OmniaScience

Journal of Technology and Science Education

JOTSE, 2026 – 16(2): 503-519 – Online ISSN: 2013-6374 – Print ISSN: 2014-5349

https://doi.org/10.3926/jotse.3696

DOCKER CONTAINER-BASED INFRASTRUCTURE PENTESTING LABORATORIES FOR CYBERSECURITY EDUCATION

William-Rogelio Marchand-Niño*, Raquel-Esther Arias-Ramos

Universidad Nacional Agraria de la Selva (Perú)

*Corresponding author: william.marchand@unas.edu.pe
raquel.arias@unas.edu.pe

Received July 2025
Accepted April 2026

Abstract

This study considers the expanding gap in cybersecurity skills and the need for practical learning environments for the training of professionals. It investigates the influence of Docker container–based infrastructure pentesting laboratories on cybersecurity learning, comparing them with virtual machine–based laboratories. The methodology that was used was a quasi-experimental design with two groups of students; one applying containers, and the other with virtual machines. The laboratories were designed with vulnerable services, such as FTP, SSH, databases and HTTP, including tools such as Metasploitable, DVWA and Kali Linux. The flexibility of the laboratory (configuration and deployment time) was measured, in addition to the learning of cybersecurity in its procedural and attitudinal knowledge dimensions. The results show that Docker containers are more convenient for configuring and preparing laboratories as compared to virtual machines, due to their lightweight nature and lower computational resource requirements. However, no statistically significant differences were found in the procedural or attitudinal learning between the two groups. The lack of a significant impact on learning is attributed to the technical familiarity of the students with the handling of containers and the complexity of the activities. However, the containers represent an attractive alternative, due to their efficiency in the use of computational resources.

Keywords – Containers, Cybersecurity, Learning, Procedural knowledge, Attitudinal knowledge.

To cite this article:

Marchand-Niño, W. R., & Arias-Ramos, R. E. (2026). Docker container-based infrastructure pentesting laboratories for cybersecurity education. *Journal of Technology and Science Education, 16*(2), 503–519. https://doi.org/10.3926/jotse.3696

---

1. Introduction

The growing importance of cybersecurity has revealed the emergence of challenges, such as the disparity between the supply and demand of professionals trained to deal with the increasing number of cyber attacks (Morgan, 2023), in which there is a skills gap and a lack of competencies required among candidates (Sausalito, 2024).

In this sense, the training of professionals with practical cyber security skills requires adequate learning environments, such as configured laboratories, whose continuous implementation can consume significant

effort and resources (Knöchel et al., 2021). These laboratories may be based on virtual machine or container environments, with some differences, primarily in the configuration and set-up times from the instructor perspective (Zhang et al., 2018).

On the other hand, a disconnection persists between theoretical and practical learning in the field of IT, and this gap reflects the difficulty of preparing students with the necessary skills demanded by today's technological industry (Dubec et al., 2023).

The technology gap also stands out due inadequate technological equipment available for learning in computer science disciplines (Kristian & Nascimento-e-Silva, 2024), particularly in geographic areas with limited economic income (Amjad et al., 2024). This scenario represents the need to design alternatives to remedy the educational inequality and the digital divide with long-term investments, such as in adaptable learning environments that are accessible to everyone (Assefa et al., 2025).

Likewise, studies have been conducted on the integration of containers, especially Docker containers, in cybersecurity education to meet the need for practical, dynamic environments with reduced complexity in terms of laboratory preparation (Sedov & Lazarev, 2024), as in the case of using containers prepared for the recovery and analysis of fraud data involving credit cards, using logistic regression algorithms, eliminating the process of configuring and obtaining data, so that the student can concentrate on the learning content (Shahriar et al., 2020).

The use of containers is relatively independent from the operating system the user employs, although the performance may be slightly better on Linux operating systems than on Windows systems (Sergeev et al., 2022); similar to this, the ease of use in the delivery and creation of software surpasses that of virtual machines (Maruszczak et al., 2022; Li, 2021), which makes it possible to transfer these advantages to cybersecurity laboratory environments.

There are cases such as the development of tools with a certain level of automation for the creation of cybersecurity scenarios for education, using Docker containers, that show better performance (CPU, RAM memory and storage usage) as compared to virtual machines (Potdar et al., 2020; Nakata & Otsuka, 2021; Ionescu et al., 2019), but it must be kept in mind that learning also depends on the familiarity of the students with the specific technology (Nakata & Otsuka, 2021). However, it must also be considered that container platforms have their own vulnerabilities that must be taken into account for the execution of practical exercises in safe scenarios (Tomar et al., 2020; Tunde-Onadele et al., 2019; Zhu & Gehrmann, 2021).

This research study revolves around the question: How do Docker container-based infrastructure pentesting laboratories influence the learning of cybersecurity skills? The dimension of the learning considered were the procedural and metacognitive knowledge, as they are related to the practical aspects of the learning process.

2. Literature Review

2.1. Virtualization and Containers

Virtualization and containers are technologies in modern computing. Virtualization is based on the principles of creating virtual instances of hardware or operating systems using specialized software referred to as a hypervisor (Mishra et al., 2020). These hypervisors allow multiple operating systems or virtual machines (VMs) to be run on a single physical server, optimizing the use of IT resources (Susen et al., 2022).

The hypervisor manages the allocation of virtualization resources based on the available host resources; there are generally two common types of hypervisors: the first one is installed directly on the hardware, with an operating system that belongs to the hypervisor itself; and the second is an application that is run on the host operating system, using its functions to abstract hardware resources and manage the virtual machines (Cardwell, 2016).

In virtualization, the physical resources are abstracted, logically representing the CPU, RAM, storage, network and software, including the operating systems, file systems and applications (Hess & Newman, 2010); this result is referred to as a virtual machine, which also generally meets the criteria of efficiency, equivalence and resource control (Popek & Goldberg, 1974).

The taxonomy of the virtualization is diverse, but the one relevant to this research study is that proposed by Buyya et al. (2013), popularly known as hyper-V hardware virtualization, the product of which is a virtual machine that consists of an operating system and applications.

On the other hand, the containers are considered to be a type of virtualization on the operating system level, since they are run on the kernel of a specific operating system, although they can be less secure than the virtualization of hypervisors (Turnbull, 2019); in contrast, the attack surface is smaller, as they are lighter than virtual machines (Mouat, 2015). Another characteristic of the containers, in addition to quick initialization, is that they run applications compatible with the host operating system, without the need for additional virtualization (Ionescu et al., 2019). This design eliminates the need for a guest operating system, making the containers more efficient in the use of IT resources (Baresi et al., 2024).

Containers are ideal for microservices and native cloud applications, permitting quick scaling and orchestration, using tools such as Kubernetes (Abhishek et al., 2022). In the field of software development, containers are used to encapsulate the resources that are needed (framework, libraries, configuration, etc.) to later route them to the testing or operational phases without any concern for the environment in which the developed application must run correctly (Schenker, 2023).

2.2. Penetration Testing (Pentesting) in Cybersecurity

Penetration tests, generally referred to as "pentesting", consist of a systematic process to evaluate the security of the systems, networks or applications of an organization by simulating real attacks. The main objective it to identify vulnerabilities in the IT structure or applications, in terms of configurations, code or design flaws that generate security weaknesses (Tyshyk & Hulak, 2024)

This evaluation process must be governed by ethics and controlled testing so that it does not affect the normal system performance (Naga-Suneetha et al., 2025), and it must have explicit consent from the target organization, ensuring the legality of the security testing (Mohan et al., 2022).

In order to identify vulnerabilities through penetration tests, real attacks are simulated using tools and techniques that are also used by cybercriminals (Lopez de Jiménez, 2016; Wylie & Crawley, 2021). This process must be carried out after signing a confidentiality agreement (Oriyano, 2017).

This study focuses on the type of testing addressed, specifically black box and grey box testing, which are implemented in the laboratories and in which the amount of information available about the object being evaluated is limited (Lopez de Jiménez, 2016).

In order to properly conduct a penetration test, it must be supported by a methodology with defined stages, such as reconnaissance, scanning, exploitation and persistence (Engebretson, 2011) which can also be replicated in learning and laboratory environments.

2.3. Competency-Based Learning in A University Setting

Learning is a long-lasting change in behavior or in the capacity to behave in a certain way, which is the result of practice or other forms of experience. Learning also involves a change in the capacity to behave, and it is inferential in nature, since it is not directly observable, but rather inferred through its products or results. In addition, it emphasizes that it must last over time (Schunk, 1997).

From the perspective of laboratory-based learning, it refers to the process through which the students acquire knowledge and skills through direct experience and practice, which makes this process highly interactive and focused on student participation, real problem solving and the reflection on the

experiences. This is measured in terms of both theoretical comprehension and the capacity to apply knowledge in practical situations (Davies, 2008).

In the university setting, learning is based on developing different ways of thinking, including analytic thinking, critical thinking, deliberative thinking, creative thinking and practical thinking to form a more orderly mind capable of structuring information and globalizing knowledge (Villa & Poblete, 2007). However, for the purposes of this study, competency-based learning was considered, which breaks the learning down into smaller units, permitting the students to learn at their own pace and move on to the next level only after demonstrating mastery of each competency. It is also known as results-based learning or skill-based learning (Wyne et al., 2021).

The skills relevant to this study are characterized by three different types of learning: learning to know, learning to do and learning to be (Benítez-Ayala, 2022). Conceptual capacities correspond to learning to know, which focuses on enabling students to develop the ability to understand the world around them, and, as a goal of human life, it is justified by the enjoyment derived from understanding, knowing and discovering. It is measured by evaluating the knowledge of content, through multiple choice questions, two-option items (true or false) and using questionnaires for the written test.

The procedural skills are related to learning to do, and take into account the capacity to deal with and solve problems, as well as the skill to create products according to the context, involving processes, procedures and strategies that students must understand in order to perform a task; it is measured by problem solving, laboratory activities, practical tests and mini-projects based on situational activities, etc. (Ilidzhev & Ibragimova, 2018).

Finally, attitudinal skills fall under learning to be; they are focused on self-awareness, values, self-control, responsibility, attitudes and the capacity to make autonomous decisions (Schunk, 1997); these are also referred to as metacognitive learning, which concerns the emotional and psychological aspect of learning, specifically, the belief that the students have about their own capacity to learn and improve a skill or knowledge, with this being an internal transformation that occurs when they transition from thinking "I can't do this" to "I am achieving it" or "I am capable of doing it". It is measured through numerous methods, such as questionnaires, interviews, think-aloud protocol analysis, observations, stimulated recall, online computer-based recording, eye-tracking recording, etc. (Veenman et al., 2006)

3. Methodology

A quasi-experimental design with a control group was used, as the participants were undergraduate students in the Computer and Systems Engineering program enrolled in two independent sections of a cybersecurity course (21 and 28 students, respectively). It was arbitrarily decided that the experimental group would be the group of 21 students and the control group would be the second group of 28 students. This condition justified the selection of a quasi-experimental approach. Figure 1 shows the stages carried out for the research study.

The students had basic knowledge of networks and servers acquired in previous courses; however, their experience in the use of virtual machines or containers was not homogeneous. The selected design made it possible to manipulate the independent variable in a controlled manner and evaluate its effect on learning through the measurement of the indicators following the intervention. Figure 1 shows the stages carried out for the research study.

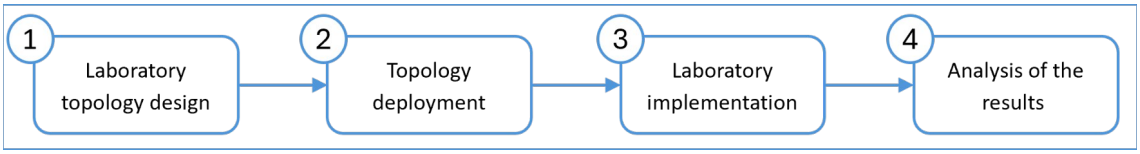

Figure 1. Execution methodology

3.1. Stage 1: Pentesting Laboratory Topology Design

The pentesting laboratory topology was designed according to two scenarios, the first based on Docker containers and the second using virtual machines. Each scenario was deployed independently on the personal computers of the students in the corresponding study groups (experimental and control), establishing communication within the local virtual networks. VMware Workstation Pro or Oracle VirtualBox hypervisors were installed on each computer, selected according to the preference of each student. They allowed a base operating system to be run to house the containers or multiple operating systems as virtual machines. The network services deployed in both scenarios were FTP, remote access (SSH), database (MySQL) and web (HTTP) services; each of these services had common, exploitable vulnerabilities.

The laboratories (scenarios) were deployed identically on each computer assigned to the students, allowing each student to conduct the tests independently, in other words, without affecting the other computers.

In the first scenario, Docker containers were deployed on each student's personal computer, which acts as the host for the laboratory environment. A base virtualized operating system (Linux) was deployed on this machine to run the following Docker containers: Metasploitable version 2; the DVWA web application version 1.10 (Damn Vulnerable Web Application); the Kali Linux operating system version 2024.1, with tools and utilities for penetration testing; and a Wireshark container for network monitoring tests. All of them are housed on a single computer and communicate with one another over an internal virtual Docker network. For this purpose, a custom network was created using the command "sudo docker network create seginformatica", and each container was assigned to the network using the "--network=seginformatica" parameter, which allowed controlled interaction between services and tools.

Likewise, the environment design was focused on the local, controlled execution of the tests, limiting communication to the exchange of traffic within the established virtual network. In this way, the host operating system only served as the execution platform, without being included in the attack targets or involved in the testing process. Consequently, all activities were carried out within each student's virtual laboratory environment. Figure 2 represents the first scenario.

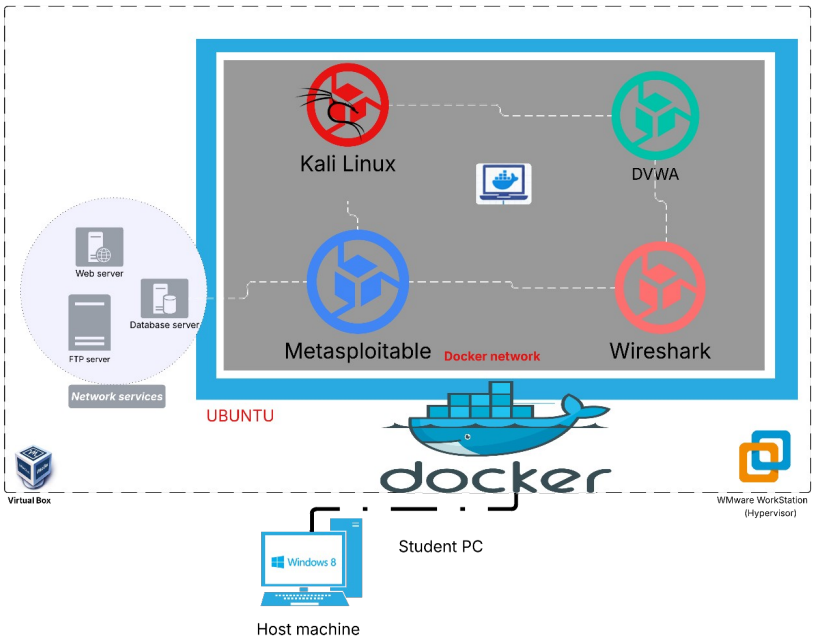

Figure 2. Pentesting laboratory topology based on Docker containers

The second scenario consisted of virtual machines, on which the same services were included as in the first scenario. Two virtual machines were used, one for the Metasploitable system, which includes the vulnerable network services, and the other for the Kali Linux system, as the attach device, integrating Wireshark and the DVWA application for web application testing. As in the first scenario, the virtual machines were deployed for each student on their assigned computer. The virtual machines were configured with network interfaces in NAT or bridge mode for communication between them in the virtual environment. Like in the first scenario, the activities were performed within the virtual laboratory in a controlled manner, without any intervention from the host operating system during the pentesting operations. Figure 3 shows the topology design.

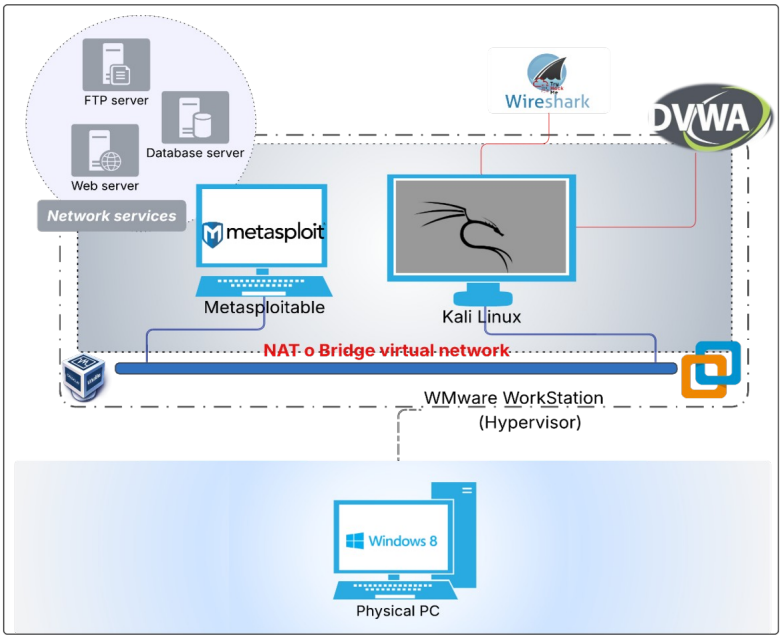

Figure 3. Pentesting laboratory topology based on virtual machines

3.2. Stage 2: Deployment of the Pentesting Laboratory Topology

This stage consisted of preparing the scenarios for the testing. For the first scenario, containers were deployed from four images downloaded from the Docker Hub repository, specifically Metasploitable version 2 (https://hub.docker.com/r/tleemcjr/metasploitable2), Kali Linux 2024.1 (https://hub.docker.com/r/raquelars/kalilinux-tools1), DVWA version 1.10 (Damn Vulnerable Web Application) (https://hub.docker.com/r/raquelars/dvwa) and Wireshark version 4.1.0 (https://hub.docker.com/r/linuxserver/wireshark). The Metasploitable container includes different vulnerable network services, such as FTP (VSFTPD v2.3.4 and ProFTPD v1.3.5), SSH and HTTP; the Kali Linux incorporates common tools, such as John the Ripper, Hydra, Medusa, Metasploit, among others; Likewise, a DVWA container was deployed for web application testing with common vulnerabilities, such as SQL injection and XSS, among others. Finally, a Wireshark container was used for network traffic monitoring.

In the second scenario, ISO were downloaded from the Metasploitable version 2 (https://sourceforge.net/projects/metasploitable/files/) and Kali Linux version 2024.1 (https://www.kali.org/get-kali/#kali-virtual-machines) systems. Two virtual machines were implemented: Metasploitable was configured as the victim machine, and the second virtual machine was MV Kali Linux. This virtual machine also included the DVWA 1.10 application (https://github.com/digininja/DVWA) and the Wireshark traffic analyzer. The virtual machines were configured within a network in NAT or bridge mode.

3.3. Stage 3: Application of the Pentesting Laboratory

In this stage, the activities were carried out to measure the study variables with the two established groups. The activities (including the installation and deployment of the scenarios) were executed by each of the students in the cybersecurity course, using a basic installation guide that was previously given to them, in both the experimental group and in the control group. In both groups, the students had basic theoretical and practical knowledge about virtualization acquired in previous courses. The tests were run measuring the indicators from the flexibility dimension, the downloading and configuration of Docker images and virtual machines, leaving the environments ready for the evaluation of the "cybersecurity learning" variable, which included practical evaluations. The experimental group worked on the scenario based on Docker containers, while the control group completed the exercises with the virtual machines. During the performance of the activities, the data corresponding to the indicators defined for each variable were recorded.

3.3.1. Measuring the "Laboratory Flexibility"

The indicators used to measure the flexibility of the laboratory were "average configuration time of laboratory components" and "service deployment time". For the measurement of the first indicator in the experimental group, the exercise was considered to begin with the students downloading the container images from Docker Hub and ending with the execution of the container. For the control group, the task was similarly defined as starting with the download of the virtual machines, followed by their installation on the selected hypervisor, configuration of the network in bridged mode, and the execution of the virtual machines.

For the "deployment time" indicator, the time required for each service to be deployed or started was measured and recorded on a data sheet. In the container-based scenario, the startup time of the containers was measured from the moment the docker start command was executed until they became fully operational. In the virtual machine-based scenario, the time required to start the virtual machines with the services already installed was recorded, taking into account the time needed for them to become fully operational.

3.3.2. Measuring the "Cybersecurity Learning"

The "cybersecurity learning" variable was evaluated while taking into consideration the procedural and attitudinal competency dimensions. Only these two dimensions were considered in this study, due to the nature of the experiment, which is focused on practice more than on theoretical learning. In order to determine whether the students achieved the competency related to the procedural knowledge, the topics listed in Table 1 were included, corresponding to the contents of a university cybersecurity course. The defined indicators were "score earned on the guided practical exercise" and "level of acquired competencies", measured using a rubric for the procedural dimension; and "level of problem-solving in response to a challenge" for the attitudinal dimension.

For the "score earned on the guided practical exercise" indicator, the three topics were considered that are set out in Table 1. Two practical exercises were administered with three evaluation criteria each, so that the maximum cumulative score per exercise was 20 (20-point scale), as shown in Table 2. The value of the indicator for each student is the mean of the two practical exercises performed. The evaluation criteria for this indicator were as follows:

- EC1: Captures HTTP, ARP and ICMP traffic packets using a network protocol analyzer.
- EC2: Identifies common vulnerabilities and attacks in IT infrastructure.
- EC3: Extracts password hashes from operating systems.
- EC4: Performs SQL injection on a MySQL database.
- EC5: Performs cookie theft and server-side storage attacks using XSS injection.
- CE6: Performs the web service attack using Clickjacking.

Topics	Activities	Evaluation criterion	Time (hours)
Defensive security	Network monitoring	EC1	3
Offensive IT infrastructure security	Common vulnerabilities and attacks on IT infrastructure	EC2	3
	Attacks and password hash extraction from operating systems	EC3	3
Application security fundamentals	Types of attacks and analysis of application vulnerabilities	EC4, EC5, EC6	3

Table 1. Topics and activities for the laboratory application

Evaluation criteria	Score			
	01 points Insufficient	2-3 points Average	4-5 points Good	6-7 points Excellent
Topics: Defensive and offensive IT infrastructure security				
EC1: Captures HTTP, ARP and ICMP traffic packets using a network protocol analyzer.	The capture was attempted, but incorrect traffic filters were applied.	Basic filters were applied (for example, "HTTP" or "ICMP"), but not accurately or in the correct context.	The proper filters were applied to capture traffic from some HTTP, ARP and ICMP packets without any errors.	The proper and effective filters were applied to capture all the HTTP, ARP and ICMP traffic packets.
EC2: Identifies common vulnerabilities and attacks in IT infrastructure.	Discovers open ports and network services.	Identifies vulnerabilities in the services associated with scanned ports.	Identifies and configures an exploit to attack a vulnerable service.	Successfully enters or takes control over the service or device being attacked.
EC3: Extracts password hashes from operating systems.	Identifies the hash location and files.	Extracts hash files, but errors occur during password decryption and plaintext recovery.	Extracts the hash files and decrypts some passwords.	Extracts the hashes using automated tools.
Topics: Application security fundamentals				
EC4: Performs SQL injection on a MySQL database.	Attempts to perform SQL execution.	Executes the SQL attack with minimal results and some frequent errors.	Successfully executes the SQL attack and obtains partial, but not particularly relevant data.	Executes the SQL injection attack precisely and successfully, obtaining relevant data (passwords, user names, etc.).
EC5: Performs cookie theft and server-side storage attacks using XSS injection.	Performs XSS injection, but does not manage to obtain the cookie or send it to the server.	Performs the XSS injection and obtains the cookie, but does not manage to save it on the server.	Obtains the cookie and saves it on a server, although some minor errors have occurred in the process.	Correctly obtains the cookie and saves it on a server.
CE6: Performs the web service attack using Clickjacking.	Does not configure the page correctly or does not expose the vulnerability to Clickjacking.	Partially configures the web page, but does not fully ensure protection against Clickjacking vulnerabilities. The attack is carried out with errors that limit its effectiveness.	Correctly configures the page and performs the attack, although with some minor errors or without achieving the desired functionality.	Correctly configures the page, exposes the vulnerability and successfully executes the Clickjacking attack, achieving the objective with evidence.

Table 2. Rubric for the indicator "score earned on the guided practical exercise"

For the indicator "level of competencies acquired", seven competencies were established, according to the definitions in the curriculum for the undergraduate degree in Computer and Systems Engineering at the university where the study was conducted. A dichotomous system was used to evaluate the criteria for this indicator, determining whether or not the competency was achieved; if it was achieved, the corresponding

value was assigned. The maximum sum was 20 points in the event that the student successfully achieved all the competencies. Table 3 shows the definition of the competencies and the score assigned.

Competencies	Was the competency achieved? YES/NO	Score earned
C1: Identifies vulnerabilities in systems and applications. Value: 3		
C2: Correctly uses multiple offensive security tools. Value: 2		
C3: Executes attacks, using different exploitation techniques on network services. Value: 3		
C4: Executes password extraction and decryption attacks on operating systems. Value: 3		
C5: Understands the technical aspects of IT infrastructure security. Value: 3		
C6: Understands the technical aspects of application security. Value: 3		
C7: Demonstrates ethical behavior. Value: 3		
Total:		

Table 3. Checklist for the indicator "level of competencies acquired"

The indicator "level of problem-solving in response to a challenge" within the attitudinal dimension was assessed by evaluating the students' ability on the selected topics, using five assessment criteria and the rubric shown in Table 4. Each criterion has a value on a 1-4 point scale, with a total sum of 20 points possible. The evaluation criteria are the following:

- D1: Identifies and accesses the vulnerable web server address.
- D2: Obtains sensitive information by uploading malware.
- D3: Inserts a new user in the database, with their own name and the proper permissions.
- D4: Captures cookies and redirects them to the target host on an external server.
- D5: Analysis and resolution efficiency.

	Score			
Evaluation criteria	**1 point Insufficient**	**2 points Average**	**3 points Good**	**4 points Excellent**
D1: Identifies and access the vulnerable web server address.	Fails to identify vulnerable ports and services on the target host.	Identifies vulnerable services and partially characterizes vulnerabilities in the target web service.	Identifies vulnerable services, characterizes vulnerabilities of the web service, but does not manage to properly access the web service.	Identifies vulnerable services, characterizes vulnerabilities of the web service, but does not manage to properly access the web service.
D2: Obtains sensitive information by uploading malware.	Uploads the file.	Uploads the malware and obtains results with process errors.	Uploads the malware file and obtains partial results.	Manages to upload the shell and obtains the requested information.

Evaluation criteria	Score			
	1 point Insufficient	2 points Average	3 points Good	4 points Excellent
D3: Inserts a new user in the database, with their own name and the proper permissions.	Unsuccessfully attempts to insert the new user.	Partially inserts the user, but it has errors and the permissions are not correctly validated.	Functionally inserts the user, but without completing the proper permission validation.	Correctly inserts and validates the user and the permissions on the system.
D4: Captures cookies and redirects then to the target host on an external server.	Does not capture the session cookie form the website or resend it to the external server.	Captures the cookie, but does not manage to send it to the external server.	Obtains the cookie and partially sends it to the external server.	Obtains the session cookie and sends it correctly to the external server.
D5: Analysis and resolution efficiency.	Performs actions without a clear or logical strategy and with little analysis.	Performs a basic analysis and follows a basic strategy.	Plans and executes the necessary steps logically and in an organized manner, although some aspects of the performance could have been optimized.	Analyzes and follows a clear strategy, demonstrating efficient resolution with a logical sequence (planning, analyzing and executing the challenge in an efficient and organized manner, optimizing time and resources, with a clear focus on the solution.)

Table 4. Rubric for the indicator "level of problem-solving in response to a challenge"

4. Results

4.1. Laboratory Evaluation

The pentesting laboratory evaluated by the indicators "configuration time of laboratory components" and "service deployment time" is shown in Table 5, where it can be seen that the highest mean time corresponds to Kali Linux, with approximately 17 minutes, in contrast with the shortest mean time recorded on Metasploitable, which was nearly 2 minutes. On the other hand, in the control group, the Kali Linux operating system shows a significantly higher mean of 36 minutes, while the Wireshark has a lower mean of approximately 5 minutes.

Components	Group	N	Minimum	Maximum	Mean	Standard Dev.
Kali Linux	GExp(A)	21	9.70	26.00	17.6871	4.378
Metasploitable			0.70	4.25	1.7333	0.948
Wireshark			0.29	13.66	5.9281	3.514
DVWA			2.33	10.51	6.2481	2.207
Kali Linux	GCtr(B)	28	25.00	56.00	36.1714	9.393
Metasploitable			9.0	37.0	20.857	6.742
Wireshark			3.0	8.0	5.814	1.558
DVWA			8.07	30.08	14.5025	5.231

Table 5. Configuration time (in minutes) of the laboratory components

Analyzing the extreme values, the experimental group registers the lowest minimum times in all services, ranging from 0.7 minutes to 26 minutes, while the control group has the highest minimum times, between 3 and 25 minutes, and maximum times of as much as 56 minutes.

Even though the operating systems require more time for their configuration, due to their size and the pre-installed tools, as occurs with the Kali Linux operating system that includes a set of offensive tools for its use, Docker images pose certain advantages, as they are smaller and easier to configure, since they do not require storage or full resource management, as they only include the libraries and dependencies

needed for their operation. In contrast, virtual machines involve longer times due to the complete installation process; this includes steps such as the assigning of computer resources (RAM, processor and storage), the installation of the operating system and network configuration, which takes longer. On the other hand, with Docker containers, it is possible to automatically generate an internal network using the Docker network, which makes it possible to work in a local laboratory environment without the need to configure heavy virtual machines.

The range of values and the standard deviation observed reflect the differences among students when configuring the environment; some took less time to complete the configuration, while others required more time to finish the same process. This difference was related to the level of familiarization and prior practical experience with the use of visualization tools and service configuration, which is reflected in the recorded times.

Table 6 shows the comparison of the means of the total time required to configure the resources, evidencing that the configuration times in experimental group (A) were relatively homogeneous, while in the control group (B) the mean time had a standard deviation that was greater than in Group A, which leads us to conclude that some students took longer to complete the configuration of the virtual machines. This difference is due to the fact that virtual machines require additional procedures as compared to Docker containers that optimize the time for laboratory preparation, since their downloading and generation take less time.

Indicator/Group	Mean	Standard error	Standard deviation	Variance
Configuration time Group A	31.5967	1.46598	6.71798	45.131
Configuration time Group B	77.3454	2.20401	11.66251	136.014

Table 6. Comparison of the means of the configuration time (in minutes) between the experimental and control groups

The results of the measure of the indicator "deployment time" evidence that the operating systems in Docker containers take less time to deploy than with the virtual machines. This is because the containers are started via commands that allow the operating system to launch directly in just a few seconds and become ready for use. In contrast, virtual machines require more time, since they must completely load the operating system and its interface before they are ready for use.

Table 7 shows the comparison of means for the indicator "service deployment time", in which it is evident that the mean time is similar in both groups, but slightly greater in the experimental group. This indicates that the students in Group A took a little longer to deploy the Docker containers, perhaps due to the novelty of this tool for the students.

Indicator/Group	Mean	Standard error	Standard deviation	Variance
Deployment time Group A	5.8952	0.31354	1.43683	2.862
Deployment time Group B	5.5643	0.44571	2.35847	7.030

Table 7. Comparison of the mean times (in minutes) of deployment between the experimental and control groups

4.2. Learning Evaluation

This section analyzes the results of the procedural and attitudinal knowledge dimensions. For the procedural knowledge dimension, the indicators "score earned on the guided practical exercise" and "level of competencies acquired" were used. In both cases, a 20-point scale was used to maintain uniformity in the data processing.

Table 8 shows the results of the evaluations applied to measure the procedural knowledge dimension with both indicators. The group that used Docker containers attained a mean of 9.61, which was slightly above the mean of 9.50 obtained by the group that used virtual machines. The standard deviation explains that

the variability is due to the fact that the use of the Docker containers does not impact all the students in the same way, since some encountered difficulties in adapting to the command-based environment, which could have affected their performance.

Dimension	Group	N	Mean	Mode	Standard deviation	Minimum	Maximum
Procedural knowledge	A	21	9.61	3.93	4.87	2.43	19.0
	B	28	9.50	8.78	3.39	3.43	16.3

Table 8. Comparison of the means for the indicator "score earned on the guided practical exercise"

The attitudinal dimension considered the indicators "level of problem-solving in response to a challenge"

The results of the measurement for the indicator "level of problem-solving in response to a challenge" are shown in Table 9. The experimental group has a mean of 9.38, while that of the control group is 9.14, which indicates that the use of Docker containers did not significantly influence the development of attitudinal knowledge as compared to the virtual machine method. In terms of variability, the groups presented a similar dispersion, which suggests that most of the students had similar performance, regardless of the focus used. Altogether, these results indicate that the use of Docker containers did not produce significant improvements in the attitudinal knowledge due to the magnitude of the practical exam, since the use of the Docker containers was not limited to specific exercises, but rather implied approaching the challenge using more tools in an integrated manner, in which the students were required to apply previous knowledge.

Dimension/Indicator	Group	N	Mean	Mode	Standard deviation	Minimum	Maximum
Attitudinal knowledge / level of problem-solving in response to a challenge	A	21	9.33	8.00	3.21	3.00	17.0
	B	28	9.14	8.00	2.97	0.00	14.0

Table 9. Comparison of the means for the indicator "level of problem-solving in response to a challenge"

4.3. Testing the Hypotheses

The hypothesis system consisted of two specific hypotheses that correspond to the dimensions of the learning variable considered for this study, in addition to the general hypothesis.

The normality of the data obtained in the learning measurement was first determined through the application of the Shapiro-Wilk test, since the sample size was less than 50. The results indicated that the procedural knowledge dimensions showed a normal distribution ($p > 0.05$), and so the parametric Student's t-test was applied for independent samples. In contrast, the attitudinal knowledge dimensions did not meet the assumption of normality ($p < 0.05$), for which reason the non-parametric Mann-Whitney U test was used. Likewise, the data corresponding to the general learning variable showed a normal distribution, and the Student's t-test was also applied.

The first specific hypothesis corresponds to the procedural knowledge dimension, and the statistical hypotheses were the following:

- $SH1_0$: The infrastructure pentesting laboratory based on Docker containers does not have a positive and significant influence on procedural knowledge in cybersecurity learning.
- $SH1_1$: The infrastructure pentesting laboratory based on Docker containers has a significant positive influence on procedural knowledge in cybersecurity learning.

The results are shown in Table 10, where the p-value is equal to 0.923. Since this value is greater than the level of significance (0.05), the null hypothesis is not rejected, which indicates that there are no statistically significant differences between the two groups; therefore, it cannot be stated that the infrastructure

pentesting laboratory based on Docker containers has a significant positive influence on the procedural knowledge in the learning of cybersecurity.

Dimension	Statistic	gl	p-value
Procedural knowledge	0.0975	47	0,923

Table 10. Student's t-test for independent samples for the procedural learning dimension

The second specific hypothesis corresponds to the attitudinal knowledge dimension of cybersecurity learning. The statistical hypotheses were as follows:

- $SH2_0$: The infrastructure pentesting laboratory based on Docker containers does not influence in a significant positive manner the attitudinal knowledge in cybersecurity learning.
- $SH2_1$: The infrastructure pentesting laboratory based on Docker containers influences in a significant positive manner the attitudinal knowledge in cybersecurity learning.

The result of the Mann-Whitney U test for the attitudinal knowledge dimension that is shown in Table 11 has a p-value of 0.959, which indicates that there are no significant differences in the scores obtained in both groups. This means that the null hypothesis is not rejected since sufficient evidence was not found to conclude that the attitudinal knowledge of the experimental group is greater than that of the control group, since both groups show similar scores in this dimension. Therefore, the infrastructure pentesting laboratory based on Docker containers does not have a significant positive influence on the attitudinal knowledge in cybersecurity learning.

Dimension	Statistic	p-value
Attitudinal knowledge	291	0.959

Table 11. Mann-Whitney U test for independent samples for the attitudinal knowledge dimension

Finally, the general hypothesis was formulated in the following manner:

- H_0: The infrastructure pentesting laboratory based on Docker containers does not have a significant positive influence on the learning of cybersecurity.
- H_1: The infrastructure pentesting laboratory based on Docker containers has a significant positive influence on the learning of cybersecurity.

Table 12 shows the result, indicating that there is not enough statistical evidence to reject the null hypothesis. In this context, it is established that there are no significant differences in the score and the level of analysis of the "learning" variable in the two groups. Therefore, the null hypothesis is accepted, which establishes that the infrastructure pentesting laboratories based on Docker containers do not have a significant positive influence on cybersecurity learning.

Variable	Statistic	p-value
Learning - CS	0.165	0.870

Table 12. Student's t-test for the analysis of the cybersecurity learning variable

5. Discussion

The results obtained show that the infrastructure pentesting laboratory based on Docker containers did not produce statistically significant differences in the learning in the evaluated dimensions, as compared to the traditional laboratory based on virtual machines.

The observed variability is associated with the students' adaptation to command-based configuration and execution environments, which is consistent with the notion that some students experienced technical difficulties that affected their performance. These findings are partially in line with what has been suggested by Potdar et al. (2020) and Ionescu et al. (2019), who indicate that the environments based on containers facilitate the reduction of the configuration time, but the impact on learning also depends on the familiarity with the technology and the specific prior experience of the users (Nakata & Otsuka, 2021).

From the perspective of laboratory preparation, the use of containers is lighter and less complex than an environment based on virtual machines, which is consistent with the findings of Sedov and Lazarev (2024), Maruszczak et al. (2022) and Li (2021). Even though this study did not test container performance on Windows systems, building the laboratory in a Linux environment has proven to be flexible and free of any computational resource issues; therefore, this aligns with the findings reported by Sergeev et al. (2022). On the other hand, in contrast with what was found by Shahriar et al. (2020), this study has shown that there is a technical dependency in order for the container-based laboratory to be truly effective, enabling students to concentrate on the learning content rather than the laboratory environment.

6. Conclusions

It was determined that the pentesting laboratories based on Docker containers did not generate a significant difference in general cybersecurity learning as compared to laboratories based on virtual machines. The results showed similar means in both groups of the experiment, and in both the procedural and the attitudinal knowledge dimensions. The inferential statistical tests confirmed that there was no significant influence by container use as compared to the use of virtual machines in this type of learning.

This lack of impact is related to the nature of the designed activities and the students' technical familiarity with the handling of containers, which affected the effectiveness of the laboratory. However, it represents a potentially advantageous alternative because it requires fewer computational resources (RAM, processing power and storage) than virtual machines.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

Authors' contributions

William-Rogelio Marchand-Niño: conceptualization, methodology, supervision, validation, writing–original draft, writing–review.

Raquel-Esther Arias-Ramos: investigation, methodology, formal analysis, writing–original draft.

Data availability

Data included in the article itself or supplementary material

Use of Artificial Intelligence

The authors declare that the content of the article has not been developed using Artificial Intelligence.

References

Abhishek, M. K., Rajeswara-Rao, D., & Subrahmanyam, K. (2022). Framework to deploy containers using Kubernetes and CI/CD pipeline. *International Journal of Advanced Computer Science and Applications (IJACSA), 13*(4). https://doi.org/10.14569/IJACSA.2022.0130460

Amjad, A. I., Aslam, S., Tabassum, U., Sial, Z. A., & Shafqat, F. (2024). Digital Equity and Accessibility in Higher Education: Reaching the Unreached. *European Journal of Education, 59*(4), e12795. https://doi.org/10.1111/ejed.12795

Assefa, Y., Gebremeskel, M. M., Moges, B. T., Tilwani, S. A., & Azmera, Y. A. (2025). Rethinking the digital divide and associated educational in(equity) in higher education in the context of developing countries: The social justice perspective. *The International Journal of Information and Learning Technology, 42*(1), 15–32. https://doi.org/10.1108/IJILT-03-2024-0058

Baresi, L., Quattrocchi, G., & Rasi, N. (2024). A qualitative and quantitative analysis of container engines. *Journal of Systems and Software, 210,* 111965. https://doi.org/10.1016/j.jss.2024.111965

Benítez-Ayala, D. A. (2022). Evaluación del aprendizaje y el enfoque por competencias. Revisión de antecedentes teóricos. *Ciencia Latina Revista Científica Multidisciplinar, 6*(6), 10402–10434. https://doi.org/10.37811/cl_rcm.v6i6.4136

Buyya, R., Vecchiola, C., & Selvi, S. T. (2013). Chapter 3—Virtualization. In R. Buyya, C. Vecchiola, & S. T. Selvi (Eds.), *Mastering Cloud Computing* (pp. 71–109). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-411454-8.00003-6

Cardwell, K. (2016). *Building virtual pentesting labs for advanced penetration testing*. Packt Publishing Ltd. https://www.packtpub.com/en-us/product/building-virtual-pentesting-labs-for-advanced-penetration-testing-9781785884955?srsltid=AfmBOorjKyckbNlFeGkf9KPeAyfPsFzde_zq3vW9MA58zZI_-_QRMWFH

Davies, C. (2008). *Learning and teaching in laboratories*. Higher Education Academy Engineering Subject Centre. Loughborough University. https://repository.lboro.ac.uk/articles/book/Learning_and_teaching_in_laboratories/9488273

Dubec, J., Balažia, J., Bencel, R., & Čičák, P. (2023). Docker-Based Assignment Evaluations in E-Learning. *Communication and Information Technologies (KIT)* (pp. 1–5). Slovakia. https://doi.org/10.1109/KIT59097.2023.10297093

Engebretson, P. (2011). Chapter 1–What Is Penetration Testing? In P. Engebretson (Ed.), *The Basics of Hacking and Penetration Testing* (pp. 1–14). Syngress. https://doi.org/10.1016/B978-1-59749-655-1.00001-5

Hess, K., & Newman, A. (2010). *Practical Virtualization Solutions: Virtualization from the Trenches*. Prentice Hall/Pearson Education. https://books.google.com.pe/books?id=h1M7OAAACAAJ

Ilidzhev, A. A., & Ibragimova, A. N. (2018). Procedural Component Of The Didactic System in the Competence-Based Paradigm of Education. In R. Valeeva (Ed.), *Teacher Education–IFTE 2018* (Vol. 45, pp. 594–601). Future Academy. https://doi.org/10.15405/epsbs.2018.09.69

Ionescu, V. M., Patel, M., & Hindocha, D. (2019). Alternatives for Running Linux Applications in Windows. In *2019 11th International Conference on Electronics, Computers and Artificial Intelligence* (pp. 1–4). Romania. https://doi.org/10.1109/ECAI46879.2019.9042127

Knöchel, M., Karius, S., & Wefel, S. (2021). Developing a Web-based Training Platform for IT Security Education. In *DELFI 2021* (pp. 223–228). Gesellschaft für Informatik e.V.

Kristian, A., & Nascimento-e-Silva, D. (2024). Evaluation of internet access by students at a federal educational institution in Northern Brazil. *International Journal of Professional Business Review, 9*(10), e05061. https://doi.org/10.26668/businessreview/2024.v9i10.5061

Li, Z. (2021). Comparison between common virtualization solutions: VMware Workstation, Hyper-V and Docker. In *2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC)* (pp. 701–707). USA. https://doi.org/10.1109/ICFTIC54370.2021.9647226

Lopez de Jiménez, R. E. (2016). Pentesting on Web Applications using Ethical-Hacking. In *IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI)* (pp. 1–6). Costa Rica. https://doi.org/10.1109/CONCAPAN.2016.7942364

Maruszczak, A., Walkowski, M., & Sujecki, S. (2022). Base Systems for Docker Containers – Security Analysis. In *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (pp. 1–5). Croatia. https://ieeexplore.ieee.org/document/9911523

Mishra, P., Bhatnagar, S., & Katal, A. (2020). Cloud container placement policies: A study and comparison. In S. Smys, T. Senjyu, & P. Lafata (Eds.), *Second International Conference on Computer Networks and Communication Technologies (ICCNCT 2019)* (Lecture Notes on Data Engineering and Communications Technologies, Vol. 44). Springer. https://doi.org/10.1007/978-3-030-37051-0_58

Mohan, A., Swaminathan, G. A., & Shafana, N. J. (2022). Automated tools and techniques in vulnerability assessment. In *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)* (pp. 533–540). IEEE. https://doi.org/10.1109/ICSSIT53264.2022.9716474

Morgan, S. (2023). 2023 *Cybersecurity Almanac: 100 Facts, Figures, Predictions, And Statistics*. Cybercrime Magazine. https://cybersecurityventures.com/cybersecurity-almanac-2023/

Mouat, A. (2015). *Using Docker: Developing and Deploying Software with Containers*. O'Reilly Media. https://books.google.com.pe/books?id=wpYpCwAAQBAJ

Naga-Suneetha, A. R. V., Prasanthi, J. K. V. S., Nimisha, S., & Meghna, C. H. (2025). Vulnerability assessment and penetration testing using Parrot operating system. In A. Patel, N. Kesswani, M. Mishra, & P. Meher (Eds.), *Advances in Machine Learning and Big Data Analytics I (ICMLBDA 2023)* (Springer Proceedings in Mathematics & Statistics, Vol. 441). Springer. https://doi.org/10.1007/978-3-031-51338-1_15

Nakata, R., & Otsuka, A. (2021). CyExec: A High-Performance Container-Based Cyber Range With Scenario Randomization. *IEEE Access, 9,* 109095–109114. https://doi.org/10.1109/ACCESS.2021.3101245

Oriyano, S. P. (2017). *Penetration testing essentials*. John Wiley & Sons. https://www.wiley.com/en-us/Penetration+Testing+Essentials-p-9781119419358

Popek, G. J., & Goldberg, R. P. (1974). Formal requirements for virtualizable third generation architectures. *Commun. ACM, 17*(7), 412–421. https://doi.org/10.1145/361011.361073

Potdar, A. M., D G, N., Kengond, S., & Mulla, M. M. (2020). Performance Evaluation of Docker Container and Virtual Machine. *Third International Conference on Computing and Network Communications (CoCoNet'19), 171*, 1419–1428. https://doi.org/10.1016/j.procs.2020.04.152

Sausalito, C. (2024). *Why Employers Are Struggling To Fill Cybersecurity Vacancies*. Cybercrime Magazine. https://cybersecurityventures.com/why-employers-are-struggling-to-fill-cybersecurity-vacancies/

Schenker, G. N. (2023). *The ultimate Docker container book: Build, test, ship, and run containers with Docker and Kubernetes*. Packt Publishing Ltd.

Schunk, D. H. (1997). *Teorías del aprendizaje*. Pearson Educación.

Sedov, D., & Lazarev, A. (2024). Enhancing IT Education Through Docker Integration. In *4th International Conference on Technology Enhanced Learning in Higher Education (TELE)* (pp. 252–255). Russian Federation. https://doi.org/10.1109/TELE62556.2024.10605628

Sergeev, A., Rezedinova, E., & Khakhina, A. (2022). Docker Container Performance Comparison on Windows and Linux Operating Systems. In *2022 International Conference on Communications, Information, Electronic and Energy Systems (CIEES)* (pp. 1–4). Bulgaria. https://doi.org/10.1109/CIEES55704.2022.9990683

Shahriar, H., Qian, K., & Zhang, H. (2020). Learning Environment Containerization of Machine Leaning for Cybersecurity. In *IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 1131–1132). Spain. https://doi.org/10.1109/COMPSAC48688.2020.0-105

Susen, C., Storms, S., & Herfs, W. (2022). Virtualization in control technology – Virtualization and consolidation of programmable logic controllers. *WT Werkstattstechnik, 112*(11–12), 792–796. https://doi.org/10.37544/1436-4980-2022-11-12-66

Tomar, A., Jeena, D., Mishra, P., & Bisht, R. (2020). Docker Security: A Threat Model, Attack Taxonomy and Real-Time Attack Scenario of DoS. In *10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 150–155). India. https://doi.org/10.1109/Confluence47617.2020.9058115

Tunde-Onadele, O., He, J., Dai, T., & Gu, X. (2019). A Study on Container Vulnerability Exploit Detection. *2019 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 121–127). Czech Republic. https://doi.org/10.1109/IC2E.2019.00026

Turnbull, J. (2019). *The Docker Book*. James Turnbull. https://dockerbook.com/

Tyshyk, I., & Hulak, H. (2024). Testing an organization's information system for unauthorized access. *CEUR Workshop Proceedings, 3826,* 17–29. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85210246188&partnerID=40&md5=402867e9403e4e4759d8df4c88451515

Veenman, M. V. J., Van Hout-Wolters, B. H. A. M., & Afflerbach, P. (2006). Metacognition and learning: Conceptual and methodological considerations. *Metacognition and Learning, 1*(1), 3–14. https://doi.org/10.1007/s11409-006-6893-0

Villa, A., & Poblete, M. (2007). *Aprendizaje basado en competencias: Una propuesta para la evaluación de las competencias genéricas*. Ediciones Mensajero.

Wylie, P., & Crawley, K. (2021). What Is a Pentester? In *The Pentester BluePrint: Starting a Career as an Ethical Hacker* (pp. 1–16). Wiley. https://doi.org/10.1002/9781119684367.ch1

Wyne, M. F., Zhang, L., & Farahani, A. (2021). A Review of Competency Based Education (CBE). In *2021 International e-Engineering Education Services Conference (e-Engineering)* (pp. 127–132). Jordan. https://doi.org/10.1109/e-Engineering47629.2021.9470564

Zhang, Q., Liu, L., Pu, C., Dou, Q., Wu, L., & Zhou, W. (2018). A Comparative Study of Containers and Virtual Machines in Big Data Environment. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)* (pp. 178–185). USA. https://doi.org/10.1109/CLOUD.2018.00030

Zhu, H., & Gehrmann, C. (2021). Lic-Sec: An enhanced AppArmor Docker security profile generator. *Journal of Information Security and Applications, 61,* 102924. https://doi.org/10.1016/j.jisa.2021.102924

Published by OmniaScience (www.omniascience.com)

Journal of Technology and Science Education, 2026 (www.jotse.org)