МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут автоматики і електротехніки

Кафедра комп'ютерних технологій та інформаційної безпеки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних технологій та інформаційної безпеки, к.т.н., доцент
_____ С.М. Нужний
«19» 12 2024 р.

Кваліфікаційна робота
на правах рукопису

КВАЛІФІКАЦІЙНА РОБОТА

Розгортання та налаштування Cyber ICE Box Platform для проведення CTF змагань

Ступінь вищої освіти: магістр
Галузь знань: 14 Електрична інженерія
Спеціальність: 141 «Електроенергетика, електротехніка та електромеханіка»
Освітньо-професійна програма: Системи технічного захисту інформації, автоматизація її обробки

Виконав: студент 6 курсу групи 6366м
Момченко Олександр Володимирович _____

Кваліфікаційна робота містить результати самостійного виконання навчального завдання. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Керівник:
к.т.н., доцент кафедри КТІБ

Сірівчук Андрій Сергійович _____

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут автоматики і електротехніки

ЗАТВЕРДЖУЮ
Гарант освітньої програми, к.т.н., доцент, доцент кафедри КТІБ
_____ Турти М.В.
«_19_» ____12____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу

Студент: Момченко Олександр Володимирович

Курс:6

Група: 6366м

Ступінь вищої освіти: магістр

Галузь знань: 14 Електрична інженерія

Спеціальність: 141 «Електроенергетика, електротехніка та електромеханіка»

Освітньо-професійна програма: Системи технічного захисту інформації, автоматизація її обробки

1. Тема роботи: Розгортання та налаштування Cyber ICE Box Platform для проведення CTF змагань

затверджена наказом НУК від «_14__» жовтня 2024 р. № 1075 - уч

2. Вихідні дані до роботи: система для проведення CTF змагань – Cyber ICE Box Platform; основа для розгортання – Kubernetes; тестові завдання за напрямками – веб-захист, криптографія та стненографія, інформаційні технології.

визначити роль CTF змагань в Україні; провести аналіз платформ для проведення CTF; визначити особливості роботи Cyber ICE Box Platform, розгорнути Cyber ICE Box Platform; створити тестові завдання для проведення CTF.

4. Терміни виконання завдання:
термін видачі завдання: «_04_» вересня 2024 р.
термін подання роботи: «_19_» грудня 2024 р.

6. План-графік виконання кваліфікаційної роботи

№ п/п	Заходи	Дата		Готовність	Відмітка про виконання
		Навч. тиждень	Контрольна дата		
1.	Наукове стажування	1 - 8	25.10.2024	Звіт з наукового стажування	
2.	Організаційні збори	9	31.10.2024	Попередній варіант змісту КР	
3.	Рубіжний контроль	10	05.11.2024	Попередній варіант оглядових розділів	
4.	Рубіжний контроль	13	28-29.11.2024	1. Попередній варіант повної КР. 2. Попередній варіант презентації.	
5.	Рубіжний контроль	15	12.12.2024	Доповідь на 12-ій Всеукраїнській науково-технічній конференції з міжнародною участю «СПІБТ–2024»	
6.	Перевірка на плагіат	15	12-13.12.2024	Електронний варіант кваліфікаційної роботи	
7.	КЕ на рівень « магістра»	16	17-18.12.2024	Здавання державного екзамену зі спеціальності на ступінь бакалавра	
8.	Кафедральний захист	16	19.12.2024	1. Оформлена КР (в форматі pdf) (передається студентом на кафедрі для внутрішньої рецензії). У разі отримання позитивної внутрішньої рецензії, КР подається на зовнішню рецензію. 2. Оформлена презентація (в форматі pdf).	
9.	Подання документів на захист	16	20.12.2024	1. Подання щодо захисту КР: тема, успішність, висновок керівника та висновок кафедри про КР. 2. Зовнішня рецензія (окремо від КР). Резюме на КР. 3. Електронний варіант та роздрукована презентація в кількості 5 примірників. 4. Електронний варіант КР на диску	
10.	Захист ДР	17	24,26,27, 28.12.2024	1. Приєднання студента до засідання кваліфікаційної комісії (конференції) у встановлений час для проведення процедури дистанційного захисту 2. Процедура захисту КР.	

ПРИМІТКА: Завдання додається до виконаної роботи та подається до атестаційної комісії.

Керівник
К.т.н., доцент _____ А.С. Сірівчук

Студент _____ О.В. Момченко

АНОТАЦІЯ

Момченко О. В. Розгортання та налаштування *Cyber Ice Box platform* для проведення *CTF*-змагань

Кваліфікаційна робота на здобуття ступеня вищої освіти «магістр» за спеціальністю 141 «Електроенергетика, електротехніка та електромеханіка» галузі знань 14 «Електрична інженерія». – Національний університет кораблебудування імені адмірала Макарова, Миколаїв, 2024.

Пояснювальна записка: 66 с., 15 рис., 1 табл., 36 посилань.

Дана кваліфікаційна робота присвячена створенню сучасної платформи для проведення *CTF*-змагань із кібербезпеки з використанням *Cyber ICE Box Platform*. Основна мета полягає у впровадженні контейнерних технологій та оркестрації Kubernetes для забезпечення масштабованості, безпеки та зручності налаштувань. Робота охоплює аналіз принципів роботи контейнерних програм, дослідження інфраструктури *Cyber ICE Box* та тестування функціональності платформи через реальні сценарії.

Висновки та рекомендації можуть бути корисними для організаторів змагань у сфері кібербезпеки, університетів та приватних компаній, які прагнуть розвивати навички фахівців через навчальні симуляції.

Ключові слова: *CTF*, змагання, контейнеризація, *Kubernetes*, безпека.

ANNOTATION

Momchenko O. V. Deployment and configuration of the Cyber Ice Box platform for ctf competitions

Qualification work for the degree of higher education “Master” in specialty 141 “Electric power engineering, electrical engineering and electromechanics”, field of knowledge 14 “Electrical engineering.” - Admiral Makarov National University of Shipbuilding, Mykolaiv, 2024.

Explanatory note: 66 p., 15 figures, 1 table, 36 references.

This qualification work is devoted to the creation of a modern platform for conducting CTF competitions in cybersecurity using the Cyber ICE Box Platform. The main goal is to implement container technologies and Kubernetes orchestration to ensure scalability, security and ease of configuration. The work includes analyzing the principles of containerized applications, researching the Cyber ICE Box infrastructure, and testing the platform's functionality through real-world scenarios.

The conclusions and recommendations can be useful for organizers of cybersecurity competitions, universities, and private companies seeking to develop the skills of specialists through training simulations.

Keywords: CTF competition, containerization, Kubernetes, security.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

CTF - Capture The Flag

IT - Information Technology

КПІ - Київський політехнічний інститут

РНБО - Рада національної безпеки і оборони

RBAC - Role-Based Access Control

PSP - Pod Security Policies

UnionFS - Union File System

NAT - Network Address Translation

CRD - Custom Resource Definitions

ELK - Elasticsearch, Logstash, Kibana

RSA - Rivest, Shamir, Adleman

CI - Continuous Integration

CD - Continuous Delivery

ОС - Операційна Система

API - Application Programming Interface

IP – Internet Protocol

IPC - Inter-Process Communication

UTS - UNIX Time-Sharing

Cgroups - Control Groups

CPU - Central Processing Unit

HPA - Horizontal Pod Autoscaling

AES - Advanced Encryption Standard

PV - Persistent Volumes

CSRF - Cross-Site Request Forgery

SQL - Structured Query Language

VPA - Vertical Pod Autoscaling

HTTPS - HyperText Transfer Protocol Secure

OAST - Out-of-band Application Security Testing

PHP - Hypertext Preprocessor

ASCII - American Standard Code for Information Interchange

LSB - Least Significant Bit

ВСТУП

Останні роки значно зросла увага до проблем кібербезпеки у всьому світі, і Україна не є винятком. Постійний розвиток технологій та інформаційних систем супроводжується збільшенням кількості кіберзагроз, що створює виклики для фахівців у цій галузі. В умовах постійного ризику та необхідності захисту критичної інфраструктури, проведення змагань з кібербезпеки, зокрема *CTF* (*Capture The Flag*), набуло великого значення як метод навчання та оцінки навичок кіберзахисту.

CTF змагання є інструментом, який дозволяє учасникам відпрацьовувати свої знання та навички у реальних сценаріях кіберзагроз. Вони моделюють ситуації з проникненням в інформаційні системи або їх захистом, тим самим сприяючи розвитку практичних навичок. Учасники можуть вирішувати завдання з криптографії, мережевої безпеки, аналізу вразливостей, зворотного інжинірингу та інших напрямків кібербезпеки.

Розвиток таких платформ для проведення *CTF* змагань, як *Cyber Ice Box Platform*, є важливим кроком у вдосконаленні процесу навчання спеціалістів з кібербезпеки. *Cyber Ice Box Platform* забезпечує гнучкість, масштабованість та інтеграцію з сучасними технологіями, такими як контейнеризація та *Kubernetes*. Ця платформа дозволяє проводити змагання на будь-якому рівні складності, як для початківців, так і для професіоналів, завдяки зручності налаштувань та можливості створення реалістичних сценаріїв атак та захисту.

У цій дипломній роботі буде розглянуто процес розгортання та налаштування платформи *Cyber Ice Box* для проведення *CTF* змагань. Дослідження охоплює аналіз принципів роботи контейнерних програм, а також використання *Kubernetes* як базової інфраструктури для забезпечення стабільної та безпечної роботи платформи.

Об’єктом магістерської роботи є ознайомлення з можливостями та засобами встановлення власних платформ для *CTF* змагань.

Предметом дослідження є розгортання платформи *Cyber Ice Box* для створення власних систем перевірки знань.

Метою роботи є розгортання та створення тестових завдань на платформі *Cyber Ice Box*.

1. ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ CTF ЗМАГАННЬ

1.1 Поняття *CTF* та їх види

Capture The Flag (*CTF*) — це змагання з кібербезпеки, що імітують реальні сценарії кіберзагроз і дозволяють учасникам розвивати свої навички у виявленні вразливостей, експлуатації та захисті систем. Назва "*Capture The Flag*" (з англ. "захопи прапор") походить від традиційної мети таких змагань — знайти прихований цифровий "прапор" (*flag*), який зазвичай являє собою рядок тексту або код, що підтверджує виконання завдання.

Основні завдання *CTF* змагань включають:

- Навчання учасників реальним кібербезпековим навичкам у різних галузях, таких як криптографія, мережевий захист, зворотний інжиніринг і веб-безпека.
- Оцінка швидкості реакції та здібності до аналізу у випадках кіберзагроз.
- Спрощення процесу навчання та взаємодії між фахівцями в галузі кібербезпеки.

CTF змагання проводяться як серед професіоналів, так і серед студентів навчальних закладів або ентузіастів, що лише починають свій шлях у галузі кібербезпеки. Завдяки гнучкому підходу, *CTF* можуть бути налаштовані для різних рівнів складності, що робить їх популярним форматом як для навчання, так і для змагань.

Види *CTF* змагань

CTF змагання поділяються на кілька основних видів залежно від їхнього формату, складності та напрямків завдань. Найбільш поширеними є наступні формати:

1. *Jeopardy-style CTF*

Jeopardy-style CTF — це класичний формат змагань, що складається з набору окремих завдань, згрупованих за категоріями. Учасники можуть вибирати будь-

яке завдання в довільному порядку, і за кожне вирішене завдання нараховуються бали. Завдання у цьому форматі можуть охоплювати різні аспекти кібербезпеки, такі як:

- Криптографія: завдання, пов'язані з розшифруванням зашифрованих повідомлень або створенням власних криптографічних алгоритмів. Учасники повинні використовувати свої знання про різні криптографічні методи, такі як шифри Цезаря, *RSA*, *AES*, або навіть сучасні криптографічні протоколи.
- Зворотний інжиніринг: тут учасники аналізують бінарні файли або програми для того, щоб з'ясувати, як вони працюють, і знайти приховані функції або вразливості. Зворотний інжиніринг допомагає учасникам зрозуміти механіку програм і викрити можливі загрози.
- Експлуатація вразливостей: ці завдання потребують від учасників використання вразливостей програмного забезпечення або операційних систем для отримання несанкціонованого доступу або виконання певних дій. Це може включати роботу з переповненням буфера, уразливістю в пам'яті або вразливостями веб-додатків.
- Форензіка: тут учасники аналізують дані для знаходження цифрових слідів, що вказують на те, що сталося з системою чи файлом. Форензіка може охоплювати аналіз мережевих трафіків, логів операційних систем, файлів з пам'яті.
- Веб-безпека: завдання у цій категорії пов'язані з пошуком і експлуатацією вразливостей у веб-додатках, таких як *SQL*-ін'єкції, *XSS* (міжсайтові скрипти), *CSRF* (міжсайтові підроблені запити) та інші.
- Програмування: тут учасники вирішують завдання шляхом написання або модифікації коду, щоб знайти рішення проблеми, пов'язаної з безпекою.

Кожне завдання у форматі *Jeopardy-style CTF* має певну кількість балів залежно від його складності. Учасники вибирають завдання зі списку, вирішують їх та надсилають "прапор", щоб отримати бали. Виграє команда, яка набрала найбільшу кількість балів до кінця змагань. Цей формат підходить для

початківців і досвідчених гравців, оскільки завдання можуть бути як простими, так і дуже складними.

2. *Attack-Defense CTF*

Attack-Defense CTF — це більш складний і реалістичний формат, у якому учасники одночасно виконують ролі атакуючих і захисників. Кожна команда отримує власну систему або мережеву інфраструктуру, яку необхідно захищати від атак інших команд, водночас намагаючись проникнути в системи суперників.

Змагання цього типу моделюють реальні сценарії кіберконфлікту, де важливо не лише вміти атакувати, але й ефективно захищатися. Учасники повинні знаходити та виправляти вразливості в своїй інфраструктурі, а також експлуатувати вразливості інших команд, щоб отримати їхні прапори. За правильну атаку або успішний захист команда отримує бали.

Основні етапи змагань *Attack-Defense*:

- Захист: команди аналізують і виправляють вразливості власних серверів, намагаючись закрити доступ до критичних систем або послуг від атак супротивників.
- Атака: учасники аналізують системи суперників, намагаючись знайти та експлуатувати уразливості для отримання доступу до систем або перехоплення даних.
- Оцінка результатів: зазвичай, команди отримують бали за успішні атаки на суперників і за те, як довго вони змогли зберегти свої системи захищеними.

Такий формат вимагає від учасників не тільки технічних знань, але й умінь швидко реагувати на атаки, забезпечувати стійкість систем до зламів і підтримувати працездатність інфраструктури в умовах високого навантаження.

3. *Mixed CTF* (гібридний формат)

Mixed CTF поєднує елементи *Jeopardy-style* та *Attack-Defense* форматів. Спочатку учасники вирішують завдання з *Jeopardy-style* частини, що допомагає їм підготуватися до наступної фази змагань — *Attack-Defense*. Такий підхід

дозволяє створити комплексні змагання, які не тільки оцінюють технічні навички учасників, але й перевіряють їх здатність до атак та захисту в реальних умовах. Цей формат часто використовується для великих змагань з кібербезпеки, де метою є створення найбільш реалістичного середовища, що максимально наближене до реальних сценаріїв кіберзагроз.

4. *Boot2Root CTF*

Boot2Root – це формат змагань, де учасники повинні пройти серію завдань для отримання привілеїв "*root*" у системі, зазвичай у вигляді віртуальної машини з попередньо налаштованими вразливостями. Завдання часто починаються з отримання мінімальних привілеїв у системі та поступового отримання контролю над усією інфраструктурою.

Кроки, що часто виконуються у *Boot2Root*:

- Первинний доступ: учасники знаходять початкову уразливість для отримання доступу до системи.
- Привілеї: після отримання початкового доступу учасники повинні знайти спосіб підвищити свої привілеї, отримати доступ до інших користувачів або до адміністративних функцій системи.
- Повний контроль: кінцевою метою є отримання повного root-доступу до системи.

5. Стеганографічні та криптографічні *CTF*

Стеганографія та криптографія є важливими напрямками у сфері кібербезпеки, і тому часто використовуються як окремі категорії завдань на *CTF* змаганнях. У стеганографічних завданнях учасники мають знаходити приховану інформацію всередині файлів (зображень, аудіо, відео). Криптографічні *CTF* завдання зосереджені на розшифруванні даних, зломі алгоритмів шифрування або проведенні атак на криптографічні системи.

CTF змагання надають широкі можливості для розвитку навичок у сфері кібербезпеки, незалежно від рівня підготовки учасників. Кожен формат *CTF* — від *Jeopardy-style* до *Attack-Defense* — орієнтований на певні аспекти інформаційної безпеки, дозволяючи учасникам підвищувати як технічні, так і

стратегічні вміння. Такі змагання допомагають спеціалістам залишатися на передовій кібербезпеки, оскільки забезпечують реалістичне середовище для навчання та вдосконалення.

1.2 Роль *CTF* змагань в Україні

Capture The Flag (*CTF*) змагання в Україні відіграють критично важливу роль у формуванні кібербезпекової спільноти, розвитку освітніх ініціатив та зміцненні національної кібероборони. У зв'язку зі стрімким зростанням кібератак на державний та приватний сектори, а також на інфраструктурні об'єкти, змагання *CTF* стали ефективним інструментом не тільки для навчання та вдосконалення фахівців, але й для підвищення загальної обізнаності про актуальні кіберзагрози. Участь України у глобальних *CTF* змаганнях та організація національних подій сприяє підвищенню кваліфікації фахівців і розробці передових методів кіберзахисту.

Останніми роками змагання *CTF* стали інтегрованою частиною навчальних програм у вищих навчальних закладах України, що спеціалізуються на *IT* та кібербезпеці. Від університетів до професійних технічних закладів, *CTF* змагання використовуються як практичний метод навчання студентів, що дозволяє їм отримати реальні навички, необхідні для роботи у сфері кібербезпеки.

Українські вищі навчальні заклади, такі як Київський політехнічний інститут (*КПІ*), Львівська політехніка, Харківський національний університет радіоелектроніки, а також інші технічні вузи активно впроваджують *CTF* у навчальні програми. Студенти мають можливість брати участь у локальних змаганнях, а також готуватися до міжнародних турнірів. Учасники таких змагань здобувають знання у різних напрямах: від пошуку вразливостей у веб-додатках до реверс-інжинірингу, криптографії та аналізу безпеки мереж.

Ці змагання також допомагають закласти фундамент для міждисциплінарної співпраці між програмістами, аналітиками,

адміністраторами систем, фахівцями з кібербезпеки та інженерами. Така співпраця дозволяє студентам краще розуміти загрози і підходи до їх вирішення в реальних умовах.

CTF змагання сприяють створенню та зміцненню кібербезпекової спільноти в Україні. Завдяки таким подіям формується мережа фахівців, що обмінюються досвідом та знаннями у сфері кібербезпеки. Спільнота *CTF* гравців включає в себе як новачків, так і досвідчених фахівців, що допомагає створити безперервний процес передачі знань і практичного досвіду.

Змагання надають учасникам можливість зустрічатися з професіоналами галузі, обговорювати нові виклики, вивчати інструменти та методи атак і захисту, а також співпрацювати над вирішенням складних завдань. Багато команд, що формуються для участі в *CTF*, згодом стають відомими і виступають на міжнародних аренах, представляючи Україну на світових турнірах.

Інформаційно-технічні конференції, такі як *UAHACK, HackIT, IT Weekend Ukraine, NoNameCon*, часто включають у свою програму *CTF* змагання, на яких учасники можуть показати свої навички перед потенційними роботодавцями або замовниками. Ці заходи стають платформою для зростання кібербезпекової спільноти і розширюють можливості для обміну досвідом.

У світлі поточних геополітичних викликів та підвищеної загрози з боку кібератак, *CTF* змагання допомагають підвищити загальний рівень кіберсвідомості серед громадян, державних установ і приватного сектору в Україні. Завдяки проведенню таких змагань, широка аудиторія отримує доступ до інформації про потенційні кіберзагрози та способи захисту від них.

Змагання *CTF* також виконують важливу роль у підготовці кадрів для державних кіберструктур, таких як Державна служба спеціального зв’язку та захисту інформації України (Держспецзв'язку) або Національний координаційний центр кібербезпеки при *РНБО*. Навчання в рамках *CTF* дає можливість державним службовцям отримувати практичні навички для захисту національної інфраструктури, критичних об’єктів та урядових систем від кібератак.

У контексті кіберконфлікту, *CTF* змагання допомагають підготувати професіоналів, які можуть оперативно реагувати на загрози, моделювати можливі атаки та здійснювати ефективний захист у реальному часі. Це дозволяє підвищувати готовність країни до нових видів загроз, включаючи хакерські атаки з боку іноземних урядів чи злочинних угруповань.

Україна активно бере участь у міжнародних *CTF* змаганнях, і українські команди вже отримали визнання на світовій арені. Участь у таких турнірах, як *DEF CON CTF*, *Google CTF*, *Hack the Box* та інші, дозволяє українським фахівцям не тільки демонструвати високий рівень професіоналізму, але й постійно вдосконалювати свої навички, беручи участь у нових та складніших викликах.

Українська команда *dcua*, відома своєю активною участю та успіхами на міжнародних змаганнях. Такі досягнення не тільки підвищують престиж українських фахівців на міжнародному рівні, але й залучають нових учасників до кіберспільноти. Участь у міжнародних змаганнях також відкриває можливості для співпраці з іншими країнами та обміну досвідом у боротьбі з кіберзагрозами.

CTF змагання в Україні не тільки популяризують кібербезпеку, але й створюють прямі можливості для учасників вийти на ринок праці. Багато *IT*-компаній в Україні, що працюють у галузі кібербезпеки, організовують або підтримують *CTF* змагання, щоб знайти нові таланти та запросити їх на роботу. Переможці або учасники змагань часто отримують запрошення на співбесіди чи стажування у провідних компаніях.

Для молодих фахівців це можливість не лише навчитися новим технологіям, але й створити портфоліо реальних проєктів, які можна показати потенційним роботодавцям. Крім того, *CTF* змагання є чудовою платформою для нетворкінгу та встановлення професійних контактів, що може значно допомогти у побудові кар'єри.

Завдяки активній участі українських фахівців у міжнародних *CTF* змаганнях, Україна поступово стає частиною глобальної кіберспільноти. Це важливо не тільки з точки зору підвищення кваліфікації, але й для обміну інформацією та координації з іншими країнами у сфері кібербезпеки. Українські

команди активно взаємодіють з фахівцями з усього світу, що дозволяє адаптувати новітні методики та стратегії кіберзахисту.

CTF змагання відіграють важливу роль в Україні, сприяючи розвитку нових кадрів у сфері кібербезпеки, покращенню професійних навичок фахівців та підвищенню обізнаності про сучасні кіберзагрози. Вони не лише допомагають вітчизняним фахівцям підтримувати високий рівень компетенцій, але й сприяють інтеграції України в глобальну кіберспільноту.

1.3 Аналіз платформ для проведення *CTF*

CTF змагання стали популярними в світі кібербезпеки, завдяки своїй здатності навчати учасників основам безпеки, а також розвитку практичних навичок. Для проведення таких змагань використовуються різні платформи, кожна з яких має свої особливості, переваги та недоліки. У цьому розділі ми розглянемо основні платформи, які активно використовуються для організації *CTF*.

CTFd

CTFd є однією з найпопулярніших платформ для проведення *CTF* змагань. Вона дозволяє організаторам легко створювати та налаштовувати завдання, а також стежити за результатами учасників.

Основні характеристики *CTFd*:

- Модульність: *CTFd* підтримує плагіни, що дозволяє організаторам додавати нові функції відповідно до своїх потреб.
- Адаптивний дизайн: Інтерфейс платформи адаптується до різних пристроїв, що забезпечує зручність використання на мобільних телефонах і комп'ютерах.
- Безпека: *CTFd* реалізує сучасні методи безпеки, такі як шифрування паролів, що підвищує надійність системи.

CTFd є відкритим програмним забезпеченням, що дозволяє розгортати платформу на власних серверах та кастомізувати її.

Hack The Box

Hack The Box — це не просто платформа для *CTF* змагань, а й величезна онлайн-спільнота, що пропонує доступ до численних віртуальних машин і завдань.

Основні особливості *Hack The Box*:

- Різноманітність завдань: Платформа пропонує велику кількість завдань різного рівня складності, що дозволяє учасникам обирати відповідні для свого рівня підготовки.
- Тренувальні кімнати: *Hack The Box* має спеціальні тренувальні кімнати, де учасники можуть вдосконалити свої навички перед змаганнями.
- Соціальна складова: Спільнота активно взаємодіє, обмінюючись досвідом і порадами.

Ця платформа також має функцію проведення *CTF* у командному форматі, що робить її особливо корисною для командного навчання та змагань.

PicoCTF

PicoCTF — це платформа, створена для навчання учнів і студентів через *CTF* змагання.

Основні характеристики:

- Інтуїтивно зрозумілий інтерфейс: *PicoCTF* пропонує простий і зрозумілий інтерфейс, що дозволяє новачкам легко орієнтуватися.
- Освітні ресурси: Платформа надає навчальні матеріали, що допомагають учасникам розвивати свої навички.
- Різноманітність завдань: Завдання охоплюють різні аспекти інформаційної безпеки.

PicoCTF допомагає молоді вчитися у веселій та інтерактивній формі.

Root Me

Root Me — це ще одна популярна платформа, яка надає доступ до великої кількості завдань з кібербезпеки.

Основні особливості:

- Велика бібліотека завдань: Платформа містить тисячі завдань різного рівня складності.
- Система рейтингів: Учасники можуть змагатися за найвищі позиції на платформі, що підвищує мотивацію.
- Спільнота: *Root Me* має активну спільноту, що підтримує новачків.

OverTheWire

OverTheWire — це платформа, яка пропонує навчальні курси у формі *CTF*.

Основні характеристики:

- Гра на базі завдань: Кожен рівень представляє окреме завдання, що робить навчання інтерактивним.
- Відкритий доступ: Платформа є безкоштовною, що дозволяє всім охочим отримати доступ до навчальних матеріалів.
- Фокус на основах: *OverTheWire* зосереджується на базових поняттях кібербезпеки.

OverTheWire підходить для новачків та тих, хто хоче повторити основи.

Cyber Ice Box Platform

Cyber Ice Box Platform — це сучасна платформа, спеціально розроблена для організації *CTF* змагань, що використовує контейнерні технології.

Основні характеристики:

- Контейнеризація: Використання *Docker* дозволяє ізолювати завдання та забезпечити безпеку під час змагань. Це дозволяє створювати безпечні середовища для виконання завдань.
- Гнучкість: *Cyber Ice Box* дозволяє організаторам легко налаштовувати середовища і завдання відповідно до своїх потреб, що робить її ідеальною для специфічних змагань.
- Інтуїтивно зрозумілий інтерфейс: Платформа має зручний інтерфейс, що дозволяє учасникам легко орієнтуватися у завданнях.
- Адаптивність: *Cyber Ice Box* може адаптуватися до потреб організаторів, пропонуючи різноманітні формати змагань і типи завдань.

- Безпека: Використання контейнерних технологій забезпечує високий рівень безпеки, оскільки кожне завдання виконується в ізольованому середовищі.

Порівняння платформ для проведення *CTF* змагань допомагає організаторам визначити, яка з них найкраще підходить для їхніх цілей. Ось кілька важливих критеріїв для порівняння в таблиці (табл. 1.1).

Таблиця 1.1 – Порівняльна таблиця платформ

Параметр	***CTFd***	***Hack The Box***	***PicoCTF***	***Root Me***	***OverTheWire***	***Cyber Ice Box Platform***
Тип	Веб-платформа для проведення *CTF*	Онлайн-платформа з віртуальними машинами	Освітня платформа для *CTF*	Онлайн-платформа з великою бібліотекою завдань	Платформа для навчання з *CTF* завданнями	Платформа для організації *CTF* змагань
Користувацький інтерфейс	Інтуїтивно зрозумілий	Інтуїтивно зрозумілий, але з акцентом на віртуальні машини	Простий та зрозумілий	Зрозумілий та зручний	Простий та навчальний	Інтуїтивно зрозумілий, адаптивний
Види завдань	Різноманітні (веб-безпека, реверс-інжиніринг, криптографія тощо)	Різноманітні (включаючи реверс-інжиніринг, експлуатацію вразливостей)	Основи кібербезпеки	Велика бібліотека завдань з різного рівня складності	Основи кібербезпеки, покрокові завдання	Завдання різних типів з використанням контейнерів
Спільнота	Активна, можливість обміну досвідом	Велика та активна, соціальна взаємодія	Підтримка молоді та початківців	Активна спільнота, змагання	Активна, але більше навчальна	Активна спільнота, підтримка організаторів

Продовження таблиці 1.1

Підтримка команд	Так, можливість створення команд	Так, можливість участі у командних змаганнях	Так, з акцентом на навчання	Так, з можливістю участі команд	Ні, індивідуальне навчання	Так, підтримка командних змагань
Безкоштовність	Безкоштовна, відкритий код	Обмежений безкоштовний доступ, платний контент	Безкоштовна	Безкоштовна	Безкоштовна	Зазвичай безкоштовна, залежить від конфігурації
Можливість кастомізації	Висока, підтримка плагінів	Немає можливості кастомізації	Немає	Обмежена	Немає	Висока, можливість налаштування за потребами

Вибір платформи для проведення *CTF* змагань залежить від кількох факторів, зокрема:

- Цілі змагань: Якщо змагання орієнтовані на навчання, такі платформи як *PicoCTF* чи *OverTheWire* можуть бути найкращим вибором. Для професійних змагань підходять *CTFd* та *Hack The Box*.
- Рівень підготовки учасників: Платформи, які пропонують різноманітні завдання, як-от *Hack The Box* або *Root Me*, можуть бути ідеальними для учасників з різним рівнем підготовки.
- Бюджет: Вартість використання платформи може бути критично важливою. Деякі платформи, такі як *Cyber Ice Box*, можуть вимагати ресурсів для розгортання, але в результаті надають більше можливостей кастомізації.
- Спільнота та підтримка: Наявність активної спільноти може допомогти в навчанні та покращенні навичок учасників. Платформи з активними форумами та групами, такі як *Hack The Box* та *Root Me*, можуть бути корисними.

- Технічні можливості: Технології контейнеризації в *Cyber Ice Box Platform* надають додаткові можливості для організації змагань, зокрема, підвищену безпеку та адаптивність.

Вибір платформи для проведення *CTF* змагань є критично важливим етапом організації. Кожна з платформ має свої особливості та переваги, що робить їх більш або менш придатними в залежності від конкретних цілей та потреб. *Cyber Ice Box Platform* з її акцентом на контейнеризацію та безпеку може стати ідеальним рішенням для організаторів, які прагнуть забезпечити високий рівень безпеки та гнучкості під час проведення змагань.

Висновки до розділу

CTF змагань є ефективним інструментом для перевірки знань в сфері кібербезпеки, що підтверджується великою кількістю проведення змагань в Україні, що пдтверджує актуальність даної роботи. Аналіз платформ для таких змагань показує, що перпективним для учбового процесу є платформа *Cyber Ice Box Platform.*

2. АНАЛІЗ *CYBER ICE BOX PLATFORM*

2.1 Загальний опис *Cyber Ice Box Platform*

Cyber Ice Box Platform — це інноваційна платформа, призначена для проведення *CTF* (*Capture the Flag*) змагань із використанням сучасних контейнерних технологій. Вона є результатом розвитку інструментів та технологій, які орієнтовані на підвищення безпеки, ізоляції середовищ виконання, а також гнучкості та масштабованості змагань. Платформа була розроблена з урахуванням потреб організаторів, яким необхідний зручний і безпечний інструмент для проведення змагань різного рівня складності. У даному розділі розглянемо основні аспекти та особливості *Cyber Ice Box Platform*.

Cyber Ice Box Platform створена для того, щоб забезпечити учасникам *CTF* змагань зручний і безпечний доступ до завдань у контейнеризованих середовищах. Основна ідея полягає у тому, щоб забезпечити ізоляцію середовищ, що дозволяє уникнути взаємодії між різними завданнями, підвищуючи безпеку та контроль над виконанням завдань.

Основні цілі платформи:

1. Безпека: Використання контейнерів забезпечує високий рівень ізоляції завдань, що гарантує безпеку під час змагань.
2. Масштабованість: Платформа може підтримувати як невеликі змагання з декількома учасниками, так і великі міжнародні змагання з сотнями чи навіть тисячами команд.
3. Гнучкість: *Cyber Ice Box* дозволяє організаторам легко налаштовувати середовище під конкретні вимоги завдань, забезпечуючи високу кастомізацію.
4. Зручність використання: Інтерфейс платформи спроєктовано таким чином, щоб учасники могли легко орієнтуватися в системі, а організатори — зручно налаштовувати завдання.

Cyber Ice Box Platform базується на сучасних контейнерних технологіях, таких як *Docker* та *Kubernetes*, що дозволяють створювати ізольовані середовища для виконання завдань.

Використання цих технологій забезпечує такі переваги:

1. Ізоляція середовищ: Кожне завдання виконується в окремому контейнері, що дозволяє повністю ізолювати його від інших завдань та системи загалом. Це не тільки підвищує безпеку, але й дозволяє більш гнучко керувати ресурсами.

2. Масштабованість: Використання *Kubernetes* як базової інфраструктури дозволяє масштабувати систему в залежності від кількості учасників і завдань. Організатори можуть додавати чи зменшувати кількість контейнерів у реальному часі, що робить платформу дуже гнучкою і придатною для великих змагань.

3. Автоматизація: Завдяки контейнерним технологіям, *Cyber Ice Box Platform* забезпечує можливість автоматизованого розгортання завдань та управління ними. Це значно знижує навантаження на організаторів та дозволяє зосередитися на інших аспектах змагання.

4. Гнучкість середовищ: Контейнери дозволяють організаторам створювати середовища з різноманітними налаштуваннями та програмним забезпеченням. Це робить можливим створення специфічних завдань, які вимагають нестандартних конфігурацій.

Один з ключових елементів *Cyber Ice Box Platform* — це інтуїтивно зрозумілий інтерфейс, який робить процес участі в *CTF* змаганнях легким і зручним для користувачів.

Інтерфейс побудований так, щоб:

- Учасники могли легко отримувати доступ до завдань та стежити за своїм прогресом.

- Організатори могли легко додавати нові завдання, відстежувати результати учасників і керувати змаганням в режимі реального часу.

Користувацький інтерфейс включає в себе:

- Панель керування для учасників, де можна переглядати доступні завдання, отримувати підказки та здавати рішення.
- Панель для організаторів, яка надає інструменти для управління завданнями, моніторингу стану системи та аналізу продуктивності учасників.

Переваги використання *Cyber Ice Box Platform*:

1. Безпека та ізоляція: Кожне завдання виконується в окремому контейнері, що мінімізує можливість взаємодії з іншими завданнями чи середовищами. Це особливо важливо для захисту від потенційних атак під час змагань.

2. Масштабованість: Завдяки *Kubernetes*, платформа може бути легко масштабована для підтримки великої кількості учасників. Це робить її ідеальною для проведення як локальних, так і міжнародних змагань.

3. Автоматизація процесів: Організаторам не потрібно вручну налаштовувати кожне середовище. Використання автоматизованих процесів розгортання завдань значно полегшує адміністрування змагань.

4. Підтримка різних типів завдань: Платформа підтримує різноманітні типи завдань, включаючи веб-безпеку, реверс-інжиніринг, криптографію та інші напрямки кібербезпеки.

5. Гнучкість налаштувань: Організатори можуть кастомізувати середовища відповідно до специфіки завдань, включаючи налаштування програмного забезпечення, бібліотек, мережевих параметрів тощо.

Cyber Ice Box Platform має вбудовану підтримку командних змагань, що дозволяє учасникам створювати команди та брати участь у змаганнях спільно. Це забезпечує координацію між членами команди і можливість спільного вирішення завдань. Організатори можуть налаштовувати правила змагань таким чином, щоб дозволяти командну роботу або обмежувати її.

Однією з важливих переваг *Cyber Ice Box Platform* є її можливості інтеграції з іншими системами та сервісами. Це дозволяє організаторам використовувати платформу як частину більшого середовища для проведення кібербезпекових тренувань або змагань.

Основні аспекти інтеграції:

- Інтеграція з *CI/CD* системами: Організатори можуть налаштовувати автоматизоване розгортання завдань через системи безперервної інтеграції та доставки (*CI/CD*), такі як *Jenkins* або *GitLab CI*. Це забезпечує швидке оновлення завдань та їхній постійний моніторинг.

- *API* для керування платформою: *Cyber Ice Box* надає *API* для взаємодії з платформою, що дозволяє організаторам автоматизувати управління завданнями, моніторингом учасників, а також інтегрувати платформу з іншими сервісами, такими як системи обліку часу чи зовнішні бази даних.

- Інтеграція з системами обліку результатів: Платформа може бути інтегрована з системами для відстеження прогресу учасників, оцінювання їхніх результатів та відображення лідерборду в режимі реального часу. Це дозволяє створювати динамічний досвід для учасників і глядачів.

Безпека є критичним аспектом для будь-якої платформи *CTF*, і *Cyber Ice Box* забезпечує високий рівень захисту через кілька ключових механізмів:

- Контейнеризація: Контейнери забезпечують повну ізоляцію середовищ, що гарантує, що учасники не можуть взаємодіяти з іншими завданнями чи системними ресурсами за межами своїх контейнерів. Це мінімізує ризик впливу на систему через помилки чи навмисні атаки.

- Система ролей і доступу: *Cyber Ice Box* має вбудовану систему контролю доступу на основі ролей, що дозволяє організаторам призначати різні рівні доступу для учасників, адміністраторів та інших залучених осіб. Це забезпечує розмежування прав доступу до різних частин платформи.

- Моніторинг безпеки: Платформа підтримує інтеграцію з системами моніторингу безпеки, які дозволяють в реальному часі відстежувати потенційні загрози або аномальну активність учасників.

Однією з ключових технічних переваг *Cyber Ice Box* є її здатність масштабуватися відповідно до потреб змагання:

- Автоматичне масштабування: Використання *Kubernetes* дозволяє автоматично масштабувати кількість контейнерів у залежності від кількості

учасників або навантаження на платформу. Це робить платформу придатною для великих заходів із сотнями чи тисячами учасників.

- Резервування та відновлення: Платформа підтримує механізми резервного копіювання та відновлення, що дозволяє швидко відновлювати роботу у разі технічних несправностей. Це забезпечує високу надійність і стабільність під час проведення тривалих змагань.

- Балансування навантаження: *Kubernetes* забезпечує ефективне балансування навантаження між контейнерами, що дозволяє уникнути перевантаження окремих компонентів системи і забезпечити рівномірний розподіл ресурсів.

Cyber Ice Box є відкритою платформою, що дозволяє організаторам здійснювати глибоку кастомізацію відповідно до своїх потреб. Це особливо корисно для проведення змагань із специфічними вимогами або для інтеграції з іншими інструментами.

Основні можливості кастомізації:

- Налаштування середовищ: Організатори можуть налаштовувати кожен контейнер під конкретні вимоги завдань, включаючи встановлення необхідного програмного забезпечення, бібліотек або створення спеціальних конфігурацій.

- Гнучка система завдань: Платформа дозволяє створювати завдання будь-якої складності, з різними видами форматів та варіантами їх вирішення. Це може бути як простий веб-сервіс для тестування вразливостей, так і складні багаторівневі системи для реверс-інжинірингу або криптоаналізу.

- Інтеграція з іншими платформами: Завдяки відкритому *API*, *Cyber Ice Box* може бути інтегрована з іншими системами, такими як інструменти для моніторингу кіберзагроз, аналітики безпеки чи навіть освітніми платформами.

Cyber Ice Box Platform вже активно використовується на різних етапах змагань із кібербезпеки. Вона підходить як для внутрішніх корпоративних тренінгів, так і для міжнародних змагань з кібербезпеки.

Деякі приклади використання включають:

1. Освітні програми: Платформа може бути інтегрована в університетські курси з кібербезпеки, де студенти вирішують завдання у форматі *CTF*, отримуючи практичний досвід.

2. Корпоративні тренінги: Компанії можуть використовувати платформу для проведення внутрішніх тренінгів з кібербезпеки для своїх співробітників. Завдання можуть бути налаштовані для відображення реальних загроз і ситуацій, з якими компанії стикаються на практиці.

3. Міжнародні змагання: Завдяки своїй масштабованості та надійності, *Cyber Ice Box* використовується для проведення великих міжнародних змагань з кібербезпеки, де беруть участь команди з усього світу.

Cyber Ice Box Platform — це потужний інструмент для організації та проведення *CTF* змагань різного рівня складності. Вона пропонує високий рівень безпеки, масштабованість та можливості кастомізації, що робить її ідеальним вибором для організаторів, які прагнуть створити професійне та безпечне середовище для своїх змагань. Використання контейнерних технологій та інтеграція з *Kubernetes* дозволяє досягти високої продуктивності та надійності, що особливо важливо під час проведення великих заходів.

2.2 Аналіз принципів роботи контейнерних програм

Контейнерні програми стали фундаментальною технологією у сучасному розвитку програмного забезпечення та управлінні інфраструктурою. Вони дозволяють ізолювати додатки, їхні залежності та конфігурації в окремих середовищах, що робить можливим запуск багатьох додатків на одній операційній системі без конфліктів між ними. Принципи роботи контейнерних програм базуються на кількох ключових концепціях: ізоляція процесів, контроль ресурсів, віртуалізація файлових систем і мереж, а також керування контейнерними образами. Нижче ми розглянемо основні принципи роботи контейнерних програм.

1. Ізоляція процесів і простори імен (*Namespaces*)

Один із найважливіших принципів роботи контейнерів — це ізоляція процесів за допомогою просторів імен, або *namespaces*. Простори імен — це механізм у ядрах операційних систем (найчастіше *Linux*), що дозволяє ізолювати різні компоненти *ОС* для окремих додатків чи процесів. Завдяки *namespaces* кожен контейнер може мати власне ізольоване середовище, не заважаючи роботі інших контейнерів на тій самій операційній системі.

Основні типи просторів імен, які використовуються в контейнерах:

- *PID namespace*: Ізолює ідентифікатори процесів. Це дозволяє кожному контейнеру мати власну таблицю процесів, тобто процеси всередині контейнера не "бачать" процеси в інших контейнерах чи на хості.
- *Mount namespace*: Ізолює файлові системи. Контейнер має власну кореневу файлову систему, що робить неможливим доступ до файлових систем інших контейнерів чи хост-системи.
- *Network namespace*: Ізолює мережеві інтерфейси та налаштування мережі. Це дозволяє кожному контейнеру мати власні *IP*-адреси, маршрутизацію та правила брандмауера.
- *IPC namespace*: Ізолює міжпроцесні взаємодії, такі як спільна пам'ять, семафори і черги повідомлень.
- *UTS namespace*: Відповідає за ізоляцію імен хостів і доменів, що дозволяє контейнерам мати власні хостнейми.
- *User namespace*: Ізолює ідентифікатори користувачів та груп, дозволяючи запускати процеси з певними правами в контейнері, навіть якщо вони мають різні права на хості.

Ізоляція за допомогою *namespaces* забезпечує, що кожен контейнер працює в умовах окремого середовища, не впливаючи на інші контейнери або хост-систему. Це дозволяє безпечно запускати кілька різних додатків в одній операційній системі без ризику виникнення конфліктів.

2. Контроль ресурсів за допомогою *Cgroups* (*Control Groups*)

Другим ключовим принципом роботи контейнерних програм є контроль ресурсів, що здійснюється через *cgroups* (*Control Groups*). *Cgroups* — це

механізм у *Linux*, який дозволяє обмежувати та контролювати використання ресурсів (таких як *CPU*, пам'ять, мережевий трафік, ввід/вивід з диску) для кожного окремого контейнера. Це важливо для забезпечення стабільної роботи системи, особливо коли на одному сервері працюють кілька контейнерів.

Основні функції *cgroups* включають:

- Лімітування використання ресурсів: Наприклад, можна обмежити кількість пам'яті, доступної для конкретного контейнера, щоб запобігти ситуації, коли один контейнер споживає всі ресурси системи.
- Пріоритезація використання *CPU*: Контейнери можуть мати різні рівні пріоритету при доступі до *CPU*. Це забезпечує, що критично важливі додатки отримують більше процесорного часу, ніж менш важливі.
- Ізоляція вводу/виводу: Можна контролювати кількість вводу/виводу, що виконується контейнером на дискових носіях, що важливо для балансування навантаження між кількома контейнерами.
- Моніторинг та обмеження використання мережевих ресурсів: *Cgroups* дозволяє контролювати кількість мережевого трафіку, який генерує кожен контейнер.

Використання *cgroups* дозволяє гарантувати, що навіть якщо один контейнер перевантажується, це не вплине на інші контейнери або загальну стабільність системи.

3. Контейнерні образи (*Images*) та *UnionFS*

Контейнерні образи є основою для створення контейнерів. Образ — це статичний файл, який містить усе необхідне для запуску додатка: код, бібліотеки, конфігураційні файли та всі залежності. Контейнерні образи будуються за допомогою спеціальних файлів конфігурації (наприклад, *Dockerfile* для *Docker*), де прописані всі інструкції для створення контейнера.

UnionFS (*Union File System*) — це механізм, що дозволяє об'єднувати кілька шарів файлових систем у одну логічну файлову систему. Це забезпечує зручний і ефективний спосіб управління файлами та залежностями у контейнері. Образ контейнера складається з кількох шарів, де кожен шар представляє зміни на

конкретному етапі побудови контейнера. Наприклад, базовий шар може бути операційною системою, наступні шари — це встановлені пакети, бібліотеки та власне додаток.

Переваги використання *UnionFS*:

- Ефективність зберігання: Оскільки образ контейнера складається з кількох шарів, які можуть бути спільними для різних контейнерів, це дозволяє уникнути дублювання даних. Один і той самий базовий образ може використовуватися для багатьох контейнерів.
- Модифікованість: Зміни в образі контейнера (наприклад, встановлення нового додатку) додаються як новий шар, що не змінює попередні шари. Це дозволяє легко відслідковувати зміни та повертатися до попередніх версій.

Це означає, що один і той самий образ може використовуватися як базовий для кількох різних контейнерів, що робить процес створення контейнерів більш ефективним і швидким.

4. Мережа контейнерів

Кожен контейнер має власну ізольовану мережу завдяки network namespaces. Контейнери можуть взаємодіяти між собою за допомогою створення внутрішніх віртуальних мереж або можуть бути підключені до зовнішніх мереж через мережеві інтерфейси хост-системи.

Мережа контейнерів включає такі елементи:

- Мережеві інтерфейси: Кожен контейнер може мати власні віртуальні інтерфейси (*veth*), які можуть бути з'єднані з іншими контейнерами або зовнішніми мережами через мостові інтерфейси (*bridge*).
- Мережеві адреси: Контейнери отримують власні *IP*-адреси, що дозволяє їм функціонувати як незалежні вузли в мережі. Це може бути корисно при налаштуванні складних архітектур з використанням декількох контейнерів.
- Проксі-сервери та *NAT*: Для забезпечення доступу до зовнішніх мереж або з зовнішніх мереж до контейнерів використовуються механізми *NAT* (*Network Address Translation*) і проксі-сервери.

Контейнеризація спрощує роботу з мережами, дозволяючи налаштовувати ізольовані мережеві середовища для додатків.

5. Оркестрація контейнерів

У реальному світі більшість програм складаються не з одного контейнера, а з багатьох, які взаємодіють між собою. Для управління та автоматизації цих взаємодій використовується концепція оркестрації контейнерів. Оркестрація включає управління контейнерами на різних рівнях: від розгортання до масштабування, моніторингу та управління життєвим циклом контейнерів. Найбільш популярним інструментом для оркестрації є *Kubernetes*.

Kubernetes — це платформа для автоматизації розгортання, управління та масштабування контейнеризованих додатків. Основні принципи роботи *Kubernetes* полягають у таких концепціях:

- Поди: *Pods* — це основний об'єкт в *Kubernetes*, який представляє собою один або кілька контейнерів, що запускаються разом та мають спільні ресурси, такі як мережа та файлові системи. *Pod* є найменшою одиницею, яку можна керувати в *Kubernetes*.

- Ноди (*Nodes*): *Kubernetes* запускає контейнери на віртуальних або фізичних машинах, які називаються "нодами". Ноди можуть бути розгорнуті на будь-якій інфраструктурі, що підтримує *Kubernetes* (локальні сервери, хмарні сервіси тощо).

- Кластер: Кластер — це сукупність нод, керованих одним або декількома контрольними компонентами (*master nodes*). Усі операції в кластері *Kubernetes* здійснюються через контрольну площину (*control plane*), яка розподіляє навантаження, управляє мережами та контейнерами.

- Namespace: *Kubernetes* використовує концепцію *namespaces* для організації ресурсів усередині кластера. Це дозволяє розподіляти додатки по різних середовищах (наприклад, продакшн і тестове середовище), а також забезпечує ізоляцію.

- Сервіси та мережеві політики: *Kubernetes* автоматично створює сервіси для забезпечення доступу до *Pod*'ів та управління мережею між ними. Мережеві

політики дозволяють визначати правила взаємодії між *Pod*'ами та зовнішніми сервісами.

- Масштабування: *Kubernetes* дозволяє автоматично масштабувати додатки залежно від навантаження, збільшуючи або зменшуючи кількість *Pod*'ів. Це важливо для підтримки стабільності та оптимізації використання ресурсів.

- Управління станами: *Kubernetes* постійно моніторить стан *Pod*'ів і перезапускає їх, якщо вони виходять з ладу або не відповідають бажаному стану (*desired state*). Це забезпечує надійність і безперервність роботи додатків.

6. Безпека контейнерів

Ізоляція контейнерів не тільки важлива для розподілу ресурсів, але й для безпеки. Контейнери забезпечують рівень ізоляції, що захищає додатки від взаємного впливу, проте безпека контейнерів потребує додаткових заходів:

- Ізоляція через *namespaces* та *cgroups*: Як уже було зазначено, контейнери використовують простори імен для ізоляції процесів і ресурсів, а cgroups обмежують доступ до апаратних ресурсів. Це мінімізує ризики доступу до даних або ресурсів інших контейнерів.

- *Rootless* контейнери: Для підвищення безпеки контейнерів використовуються *rootless* контейнери, які працюють без привілеїв адміністратора (*root*). Це значно знижує ризики, пов'язані з виконанням шкідливого коду в контейнері.

- Контроль доступу: Ще один важливий аспект безпеки контейнерів — це контроль доступу. Системи аутентифікації та авторизації, такі як *Role-Based Access Control* (*RBAC*) у *Kubernetes*, допомагають обмежити доступ до управління контейнерами і ресурсами на рівні ролей і прав користувачів.

- Секрети та конфігурації: *Kubernetes* використовує механізм "секретів" для безпечного зберігання конфіденційної інформації (паролі, токени доступу тощо). Це дозволяє безпечно передавати такі дані у контейнер без їх явного включення у конфігураційні файли чи образи.

- Використання незмінних образів: Контейнерні образи повинні бути зафіксовані на певному рівні (*immutable*), тобто після їх створення вони не

повинні змінюватися. Це дозволяє уникнути ризиків, пов'язаних із модифікацією образів під час роботи, і забезпечує більш стабільне та передбачуване середовище.

7. *CI/CD* та контейнеризація

Одним із основних застосувань контейнерів є інтеграція з процесами безперервної інтеграції та безперервного розгортання (*CI/CD*). Контейнерні технології, такі як *Docker*, дозволяють автоматизувати весь життєвий цикл розробки та тестування програмного забезпечення, що значно прискорює процес релізу нових версій:

- Контейнери як середовище для тестування: Завдяки ізоляції та фіксованому набору залежностей контейнери ідеально підходять для автоматизованого тестування додатків. Вони дозволяють створювати однакові тестові середовища незалежно від того, де тестується програма — на локальному комп'ютері чи на сервері *CI*.

- Контейнерні реєстри: Образи контейнерів зберігаються в контейнерних реєстрах, таких як *Docker Hub* або приватні реєстри, де вони можуть бути легко завантажені та повторно використані у будь-якому середовищі.

- Автоматизоване розгортання: *CI/CD* системи, такі як *Jenkins* або *GitLab CI,* можуть автоматично створювати, тестувати та розгортати контейнерні образи у різних середовищах. Це значно скорочує час між внесенням змін у код і їх виходом у продакшн.

Контейнеризація стала ключовою технологією у сучасній інфраструктурі та розробці програмного забезпечення. Основні принципи роботи контейнерів, такі як ізоляція через namespaces та контроль ресурсів через cgroups, забезпечують безпечне та ефективне використання ресурсів системи. Оркестраційні інструменти, такі як *Kubernetes*, дозволяють автоматизувати управління контейнерами та забезпечувати стабільну роботу додатків у різних середовищах.

Контейнери не тільки спрощують процес розробки та тестування, але й покращують безпеку та надійність додатків завдяки ізоляції та використанню

незмінних образів. Завдяки цим властивостям контейнеризація стає важливим елементом будь-якої сучасної інфраструктури.

2.3 Аналіз *Kubernetes*, як основи для *Cyber Ice Box*

Kubernetes є однією з найпопулярніших і найпотужніших платформ для управління контейнеризованими додатками, і її роль у *Cyber Ice Box* платформі є ключовою для забезпечення стабільності, масштабованості та безпеки. *Cyber Ice Box*, як платформа для проведення *CTF*-змагань, має високу вимогу до можливості динамічного управління великою кількістю різних завдань, учасників і контейнерів, що робить *Kubernetes* ідеальним вибором для цієї архітектури.

1. Основи *Kubernetes*

Kubernetes (*K8s*) — це відкрита система оркестрації контейнерів, розроблена *Google*, яка автоматизує розгортання, масштабування і управління контейнеризованими додатками. *Kubernetes* забезпечує управління ресурсами через кластери, що складаються з вузлів (*nodes*), на яких працюють контейнери.

Головні компоненти *Kubernetes* включають:

- *Pod*: Основна одиниця, що управляється *Kubernetes*, це *Pod*. *Pod* може містити один або кілька контейнерів, які запускаються разом і мають спільне середовище.
- *Service*: Логічне представлення групи *Pods*, які забезпечують певний тип сервісу, наприклад, доступ до веб-додатка або бази даних.
- *Node*: Фізичний або віртуальний сервер, який є частиною кластера *Kubernetes* і виконує запущені *Pods*.
- *Kube-scheduler* і *Kube-controller-manager*: Основні компоненти, що відповідають за управління життєвим циклом *Pods*, розподіл завдань між вузлами та підтримання стану кластера.

Cyber Ice Box базується на контейнерній архітектурі, де кожне завдання для учасників може бути ізольоване в окремому контейнері, що виконується в *Pod*.

Завдяки цьому, платформа забезпечує незалежність виконання завдань, підвищену безпеку і можливість легкої ізоляції різних команд та їхніх рішень.

2. Гнучкість і модульність архітектури

Однією з головних переваг *Kubernetes* є його модульна архітектура, яка дозволяє гнучко налаштовувати кожен компонент під потреби платформи, як це реалізовано в *Cyber Ice Box*. Кожен з учасників або команд у *CTF*-змаганнях може отримувати індивідуальні середовища виконання завдань, що дозволяє організаторам швидко розгортати нові сценарії та оновлення під час змагань.

Kubernetes надає кілька важливих можливостей для цього:

- *Namespace*: *Kubernetes* підтримує ізоляцію ресурсів між різними середовищами за допомогою namespace, що дозволяє розподіляти ресурси між різними командами або типами завдань. У *Cyber Ice Box* кожна команда може бути відокремлена в свій власний namespace, що гарантує ізоляцію і безпеку.

- *ConfigMaps* і *Secrets*: Ці механізми дозволяють зберігати конфігураційні дані або чутливу інформацію (наприклад, ключі шифрування чи паролі) і передавати їх в контейнер в безпечний спосіб. Це особливо важливо для *CTF*-платформ, де захист даних критичний для чесності змагань.

- *Pod Security Policies* (*PSP*): *PSP* — це набір політик, що визначають, які дії можуть виконувати контейнери, наприклад, доступ до файлів, використання привілейованих режимів, можливість зміни мережевих параметрів тощо. *Cyber Ice Box* використовує ці політики для контролю доступу та запобігання небажаним діям з боку учасників, які можуть впливати на загальну безпеку платформи.

3. Контейнеризація і ізоляція завдань

Однією з головних причин використання *Kubernetes* у *Cyber Ice Box* є його можливість надавати ізольовані середовища для виконання завдань через контейнери. Контейнеризація дозволяє ізолювати середовище виконання кожної команди або завдання від інших, запобігаючи впливу помилок або шкідливих дій однієї команди на роботу всієї платформи.

- *Docker* або інші *runtime*: *Kubernetes* може використовувати різні контейнерні runtime для запуску контейнерів, найпопулярнішим з яких є *Docker*. У *Cyber Ice Box* кожне завдання або набір завдань може бути упаковане в окремий *Docker*-контейнер, що робить систему дуже гнучкою та легкою в налаштуванні. Наприклад, завдання можуть бути різного рівня складності, і кожен контейнер може бути оптимізований під конкретну задачу, будь то тестування вразливостей, робота з мережевими протоколами або аналіз даних.

- Відновлення стану: *Kubernetes* автоматично перезапускає контейнер у разі його падіння, забезпечуючи стабільність системи. Це критично для проведення *CTF*-змагань, де кожна хвилина важлива, а учасники повинні мати безперервний доступ до завдань. У випадку, якщо контейнер із завданням виходить з ладу, *Kubernetes* автоматично запускає його заново, забезпечуючи учасників стабільною платформою.

4. Оркестрація контейнерів і управління життєвим циклом завдань

Kubernetes дозволяє легко управляти великими масивами контейнерів, що є необхідним для масштабних *CTF*-змагань. Кожне завдання в *Cyber Ice Box* може бути виконане в окремому контейнері, а *Kubernetes* забезпечує автоматичне управління їхнім життєвим циклом: від розгортання до завершення виконання.

- Автоматичне масштабування: *Kubernetes* підтримує автоматичне масштабування контейнерів залежно від навантаження. У контексті *Cyber Ice Box* це означає, що кількість контейнерів для виконання завдань може бути автоматично збільшена або зменшена в залежності від кількості учасників і поточного навантаження на систему. Наприклад, якщо на платформі одночасно працюють кілька сотень команд, *Kubernetes* може автоматично додати нові екземпляри завдань для обробки запитів без затримок або збоїв.

- Живі оновлення (*Rolling Updates*): *Kubernetes* підтримує механізм живих оновлень, що дозволяє оновлювати контейнери без зупинки сервісу. У *Cyber Ice Box* це використовується для оновлення або додавання нових завдань під час змагань, без необхідності переривати процес для учасників. Таким чином,

платформа може постійно оновлювати завдання або виправляти помилки під час змагання, що підвищує гнучкість і зручність.

5. Забезпечення високої доступності та відмовостійкості

Kubernetes має вбудовані механізми для забезпечення високої доступності (*High Availability, HA*) і відмовостійкості (*Fault Tolerance*), що робить його надійним вибором для платформ з високими вимогами до стабільності, таких як *Cyber Ice Box.*

- *HA*-кластери: *Kubernetes* може працювати у режимі високої доступності, забезпечуючи мінімальні перерви у роботі системи навіть у разі виходу з ладу окремих вузлів. Це дозволяє *Cyber Ice Box* функціонувати без збоїв навіть при високому навантаженні або під час технічних проблем на окремих серверах.

- Розподілені репліки *Pods*: *Kubernetes* дозволяє створювати кілька реплік одного і того ж *Pod*, що підвищує надійність і швидкість виконання завдань. У *Cyber Ice Box* це може бути використано для паралельного запуску одного завдання для кількох учасників або команд, що дозволяє уникнути перевантаження окремих ресурсів і забезпечує кращу масштабованість системи.

6. Масштабованість та динамічне управління ресурсами

Одна з основних переваг *Kubernetes* — це його можливості щодо масштабування та ефективного управління ресурсами. Це дозволяє платформам, як-от *Cyber Ice Box,* працювати на значному рівні навантаження, особливо коли мова йде про великі *CTF*-змагання, де участь можуть брати сотні команд одночасно. Масштабування в *Kubernetes* дозволяє динамічно змінювати кількість активних контейнерів та використовуваних ресурсів у відповідь на зміни в навантаженні.

- *Horizontal Pod Autoscaling* (*HPA*): Ця функція *Kubernetes* дозволяє автоматично масштабувати кількість *Pods* у кластерах на основі використання ресурсів, таких як *CPU* або пам'ять. У *Cyber Ice Box* це може бути корисним для динамічного регулювання кількості контейнерів залежно від кількості учасників або інтенсивності завдань. Наприклад, при великому навантаженні (під час

пікових моментів змагань) система може автоматично збільшити кількість ресурсів, щоб забезпечити стабільну роботу платформи.

- *Vertical Pod Autoscaling* (*VPA*): Ця функція автоматично регулює виділення ресурсів для окремих контейнерів, дозволяючи змінювати об'єм доступної пам'яті або *CPU*. Це допомагає оптимізувати використання ресурсів для кожного завдання у *Cyber Ice Box*, забезпечуючи виконання завдань в межах виділених ресурсів і запобігаючи перевантаженню.

- Керування вузлами (*Node Management*): *Kubernetes* може автоматично переміщати *Pods* між різними вузлами в кластері для забезпечення рівномірного розподілу навантаження або у випадку, якщо один із вузлів вийде з ладу. У *Cyber Ice Box* це дозволяє зберегти безперебійну роботу платформи навіть у випадку збоїв у окремих вузлах інфраструктури.

7. Безпека в *Kubernetes*

Оскільки безпека є ключовим аспектом у проведенні *CTF*-змагань, *Kubernetes* пропонує низку інструментів для забезпечення ізоляції, шифрування і контролю доступу, що є критично важливими для платформи *Cyber Ice Box*. Для гарантування захищеності учасників і чесності змагань, Kubernetes надає наступні можливості:

- *Network Policies*: Це механізм, що дозволяє встановлювати правила для мережевої взаємодії між *Pods*. *Cyber Ice Box* може використовувати мережеві політики для обмеження доступу учасників до небажаних ресурсів і контролювати внутрішні та зовнішні підключення, що підвищує безпеку та ізоляцію між командами.

- *Role-Based Access Control* (*RBAC*): *RBAC* використовується для управління доступом до ресурсів кластера Kubernetes. У контексті *Cyber Ice Box* це дозволяє адміністраторам платформи контролювати, хто має доступ до конкретних частин системи (завдань, контейнерів, конфігурацій) і надавати відповідні дозволи лише тим, хто їх потребує.

- *Secrets Management*: *Kubernetes* має вбудований механізм для управління секретами, такими як паролі, ключі *API* або конфігураційні дані. Це

забезпечує надійне зберігання конфіденційної інформації та безпечну її передачу в контейнери *Cyber Ice Box*. Завдяки цьому забезпечується захист від потенційних витоків даних під час змагань.

- *Pod Security Policies* (*PSP*): Це ще один рівень захисту, який дозволяє обмежити привілеї контейнерів, що запускаються в *Pod*. Важливим аспектом у *Cyber Ice Box* є обмеження дій контейнерів для запобігання небажаним втручанням учасників у роботу системи або виконання несанкціонованих дій, таких як запуск зловмисного коду на платформі.

8. Інтеграція з іншими сервісами

Kubernetes забезпечує легку інтеграцію з зовнішніми сервісами, що може бути корисним для побудови складної архітектури платформи, як *Cyber Ice Box*. Зовнішні сервіси можуть включати бази даних, системи логування, аналітичні сервіси, і всі вони можуть бути інтегровані через *API Kubernetes*.

- *Persistent Volumes*: У випадках, коли необхідно зберігати дані тривалий час, наприклад, результати завдань або записи про дії учасників, Kubernetes використовує механізм *Persistent Volumes* (*PV*), який дозволяє зберігати дані навіть після зупинки контейнерів. Це забезпечує надійне зберігання даних у *Cyber Ice Box* під час тривалих змагань.

- *Ingress Controllers*: *Kubernetes* підтримує керування вхідним трафіком через механізм *Ingress*, що дозволяє направляти запити на правильні сервіси усередині кластера. У *Cyber Ice Box* це може бути використано для керування доступом до різних завдань, забезпечення балансування навантаження та створення безпечних підключень через *HTTPS*.

9. Можливості для розширення та кастомізації

Однією з найсильніших сторін *Kubernetes* є можливість створення кастомних розширень і використання додаткових модулів для розширення функціональності кластера. Для *Cyber Ice Box* це відкриває безліч можливостей для інтеграції з іншими інструментами, автоматизації процесів та гнучкого налаштування платформи під специфічні потреби організаторів *CTF*-змагань.

- *Custom Resource Definitions* (*CRD*): *CRD* дозволяють розширювати можливості *Kubernetes*, створюючи власні об'єкти і ресурси. У *Cyber Ice Box* це може використовуватися для розробки специфічних компонентів платформи, таких як нові типи завдань або механізми для оцінки результатів учасників.

- *Operators*: *Kubernetes* підтримує концепцію операторів, які автоматизують складніші завдання управління додатками. Оператори можуть автоматично управляти життєвим циклом компонентів *Cyber Ice Box*, що значно спрощує процеси розгортання та управління платформою під час великих змагань.

10. Можливість моніторингу та логування

Моніторинг та логування є важливими аспектами будь-якої платформи для проведення *CTF*-змагань, оскільки вони дозволяють слідкувати за роботою системи, виявляти проблеми і збирати дані для аналітики. *Kubernetes* інтегрується з багатьма популярними рішеннями для моніторингу та логування, що надає *Cyber Ice Box* важливі інструменти для аналізу та покращення платформи.

- *Prometheus* і *Grafana*: *Prometheus* є стандартним інструментом для моніторингу *Kubernetes*. Він дозволяє відстежувати показники використання ресурсів, статус *Pods* та інші критично важливі метрики. У поєднанні з *Grafana* ці дані можуть бути візуалізовані для надання зрозумілих і детальних графіків продуктивності платформи.

- *ELK Stack* (*Elasticsearch, Logstash, Kibana*): Цей набір інструментів широко використовується для збору та аналізу логів у *Kubernetes*-кластерах. Для *Cyber Ice Box* це означає можливість централізованого зберігання і аналізу логів, що дозволяє швидко реагувати на можливі проблеми та надавати зворотній зв'язок учасникам.

Kubernetes дозволяє *Cyber Ice Box* ефективно управляти контейнерами, масштабувати ресурси, забезпечувати стабільність і безпеку під час проведення *CTF*-змагань.

Висновки до розділу

Cyber Ice Box має контейнерну структуру на основі докера *Kubernetes*. Це дозволяє забезпечити розподілення ресурсів між учасниками та забезпечити безпеку та спостережність при проведенні змагань.

3. РОЗГОРТАННЯ ПЛАТФОРМИ

3.1 Підготовка до розгортання платформи

Платформа *Cyber ICE Box* не може працювати в повному обсязі без призначеного домену, який дозволяє адміністратору налаштувати доступ до неї для всіх зацікавлених осіб з Інтернету, для цього було зареєстровано новий домен (*рис*. 3.1).

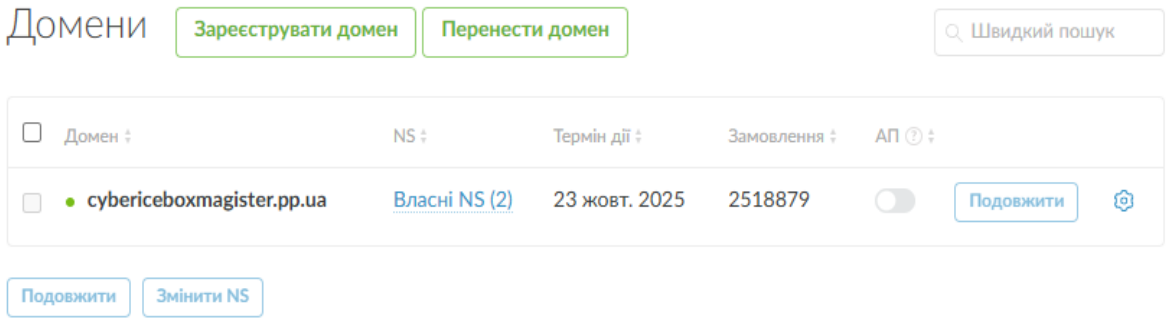

Рисунок 3.1 – Зареєстрований домен

Для забезпечення безпеки та захисту платформи від автоматизованих атак необхідно обов'язково налаштувати *Google reCAPTCHA*. Це допоможе запобігти несанкціонованому доступу і підвищить надійність форми авторизації. Налаштування та інтеграція *reCAPTCHA* для платформи *Cyber ICE Box* здійснюється через офіційний сайт *reCAPTCHA*. Основні кроки для створення та налаштування сервісів наведено нижче:

1. Активація *reCAPTCHA*:

- Перейдіть до *Google reCAPTCHA*.
- Згенеруйте публічний та секретний ключі (*рис*. 1.2).
- Використайте створені ключі для *config.yml*.

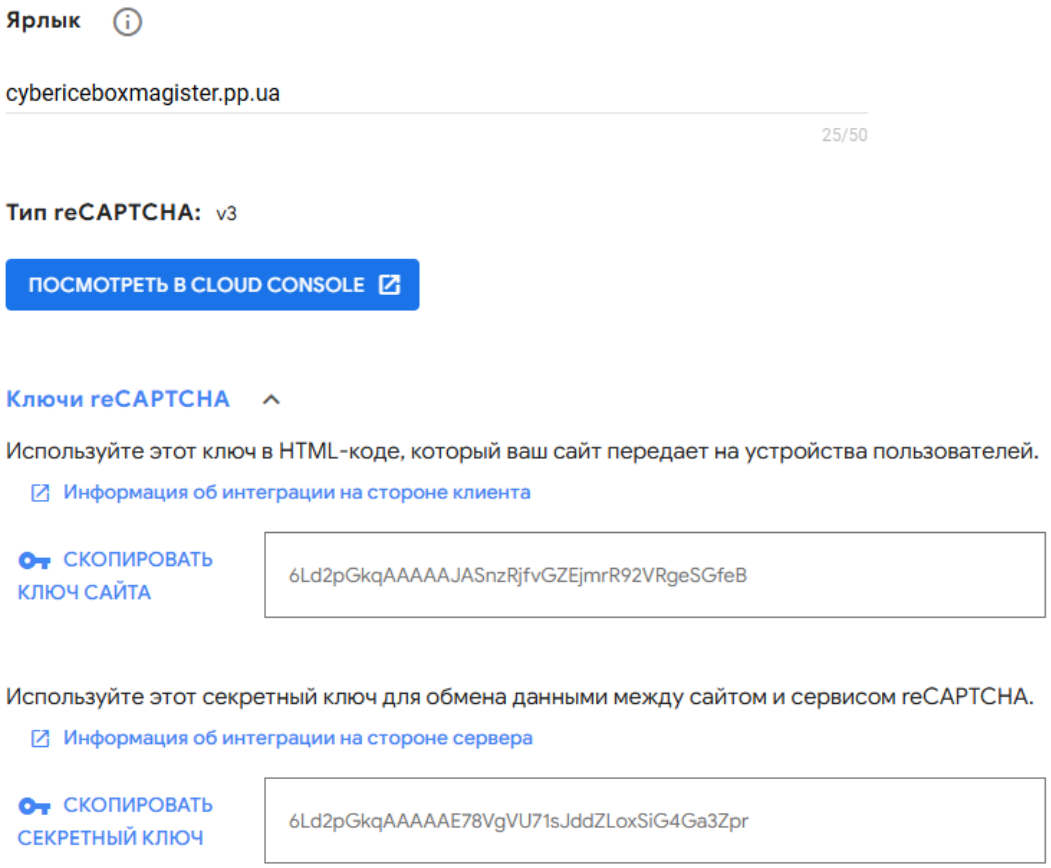

Рисунок 3.2 – Публічний та секретний ключі

2. Активація авторизації за допомогою *Google*:

- Перейдіть до *Google Cloud Console*.
- Створіть новий проєкт або виберіть існуючий.
- У розділі "*APIs & Services*" згенеруйте індетифікатор клієнта *OAuth2.0* та секретний ключ (*рис*. 1.3).
- Необхідною умовою для створення індетифікатора є заповнення параметру "*Authorized redirect URIs*". Для цього використайте цей шаблон: *https://cybericeboxmagister.pp.ua/api/auth/google/callback*
- Використайте створені ідентифікатор та секретний ключ для *config.yml*.

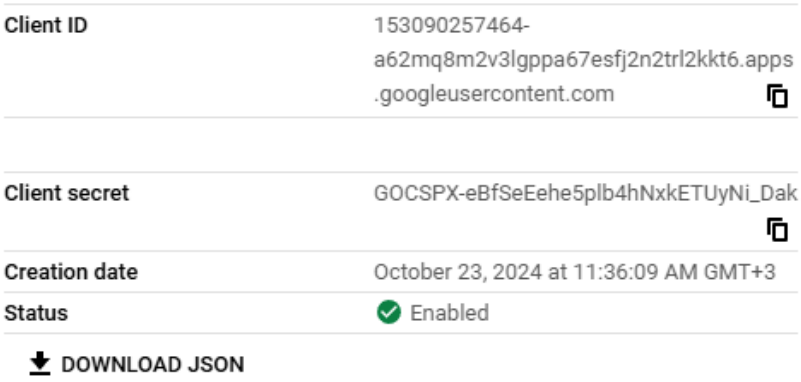

Рисунок 3.3 – Ідентифікатор та секретний ключ

3.2 Встановлення платформи *Kubernetes*

Основою для платформи є технологія *k0s.0 v1.30.2* — мінімалістичний дистрибутив *Kubernetes* з відкритим кодом, налаштований з усіма необхідними функціями для створення кластера *Kubernetes*.

Виконаємо команди для завантаження *k0s*, встановимо керуючий вузол кластера, запустимо створений кластер *Kubernetes*, під керуванням *k0s* та виконаємо перевірку поточного стану кластеру (*рис*. 1.4):

curl -sSLf https://get.k0s.sh | sudo sh

sudo k0s install controller –single

sudo k0s start

sudo k0s status

```
alex@alex-VirtualBox:~$ sudo k0s status
Version: v1.31.1+k0s.1
Process ID: 3942
Role: controller
Workloads: true
SingleNode: true
Kube-api probing successful: true
Kube-api probing last error:
```

Рисунок 3.4 – Перевірка стану

Наступні команди встановлюють мережевий плагін *Calico* для *Kubernetes*. Виконуючи команди, ви завантажуєте та застосовуєте маніфести для *Calico*, що налаштовують мережеву політику та маршрутизацію в кластері *Kubernetes*. Це забезпечує надійну мережеву інфраструктуру для взаємодії між подами та іншими компонентами в кластері:

sudo k0s kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/tigera-operator.yaml

sudo k0s kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manifests/custom-resources.yaml

Команда налаштовує доступ до ресурсів кластеру з зовнішньої мережі:

sudo k0s kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.14.5/config/manifests/metallb-native.yaml

Команда завантажує файл конфігурації платформи *config.yml*:

curl https://raw.githubusercontent.com/cybericebox/docs/main/config.yml -O

Певні параметри цього файлу необхідно змінити відповідно до підготовчого етапу (*рис*. 3.5).

config.yml: Блокнот

Файл Редагування Формат Вигляд Довідка

```
apiVersion: v1
kind: Namespace
metadata:
  name: cybericebox
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: config
  namespace: cybericebox
data:
  DOMAIN: cybericeboxmagister.pp.ua # кореневий домен платформи
  RECAPTCHA_KEY: 6Ld2pGkqAAAAAJASnzRjfvGZEjmrR92VRgeSGfeB # ключ сайту
  RECAPTCHA_SECRET:  6Ld2pGkqAAAAAE78VgVU71sJddZLoxSiG4Ga3Zpr
  JWT_TOKEN_SIGNATURE: "123123123" # ключ підпису токенів аутентифікації
  GOOGLE_CLIENT_ID: 153090257464-a62mq8m2v3lgppa67esfj2n2trl2kkt6.apps.googleusercontent.com
  GOOGLE_SECRET: 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
  EMAIL_HOST: smtp.gmail.com
  EMAIL_USERNAME: noreply.cybericebox@gmail.com
  EMAIL_PASSWORD: ...
  EMAIL_SENDER_NAME: "Cyber ICE Box Platform"
  EMAIL_SENDER_EMAIL: noreply@cybericebox.com
  EMAIL_REPLY_TO_NAME: "Support Cyber ICE Box Platform"
  EMAIL_REPLY_TO_EMAIL: support@cybericebox.com
  POSTGRES_HOSTNAME: ep-hidden-unit-a2yfrw6a.eu-central-1.aws.neon.tech
  POSTGRES_USER: cybericebox_mag_test_owner
  POSTGRES_PASSWORD: uG0CcvjkNEx1
  POSTGRES_DB: cybericebox_mag_test
  POSTGRES_SSL_MODE: require
  WG_GRPC_AUTH_KEY: "123"
  WG_GRPC_SIGN_KEY: "124"
  VPN_ENDPOINT: cybericeboxmagister.pp.ua
  AGENT_GRPC_AUTH_KEY: "123"
  AGENT_GRPC_SIGN_KEY: "123"
  LABS_CIDR: 128.0.0.0/8
---
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: subnet-address
  namespace: metallb-system
spec:
  addresses:
```

Рисунок 3.5 – Змінені параметри в файлі

Команда застосовує конфігурацію платформи з файлу *config.yml* до *Kubernetes*-кластеру:

sudo k0s kubectl apply -f config.yml

Наступна команда виконує розгортання інфрастриктури платформи *Kubernetes* з файлу *platform.yml*, використовуючи *kubectl*, що входить до складу *k0s*. Це означає, що *Kubernetes* об'єкти, визначені в *platform.yml*, будуть створені або оновлені в кластері.

sudo k0s kubectl apply -f https://raw.githubusercontent.com/cybericebox/docs/main/platform.yml

Команда перевіряє готовність інфрастуктури платформи:

sudo k0s kubectl get all -n cybericebox

У разі готовності всі компоненти матимуть статус "*Running*", як це показано на (*рис*. 3.6).

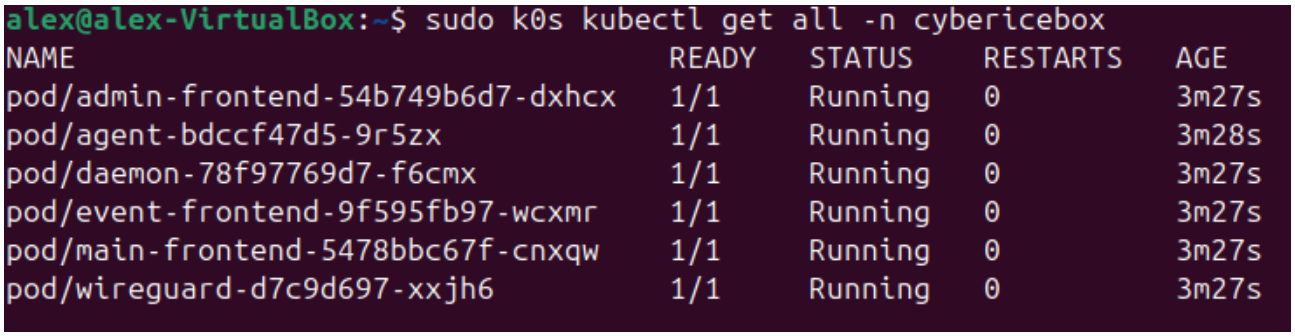

Рисунок 3.6 – Готовність всіх компонентів

3.3 Перевірка працездатності системи

Після успішного розгортання платформи переходимо до Адміністративної панелі за посиланням, https://admin.cybericeboxmagister.pp.ua.com, після переходу за посиланням відкриється сайт зі сторінкою аутентифікації (*рис*. 3.7).

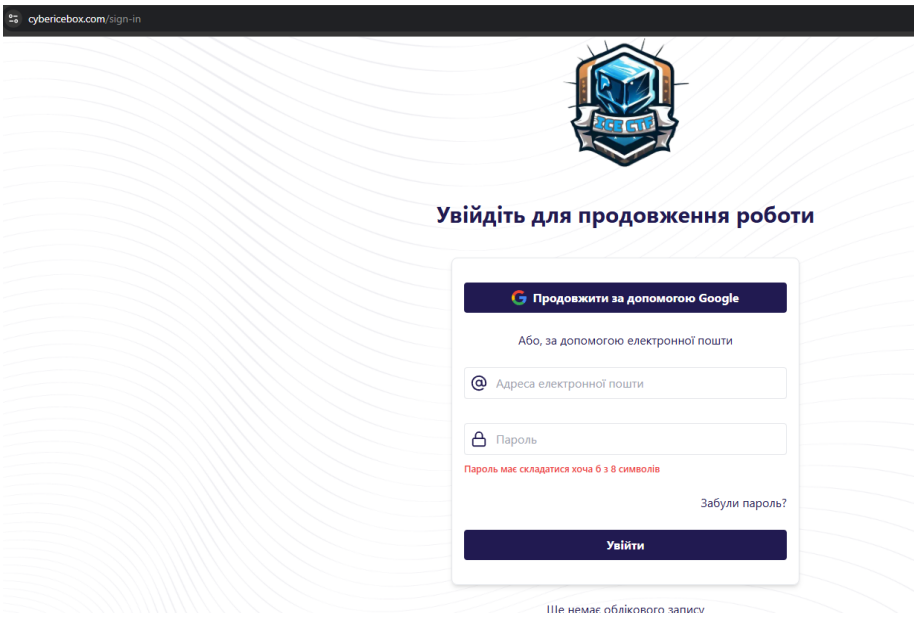

Рисунок 3.7 – Сторінка аутентифікації

Після проходження аутентифікації створимо завдання, для цього натискаємо на вкладку "Завдання", у відкритій вкладці напискаємо кнопку "Створити завдання" , після цього відкриється вкладка з налаштуванням нового завдання (*рис*. 3.8).

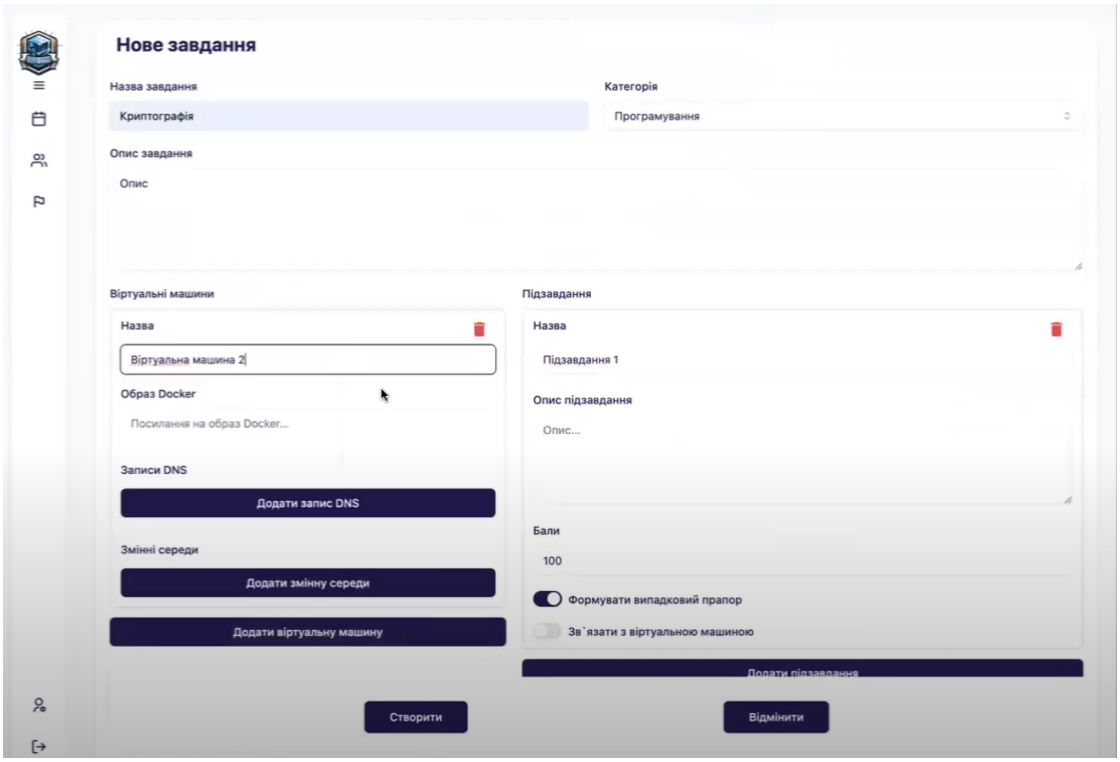

Рисунок 3.8 – Налаштування завдання

Завдання можуть бути "динамічними" і "статичні". Статичні завдання не використовують ресурсів *Kubernetes* , для створення статичних завдань треба заповнити розділ "Підзавдання", а для створення динамічних завдань треба заповнити розділ "Віртуальні машини" і "Підзавдання".

Для тестування було підготовлено створено завдання.

3.3.1 Завдання з криптографії

Короткі теоретичні відомості по завданню.

Шифр *Playfair* був першим практичним шифром заміни орграфа. Алгоритм шифрування *Playfair Cipher*:

Алгоритм складається з 2 кроків:

– згенеруйте квадрат ключа (5×5): Ключовий квадрат – це сітка 5×5 алфавітів, яка діє як ключ для шифрування відкритого тексту. Кожен із 25 алфавітів має бути унікальним, і одна літера алфавіту (зазвичай *J*) пропускається з таблиці (оскільки таблиця може містити лише 25 алфавітів). Якщо відкритий текст містить *J*, то він замінюється на *I*.

– початкові алфавіти в квадраті ключа – це унікальні алфавіти ключа в тому порядку, у якому вони з’являються, за якими йдуть інші літери алфавіту.

Алгоритм шифрування відкритого тексту: відкритий текст розбивається на пари з двох літер (диграфи). Якщо кількість літер непарна, до останньої літери додається *Z*. Додаткові вимоги:

1. пара не може бути створена з однаковою буквою. Розбийте літеру на одну та додайте фіктивну літеру до попередньої;

2. Якщо літера стоїть окремо в процесі створення пари, тоді додайте додаткову фіктивну літеру разом із окремою літерою;

Правила шифрування:

1) якщо обидві літери знаходяться в одному стовпчику : візьміть літеру під кожною (повертайтеся вгору, якщо внизу);

2) якщо обидві літери знаходяться в одному рядку : візьміть літеру праворуч від кожної (поверніться до крайньої ліворуч, якщо справа);

3) якщо жодне з наведених вище правил не відповідає дійсності : сформуйте прямокутник із двох літер і візьміть літери в горизонтальному протилежному куті прямокутника.

Назва: *Decrypt the Message*

Категорія: *Cryptography*

Опис завдання:

Тобі вдалося перехопити зашифроване повідомлення, яке використовує метод Плейфера. Використовуючи відомий ключ, розшифруй текст і знайди флаг.

Зашифроване повідомлення:

GATLMZ CLRIFY WKMNTY

Ключ: *securekey*

Пояснення методу (приклад для таблиці):

Створіть таблицю 5x5, видаливши повтори з ключа:

S E C U R

K Y A B D

F G H I L

M N O P Q

T V W X Z

Розшифруйте зашифроване повідомлення, застосовуючи правила шифру Плейфера.

Флаг: *CTF{MESSAGE_DECRYPTED}*

3.3.2 Тестування вразливості сайту

Короткі теоретичні відомості.

SQL-ін’єкція – це вразливість веб-безпеки, яка дозволяє зловмиснику втручатися в запити, які програма робить до своєї бази даних. Це може дозволити

зловмиснику переглянути дані, які вони зазвичай не можуть отримати. Це може включати дані, які належать іншим користувачам, або будь-які інші дані, до яких програма має доступ. У багатьох випадках зловмисник може змінити або видалити ці дані, викликаючи постійні зміни у вмісті або поведінці програми.

Ви можете виявити впровадження *SQL* вручну, використовуючи систематичний набір тестів для кожної точки входу в програмі. Для цього ви зазвичай надсилаєте:

– символ одинарних лапок 'і пошук помилок або інших аномалій;

– деякий спеціальний синтаксис *SQL*, який обчислює базове (оригінальне) значення точки входу та інше значення та шукає систематичні відмінності у відповідях програми;

– логічні умови, такі як *OR* 1=1і *OR* 1=2, і шукайте відмінності у відповідях програми;

– корисне навантаження, призначене для запуску затримок часу під час виконання в межах *SQL*-запиту та пошуку відмінностей у часі, необхідному для відповіді;

– корисні навантаження *OAST*, призначені для ініціювання позасмугової взаємодії з мережею під час виконання в межах запиту *SQL* і моніторингу будь-яких взаємодій, що виникають у результаті.

Крім того, ви можете швидко та надійно знайти більшість уразливостей *SQL*-ін’єкцій за допомогою *Burp Scanner*.

Однак уразливості *SQL*-ін’єкцій можуть виникати в будь-якому місці запиту та в різних типах запитів. Деякі інші поширені місця, де виникає *SQL*-ін’єкція:

– у *UPDATE* заявах, в межах оновлених значень або *WHERE* пропозиції;

– у *INSERT* заявах у межах вставлених значень;

– у *SELECT* заявах, у межах назви таблиці чи стовпця;

– у *SELECT* заявах, у *ORDER BY* пункті.

Назва: *Hack the Admin Panel*

Категорія: *Web Exploitation*

Опис завдання:

У тебе є доступ до сторінки входу в адміністративну панель. Твоя задача – знайти уразливість у системі і проникнути до адміністратора, щоб отримати флаг.

Файл для завантаження:

SQL-дамп бази даних із таблицею users.

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(50) NOT NULL
);

INSERT INTO users (username, password) VALUES
('admin', 'adminpass'),
('guest', 'guestpass'),
('flag_user', 'CTF{SQL_HACK_SUCCESS}');
```

PHP-код сторінки авторизації:

```
<?php
$conn = new mysqli("localhost", "root", "", "ctf");

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $password = $_POST['password'];

    $sql = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";
    $result = $conn->query($sql);
```

```
    if ($result->num_rows > 0) {
        echo "Welcome, " . $username;
        if ($username == "flag_user") {
            echo "<br>Flag: CTF{SQL_HACK_SUCCESS}";
        }
    } else {
        echo "Invalid login";
    }
}
?>
<form method="POST">
    Username: <input type="text" name="username"><br>
    Password: <input type="password" name="password"><br>
    <input type="submit" value="Login">
</form>
```

Інструкція для учасників:

– завантажте *PHP*-файли та розгорніть їх локально (*рис*. 3.9).

– виявіть уразливість *SQL*-ін’єкції.

– увійдіть як *flag_user*, щоб отримати флаг.

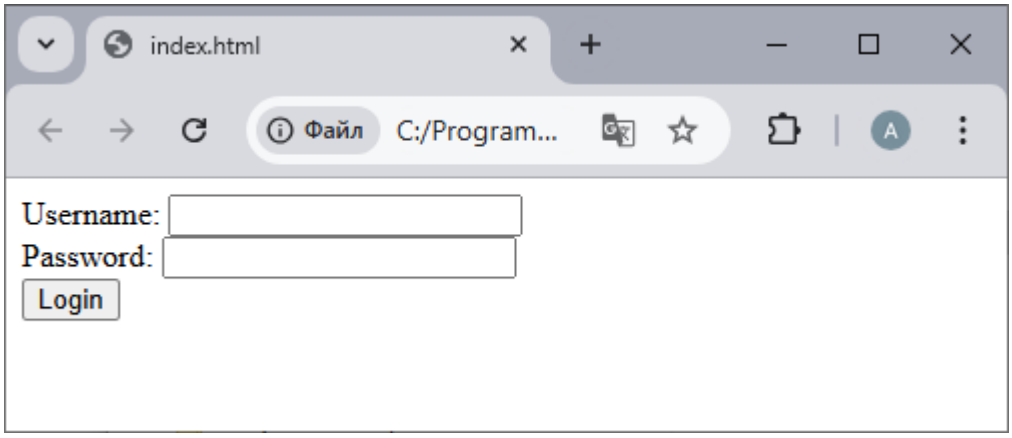

Рисунок 3.9 – Зовнішній вигляд сторінки

Рішення завдання

Аналіз запиту в коді *PHP*: У *PHP*-коді є *SQL*-запит, що використовує введення користувача без належної перевірки або захисту:

$sql = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";

Знайти можливість *SQL*-ін'єкції: Оскільки у запиті не використовується підготовлений запит або екранування вхідних даних, зловмисник може вставити власний *SQL*-код у поля "*username*" та "*password*".

Спробувати стандартні *SQL*-ін'єкції: У полі *Username* можна ввести:

admin' --

Введення ' -- перетворює решту запиту на коментар, і *SQL*-запит стає:

*SELECT * FROM users WHERE username = 'admin' -- ' AND password = '$password';*

Це дозволяє отримати доступ до користувача з ім'ям *admin* без перевірки пароля.

Рішення (отримання флагу): Коли в поле "*Username*" вводиться *admin*' --, а в поле "*Password*" залишити будь-яке значення, запит виконується успішно, і користувач автоматично авторизується як *admin*.

Отримати флаг: Після авторизації як *admin*, сторінка відобразить флаг:

Flag: CTF{SQL_HACK_SUCCESS}

Пояснення *SQL*-ін'єкції:

' -- в *SQL* означає, що після символу ' починається коментар, і решта запиту ігнорується.

Завдяки цьому, запит для користувача *admin* буде виглядати так:

*SELECT * FROM users WHERE username = 'admin' -- ' AND password = '$password';*

Частина *AND password = '$password'* буде ігнорована, оскільки після -- йде коментар.

Повний алгоритм:

– введіть в полі *Username: admin*' -- ;

– введіть у полі *Password* будь-яке значення (наприклад, 1234);

– натисніть кнопку входу.

Оскільки *SQL*-запит успішно пройде для користувача admin, ви отримаєте флаг: *Flag: CTF{SQL_HACK_SUCCESS}*

3.3.3 Аналіз дампу пам'яті

Короткі теоретичні відомості.

Дамп пам'яті - вміст робочої пам'яті одного процесу , ядра або всієї операційної системи . Також може включати додаткову інформацію про стан програми або системи, наприклад, значення регістрів процесора і вміст стека . Багато операційних систем дозволяють зберігати дамп пам'яті для налагодження програми . Як правило, дамп пам'яті процесу зберігається автоматично, коли процес завершується через критичну помилку (наприклад, через помилку сегментації). Дамп також можна зберегти вручну через відладчик чи будь-яку іншу спеціальну програму.

Назва: *Lost key*

Категорія: *Analyze the memory dump*

Опис завдання:

Ваше завдання - знайти прихований ключ у дампі пам'яті. Інженер безпеки випадково залишив зашифрований секрет у програмі, а сам ключ потрапив у пам'ять. Доступ до вихідного коду програми відсутній, тому єдиний спосіб розкрити секрет — аналіз дампу пам'яті.

Вхідні дані:

Завантажте дамп пам'яті (наприклад, *memory_dump.bin*).

Ключ зберігається у вигляді *ASCII*-рядка у пам'яті.

Структура ключа: він починається зі слова *KEY* і складається із 15 символів

Завдання:

– знайти ключ у дампі пам'яті;

– ввести ключ у форму для перевірки.

Для створення дампу пам’яті використовуємо код мовою *Python*:

```
# Генеруємо штучний дамп пам'яті
dummy_data = b"This is some random memory data.\n"
key = b"KEY1234abcd5678"  # Прихований ключ
dummy_data += key + b"\nMore random data follows."

# Запис дампу у файл
with open("memory_dump.bin", "wb") as f:
    f.write(dummy_data)
```

Інструкція по розв’язку:

– необхідно завантажити файл *memory_dump.bin*

– Для знахождення ключа можна використати код *Python*:

```
import re

# Відкриття дампу пам'яті
with open("memory_dump.bin", "rb") as f:
```

```
    data = f.read()

# Пошук ключа за допомогою регулярного виразу
matches = re.findall(rb"KEY[a-zA-Z0-9]{12}", data)
if matches:
    print("Знайдено ключ:", matches[0].decode())
else:
    print("Ключ не знайдено.")
```

Виведення ключа: Ключ, що відповідає шаблону, буде виведений у терміналі.

Перевірка на платформі: Скопіюйте знайдений ключ і вставте його в поле введення на платформі *CTF*.

3.3.4 Стеганографія

Короткі теоретичні відомості.

Стеганографія в цифрових зображеннях - розділ стеганографії , що вивчає проблему приховування даних у цифрових зображеннях. На відміну від криптографії , завдання стеганографії - приховати сам факт наявності прихованого повідомлення . .

Завдання стеганографії у зображеннях - вбудувати інформацію в цифрове зображення так, щоб і повідомлення, і сам факт його наявності були приховані. Отримане зображення з додатковою прихованою інформацією не має виглядати аномальним. Це досягається шляхом внесення змін, непомітних для людського зору. Багато методів стеганографії використовують методики, схожі з методами стиснення зображень .

Назва*: Hidden key*

Категорія: *Stenography*

Опис завдання:

Отримати приховане повідомлення з фото. Повідомлення приховано за допомогою коду *Python*. Приховування тексту у *LSB* (нижні біти) повідомлення було закодоване в нижніх бітах пікселів і завершується спеціальним стоп-символом (11111110).

Для формування завдання було обрано вхідний рисунок 3.10 та алгоритм приховування повідомлення в ньому.

Рисунок 3.10 – Вхідний рисунок

Для шифрування повідомлення використовуємо код:

```
from PIL import Image

def embed_message(image_path, message, output_path):
    img = Image.open(image_path)
    binary_message = ''.join(format(ord(i), '08b') for i in message) + '11111110'  # Стоп-символ
    data = img.getdata()
    new_data = []
    message_index = 0
```

```
    for pixel in data:
        new_pixel = list(pixel)
        for i in range(3):  # R, G, B
            if message_index < len(binary_message):
                new_pixel[i] = pixel[i] & ~1 | int(binary_message[message_index])
                message_index += 1
        new_data.append(tuple(new_pixel))
    img.putdata(new_data)
    img.save(output_path)
embed_message('original_image.png', 'flag{hidden_message_in_image}',
'hidden_image.png')
```

Отримуємо вихідний рисунок 3.11.

Рисунок 3.11 – Отриманий рисунок з прихованим повідомленням

Помістіть файл *hidden_image.png* у ту саму папку, де знаходиться скрипт:

```
from PIL import Image

def extract_message(image_path):
    img = Image.open(image_path)
    data = img.getdata()
    binary_message = ""

    # Збираємо останні біти з кожного кольору (R, G, B)
    for pixel in data:
        for value in pixel[:3]:  # Беремо лише R, G, B (ігноруємо Alpha, якщо є)
            binary_message += str(value & 1)

    # Розбиваємо бінарний рядок на байти
    byte_array = [binary_message[i:i+8] for i in range(0, len(binary_message), 8)]

    # Перетворюємо байти на символи, поки не досягнемо стоп-символу
    message = ""
    for byte in byte_array:
        if byte == "11111110":  # Стоп-символ
            break
        message += chr(int(byte, 2))
    return message
# Виклик функції для витягнення повідомлення
hidden_message = extract_message("hidden_image.png")
print("Приховане повідомлення:", hidden_message)
```

Ви побачите приховане повідомлення у консолі.

3.3.5 Реверс-інженірінг

Короткі теоретичні відомості

Зворотна інженерія — це процес дослідження пристрою або програми для розуміння принципів їхньої роботи. Основна мета такого аналізу — створення нового об'єкта, який за функціональністю подібний до оригінального, але без точного копіювання його функцій.

Цей метод застосовують переважно тоді, коли розробник оригінального продукту не надає детальної інформації про алгоритми його роботи або намагається обмежити використання своїх технологій.

Нині зворотна інженерія найбільше використовується для аналізу програмного забезпечення із закритим вихідним кодом. У цьому процесі дослідники аналізують машинний код (часто у вигляді дизасембльованого тексту), відновлюють алгоритми роботи програми та використовують ці дані для створення нових продуктів або розробки відповідних специфікацій.

Хоча зворотна інженерія зазвичай заборонена законами чи умовами ліцензійних угод, продукти, створені на основі вивчення алгоритмів, вважаються законними, оскільки кінцевий результат рідко нагадує оригінал.

Назва*: Decrypt the program*

Категорія: *Reverse engineering*

Опис завдання:

Ваша мета — проаналізувати виконуваний файл *Windows* (*decrypt_me.exe*), щоб отримати прихований флаг. Файл містить простий алгоритм перевірки пароля. Знайдіть правильний пароль і отримайте флаг! Дається 2 файли *"pyinstxtractor.py"* і *"decrypt_me.exe"*

Виконуваний файл написаний на мові Python:

```
# decrypt_me.py

def check_password():
    password = input("Введіть пароль: ")
    correct_password = "sup3r_s3cr3t"

    if password == correct_password:
```

```
        print("Флаг: CTF{" + password + "}")
    else:
        print("Неправильний пароль. Спробуйте ще раз.")

    input()

if __name__ == "__main__":
    check_password()
```

Файл python скомпільований в виконуваний файл за допомогою команди:

pyinstaller --onefile decrypt_me.py

після цього отримується файл *"decrypt_me.exe"*.

Рішення завдання

За допомогою файлу *"pyinstxtractor.py*" здобуваємо тимчасові файли, для отримання тимчасових файлі треба файли *"pyinstxtractor.py"* і *"decrypt_me.exe"* помістити в одну папку і в консолі запустити команду (рис. 3.12) :

python pyinstxtractor.py decrypt_me.exe

```
C:\Users\Oleksandr\Desktop\test>python pyinstxtractor.py decrypt_me.exe
[+] Processing decrypt_me.exe
[+] Pyinstaller version: 2.1+
[+] Python version: 3.7
[+] Length of package: 4368511 bytes
[+] Found 20 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: decrypt_me.pyc
[+] Found 76 files in PYZ archive
[+] Successfully extracted pyinstaller archive: decrypt_me.exe

You can now use a python decompiler on the pyc files within the extracted directory
```

Рисунок 3.12 – Виконання команди

Після виконання команди створится папка *"decrypt_me.exe_extracted"* в якій знаходяться тимчасові файли (*рис*. 3.13).

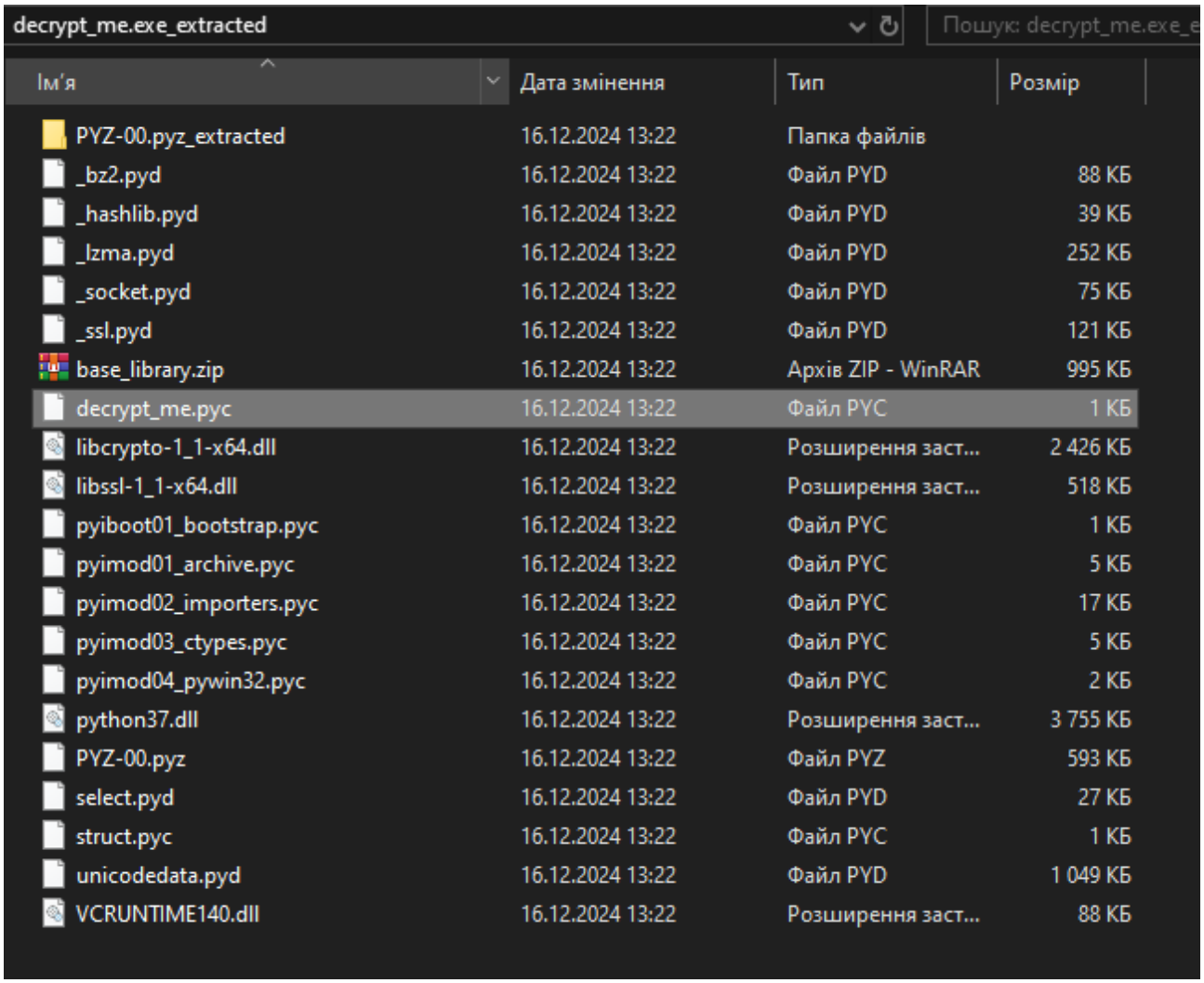

Рисунок 3.13 – Папка з тимчасовими файлам

В цій папці треба найти файл *"decrypt_me.pyc"*, за допомогою команди розкодовуємо файл:

uncompyle6 decrypt_me.exe_extracted/decrypt_me.pyc > main.py

Створюється розкодований файл *"main.py"* в якоку знаходится ключ (*рис.* 3.14).

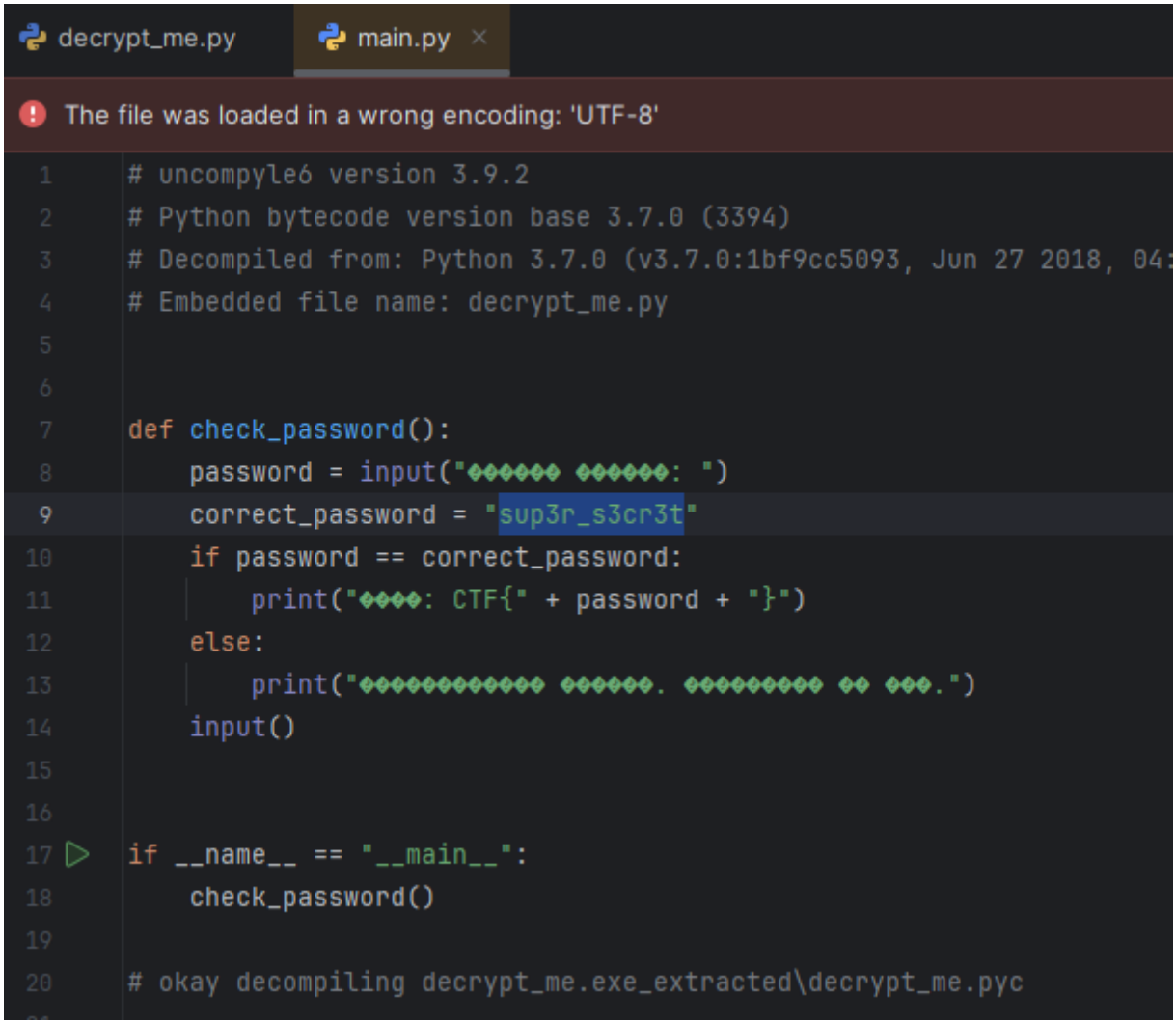

Рисунок 3.14 – Розкодований файл

Перевіряємо ключ увівши його в програму і отримаємо флаг (*рис*. 3.15)

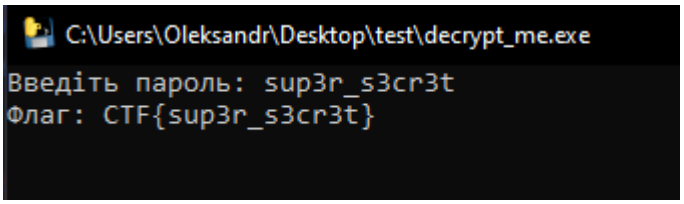

Рисунок 3.15 – Перевірка ключа

Висновки до роздулу

В розділі представлено інструкцію по розгортанню системи CYBER ICE BOX Platform та проведено аналіз інструментів роботи з завданнями та створено тестові завдання для перевірки навичок на рівні бакалаврів.

ВИСНОВОК

У кваліфікаційній роботі "Розгортання та налаштування CYBER ICE BOX Platform для проведення CTF змагань" було детально проаналізовано ключові аспекти організації Capture the Flag (CTF) змагань, їх значення для розвитку кібербезпеки, а також технічні можливості, які забезпечує платформа CYBER ICE BOX. Робота складається з кількох розділів, що розкривають як концептуальні, так і практичні питання, пов'язані з розгортанням та налаштуванням інфраструктури для CTF.

Перший розділ висвітлив роль CTF-змагань як ефективного інструменту для тестування навичок кібербезпеки, що сприяє підготовці фахівців до вирішення реальних загроз.

У другому розділі був проведений глибокий аналіз CYBER ICE BOX Platform. Ця платформа використовує контейнеризацію і Kubernetes для забезпечення високої продуктивності, гнучкості та безпеки середовища, що є критично важливим для проведення CTF-змагань.

Третій розділ роботи містить детальний опис процесу розгортання Kubernetes як основного середовища для CYBER ICE BOX, від початкового налаштування до тестування працездатності системи.

Платформа забезпечує високий рівень безпеки, гнучкість та продуктивність, що робить її одним із перспективних інструментів для організаторів і учасників змагань. Використання подібних технологій сприяє підвищенню якості підготовки фахівців з кібербезпеки, особливо в Україні, де інтерес до таких заходів продовжує зростати.

ПЕРЕЛІК ПОСИЛАНЬ

1. Capture the flag (CTF): веб-сайт. URL: https://uk.wikipedia.org/wiki/Capture_the_flag_(CTF)

2. Capture the flag (cybersecurity): веб-сайт. URL: https://en.wikipedia.org/wiki/Capture_the_flag_(cybersecurity)

3. CTF: веб-сайт. URL: https://www.linkedin.com/pulse/ctf-digialert

4. Kubernetes Documentation: веб-сайт. URL: https://kubernetes.io/docs/home/

5. Що таке контейнеризація та в чому її переваги – GigaCloud: веб-сайт. URL: https://gigacloud.ua/blog/navchannja/chomu-kontejneri-ce-majbutne-virtualizacii

6. Types of CTF challenges: веб-сайт. URL: https://snyk.io/series/ctf/ctf-types/

7. CTFtime.org / What is Capture The Flag?: веб-сайт. URL: https://ctftime.org/ctf-wtf/

8. GitHub - cybericebox/docs: Рекомендації, щодо розгортання та налаштування Cyber ICE Box Platform: веб-сайт. URL: https://github.com/cybericebox/docs?tab=readme-ov-file

9. Зворотна розробка: веб-сайт. URL: https://uk.wikipedia.org/wiki/%D0%97%D0%B2%D0%BE%D1%80%D0%BE%D1%82%D0%BD%D0%B0_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0

10. Шифр Плейфера: веб-сайт. URL: https://uk.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%9F%D0%BB%D0%B5%D0%B9%D1%84%D0%B5%D1%80%D0%B0

11. SQL ін'єкції: що це, як працює, які типи існують, небезпечність: веб-сайт. URL: https://foxminded.ua/sql-iniektsii/

12. Тестування безпеки: SQL-ін'єкції. : веб-сайт. URL: https://training.qatestlab.com/blog/technical-articles/security-testing-sql-injection/

13. Захист веб-додатків від SQL-ін'єкцій: веб-сайт. URL: https://openarchive.nure.ua/server/api/core/bitstreams/4bc35e97-c6a3-4155-bc68-70591fe4fd27/content

14. Стеганографія : веб-сайт. URL: https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%B3%D0%B0%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%8F

15. CTF Challenge Categories: веб-сайт. URL: https://docs.ctfd.io/events/challenge-categories/

16. Top 6 Platforms to Run your CTF On: веб-сайт. URL: https://cybertalents.com/blog/top-platforms-to-run-your-ctf

17. Beginner's Guide: веб-сайт. URL: https://www.csc.ac.za/?page_id=555

18. CTFd : The Easiest Capture The Flag Platform: веб-сайт. URL: https://ctfd.io/features/

19. Hack The Box: The #1 Cybersecurity Performance Center: веб-сайт. URL: https://www.hackthebox.com/

20. picoCTF - CMU Cybersecurity Competition: веб-сайт. URL: https://picoctf.org/

21. Добро пожаловать на [Root Me : plateforme d'apprentissage ... : веб-сайт. URL: https://www.root-me.org/?lang=en

22. OverTheWire: Wargames: веб-сайт. URL: https://overthewire.org/

23. Kubernetes: веб-сайт. URL: https://en.wikipedia.org/wiki/Kubernetes

24. Змагання CTF — ключ до успішної кар'єри у сфері ... : веб-сайт. URL: https://dou.ua/forums/topic/43810/

25. Hack The Box: веб-сайт. URL: https://univd.edu.ua/uk/news/8668

26. Український студентський CTF-турнір: кібервоїни з КПІ: веб-сайт. URL: https://kpi.ua/2023-kp9-iszzi

27. Українські команди перемогли у міжнародному CTF- ... : веб-сайт. URL: https://cip.gov.ua/ua/news/ukrayinski-komandi-peremogli-u-mizhnarodnomu-ctf-zmaganni1

28. Best CTF Platforms To Learn (Capture The Flag) ... : веб-сайт. URL: https://www.craw.sg/best-ctf-platforms-to-learn/

29. Review of 5 Platforms to Run Your CTF by Cyberseclabs: веб-сайт. URL: https://www.cyberseclabs.org/review-of-5-platforms-to-run-your-ctf-by-cyberseclabs/

30. CTF for Beginners: What is CTF and how to get started!: веб-сайт. URL: https://dev.to/atan/what-is-ctf-and-how-to-get-started-3f04

31. UnionFS: веб-сайт. URL: https://en.wikipedia.org/wiki/UnionFS

32. Що таке оркестрація контейнерів: пояснюємо на пальцях: веб-сайт. URL: https://itedu.center/ua/blog/articles/simple_orchestration/

33. Вербіцький О. В. Вступ до криптології : підручник. Львів : видавництво науково-тех. літ., 1998. 247 с

34. Halfond, W. G., Viegas, J., Orso A. A classification of SQL-injection attacks and countermeasures. In Proceedings of the IEEE international symposium on secure software engineering. 2006.Vol. 1, pp. 13-15). IEEE.

35. Andrew S. Tanenbaum, Todd Austin. Structured computer organization, 6th ed. Published by Pearson 2012, 801 p.

36. Франчук В. М. Захист інформаційних ресурсів : криптографічні та стеганографічні методи захисту даних: посібник для викладачів, вчителів та студентів інформатичних спеціальностей. Київ : НПУ ім. М. П. Драгоманова. Інститут інформатики, 2012. 120 с