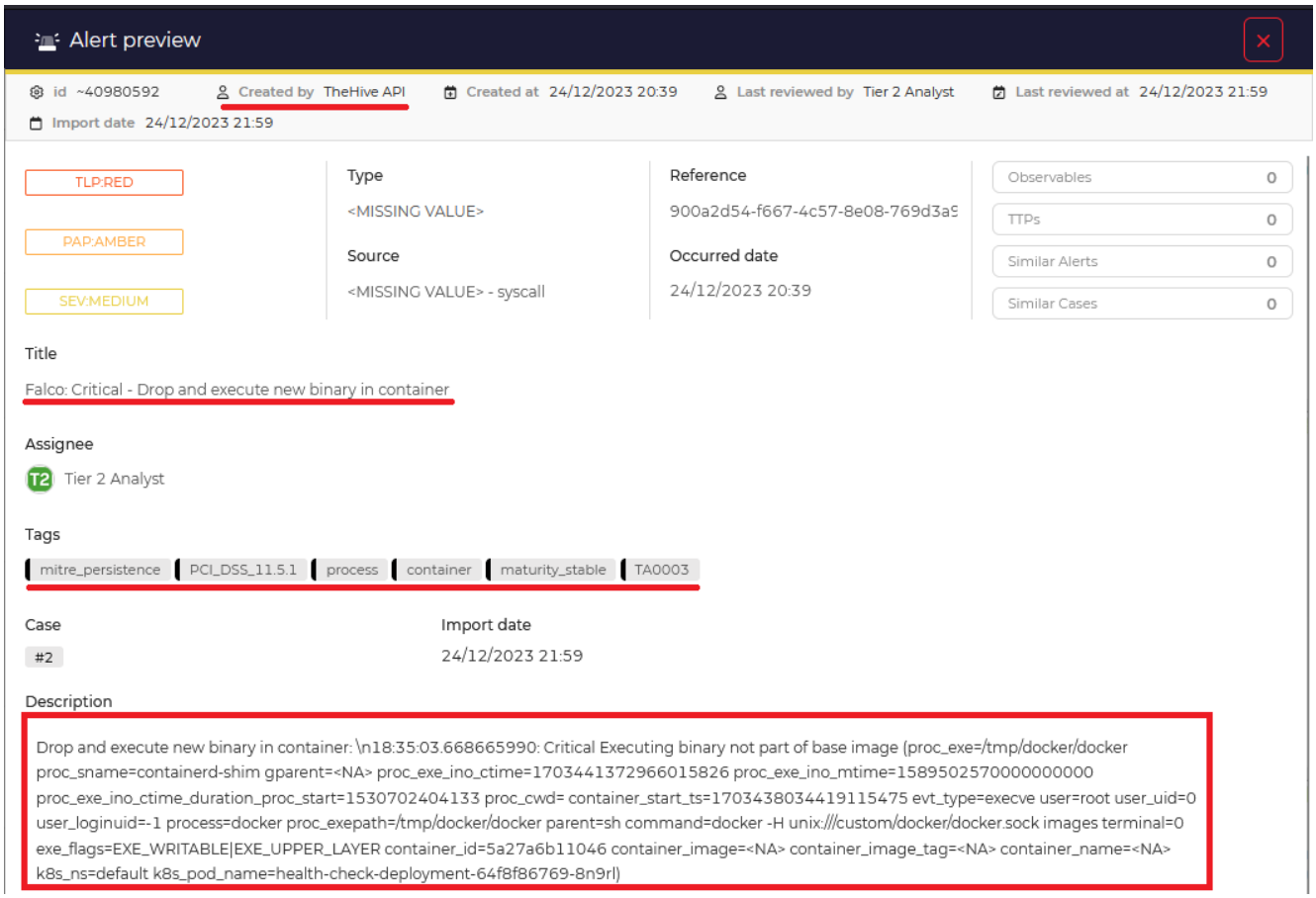


Рисунок 3.3.2 – Перший алерт

З вигляду першого алерту, можна сказати наступне:

1. Алерт створений через TheHive API, що вказує на відпрацювання правила ElastAlert;

2. Відповідний рівень критичності алерту вірно перенесений з описаного правила;

3. Поля заголовку, тегів та опису відповідним чином заповнені з подій індексу elasticsearch falco*.

Все це свідчить про коректну роботу написаних правил. Перевіримо решту алертів, щоб зрозуміти, чи Falco не пропустив жодних підозрілих дій.

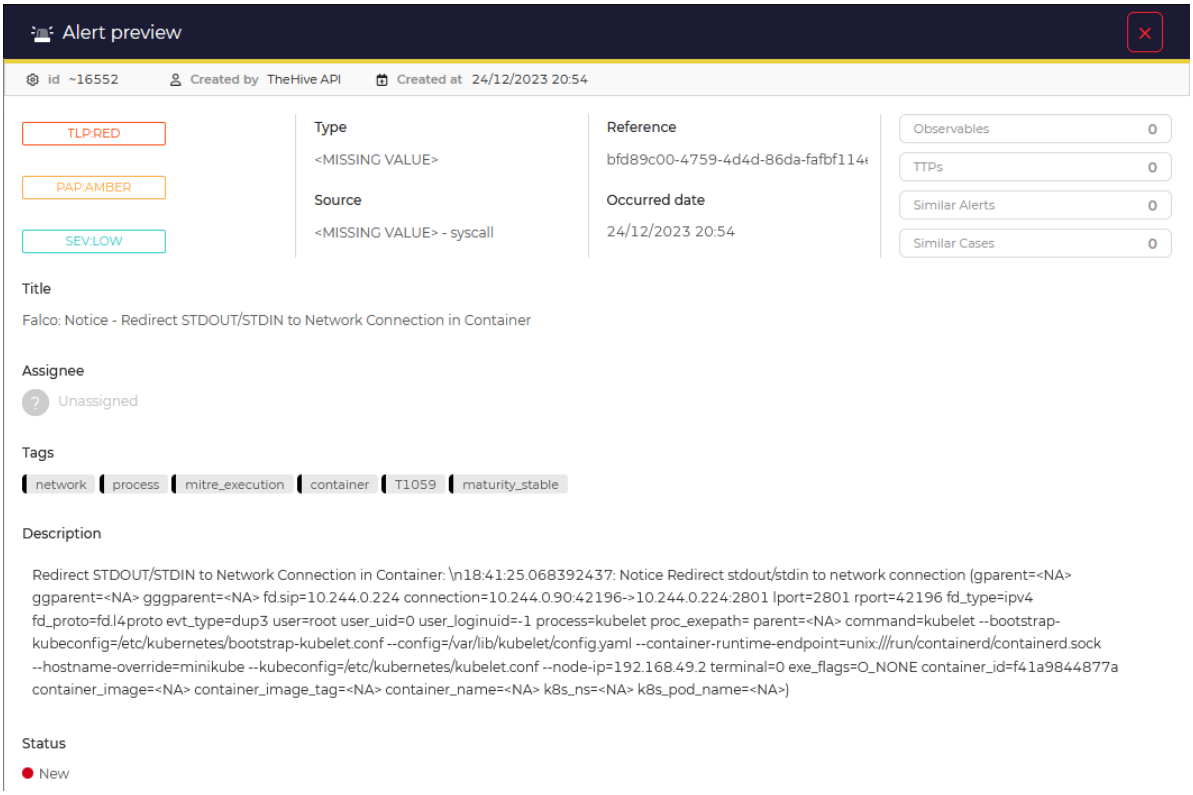


Рисунок 3.3.3 – Другий алерт, пов’язаний зі зверненням до docker.sock

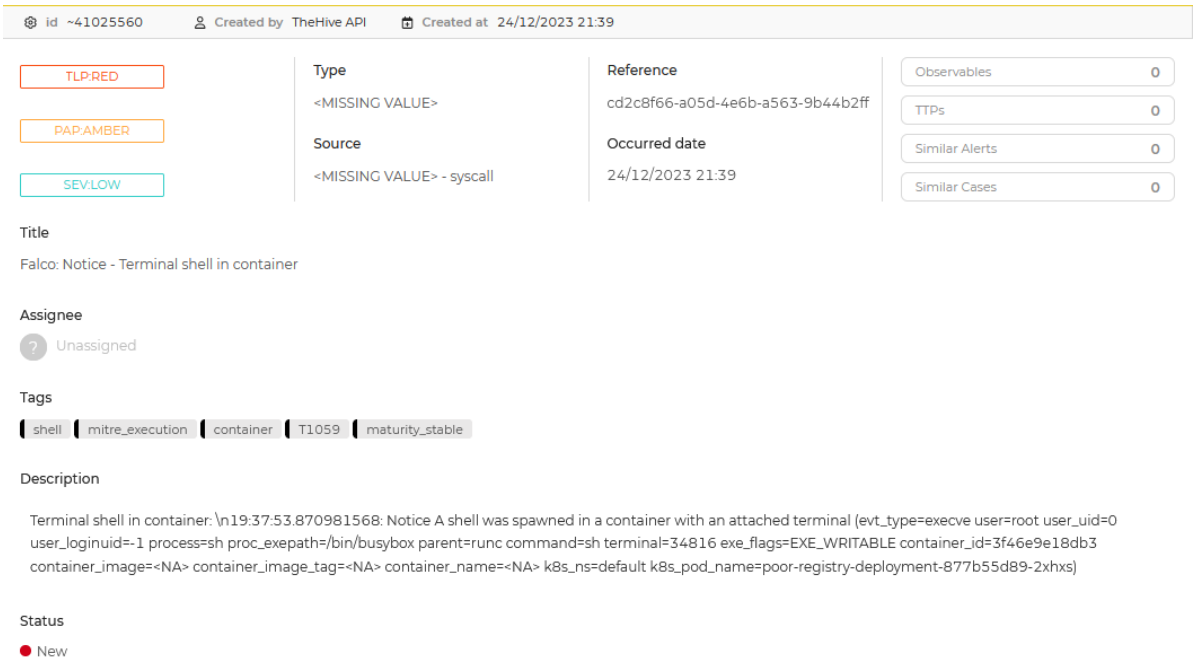


Рисунок 3.3.4 – Третій алерт, пов’язаний зі створення термінальної сесії до контейнеру

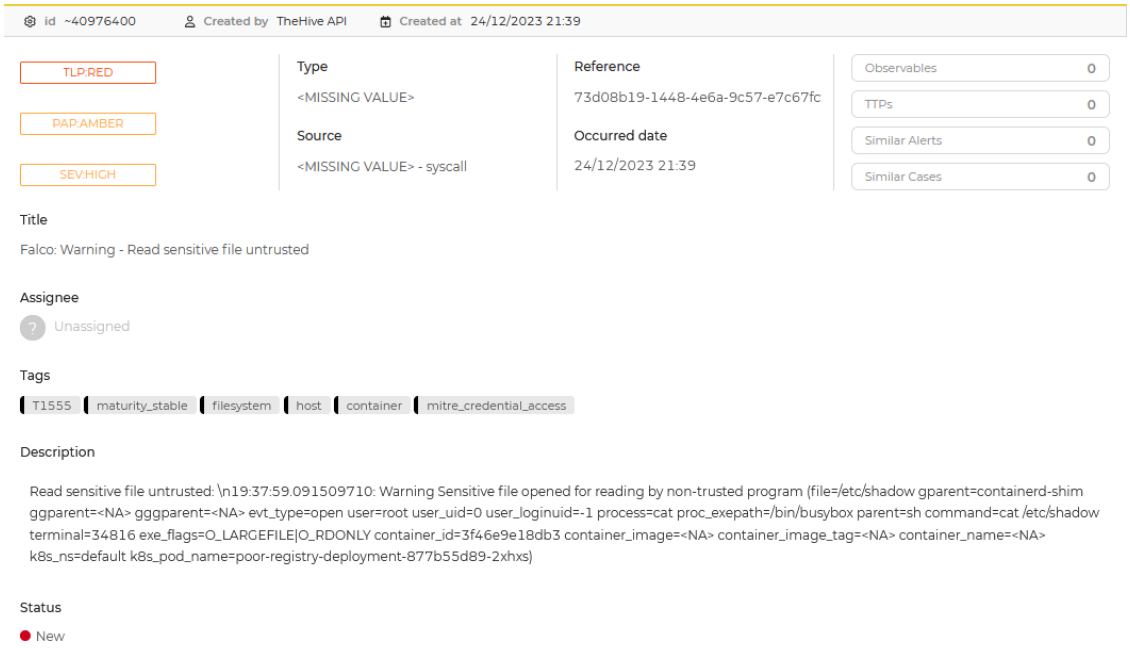


Рисунок 3.3.5 – Четвертий алерт, пов’язаний з читанням чутливого файлу ***/etc/shadow***

Виявлення зловмисних дій підсистемою моніторингу подій безпеки можна вважати вдалим, адже на кожну підозрілу дію всередині cloud-native середовища аналітик з інцидентів отримав відповідний алерт. В реальних сценаріях це дозволить вчасно та якісно відреагувати на те чи інше порушення безпеки належним чином.

Слід також зазначити, що бажано було б виявляти вразливості типу ін’єкцій Unix-команд, але це лежить поза метою цією кваліфікаційної роботи, так як більше стосується Application Security та розгортання мережевого веб-екрану, що буде в автоматичному режимі блокувати спроби експлуатації веб-вразливостей.

Говорячи про простір для вдосконалення даної підсистеми моніторингу можна виділити наступні кроки:

– Розширення моніторингу на компонент kube-apiserver, що дозволить використовувати готові правила виявлення SIEM-системи ELK та реалізовувати свої, більш комплексні правила.

– Додавання сканування в процес моніторингу безпеки контейнерів в середовищі. Це дозволить регулярно перевіряти образи контейнерів, що використо-

вуються в кластері, на предмет вразливостей та застарілих програмних залежностей. Одним з популярних інструментів з широкою підтримкою спільноти та безкоштовною моделлю поширення можна виділити Trivy [12].

– Написання аналізаторів та відповідачів Cortex, які допоможуть аналітикам належним чином розслідувати та реагувати на інциденти в таких середовищах. Для прикладу, створити аналізатор, в який при передачі імені простору імен/поду/контейнера буде виведено розширено інформацію про цей об’єкт, аналогічно команді kubectl describe. Або створення відповідача, який дозволить видаляти заражені чи скомпрометовані об’єкти кластеру для уникнення повної компрометації всього середовища.

– Інтеграція елементів Cortex з платформою Shuffle, що дозволить аналітикам безпеки за допомогою зручного графічного інтерфейсу та блок-схем створювати алгоритми (сценарії) автоматизації, що візьмуть на себе більшість рутинних завдань SOC. До них можна віднести сканування мережі, робочих станцій та сайтів на вразливості або виконання розвідки загроз для виявлення глобальних індикаторів компрометації (IP адрес, хешів файлів, URL адрес, що використовуються зловмисниками) у власному середовищі.

Висновки до 3-го розділу:

В останньому розділі створено правила безпеки, що пов'язують системи SIEM, SOAR та інструмент безпеки виконання контейнерів між собою.

Після цього проведено імітацію зловмисної активності для дослідження можливих сценаріїв компрометації cloud-native середовища та насамперед для перевірки створених правил безпеки.

Проведено оцінку ефективності та повноти виявлення проведеної зловмисної активності описаними правилами. Сформовано рекомендації по покращенню спроектованої підсистеми моніторингу подій безпеки для збільшення можливостей виявлення та реагування на події та інциденти безпеки в cloud-native середовищах.

ВИСНОВКИ

В даній кваліфікаційній роботі було вирішено завдання з проектування підсистеми моніторингу подій безпеки SOC на базі безкоштовних відкритих рішень.

Основною метою було продемонструвати можливість ефективного моніторингу та реагування на інциденти безпеки в cloud-native середовищах, використовуючи безкоштовні відкриті рішення SOC.

Після детального аналізу концепції безпеки cloud-native та організації безпеки в кластерах, контейнерах та хмарі, обрано набір відкритих рішень для побудови підсистеми моніторингу. Ці рішення були вибрані з урахуванням їхньої інтеграції з концепцією "4 C's" та гнучкості їхнього використання в різних складових cloud-native середовищ.

Детально проаналізовано сучасний ринок безкоштовних відкритих рішень для підсистеми моніторингу подій безпеки. Обрані рішення відповідають вимогам концепції безпеки cloud-native та забезпечують потрібний рівень функціональності та інтеграції.

Створено віртуальне мережеве середовище, в якому емульовано реальні умови cloud-native середовища. Взаємодія з емулятором GNS3, інтеграція SIEM та SOAR систем та створення моделі загроз визначили основні компоненти підсистеми моніторингу подій безпеки.

Проведено імітацію зловмисної активності та перевірено виявлення атак в розробленому середовищі. За допомогою підсистем моніторингу безпеки та правил безпеки ефективно виявлені та оброблені події безпеки, що дозволяє вдосконалити стратегії реагування на інциденти.

У результаті виконаної магістерської роботи було досягнуто поставленої мети – продемонструвати та довести, що моніторинг та реагування на інциденти безпеки в cloud-native середовищах можливе з використанням безкоштовних відкритих рішень SOC. Отримані результати можуть бути використані для подальших

досліджень та розвитку систем моніторингу в сфері кібербезпеки cloud-native середовищ.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Binnie C., McCune R. Cloud Native Security. Wiley & Sons, Limited, John, 2021. 336 с.

2. Grasso L., Degioanni L. Practical Cloud Native Security with Falco: Risk and Threat Detection for Containers, Kubernetes, and Cloud. O'Reilly Media, Incorporated, 2022.

3. Pollitt A., Sampat M. Kubernetes Security and Observability: A Holistic Approach to Securing and Troubleshooting Cloud Native Applications. O'Reilly Media, Incorporated, 2021. 165 с.

4. Rice L. Container Security: Fundamental Technology Concepts That Protect Containerized Applications. O'Reilly Media, Incorporated, 2020. 200 с.

5. Dotson C. Practical Cloud Security: A Guide for Secure Design and Deployment. O'Reilly Media, Incorporated, 2019. 196 с.

6. Vehent J. Securing DevOps: Security in the Cloud. Manning Publications, 2018. 384 с.

7. Armitage J. Cloud Native Security Cookbook: Recipes for a Secure Cloud. O'-Reilly Media, Incorporated, 2022.

8. Don M. G. #. Blue Team Handbook : SOC, SIEM, and Threat Hunting: A Condensed Guide for the Security Operations Team and Threat Hunter. Independently published, 2019. 258 с.

9. Shukla S. K., Negi R., Handa A. Implementing Enterprise Cybersecurity with Open-Source Software and Standard Architecture. River Publishers, 2021. 300 с.

10. Kovacevic B. Security Orchestration, Automation, and Response for Security Analysts. Packt, 2023. 338 с.

11. Chuvakin A. SOCstock 2021 The Cloud-native SOC [Електронний ресурс] / Anton Chuvakin. – 2021. – Режим доступу до ресурсу: https://www.slideshare.net/anton_chuvakin/the-cloudnative-soc.

12. State of Cloud Native Application Security [Електронний ресурс] // Snyk. – 2023. – Режим доступу до ресурсу: https://snyk.io/reports/state-of-cloud-native-application-security/.

13. 2023 Cloud Risk Report [Електронний ресурс] // CrowdStrike. – 2023. – Режим доступу до ресурсу: https://go.crowdstrike.com/rs/281-OBQ-266/images/CrowdStrike2023CloudRiskReport.pdf.

14. Understanding the 4Cs of Cloud Native Security: Code, Container, Cluster, and Cloud [Електронний ресурс] // Team Cloud4C. – 2023. – Режим доступу до ресурсу: https://www.cloud4c.com/blogs/4cs-of-cloud-native-security-blog.

15. Lakshmi Narayanan Kaliyaperumal. The Evolution of Security Operations and Strategies for Building an Effective SOC [Електронний ресурс] / Lakshmi Narayanan Kaliyaperumal // ISACA. – 2021. – Режим доступу до ресурсу: https://www.isaca.org/resources/isaca-journal/issues/2021/volume-5/the-evolution-of-security-operations-and-strategies-for-building-an-effective-soc.

16. Elastic Stack Meet the search platform that helps you search, solve, and succeed [Електронний ресурс] // Elastic. – 2023. – Режим доступу до ресурсу: https://www.elastic.co/elastic-stack.

17. TheHive technical documentation [Електронний ресурс] // StrangeBee. – 2023. – Режим доступу до ресурсу: https://docs.strangebee.com/thehive/setup/.

18. Threat matrix for Kubernetes [Електронний ресурс] // Microsoft. – 2020. – Режим доступу до ресурсу: https://www.microsoft.com/en-us/security/blog/2020/04/02/attack-matrix-kubernetes/.

19. Falco. Getting Started [Електронний ресурс] // Falcosecurity. – 2023. – Режим доступу до ресурсу: https://falco.org/docs/getting-started/.

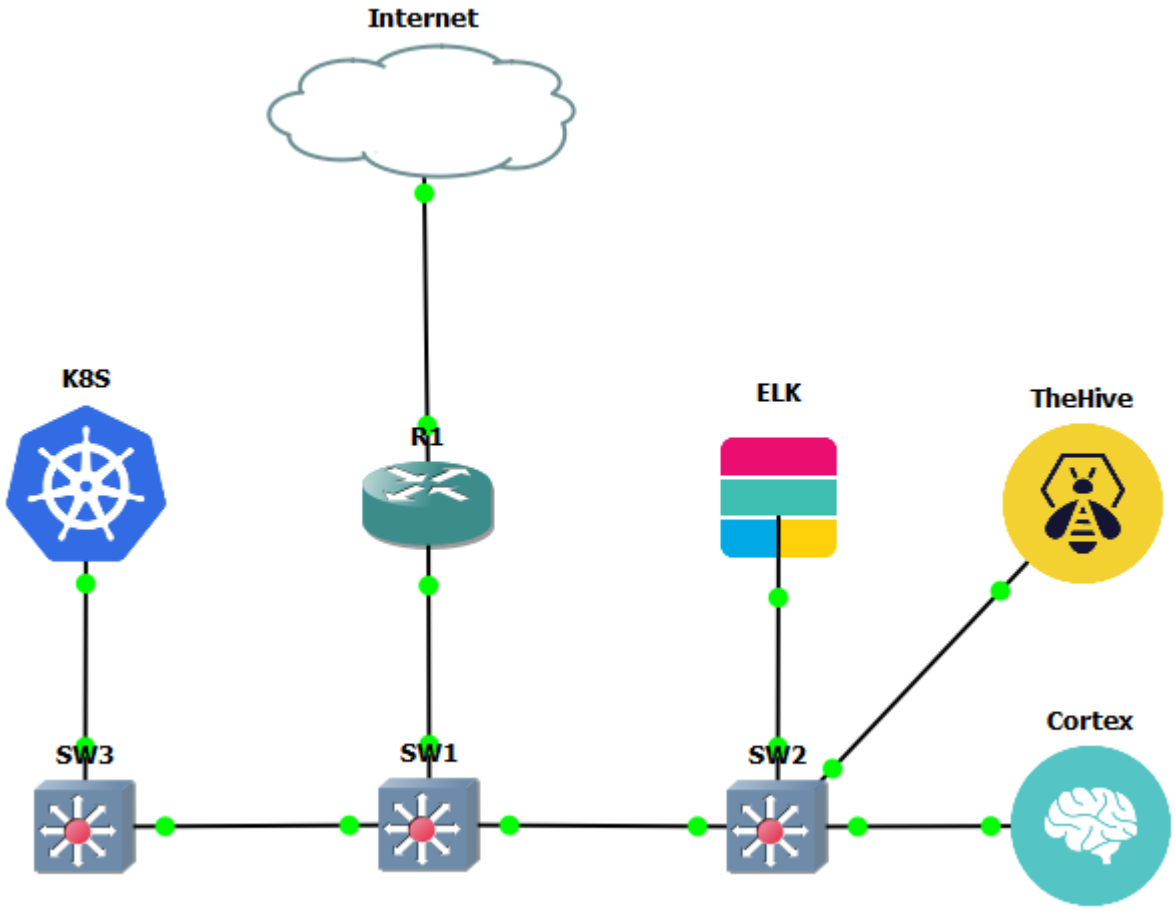
20. jertel. ElastAlert 2 [Електронний ресурс] / jertel. – 2023. – Режим доступу до ресурсу: https://elastalert2.readthedocs.io/en/latest/running_elastalert.html.

21. MITRE ATT&CK framework for container runtime security with Falco. [Електронний ресурс] // Falcosecurity. – 2019. – Режим доступу до ресурсу: https://sysdig.com/blog/mitre-attck-framework-for-container-runtime-security-with-sysdig-falco/.

ДОДАТКИ

Додаток А

Топологічна схема мережі cloud-native середовища



Додаток Б

Мінімальні системні вимоги ОС Ubuntu Server

ОЗП: 1+ ГБ

Місце на диску: 2.5 ГБ

ЦП: 1+ ГГц

Клавіатура (чи інші сумісні вказуючі пристрої)

Мінімальні системні вимоги рішення ELK Stack 8.11

ОЗП: 4-8 ГБ

Місце на диску: 50-100 ГБ

ЦП: 1+ ядра

Платформа: RHEL/CentOS 7/8, Oracle Linux 6, Windows Server 2016/2019/2022, Debian 10/11, Ubuntu 18/20/22.04, Docker, Kubernetes

Мінімальні системні вимоги рішення The Hive 5

ОЗП: 6+ ГБ

Місце на диску: 50-80 ГБ

ЦП: 2+ ядер

Платформа: Ubuntu 20.04 LTS, Debian 11, RHEL 8, Fedora 35, Docker, Kubernetes

Мінімальні системні вимоги рішення Cortex

ОЗП: 6+ ГБ

Місце на диску: 40-60 ГБ

ЦП: 2+ ядер

Платформа: Ubuntu 20.04 LTS, Debian 11, RHEL 8, Fedora 35, Docker, Kubernetes

Мінімальні системні вимоги рішення Minikube

ОЗП: 2+ ГБ

Місце на диску: 20+ ГБ

ЦП: 2+ ядер

Платформа: Docker / Virtual Machine Manager (KVM, Hyper-V, Oracle Virtualbox, VMWare)

Додаток В

Файл поточної конфігурації маршрутизатора R1:

```
! Last configuration change at
13:56:17 EET Sun Dec 24 2023
! NVRAM config last updated at
13:59:43 EET Sun Dec 24 2023
!
version 15.5
service timestamps debug datetime msec
service timestamps log datetime
no service password-encryption
!
hostname R1
!
boot-start-marker
boot-end-marker
!
no aaa new-model
!
bsd-client server url
https://cloudsso.cisco.com/as/to-
ken.oauth2
clock timezone EET 2 0
clock summer-time EEST recurring
mmi polling-interval 60
no mmi auto-configure
no mmi pvc
mmi snmp-timeout 180
```

```
!
no ip icmp rate-limit unreachable
!
ip domain list raccoon.net
ip domain list soc.raccoon.net
ip domain list dev.raccoon.net
ip domain name raccoon.net
ip host r1.raccoon.net 192.168.44.62
ip host sw1.raccoon.net 192.168.44.11
ip host sw2.raccoon.net 192.168.44.12
ip host elk.soc.raccoon.net
192.168.44.65
ip host sw3.raccoon.net 192.168.44.13
ip host k8s.dev.raccoon.net
192.168.44.129
ip host thehive.soc.raccoon.net
192.168.44.66
ip name-server 192.168.1.1
ip name-server 8.8.8.8
ip cef
login on-failure log
login on-success log
no ipv6 cef
!
multilink bundle-name authenticated
!
cts logging verbose
!
username admin privilege 15 secret 8
$8$y/oE17PALRryb1$k9GuY6OWKmU.IwMWutQ6
GV77keGmhFN5UkLAiDK1IhY
!
redundancy
!
ip tcp synwait-time 5
ip ssh version 2
!
interface Ethernet0/0
 ip address dhcp
 ip flow ingress
 ip flow egress
!
interface Ethernet0/1
 no ip address
!
interface Ethernet0/1.20
 encapsulation dot1Q 20
 ip address 192.168.44.62
255.255.255.192
!
interface Ethernet0/1.30
 encapsulation dot1Q 30
 ip address 192.168.44.126
255.255.255.192
!
interface Ethernet0/1.40
 encapsulation dot1Q 40
 ip address 192.168.44.190
255.255.255.192
!
interface Ethernet0/2
 no ip address
 shutdown
!
interface Ethernet0/3
 no ip address
 shutdown
!
interface Ethernet1/0
 no ip address
 shutdown
!
interface Ethernet1/1
 no ip address
 shutdown
!
interface Ethernet1/2
 no ip address
 shutdown
!
interface Ethernet1/3
 no ip address
 shutdown
!
interface Serial2/0
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/1
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/2
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial2/3
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/0
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/1
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/2
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial3/3
 no ip address
 shutdown
 serial restart-delay 0
!
ip forward-protocol nd
!
ip flow-export source Ethernet0/1.30
ip flow-export version 9
```

```
ip flow-export template options ex-
port-stats
ip flow-export template options
timeout-rate 120
ip flow-export template options re-
fresh-rate 25
ip flow-export template timeout-rate
90
ip flow-export template refresh-rate
15
ip flow-export interface-names
ip flow-export destination
192.168.44.65 2055
!
no ip http server
no ip http secure-server
ip dns server
ip dns primary soc.raccoon.net soa
r1.raccoon.net mail.raccoon.net 21600
900 7776000 86400
ip dns primary dev.raccoon.net soa
r1.raccoon.net mail.raccoon.net 21600
900 7776000 86400
ip dns primary raccoon.net soa r1.rac-
coon.net mail.raccoon.net 21600 900
7776000 86400
ip route 0.0.0.0 0.0.0.0 192.168.1.1
!
logging trap notifications
logging host 192.168.44.65 transport
udp port 9002
!
control-plane
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 login local
 transport input ssh
!
ntp master 5
ntp update-calendar
ntp server pool.ntp.org
!
end
```

Файл поточної конфігурації комутатора SW1:

```
!
! Last configuration change at
22:36:48 EET Sat Dec 23 2023
! NVRAM config last updated at
22:39:05 EET Sat Dec 23 2023
!
version 15.2
service timestamps debug datetime msec
service timestamps log datetime
no service password-encryption
service compress-config
!
hostname SW1
!
boot-start-marker
boot-end-marker
!
logging discriminator EXCESS severity
drops 6 msg-body drops EXCESSCOLL
logging buffered 50000
logging console discriminator EXCESS
!
username admin privilege 15 secret 8
$8$fBu6SrFg3dV97a$nfkKLzWQRu-
RANszInBTF7nGPErI7MopU6GrxokGCtwI
no aaa new-model
clock timezone EET 2 0
clock summer-time EEST recurring
!
no ip routing
no ip icmp rate-limit unreachable
!
ip domain-name raccoon.net
ip name-server 192.168.44.62
no ip cef
no ipv6 cef
!
spanning-tree mode rapid-pvst
spanning-tree extend system-id
!
vlan internal allocation policy as-
cending
!
ip tcp synwait-time 5
!
interface Ethernet0/0
 switchport trunk allowed vlan
20,30,40
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport nonegotiate
!
interface Ethernet0/1
 switchport trunk allowed vlan
20,30,40
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport nonegotiate
!
interface Ethernet0/2
 switchport trunk allowed vlan
20,30,40
 switchport trunk encapsulation dot1q
```

```
 switchport mode trunk
 switchport nonegotiate
!
interface Ethernet0/3
!
interface Ethernet1/0
!
interface Ethernet1/1
!
interface Ethernet1/2
!
interface Ethernet1/3
!
interface Ethernet2/0
!
interface Ethernet2/1
!
interface Ethernet2/2
!
interface Ethernet2/3
!
interface Ethernet3/0
!
interface Ethernet3/1
!
interface Ethernet3/2
!
interface Ethernet3/3
!
interface Vlan1
 no ip address
 no ip route-cache
 shutdown
!
interface Vlan20
 ip address 192.168.44.11
255.255.255.192
 no ip route-cache
!
ip default-gateway 192.168.44.62
ip forward-protocol nd
!
no ip http server
no ip http secure-server
!
logging trap notifications
logging host 192.168.44.65 transport
udp port 9002
!
control-plane
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 exec-timeout 20 0
 logging synchronous
 login local
!
ntp server 192.168.44.62
ntp server 0.ua.pool.ntp.org
!
End
```

Файл поточної конфігурації комутатора SW2

```
!


! Last configuration change at
01:13:37 EET Thu Dec 21 2023
! NVRAM config last updated at
02:09:09 EET Thu Dec 21 2023
!
version 15.2
service timestamps debug datetime msec
service timestamps log datetime
no service password-encryption
service compress-config
!
hostname SW2
!
boot-start-marker
boot-end-marker
!
logging discriminator EXCESS severity
drops 6 msg-body drops EXCESSCOLL
logging buffered 50000
logging console discriminator EXCESS
!
no aaa new-model
clock timezone EET 2 0
clock summer-time EEST recurring
!
no ip routing
no ip icmp rate-limit unreachable
!
ip name-server 192.168.44.62
no ip cef
no ipv6 cef
!
spanning-tree mode rapid-pvst
spanning-tree extend system-id
!
vlan internal allocation policy as-
cending
```

```
!
ip tcp synwait-time 5
!
interface Ethernet0/0
 switchport trunk allowed vlan
20,30,40
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport nonegotiate
!
interface Ethernet0/1
 switchport access vlan 30
 switchport mode access
!
interface Ethernet0/2
 switchport access vlan 30
 switchport mode access
!
interface Ethernet0/3
 switchport access vlan 30
 switchport mode access
!
interface Ethernet1/0
 switchport access vlan 30
 switchport mode access
!
interface Ethernet1/1
 switchport access vlan 30
 switchport mode access
!
interface Ethernet1/2
 switchport access vlan 30
 switchport mode access
!
interface Ethernet1/3
 switchport access vlan 30
 switchport mode access
!
interface Ethernet2/0
 switchport access vlan 30
 switchport mode access
!
interface Ethernet2/1
 switchport access vlan 30
 switchport mode access
!
interface Ethernet2/2
 switchport access vlan 30
 switchport mode access
!
interface Ethernet2/3
 switchport access vlan 30
 switchport mode access
!
interface Ethernet3/0
!
interface Ethernet3/1
!
interface Ethernet3/2
!
interface Ethernet3/3
!
interface Vlan1
 no ip address
 no ip route-cache
 shutdown
!
interface Vlan20
 ip address 192.168.44.12
255.255.255.192
 no ip route-cache
!
ip default-gateway 192.168.44.62
ip forward-protocol nd
!
no ip http server
no ip http secure-server
!
logging trap notifications
logging host 192.168.44.65 transport
udp port 9002
!
control-plane
!
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 login
!
ntp server 192.168.44.62 prefer
!
end
```

Файл поточної конфігурації комутатора SW3:

```
!
! Last configuration change at
23:49:41 EET Sat Dec 23 2023
!
version 15.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
service compress-config
```

```
!
hostname SW3
!
boot-start-marker
boot-end-marker
!
logging discriminator EXCESS severity
drops 6 msg-body drops EXCESSCOLL
logging buffered 50000
logging console discriminator EXCESS
!
no aaa new-model
clock timezone EET 2 0
clock summer-time EEST recurring
!
no ip routing
no ip icmp rate-limit unreachable
!
ip name-server 192.168.44.62
no ip cef
no ipv6 cef
!
spanning-tree mode rapid-pvst
spanning-tree extend system-id
!
vlan internal allocation policy as-
cending
!
ip tcp synwait-time 5
!
interface Ethernet0/0
 switchport trunk allowed vlan
20,30,40
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport nonegotiate
!
interface Ethernet0/1
 switchport access vlan 40
 switchport mode access
!
interface Ethernet0/2
 switchport access vlan 40
 switchport mode access
!
interface Ethernet0/3
 switchport access vlan 40
 switchport mode access
!
interface Ethernet1/0
 switchport access vlan 40
 switchport mode access
!
interface Ethernet1/1
 switchport access vlan 40
 switchport mode access
!
interface Ethernet1/2
 switchport access vlan 40
 switchport mode access
!
interface Ethernet1/3
 switchport access vlan 40
 switchport mode access
!
interface Ethernet2/0
 switchport access vlan 40
 switchport mode access
!
interface Ethernet2/1
 switchport access vlan 40
 switchport mode access
!
interface Ethernet2/2
 switchport access vlan 40
 switchport mode access
!
interface Ethernet2/3
 switchport access vlan 40
 switchport mode access
!
interface Ethernet3/0
!
interface Ethernet3/1
!
interface Ethernet3/2
!
interface Ethernet3/3
!
interface Vlan1
 no ip address
 no ip route-cache
 shutdown
!
interface Vlan20
 ip address 192.168.44.13
255.255.255.192
 no ip route-cache
!
ip default-gateway 192.168.44.62
ip forward-protocol nd
!
no ip http server
no ip http secure-server
!
control-plane
!
line con 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line aux 0
 exec-timeout 0 0
 privilege level 15
 logging synchronous
line vty 0 4
 login
!
ntp peer 192.168.44.11
ntp peer 192.168.44.12
ntp server 192.168.44.62
ntp server 0.ua.pool.ntp.org
!
end
```