УДК 004.056:37.018.43
DOI: 10.18523/2617-3808.2025.8.174-179

Глибовець А. М., Бабич Т. А.

РЕАЛІЗАЦІЯ ПРАКТИКО-ОРІЄНТОВАНОЇ СИСТЕМИ КІБЕРПОЛІГОНУ З ІНДИВІДУАЛІЗОВАНИМ ЗАПУСКОМ СЕРЕДОВИЩ

У статті описано архітектуру й технічну реалізацію контейнерно-орієнтованого кіберполігону, інтегрованого з LMS Moodle через LTI. Показано переваги підходу щодо швидкодії, масштабованості й безпеки. Проведено порівняння з альтернативними рішеннями та визначено подальші напрями розвитку платформи.

Ключові слова: кіберполігон, кібербезпека, практико-орієнтоване навчання, Capture the Flag (CTF), контейнеризація, Docker, LTI інтеграція, Moodle, інтерактивне навчання, імітаційні сценарії.

Вступ

Зростання кількості та складності кібератак у сучасному світі вимагає підвищення ефективності освіти у сфері кібербезпеки. Традиційні освітні підходи, що зосереджені на теорії, сертифікації та відповідності вимогам, не забезпечують достатнього практичного досвіду для протидії реальним загрозам. Як наслідок, виникає потреба у впровадженні практико-орієнтованих методів навчання, які надають слухачам можливість отримати практичні навички у безпечному, контрольованому середовищі. Саме таким інструментом є кіберполігони (англ. cyber ranges) — інтерактивні віртуальні середовища, що імітують мережі, системи та додатки і дозволяють відпрацьовувати навички кіберзахисту на реальних сценаріях. Кіберполігони сприяють формуванню практичних умінь, підвищують обізнаність і готовність фахівців шляхом моделювання реальних інцидентів без ризику для продуктивних систем.

Однак розробка та підтримка власного кіберполігону є ресурсномісткими: потрібні значні фінансові витрати, обчислювальні ресурси й час на розгортання складної інфраструктури. Традиційні лабораторні середовища на базі фізичних чи віртуальних машин також мають обмеження: вони важко масштабуються на велику кількість студентів, потребують ручного налаштування. У стандартних навчальних програмах часто бракує реалізму та інтерактивності: студенти вивчають теорію і здають тести, але майже не отримують практичного досвіду роботи з актуальними атаками чи інцидентами. Це призводить до розриву між теоретичними знаннями та реальними навичками, необхідними для кібербезпеки [3].

Відповіддю на ці виклики стало впровадження кіберполігонів як навчального інструменту. Вони забезпечують реалістичне навчання через імітацію кібератак та захисних заходів у безпечному середовищі. Дослідження показують, що практичні вправи суттєво підвищують мотивацію і рівень підготовки учасників [3; 4].

У цьому контексті актуальним є розроблення практико-орієнтованих платформ, що легко інтегруються в освітній процес. У статті представлено реалізацію такої системи кіберполігону, побудованої за контейнерним підходом з індивідуалізованим запуском середовищ для кожного користувача. Платформа інтегрована в навчальну LMS (Moodle) через стандарт LTI, що забезпечує єдиний освітній простір для теорії та практики. Нижче розглянуто сучасні підходи до створення кіберполігонів, детально описано архітектуру нашого рішення, проаналізовано технічні виклики і шляхи їх вирішення, а також обговорено переваги і обмеження підходу [3; 1; 6].

Огляд підходів до побудови кіберполігонів

Існує декілька концептуальних підходів до організації навчальних кіберполігонів. Розглянемо ключові з них:

- Полігони з теоретичними матеріалами та симуляційними завданнями. У таких системах навчальна платформа поєднує текстові або мультимедійні матеріали з кібербезпеки і інтерактивні симуляції або вправи. Студенти спочатку вивчають теорію, після чого виконують практичні завдання в середовищі, що моделює певні ситуації (наприклад, пошук вразливостей, аналіз трафіку тощо). Подібна інтеграція теорії з практикою покликана забезпечити міцніший зв'язок між знаннями і навичками. Дослідження показують, що інтерактивне навчання, яке підкріплює теоретичний матеріал практичними вправами, підвищує засвоєння і зацікавленість студентів. Прикладом можуть бути онлайн-лабораторії, де кожен розділ курсу супроводжується вбудованою практичною вправою або симулятором.
- Системи з інтерактивними підказками (віртуальні асистенти): більш просунуті кіберполігони впроваджують віртуальних наставників — інтерактивні підказки чи боти, що допомагають користувачам у процесі виконання завдань. Такий підхід часто використовує елементи штучного інтелекту або сценарних підказок, які реагують на дії учня. Змагання Capture The Flag (CTF) традиційно застосовують систему підказок із вирахуванням балів, що, як показує досвід, мотивує учасників знаходити рішення самостійно, але може стримувати тих, хто потребує допомоги. Інтерактивні асистенти всередині навчального середовища здатні адаптивно скеровувати студента, пояснювати помилки і пропонувати додаткові ресурси, що підвищує ефективність навчання. Подібні можливості починають з'являтися у сучасних платформах кібербезпеки, поєднуючи практику з елементами інтелектуальної підтримки [3; 4].

Історично перші кіберполігони будували з використанням віртуалізації на рівні гіпервізора. Кожному студенту надається одна або кілька віртуальних машин (ВМ), що містять необхідні інструменти або вразливі системи. Перевагою цього підходу є повна ізоляція середовища для кожного користувача та можливість емулювати різні операційні системи (Linux, Windows тощо). ВМ забезпечують високий рівень реалізму: студент працює ніби з окремим фізичним комп'ютером. Проте недоліками є значні витрати ресурсів і часу на розгортання таких середовищ. Кожна ВМ містить повноцінне ядро ОС та усі служби, тому десятки паралельних ВМ швидко споживають ресурси сервера. Час запуску ВМ також довший порівняно з контейнерами. За даними досліджень, Docker-контейнери демонструють принаймні на 50 % швидший старт і обслуговування запитів, ніж еквівалентні віртуальні машини. Контейнеризація загалом забезпечує кращу ефективність виконання сценаріїв порівняно з класичними ВМ за рахунок меншого накладного навантаження. Отже, ВМ-полігони добре підходять для складних сценаріїв, що потребують повної емуляції систем, але їх масштабування на велику кількість слухачів обмежене та потребує суттєвих потужностей або хмарних ресурсів [3; 1].

Сучасним трендом у побудові кіберполігонів є використання контейнеризації — ізоляції середовищ на рівні операційної системи. Контейнери (наприклад, Docker) дають можливість запускати декілька ізольованих середовищ на спільному ядрі ОС, що значно знижує витрати ресурсів на кожне середовище. На відміну від ВМ, контейнер не містить власного ядра, а використовує ядро хостової системи, завдяки чому він «легший» і запускається майже миттєво. Це дає змогу на одному фізичному сервері розгорнути набагато більше одночасних ізольованих середовищ без втрати продуктивності. Контейнерні образи легко переносити між різними машинами, що спрощує розгортання навчальних сценаріїв на різних майданчиках. Таким чином, контейнерний підхід здатний забезпечити як високу щільність розміщення середовищ, так і швидке їх оновлення чи скидання до початкового стану. Відомим прикладом є фреймворк Labtainers, що пропонує набір навчальних лабораторій із кібербезпеки на базі Docker-контейнерів; він спрощує підготовку середовищ для викладача, пакуючи всі налаштування в образ, однак не підтримує деякі розширені можливості (командну роботу, аналітику прогресу тощо) [3; 1; 5].

Рівень автоматизації і масштабованості платформ

Кіберполігони також класифікують за ступенем автоматизації розгортання середовищ і управління ними. Прості реалізації можуть потребувати ручного налаштування кожного середовища: наприклад, інструктор заздалегідь готує образ ВМ чи контейнера і видає його студентам. Такі рішення не масштабуються на великі аудиторії, оскільки адміністрування десятків / сотень копій середовища стає трудомістким і схильним до помилок. Натомість сучасні платформи впроваджують оркестрацію — автоматизоване управління життєвим циклом середовищ: їх розгортання, налаштування, запуск, зупинку та очищення. Дослідження показують, що рішення, побудовані з опорою на інфра-

структурні платформи та оркестрацію, забезпечують кращу ізоляцію і надійність середовищ, тоді як полігони на основі ad-hoc інфраструктури без оркестрації можуть мати слабші механізми ізоляції і потребують більше ручного втручання. Отже, за критерієм автоматизації платформи варіюються від локальних скриптів розгортання окремих лабораторій до комплексних хмарних сервісів, здатних автоматично масштабуватися під навантаження. При виборі підходу важливо врахувати баланс між контролем над середовищем і витратами часу на його підтримку: високоавтоматизовані рішення знижують навантаження на адміністраторів і мінімізують людський фактор, проте можуть вимагати складнішого початкового налаштування [3; 1].

Опис реалізованої системи

Розроблена нами система кіберполігону призначена для інтеграції в освітній процес і забезпечення кожному студенту ізольованого практичного середовища. Архітектурно рішення складається з вебзастосунку (серверної частини), що реалізований на мові PHP із використанням фреймворку Yii, та системи контейнеризації Docker для запуску навчальних середовищ. Вебзастосунок розгорнуто окремо від LMS, але інтегрований у Moodle за допомогою LTI (Learning Tools Interoperability). LTI — це стандарт IMS Global, який дозволяє підключати зовнішні навчальні інструменти до платформ на кшталт Moodle, Canvas тощо. У нашому випадку Moodle виступає як LTI-споживач (Platform), а вебзастосунок кіберполігону — як LTI-інструмент (Tool). Така інтеграція забезпечує єдину автентифікацію: користувачі заходять до Moodle під своїми обліковими записами і запускають кіберполігон через активність зовнішнього інструменту, без потреби окремого введення логіна та пароля на стороні полігону. Moodle передає нашому застосунку необхідні дані про користувача та курс (через підписаний LTI-запит), завдяки чому відбувається SSO (Single Sign-On) — студент одразу потрапляє у свій особистий кібер-лаб без додаткової авторизації. Перевагою такого підходу є безшовний користувацький досвід: із інтерфейсу LMS студент одним кліком відкриває практичну лабораторію, а викладач може за необхідності отримувати зворотні дані (результати, оцінки) назад у Moodle через сервіси LTI Advantage, зокрема Assignment and Grade Services [3; 1; 6].

Після переходу користувача до завдання кіберполігону система автоматично розгортає для нього індивідуальне середовище у вигляді Docker-контейнера. Для кожного завдання підготовлено шаблонний контейнерний образ, що містить усе необхідне програмне забезпечення, вразливі вебдодатки, скрипти тощо — залежно від сценарію. Вебзастосунок через Docker API запускає новий контейнер із цього образу, ізолюючи його мережеві та файлові ресурси. В результаті студент отримує власний екземпляр середовища для виконання завдання. Індивідуалізація реалізована через запуск кожного контейнера з унікальними параметрами, згенерованими для конкретного користувача. Зокрема, реалізовано механізм передання у контейнер секретного прапорця (flag) та інших змінних оточення. Прапорець — це спеціальний рядок символів, який слугує індикатором успішного виконання завдання (типово використовується у CTF-завданнях). У нашій системі для кожного запуску генерується випадковий унікальний flag, який передається в контейнер як змінна середовища (FLAG=<значення>). Всередині контейнера цей прапорець може бути автоматично записаний у певне місце (наприклад, файл у системі або рядок у банері програми) чи використовуватися сервісом перевірки. Студент, успішно виконавши завдання (знайшовши вразливість або експлуатувавши її), повинен виявити цей секретний рядок і подати його як доказ виконання. Модель прапорців гарантує, що кожен студент працює з унікальним розв'язком: навіть якщо двоє користувачів спробують обмінятися відповідями, у кожного прапорець інший, отже чужий результат не зарахується. Такий підхід широко використовується в навчальних CTF-іграх і довів ефективність у запобіганні шахрайству, дозволяючи при цьому співпрацю (студенти можуть обмінюватися ідеями, але не готовими відповідями) [3; 1; 6].

Технічні виклики та рішення

У типовому навчальному сценарії десятки студентів можуть запускати свої контейнерні середовища одночасно (наприклад, під час лабораторної роботи або заліку). Це висуває підвищені вимоги до серверної інфраструктури. Контейнери значно ефективніше використовують ресурси порівняно з ВМ, тому наш сервер здатен підтримувати в рази більше паралельних середовищ без деградації продуктивності. Проте апаратні обмеження все одно існують — насамперед за оперативною пам'ят-

тю та процесорними ядрами. Щоб запобігти перевантаженню, у системі встановлено ліміти на ресурси для кожного контейнера (через cgroups Docker конфігурується максимальний обсяг RAM і доля CPU).

Наприклад, типовому вебзастосунку може виділятися не більше ніж 512 МБ пам'яті і 20 % від одного процесорного ядра. Це унеможливлює ситуацію, коли один некоректно працюючий або зламаний контейнер вичерпує всі ресурси хоста. В перспективі, інтеграція з оркестраторами на кшталт Kubernetes або OpenStack дозволить автоматично додавати вузли при збільшенні числа користувачів, тобто реалізувати еластичне масштабування кіберполігону [3; 1].

Управління життєвим циклом контейнерів

Автоматизований запуск персонального середовища має доповнюватись так само автоматизованим зняттям цих середовищ по завершенні роботи. Якщо залишати контейнери запущеними невизначений час, це призведе до витоку ресурсів (невикористані контейнери займатимуть пам'ять і портів), а також потенційних ризиків безпеки. В нашій реалізації для кожного контейнера встановлено таймаут сесії — 2 години з моменту запуску. Після спливу цього часу контейнер автоматично зупиняється і видаляється, якщо студент не запросив продовження. Крім того, при добровільному виході користувача із завдання (натисканні кнопки «Завершити лабораторію») його середовище також згортається, щоб уникнути залишення неактивних або незавершених контейнерів у системі. Для реалізації цього механізму серверний застосунок має планувальник завдань, який періодично перевіряє час життя активних контейнерів. Управління здійснюється через Docker API: виконується команда зупинки і видалення контейнера. Важливо, що усі дані всередині контейнера після його видалення втрачаються (система побудована за принципом ephemeral environments — кожен запуск дає чистий стан). Якщо студенту потрібно повторити завдання, запускається новий екземпляр із початковим станом. Такий підхід спрощує підтримку (не треба зберігати проміжний стан) і усуває проблему накопичення змінених середовищ. Автоматизація більшості таких операцій значно знижує навантаження на адміністратора полігону та забезпечує стабільну роботу платформи [1; 2].

Одним із пріоритетів при розробці було мінімізувати час між натисканням студентом кнопки «Запустити лабораторію» і готовністю середовища до роботи. За рахунок використання контейнерів цей час вдалося звести до десятків секунд або менше (для порівняння, запуск повноцінної VM може тривати кілька хвилин). Щоб прискорити cold start (перший запуск образу), всі необхідні контейнерні образи завчасно завантажуються та кешуються на сервері. Зокрема, перед початком курсу адміністратор розгортає (або оновлює) останні версії образів завдань, тож під час заняття не витрачається час на скачування образу з реєстру. Для великих образів (>1 GB) ми використовували шарувату структуру: спільні базові шари (наприклад, ОС Ubuntu) шаряться між різними образами і завантажуються одноразово. Крім того, Docker демон підтримує копію-на-запис (CoW), що дозволяє запускати багато контейнерів із одного образу, використовуючи пам'ять ефективно: незмінні частини образу займають пам'ять тільки раз, а для кожного контейнера виділяються тільки його унікальні змінені блоки. Це особливо корисно, коли десятки студентів працюють з ідентичним середовищем: RAM витрачається значно менше, ніж якби кожен мав окрему VM. Warm start — повторні запуски того самого образу — здійснюються майже миттєво завдяки кешуванню. Таким чином, продуктивність системи в реальному часі оптимізована для інтенсивної роботи у класі [1].

Особливу увагу приділено безпеці ізольованих середовищ. Хоча Docker-контейнери ізольовані від хостової системи, теоретично існують ризики escape-атак, коли зловмисник усередині контейнера може отримати доступ до системи хоста через уразливості ядра або неправильну конфігурацію. Зважаючи на це, ми дотримувалися принципу найменших привілеїв: усі контейнери запускаються від непривілейованого користувача (відсутність root-доступу всередині контейнера, якщо це не потрібно за сценарієм), без підвищених привілеїв Docker (опція --cap-drop=ALL відсікає небезпечні привілеї ядра), без монтування зовнішніх директорій хоста. Мережевий рушій налаштовано так, що кожен контейнер працює в окремому Docker network, і між контейнерами різних користувачів немає прямого трафіку. Це унеможливлює атаки між студентами або перехоплення трафіку. Кожне середовище міститься у своєму ізольованому просторі адрес, що моделює реальну мережеву сегментацію. Усі ці заходи підвищують рівень ізоляції і забезпечують, що навчальні атаки залишаються в межах «пісочниці» полігону. Практика показала, що за належних налаштувань контейнерний підхід

є достатньо безпечним для освітніх цілей, тоді як його переваги у швидкодії та легкості розгортання суттєво переважують залишкові ризики [1].

Інтеграція з Moodle та підтримка безперервного користувацького досвіду

При впровадженні системи важливо було зробити її максимально непомітною для кінцевого користувача — студента. Інтеграція через LTI забезпечила єдиний вхід, але окрім цього ми подбали про узгодженість інтерфейсу та навігації. Застосунок кіберполігону вбудовується у вікно Moodle за допомогою iframe, що забезпечує безперервність інтерфейсу для студента та збереження контексту LMS. Верхня панель полігону містить навігаційні елементи Moodle (логотип, назву курсу тощо) для збереження контексту. LTI-запит передає інформацію про курс і завдання, тому наш застосунок може відображати назву поточної лабораторної роботи, інструкції і навіть оцінку, отриману студентом, синхронно з курсом. Викладачі інтегрують кіберполігон як звичайний модуль курсу Moodle (через активність "External Tool"), що дозволяє використовувати всі стандартні можливості LMS — обмеження за датою, умови відкриття, журнали оцінок тощо. З технічного боку, нам довелося вирішити питання відповідності протоколу LTI останньої версії (1.3 / LTI Advantage) і сумісності з Moodle. Ми реалізували необхідний OAuth 2.0 потік і перевірку підписів JWT, якими Moodle постачає дані про користувача при запуску інструменту. Безперервність користувацького досвіду також означає швидке повернення з полігону до інших матеріалів курсу. Цього досягнуто завдяки тому, що полігон відкривається в тій самій вкладці браузера і має кнопку виходу, яка просто закриває фрейм. Із погляду студента, робота в кіберполігоні нічим не відрізняється від роботи з вбудованим модулем курсу [3; 6].

Висновки та подальші напрями роботи

У статті описано реалізацію практико-орієнтованої системи кіберполігону, інтегрованої в освітнє середовище через Moodle і LTI, що дозволяє надавати кожному студенту персональне ізольоване середовище на основі Docker-контейнерів. Такий підхід вирішує одразу кілька задач: студенти отримують реалістичний досвід роботи з вразливими системами та атаками у безпечних умовах; викладачі можуть масово розгортати вправи без ручної підготовки кожної машини; навчальний процес стає більш інтерактивним і наближеним до реальних викликів кібербезпеки. Інтеграція через LMS забезпечує зручність для кінцевих користувачів і вписує практичні заняття в навчальні плани без швів [3; 1; 6].

Перспективи розвитку системи охоплюють декілька напрямів. Сценарії командної гри (Red/Blue team) — наступний логічний крок у еволюції полігону. Як показує досвід, ефективна підготовка фахівців із кібербезпеки має передбачати командну взаємодію і змагальний елемент. Ми плануємо реалізувати можливість створення спільних віртуальних середовищ для двох і більше груп студентів, де одна команда виконує роль атакуючих (Red Team), а інша — захисників (Blue Team). Технічно це потребуватиме розгортання багатокомпонентних середовищ: декількох контейнерів (або й комбінації контейнерів із повноцінними ВМ) в одній віртуальній мережі, з розподілом доступу між командами. В рамках такої розширеної моделі ми розглядаємо інтеграцію з інструментами оркестрації і моніторингу, щоб масштабні навчальні кібербитви залишалися керованими і відтворюваними [1].

Другим напрямом є розробка модуля аналітики для викладача. Зараз інструктор отримує досить обмежений зворотний зв'язок. Такий інструмент міг би збирати анонімізовані дані з контейнерів (наприклад, командні логи або журнали подій системи) і візуалізувати їх у зручній формі [1].

На завершення, розвиток кіберполігонів як навчального інструменту відкриває нові горизонти для ІТ-освіти. Представлена система показує, як сучасні технології контейнеризації та стандарти інтеграції (LTI) дають можливість створити гнучке, масштабоване і персоналізоване середовище для здобуття практичних навичок. Подальше вдосконалення платформи — впровадження командних сценаріїв, розумної аналітики і глибшої інтеграції з інфраструктурою безпеки — зробить навчальні кіберполігони ще більш наближеними до реальних умов роботи кіберфахівців. Це сприятиме підготовці нового покоління спеціалістів, озброєних не лише теоретичними знаннями, а й досвідом реагування на сучасні кіберзагрози у безпечному навчальному просторі [3; 1; 7; 6].

Список літератури

1. Chouliaras N. A novel autonomous container-based platform for cybersecurity training and research / N. Chouliaras, N. Karanikolas, T. Diamantopoulos, S. Gritzalis // PeerJ Computer Science. — 2023. — Vol. 9. — e1574.
2. Cloud Range. What is a Cyber Range? [Electronic resource]. — FAQ. — 2023. — Mode of access: https://cloudrangecyber.com/ (date of access: 25.06.2025).
3. Lazarov W. Lessons Learned from Using Cyber Range to Teach Cybersecurity at Different Levels of Education [Electronic resource] / W. Lazarov // Technology, Knowledge and Learning. — 2025. — Mode of access: https://doi.org/10.1007/s10758-025-09840-y.
4. O'Rourke G. Unique Challenges for CTFd – GitHub repository plugin [Electronic resource] G. / O'Rourke. — 2021. — Mode of access: https://github.com/ (date of access: 25.06.2025).
5. Thompson M. F. Labtainers: A Docker-based framework for cybersecurity labs / M. F. Thompson, C. E. Irvine // Proceedings of the 2021 ACM SIGCSE. — 2021.
6. Virginia Cyber Range. Cyber Range LTI Integration. — Knowledge Base. — 2023.
7. What Is Missing in Traditional Cybersecurity Training Programs? [Electronic resource]. — Cloud Range Cyber. — 16.08.2024. — Mode of access: https://cloudrangecyber.com/ (date of access: 25.06.2025).

References

Chouliaras, N., Karanikolas, N., Diamantopoulos, T., & Gritzalis, S. (2023). A novel autonomous container-based platform for cybersecurity training and research. *PeerJ Computer Science*, *9*, e1574. https://doi.org/10.7717/peerj-cs.1574.

Cloud Range. (2023). What is a cyber range? — FAQ. *Cloud Range Cyber*. https://cloudrangecyber.com/.

Cloud Range. (2024, August 16). What is missing in traditional cybersecurity training programs? *Cloud Range Cyber*. https://cloudrangecyber.com/.

Lazarov, W. (2025). Lessons learned from using cyber ranges to teach cybersecurity at different levels of education. Technology, Knowledge and Learning. Advance online publication. https://doi.org/10.1007/s10758-025-09840-y.

O'Rourke, G. (2021). Unique challenges for CTFd [Computer software]. *GitHub*. https://github.com/CTFd/CTFd.

Thompson, M. F., & Irvine, C. E. (2021). Labtainers: A Docker-based framework for cybersecurity labs. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 123–129). Association for Computing Machinery. https://doi.org/10.1145/3408877.3432403.

Virginia Cyber Range. (2023). Cyber Range LTI integration — Knowledge base. https://virginiacyberrange.org/knowledgebase/lti.

A. Hlybovets, T. Babych

IMPLEMENTATION OF A PRACTICE-ORIENTED CYBER RANGE SYSTEM WITH INDIVIDUALIZED ENVIRONMENT DEPLOYMENT

The growing complexity and frequency of cyberattacks require a transformation in cybersecurity education. Traditional approaches, which focus primarily on theoretical knowledge, certification, and compliance, often fail to provide learners with sufficient hands-on experience. To address this gap, cyber ranges have emerged as an effective tool for practical, scenario-based training in a safe and controlled environment. This paper presents the implementation of a modern cyber range system designed specifically for educational purposes. The platform is container-based and integrates seamlessly into the learning process via the LTI standard, enabling single sign-on and data exchange with the Moodle LMS. Each student receives an isolated environment on demand, with auto-generated flags embedded for individual task verification. The system automatically deploys and destroys containers based on session activity, ensuring efficient use of server resources and maintaining a secure, ephemeral environment for practice. The solution also includes network segmentation and privilege restrictions for each container, enhancing security while simulating real-world conditions. The proposed architecture supports scalability and rapid deployment, making it suitable for use in classrooms, exams, and self-paced learning. In this article, we review existing approaches to building cyber ranges, discuss the design and implementation of our system, and highlight the technical and pedagogical benefits. Future developments includes support for team-based Red/Blue training, instructor analytics modules, and orchestration integration. The presented platform bridges the gap between theoretical training and real-world cybersecurity skills development, helping to prepare students for modern digital threats.

Keywords: cyber range, cybersecurity education, practical training, Capture the Flag (CTF), containerization, Docker, Moodle, LTI integration, interactive learning, educational platform.

Матеріал надійшов 01.07.2025