Article 25fa pilot End User Agreement

This publication is distributed under the terms of Article 25fa of the Dutch Copyright Act (Auteurswet) with explicit consent by the author. Dutch law entitles the maker of a short scientific work funded either wholly or partially by Dutch public funds to make that work publicly available for no consideration following a reasonable period of time after the work was first published, provided that clear reference is made to the source of the first publication of the work.

This publication is distributed under The Association of Universities in the Netherlands (VSNU)'Article 25fa implementation' pilot project. In this pilot research outputs of researchers employed by Dutch Universities that comply with the legal requirements of Article 25fa of the Dutch Copyright Act are distributed online and free of cost or other barriers in institutional repositories. Research outputs are distributed six months after their first online publication in the original published version and with proper attribution to the source of the original publication.

You are permitted to download and use the publication for personal purposes. Please note that you are not allowed to share this article on other platforms, but can link to it. All rights remain with the author(s) and/or copyrights owner(s) of this work. Any use of the publication or parts of it other than authorised under this licence or copyright law is prohibited. Neither Radboud University nor the authors of this publication are liable for any damage resulting from your (re)use of this publication.

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please contact the Library through email: copyright@ubn.ru.nl, or send a letter to:

University Library
Radboud University
Copyright Information Point
PO Box 9100
6500 HA Nijmegen

You will be contacted as soon as possible.

A Virtual Classroom for Cybersecurity Education

Jens Haag¹, Harald Vranken^{2,3(✉)}, and Marko van Eekelen^{2,3}

¹ Cologne University of Applied Sciences, Cologne, Germany
Jens.Haag@ou.nl

² Open Universiteit, Heerlen, The Netherlands
{Harald.Vranken, Marko.vanEekelen}@ou.nl

³ Radboud University, Nijmegen, The Netherlands

Abstract. Education in general and cybersecurity education in particular can be made more attractive by adding hands-on experience to classrooms. This requires new technology, such as virtualisation, to be developed fully geared towards the needs of educational purposes. Over the years, several techniques have been developed by the authors. In this paper, the authors first give a full account of their earlier work on a distributed virtual computer lab for cybersecurity education. Then, this virtual lab is extended with educational enhancements, such as an intelligent tutoring system, which resulted in a prototype for a virtual classroom for cybersecurity education.

Keywords: Virtual lab · Distributed lab · Cybersecurity · Education

1 Introduction

Knowledge distribution and knowledge acquisition are essentials for educational establishments. Usually they are built on two components: *theory* and *hands-on experience*, where “hands-on” refers to tasks in which learners observe or manipulate real objects or material. We find this in many academic disciplines, amongst others the field of information technology (IT) resp. computer science. Here, the theory is traditionally taught in lessons or by textbooks. The knowledge that students learn, is often illustrated, deepened and anchored by carrying out practical (hands-on) exercises.

While the content of a discipline is usually fixed in the curriculum, a challenge is how to organize the hands-on experience. Especially in the field of communication technology and IT security, which are sub-disciplines of computer science, special requirements occur with respect to the hands-on learning environment, e.g. the isolated real-world character and manageability.

Usually, an exercise is derived from and also targeted at a real-world setting, e.g. setting up and configuring hosts and networks or attacking a system within a network. This requires that the learning environment feels like and behaves

as close as the real-world setting to the learner. To gather experiences, learners should be able to try out things they learned in theory. This includes doing things right as well as doing things wrong. In the scope of communication technology, this could mean that students will set up a network environment, either with a valid or an invalid configuration. With respect to IT security, this could mean that a student will protect a system and another one tries to attack it. Depending on the aim of a certain task, everything is working fine, the environment gets down or damaged, or something in between can happen. However, this requires that the learning environment is isolated from the real world in order to prevent interferences. Also, the manageability of the hands-on learning environment is crucial for both lecturer as well as learner. E.g. the lecturer may use the environment in different courses with several learners, so a time-saving clean-up and reset process of the environment is required to get an equal initial state for each learner in each course. For the learner, a great user experience as well as a short training period might result in an increased acceptance level which finally can lead to an improved learning outcome.

A common way to organise hands-on experiences is to use a *computer laboratory* or *computer security laboratory* for education. A computer lab in the scope of communication technology and cybersecurity typically consists of a computer or a group of computer systems usually connected into a network. The systems as well as the network behave as in the real world, but they are not connected to it - they are neither part of a critical infrastructure nor can connect to any. In other words the system and also the network are *isolated* from the outside world. Many tasks can be realised using a computer lab which otherwise would not be possible, are forbidden or can cause damage. In the scope of communication technology and cybersecurity, the use of a computer lab is a suitable and common way for hands-on experiences. Nevertheless, this situation still holds challenges and thus can be improved.

This paper introduces a virtualised classroom environment for use in cybersecurity education. We address the technical details of the distributed virtualised computer lab that we developed, and we also show how education is organised and supported in this lab. We give a complete overview of our work in this field by including the essential parts of our previous work. Evaluations of students' experiences with our lab show that students enjoy the hands-on experience, which nicely implements the concept of edutainment.

Educational Context

The Open Universiteit (Netherlands) as well as the Cologne University of Applied Sciences (Germany) offer courses in the field of computer networks and cybersecurity. While the Open Universiteit is a university for distance education, the Cologne University of Applied Sciences is a traditional on-campus university. Both universities offer practical courses in networking and cybersecurity, but the setting is mostly different.

Research Challenge

Providing a safe playground is only one requirement of a virtual classroom. An additional challenge is to figure out what environment fits best in a certain learning scenario. This requires to find a balance between technical opportunities, educational requirements and also the demands of the learners.

Overview

Section 2 presents our distributed virtual computer lab, which provides the basis for the virtual classroom to develop. In Sect. 3 an intelligent tutor system is described which extends the basis with an exercise assistant working towards a virtual classroom. With the educational enhancements of Sect. 4 the virtual classroom prototype emerges. In Sect. 5 we briefly discuss implementation and evaluation details. We describe related work in Sect. 6 and we conclude the paper in Sect. 7.

2 A Distributed Virtual Computer Lab

2.1 The Basis: A Virtual Computer Lab

Initially we developed a Virtual Computer Lab (VCL) [18] as a stand-alone environment, composed of two nested software virtualisation layers, that each student can install on his/her computer. The software components to build the VCL are freeware or open source, and are distributed to students through a DVD or the university's cloud storage.

Fig. 1. Architecture of the VCL

The VCL is composed of one physical host and two virtualisation layers, as shown in Fig. 1. The host machine is the student's computer, which runs an arbitrary operating system, i.e., the host operating system. The first virtualisation

layer creates the virtual host machine. It consists of virtualisation software such as VMware Player (freeware) or Oracle VM VirtualBox (open source), which runs on the host machine just like an ordinary application. Versions of this software are available for a large range of platforms. VirtualBox for instance runs on host machines with either Windows, Linux or Mac OS X. This first virtualisation layer therefore runs on nearly all student computers, regardless of the hardware and the host operating system. The virtual host machine runs the virtual host operating system. For the VCL we selected Linux, since it is open source and can be distributed to students without licensing costs. In fact, we selected Knoppix, a bootable live Linux system containing a collection of GNU/Linux applications and the KDE graphical desktop environment.

The second virtualisation layer is a Linux application, called Netkit [17], that runs inside the virtual host machine. This second virtualisation layer allows to instantiate multiple virtual machines that all run Linux. Netkit applies virtualisation based upon User Mode Linux (UML) [33,34] and allows to setup and configure UML virtual machines with virtual network interfaces, and to connect these into virtual networks.

The hardware requirements for running the VCL are very modest. A few UML virtual machines can already be run smoothly on a PC with a Pentium-4 processor and 256 MB memory. The VCL has been used successfully in our security courses by hundreds of students with only few minor problems.

2.2 Adding Distributivity: A Distributed Virtual Computer Lab

Our next step was to extend the VCL into a Distributed Virtual Computer Lab (DVCL) [26] that enables two or more geographically distant students to connect their local VCLs across an external intermediate network, which is usually the internet. Figure 2 shows such a setup. To preserve the isolated property of a computer security lab, the DVCL system ensures that the intermediate network cannot intercommunicate with the virtual network. Within their VCLs, the students are able to work synchronously in a group, and assign roles and tasks for working on an assignment (e.g. hacker, administrator, and user). As result, the working scenarios are much closer to the situations in real networks.

2.3 Adding Central Authority: DVCL with Central Authority

Our DVCL with central authority [27] optimises the organisation and management when students work together. Initially, the DVCL used Peer-2-Peer (P2P) connections to connect students' VCLs with each other. For that reason, the students need to know the communication endpoint (IP address and TCP port) of their remote partner(s). Furthermore, additional network configuration (e.g. port forwarding) is mandatory. Connecting three or more VCLs is possible, but can lead into infinite circular flow of network packets in the case of a misconfiguration. Overall, our experiences with the DVCL showed that a significant effort was needed by the students to manage their VCL. To optimise the organisation and management, we developed a central administration point ('central

Fig. 2. Architecture of the DVCL

authority') hosted by the university to connect remote VCLs into sessions and to control the packet flow between the VCLs. Now, each VCL must only know the address of the university server which is usually static and therefore can be easily provided by using a configuration file. Figure 3 illustrates two students connecting their VCLs by using the central administration point. Students do not need to know the communication endpoint of each other anymore; they simply join a distributed virtual network session or can create a new one.

Fig. 3. Architecture of the DVCL with central authority

3 Adding an Intelligent Tutoring System

Our evaluation showed that students of a traditional on-campus networking course deem it crucial for their learning success to be able to get support from

a course advisor [29]. While an on-campus university will be able to provide course advisors, which can support students in so-called guided learning hours, this support is no longer feasible if students work e.g. at home in the evening hours using a virtual lab.

A common way to resolve this issue is an intelligent tutoring system (ITS). An intelligent tutoring system is a computer system that aims to provide instruction or feedback to learners, usually without requiring intervention from a human advisor [6, 7]. We introduced a tutoring system called Electronic Exercise Assistant (EEA) for our DVCL environment. The aim of this EEA is to offer support and guidance for students while working out an exercise, even if a human course advisor is not available.

3.1 A Typical Exercise Example

A very common task for future IT network administrators is to setup, configure, and secure a network. Therefore, an example assignment for students participating in a networking course could be:

“Setup and configure a scenario with at least three hosts (client, router, server). Client and server should be located within different subnets. The client should be able to intercommunicate with the server by using the intermediate router. The routing should be based on static routing tables.”

The minimal requirement for this setup is shown in Fig. 4, consisting of at least three hosts. The client and the server have one network interface; the router is equipped with two network interfaces: one interface connects to a network with the server, the other one with the client.

Fig. 4. Example network setup

In this example exercise, students will have to set up hosts and interconnect them accordingly within two different networks. They will have to assign appropriate IP addresses to these hosts and ultimately configure the routing by altering the routing tables on the hosts. Once the setup is configured properly, students can demonstrate the validity of their solution, e.g. by sending network packets between client and server.

While setting up and configuring this scenario, the students can gather several basic and advanced practical experiences, like designing, preparing, and setting

up an example network scenario according to the assignment, configuring the hosts and network interfaces, configuring the routing, and discovering different behaviour caused by different configurations.

Solve the Exercise

To solve this exercise, students need to know at least the basic commands to start and to administrate the virtual lab (Netkit). In addition, they have to apply their networking knowledge to configure the hosts according to the assignment. Using the VCL, a valid and straightforward configuration to solve the example assignment may look like stated in Listing 1.1.

Listing 1.1. Valid solution using Netkit

```

1 // Create the hosts and networks in Netkit
  vstart client --eth0=n1
3 vstart router --eth0=n1 --eth1=n2
  vstart server --eth0=n2

  // Assign IP address on the client
7 ifconfig eth0 10.0.0.1 up

9 // Assign IP address on the router
  ifconfig eth0 10.0.0.2 up
11 ifconfig eth1 11.0.0.2 up

13 // Assign IP address on the server
  ifconfig eth0 11.0.0.1 up

  // Set default gateway on the client
17 route add default gw 10.0.0.2

19 // Set default gateway on the server
  route add default gw 11.0.0.2

  // Connection test on client to the server
23 ping 11.0.0.1

```

3.2 Exercise Modelling

In the following we show how the exercises can be transferred into a formal representation, in order to be processed by a computer program. First we will show the partition of our example exercise into activities that will then be organised in a graph structure. This graph will then be extended with conditions that will

make the activities verifiable. We also show a way to add feedback attributes to the graph in order to model a certain feedback strategy. Finally we introduce probing, a mechanism to improve the verifiability of activities.

Activities

Typically, exercises will start with an empty lab. Students have to perform activities that result in a working network environment, configured according to the requirements of the given exercise. While Listing 1.1 shows the commands needed to solve the exercise in Netkit, the minimal conceptual activities needed for solving this exercise are listed in Table 1.

Table 1. Activities needed to solve the example exercise

Activity	ID
The client network has to be created	A1
The server network has to be created	A2
The client has to be connected to the client network and an appropriate IP address has to be assigned	A3
The server has to be connected to the server network and an appropriate IP address has to be assigned	A4
One NIC of the router has to be connected to the client network and an IP address from the client network has to be assigned	A5
One NIC of the router has to be connected to the server network and an IP address from the server network has to be assigned	A6
The client has to be configured to use the router's NIC in the client network as default gateway	A7
The server has to be configured to use the router's NIC in the server network as default gateway	A8
Routing has to be enabled on the router	A9
Client and server must intercommunicate via the intermediate router using the IP protocol	A10

While A10 is the final activity, the order of the activities A1 through A9 shows only one possible sequence. The order can vary because some activities are independent from each other (e.g. A1 and A2), while some other activities have interdependencies (e.g. A1 is a precondition for A3).

These activities and their interdependencies can be modelled as an acyclic, directed graph with exactly one sink (node N with $outdegree(N)=0$) and at least one source (node M with $indegree(M)=0$). Activities are represented by nodes. A precondition is modelled as a directed edge from the predecessor to the successor, seamlessly indicating the order of the activities. The final activity will be represented by a sink. Activities without a precondition will be represented by sources.

Fig. 5. Example graph

A valid graph for our example exercise is shown in Fig. 5. This graph is based on the activities stated in Table 1. The interdependencies and thus possible sequences of activities show a valid example. These can of course vary, depending on the exercise and the author’s intent, too.

Conditions

In order to process the graph, the activities have to be verifiable. That means that a condition is needed to detect or to decide, whether an activity is deemed passed, i.e. whether the student has successfully solved a part of the exercise.

Network packets, obtained from the student’s Netkit lab, can be used to detect and verify network properties and behavior of an Ethernet based network [28]. By modelling network specific expert knowledge as predicates and verifying these predicates using the captured network packets, it is possible to detect e.g. the presence of certain hosts and also routing behaviour. While our first prototype demonstrated the technical feasibility of that approach by using SQL queries to model predicates, we improved on it by using description logics [30].

For the terminological box (TBox) we created a network ontology for Ethernet based networks, representing the network layers 2 and above [31], including but not limited to the header and payload fields of the most commonly used protocols, e.g. Ethernet (RFC1042), ARP (RFC826), IP (RFC791), TCP (RFC793), and UDP (RFC768). In addition, we added a unique identifier for each packet and the network origin. An excerpt of our ontology for Ethernet networks is shown in Fig. 6.

Fig. 6. Ontology excerpt for ethernet networks

Using this ontology it is possible to model expert knowledge as predicates using a logic programming language, e.g. Prolog [19]. For example, the expert knowledge to describe the network behaviour “routing” is:

“Routing occurs if an OSI layer 3 IP transmission of a network packet between two hosts is based on more than one OSI layer 2 transmissions.”

The technical background is shown in Fig. 7. The client wants to communicate with the server using the IP protocol, but the server is located in a different network segment. Direct intercommunication between client and server is not possible, because the underlying Ethernet protocol does not support communication over network borders. The client has to use a known router located in the same network as itself, and thus reachable by Ethernet. The client now sends an

Fig. 7. Routing packet flow example

IP packet addressed to the IP address of the server, but the underlying Ethernet packet will be addressed to the router. When the router does receive such a packet, it will forward it to the server. While the two packets that the client and the router send do not differ on the IP layer (both are sent from the client, and addressed to the server), both differ on the Ethernet layer, with different source and destination MAC addresses.

Based on the Ethernet network ontology, this behavior can be expressed as the Prolog predicate in Listing 1.2.

Listing 1.2. Prolog predicate for routing

```

1 routing :-
    ip_packet(X,A,B),
3    ip_packet(Y,A,B),
    ethernet_packet(X,M1,M2),
5    ethernet_packet(Y,M3,M4),
    M1 \= M3, M2 \= M4.
```

This predicate can be read as “*routing occurs, when there are two IP layer packets X and Y , both sent from IP address A to IP address B , for which the source and destination addresses differ on the Ethernet layer.*”

Predicates can be used as conditions to detect activities. E.g. the predicate “routing” can be used to verify the activity A10. We extended the graph, so that every activity can be associated with a condition to verify that activity.

Routing is only one example. We successfully created predicates describing e.g. the presence of hosts and networks, the network behaviour NAT or routing, and also higher level usage. E.g. ARP spoofing behaviour can be detected if two hosts within the same subnet having different MAC addresses, pretend to own the same IP address using the ARP protocol. However, this behaviour can also be caused by a misconfiguration of the hosts. For that reason this condition requires preconditions to verify a valid and error-free setup.

We also found a trade-off between the shape of an assignment and the capabilities to design predicates. If the assignment is more tightly controlled (e.g. predefined network names and IP addresses), more precise predicates can be designed to detect activities. If the assignment is more generic, the predicates also have to be designed in a more generalised manner.

Feedback

There are various types of feedback strategies which can be used to support students working on the exercise, e.g. suggestions, complete guiding, or an exam mode. The specific shape will be either customised to match the author’s aims or customised to the learning style of the learner or a combination. Usually recent

progress the student has made in the exercise graph should trigger interaction with the student according to the feedback strategy.

Therefore we extended the graph with feedback attributes. The graph as a whole can be associated with an attribute containing the exercise description; all activities can be associated with different attributes for feedback control, i.e. text messages that give hints about what the next activity might involve (pre messages), or text messages that give feedback about detected activities (post messages). An example for activity A1 from our example exercise looks like Listing 1.3.

While our message mechanism provides the technical means for the implementation of various feedback strategies, the evaluation and choice of an appropriate strategy resides with the exercise author.

Listing 1.3. Example feedback attributes

```
pre_message = "You will need at least one host connected
               to network 'n1'."
2 post_message = "Network 'n1' detected."
```

Probing

While the verification of activities based on passively observed network packets works for many activities, there still are limitations. One such limitation occurs when an activity needs to be verified that does not have immediate results in the form of network packets.

An example for that would be A9 from our example exercise: the routing functionality has to be activated on the router. Students can do that by setting the appropriate kernel flag on the router if this flag is not enabled by default (Enable IP Forwarding: `echo 1 > /proc/sys/net/ipv4/ip_forward`). This however will not result in the occurrence of observable network packets, until packets are sent to the router for being routed. A possible solution would be to ask the student to send appropriate network packets himself. We followed a different approach. For detecting certain activities we inject special predefined network packets into the Netkit environment to provoke a certain predictable behavior. This behavior can also be expressed as a predicate. In the routing example we inject an Ethernet packet addressed to the router into the client network that is addressed to a host in the server network (which does not have to exist) on the IP level. If routing is enabled in the router, the router will try to reach that host in the server network using ARP requests. These packets can be used to verify that routing is indeed enabled on the router.

Such a “probing” packet can be assembled by strictly following the network stack, starting with an Ethernet frame. The destination MAC address must be the router’s NIC connected to network n1. In Netkit, the MAC address of a network interface is bound to the name of the client, resulting in a predictable MAC address for the router’s first NIC `eth0: 0aab64910980`. The source MAC address can be virtual, e.g. `eeba7b99bca5`, followed by an

IPv4 ethertype identifier (0x0800). The encapsulated IP packet starts with the version identifier (0x4), followed by mandatory header fields, e.g. length and checksum. The source IP address can be virtual but should be located within the IP range of network n1. The destination IP address can also be virtual but must be part of subnet n2. The IP packet encapsulates an ICMP echo request just to get a complete and valid network packet. This customized packet layout can be represented by a hexadecimal character array, e.g. 0aab6491 0980eeba 7b99bca5 08004500 001c1234 4000ff01 549c0a00 00010b00 00100800 f7fd0001 0001. Tools, e.g. PackEth, can help authors to design and validate such packets.

We extended the graph, so that every activity can be associated with a custom network “probing” packet to be sent once before verifying its condition. While that actively alters the environment, it enables the verification of additional activities.

3.3 The Electronic Exercise Assistant

In order to support a student while working on an exercise, we developed an exercise assistant, which can be used in the VCL. As shown in Fig. 8, the exercise assistant is composed of three components: reasoning engine, feedback engine, and an interface to the student’s working environment called Netkit interface.

The reasoning engine itself is composed of a reasoner and a knowledge base, which contains a TBox (“terminology box”) and an ABox (“assertion box”). The TBox contains knowledge about the domain, i.e. our ontology, in the form of predefined predicates that can be extended by the author with exercise specific extensions, while the ABox contains the concrete instantiations.

Fig. 8. Architecture of electronic exercise assistant

The data in the ABox is obtained through an interface to the “real world”, in our case the Netkit interface. The Netkit interface consists of one or more Ghost Hosts [26] that record network packets from their respective Netkit network, extract the information in them and store that information in the ABox. The Ghost Hosts can also be used to inject special network packets into the environment.

The feedback engine is the part where the activity graph will be processed. Our exercise assistant is able to read an exercise graph stored in the GraphML format [32]. Once read, the activities are continuously processed according to their interdependencies, starting at the source nodes which represent activities without preconditions. Processing the activities in this case means verifying their conditions and giving the student feedback according to the feedback attributes of that activity. Once the activity is completed it will be removed from the graph and thus as a precondition for its successors. The feedback engine can also use the Netkit interface, respectively the Ghost Hosts, to insert custom network packets into the environment in order to provoke certain network behaviour to verify an activity’s condition using the reasoning engine.

The Exercise Assistant is a software program written in the programming language *C* using SWI-Prolog [25] as the reasoning engine.

Using the VCL, the window layout of the desktop presented to the students looks like Fig. 9. The exercise assistant shell is a window where the student can keep track of the feedback generated by the feedback engine. The linux shell is a window where the student is able to administrate and use Netkit in order to e.g. create hosts and networks. Once a host is started, it will open a respective shell enabling the student to administrate the host itself. Further hosts, e.g. the router and the server, will open respective shells too.

Fig. 9. DVCL desktop draft

Example

The Figs. 10, 11, 12, 13, 14, 15, 16, 17 and 18 show the exercise assistant shell, guiding the example exercise in which we replaced the linux desktop window by a sketch to improve the readability. We authored the activities of Table 1 according to the exercise graph of Fig. 5 and added verbose feedback. The introduced routing predicate is used to verify the final activity (A10). The intermediate activities too have been modeled using our ontology, partially by utilising probing packets. Once started, the exercise assistant introduces the exercise by displaying the exercise description (see Fig. 10). Starting with the activities without precondition (A1 and A2), the exercise assistant will prompt the student using the respective `pre_messages`.

The student can start solving the exercise according to Listing 1.1. After the first command `vstart client --eth0=n1` is entered using the linux shell, the exercise assistant is able to confirm this valid activity (see output in Fig. 11).

While A1 is being marked as verified, using the respective `post_message` of A1, the remaining independent activities without preconditions will be displayed again, superseding the preceding messages. According to the exercise graph, the

Fig. 10. EA guiding example 1

student is now able to choose A2, A3 or A5 as the next activity. Starting the router connected to network n1 and n2 results in a verified presence of n2 (see output in Fig. 12).

Fig. 11. EA guiding example 2

Fig. 12. EA guiding example 3

While the presence of the two networks is verified now, the exercise assistant is not able to detect whether the student has started the server, unless its network interface card is assigned an IP address. Therefore the `pre_messages` are authored to prompt the student properly. Choosing to assign the client's IP address as next activity, using the command `ifconfig eth0 10.0.0.1 up` in the client shell, will result in a verified activity A3 (see output in Fig. 13).

Fig. 13. EA guiding example 4

Still missing IP addresses of the router's and server's NICs, the student can proceed to configure the router's NICs (see output in Fig. 14 and succeeding output in Fig. 15).

Fig. 14. EA guiding example 5

Having verified that the two NICs of the router are present, the exercise assistant is able to verify A9 using a probe packet. For the simple reason that routing is enabled per default for hosts in the Netkit environment, the condition of A9 can be verified immediately (see output in Fig. 16).

After assigning an IP address to the remaining NIC of the server, the student has to alter the routing table on the client and on the server. The exercise assistant is also able to verify these activities by using probing packets (see output in Fig. 17).

Finally, the student is asked to demonstrate the routing functionality by sending packets between the client and the server using the intermediate router. One valid solution is to use the command `ping`.

Once the final activity is verified, the exercise assistant congratulates the student and then quits (see output in Fig. 18).

Fig. 15. EA guiding example 6

Fig. 16. EA guiding example 7

Fig. 17. EA guiding example 8

Fig. 18. EA guiding example 9

4 Applicability and Educational Enhancements

4.1 Security

The DVCL with Central Authority (CA) is able to connect students' local labs, even if they are distant from each other. The system ensures, that no other hosts will be harmed by malicious network packets. The topic, that was not covered so far, is, that there is always a risk that third parties will try to unauthorized use, interfere or attack *our* system.

This risk will increase by the importance of the system. If our system, meaning the DVCL client and the CA server, is actually used in a real-life educational setting, additional requirements appear. If the system is e.g. used to grade a student's work, it will be likely that someone will try e.g. to influence the results.

Use Case Scenario

The DVCL with CA system allows that remote students can work together. While the students will use their own computer, the university has to provide the CA. In our use case scenario we assume that a student uses the DVCL client outside the university (e.g. at home) utilising an internet connection to connect to the CA. A mobile environment using a cellular data plan is also possible.

Figure 19 shows a corresponding use case diagram of the DVCL with CA. The left side represents the student's home, connected via internet (cloud in the middle) to the university's CA on the right side. A student, as a user of his desktop or laptop computer, uses the DVCL client to interact with a university's CA using a control channel connection. An additional data channel connection will be established for each remotely connected virtual network.

Fig. 19. Use case diagram of the DVCL with CA

Security Issues

During the development of the DVCL with CA system some security issues appeared [65]. Details for each issue are listed in Table 2. This table first describes the security **issue**. E.g. the first issue addresses the fact that everybody is able to use the CA if the IP address is known. This may be the desired behaviour for a general public learning environment, but the security **objective** for a university would be to limit the access, e.g. to their own students. **Affected** by this issue is the control channel which is open for everyone to connect to. An appropriate counter **measure** is to add a mechanism that users have to authenticate using a previously issued username and password first. If the credentials are valid, the CA authorises the user, otherwise the user will be rejected.

Security Measures

According to Table 2, four security issues were identified. These issues can be resolved by adding (resp. implementing) the following seven measures to our DVCL client server architecture.

Measure 1: Credentials. There are already existing, common concepts to issue and verify user credentials. For a university it could be wise to use existing directories, e.g. via Lightweight Directory Access Protocol (LDAP) or Network Information Service (NIS), to authorise students and to keep the administration effort low. For our prototype, we decided to issue and use a combination of username and password, stored and verifiable using a text file on the CA. Once a student will connect to the CA, a valid credential will be required first. For security reasons, it is recommended for final productive systems to store the passwords hashed rather than in plain text. Equipped with a **username** and a **password**, the DVCL CA tries to find a matching combination using the database. Once a valid combination is found, the user is authenticated, otherwise rejected. We successfully added this measure to our DVCL with CA prototype.

Table 2. Security issues and counter-measures

No. 1	Authenticity of the user
Issue	Everybody can use the CA if the IP address is known/public. Furthermore, users or third parties can also connect to an already started session without connecting to the control channel first. As a result it is possible to observe the network traffic of a certain session and also to inject packets
Objective	Only certain authorised users, e.g. the participants of a network course, should be able to use the CA provided by the university. Furthermore, only authenticated users should be able to connect to a session
Affected	Control channel on the CA's side, data channel on the CA's side
Measure 1	The CA shall require a username and password first to access the control channel
Measure 2	The CA shall issue an access token for authenticated users on the control channel, which will be required to use a data channel
No. 2	Authenticity of the CA and a CA session
Issue	The source code of the DVCL with CA will be public, because everybody should be able to serve a CA
Objective	The university's CA as well as their sessions should be authentic
Affected	CA, CA session
Measure 3	A verifiable certificate shall be added to the CA
Measure 4	A verifiable certificate shall be also added to the CA sessions
No. 3	Confidentiality of the control and session data
Issue	Data sent between the client and the CA resp. session can be read and also reused (e.g. the credentials) by third parties, if they are in a privileged position
Objective	The data transmitted between client and CA resp. sessions should be confidential
Affected	Control channel and data channel
Measure 5	The data transmitted on the control channel should be encrypted
Measure 6	The data transmitted on the data channel should also be encrypted
No. 4	Integrity of the control and session data
Issue	Data sent between the client and the CA can be modified by third parties, if they are in a privileged position, e.g. within the same local network using a man-in-the-middle attack
Objective	Nobody should be able to modify commands or packets from or to a student's lab. This is essential if the DVCL with CA will be used for an exam
Affected	Control channel and data channel
Measure 7	A verifiable signature for the transmitted data shall be added

Measure 2: Token. The control and the data channel are separated, independent channels and rely on different transport protocols (TCP and UDP). An authorised connection on the control channel does not involve the data channel. We decided to issue an access token on the control channel when a user is successfully authorised. This token will be required to authenticate on the data channel. Since the token will be issued and verified using an encrypted connection (see Measure 5 and 6: Encryption), it can not be captured and reused by third parties.

Measure 3 and 4: Certificate. A common concept to be able to verify the authenticity of a certain host is to use a certificate based on a public key infrastructure (PKI). A PKI involves a private and a public key pair, based on strong mathematical algorithms. A message encrypted with a private key, can only be decrypted using the corresponding public key, and vice versa. For server authentication, the client uses the server's public key to encrypt a secret key. The server can get access to this secret key only if it can decrypt the data from the client with the correct private key. A common software package that is able to deal with a PKI is Open Secure Socket Layer (OpenSSL). Using OpenSSL, it is possible to create a private key and a certificate for the server. The key `server.key` has to be private and stored on the server while the certificate `server.crt` is public and will be used by the server to identify itself, and also by the client to verify the server's authenticity.

Using and verifying a certificate can be added with some lines of code to the DVCL environment. When the DVCL client connects to the server, it is able to verify the server's authenticity. In case of an unexpected or faked certificate, a message will be printed to the student and the connection will be aborted.

Measure 5 and 6: Encryption. A common method to use encryption in software components is to use an existing, public software library, for example the Open Secure Socket Layer (OpenSSL), which provides different symmetric and asymmetric cryptography algorithms and adjustable key sizes. The implementation of such a library into an application is rather simple, compared to a self-made library. We decided to use TCP-based Transport Layer Security (TLS) [22–24] for the control channel and UDP-based Datagram Transport Layer Security (DTLS) [20, 21] for the data channel.

Measure 7: Signature. The use of OpenSSL for encryption and decryption does also ensure data integrity by calculating and verifying a message digest. This digest of the message will be calculated and appended to the encrypted data before it is sent to the network. When the message arrives at the destination node, OpenSSL recalculates the digest based on the data and compares that digest to the digest appended to the message. If the values do not match, the data has been corrupted and will not be processed. This is a built-in process of the TLS specification.

4.2 GUI

Initially the DVCL client was designed as a tool to be used in a command line interface (CLI) environment. The CLI is a text-based input-output system provided by the operating system, where a user can interact with the system or a service by entering certain commands including parameters and receiving textual responses. The DVCL client requires providing different parameters, e.g. the IP and the port of the DVCL's Central Authority (CA). The benefits include a flexible and easy way to add, remove and change parameters, the re-usability in other programs (batch-mode), and low development effort. A shortcoming however is that students first have to learn utilising the CLI including certain parameters in order to learn with the DVCL. One possibility would be to predefine values as far as possible, but this will not be feasible for certain parameters, e.g. the name of the local network. A way to improve the usability of the DVCL can be the introduction of a graphical user interface (GUI) to control the DVCL client [65]. Since the students are usually more familiar with using a GUI instead of using a CLI, a lower training period can be expected. This can result in an increased acceptance level, enables more time for learning, and finally can lead to an improved learning outcome. In order to reduce the training period, the GUI should support typical DVCL use cases. Also common misconfigurations should be prevented.

Table 3 lists typical use cases, which can occur while utilising Netkit and the DVCL client. The table starts with different cases on working with Netkit, followed by major cases occurring while administrating the CA using the client, and closes with cases originated by the interaction of Netkit and the client, respectively

Table 3. Use cases

No.	Case	GUI
1	Create a host without a network connection	☐
2	Create a host with a network connection	☒
3	Create a host with more than one network connection	☒
4	Login to the CA with username and password	☒
5	List available remote sessions	☒
6	Create a new remote session	☒
7	Delete a remote session	☒
8	Create a local network without a remote connection	☒
9	Connect a local network to a remote session	☒
10	Connect a local network to more than one remote session	☐
11	Connect more than one local network to one remote session	☒
12	Disconnect a local network from a remotely connected session	☒

Explanation:

☒ This case should be supported via GUI.

☐ This case should not be supported via GUI.

a remote connection. All use cases, except 1 and 10, should be supported in a GUI. Creating a host without a network connection does not make sense in a learning environment which focuses on networking practices. Connecting a local network to more than one remote sessions is technically possible, but increases the risk of a circle, which can lead to a failure of all involved systems.

We developed an example GUI for the DVCL client, which supports the use cases listed in Table 3. This GUI can be started as usual by clicking on a program icon. This can also be an automatic process within the DVCL environment using the autostart feature of the underlying operating system. The screenshots shown in Figs. 20, 21, 22 and 23 were captured using an Ubuntu Linux distribution. The look, fonts and the colors were set by the Ubuntu theme and will look different on other desktop environments.

At first, a logon screen will appear as shown in Fig. 20.

Fig. 20. GUI login

This screen is responsible to support use case no. 4. The address of the CA and also the path to the certificate are predefined but still editable. Also a username and a password have to be entered in order to continue. The button “Login” establishes a connection to the given CA and sends username and password for verification. The certificate will also be validated. If one of these processes fails, the connection will be aborted and the GUI will report an error.

If the credentials and also the certificate are successfully validated, the main screen will appear as illustrated in Fig. 21. This screen consists of three essential areas: The **local area** on the left, the **remote area** on the right, and the **status area** at the bottom.

The **local area** can be used to administrate local Netkit hosts and networks and thus is responsible for use case no. 2 and 3. Using the button “Create virtual Host/Network” will present a dialog shown in Fig. 22a. This dialogue can be used to create a new virtual host connected to at least one virtual network. A hostname and at least one named network are required, fulfilling use case no. 1. It is also possible to add and name additional networks, for example to create a router. The host will run in a new window, this is identical to a start via CLI (e.g. `vstart client --eth0=net1`). The new network(s) will appear in a list at the left area. This list also captures virtual networks, which are not started

Fig. 21. GUI main view

(a) Create a new Netkit host

(b) Create a new remote hub

Fig. 22. GUI options

using the GUI but the CLI. We utilise a function of the Linux kernel called *inotify* to observe a certain directory and to detect if something has changed. If a Netkit network was started, a file handle will be placed in a certain preconfigured directory. This file handle will be used by the virtual hosts to connect to a certain virtual network.

An example screen can be found in Fig. 23.

The **remote area** can be used to administrate sessions of the CA. In the GUI, the term *session* is mapped to the closer matching network term *Hub*. Already existing sessions will be listed directly after the login, according to use case no. 5. A new session can be created with the “Create Hub” dialogue, which is presented in Fig. 22b. The session first requires a name. Additionally, the creator of the session as the “owner” is able to set a password. Other participants, e.g. the learning group members, have to know and also have to enter this password in order to connect to this session. Finally, a new remote session will be created on the CA. This session will appear immediately in the remote area. An example screen can also be found in Fig. 23.

Local networks can stay local, according to use case no. 8. A local network can also be connected to a remote session (use case no. 9). This requires to select a previously created network in the local area and an existing session in the remote area first. An established connection between a local network and a remote session will be displayed in the **status area**. This connection can be terminated by selecting the related entry and using the button “Disconnect” (use case no. 12). It is possible to connect more than one local network to the same remote session (use case no. 11). An additional connection attempt from an already connected local network will be aborted and a message will pop up (use case no. 10).

Fig. 23. GUI showing a connected network

4.3 Virtual Classroom Prototype

Our virtual classroom prototype [65] is based on, utilises and extends our existing DVCL (cf. Sect. 2.2) with Central Authority (cf. Sect. 2.3), including the Intelligent Tutoring System (cf. Sect. 3), security enhancements (cf. Sect. 4.1), and the GUI (cf. Sect. 4.2).

A student has first to login to use the DVCL (see Fig. 20). Once logged in, the main screen with the ‘lobby’, resp. the classroom chat will be presented as illustrated in Fig. 24. The lobby consists of three areas: An area where messages will appear to the left, an area with a list of users currently logged in to the right, and an area with a text input box to create and send own messages at the bottom.

Fig. 24. DVCL classroom main screen

The student will have more than one options to proceed. He can start a private chat with another student (communication activity), he can invite another student to create a group or will be invited by another student to join a group (both are organisational activities), or he can proceed to the exercise selection area (educational activity). Starting a private chat will open a new tab similar to the lobby chat, except that messages will only be readable by the selected

dialogue partner. A private chat is independently of other states. For manual group forming, the student can select another student to invite him into a group (see Fig. 25a), or he will be invited to join a group. A pending invitation has to be accepted or rejected in a dialogue prompt (see Fig. 25b). If a group with at least two students is formed, a new tab will appear for the group chat. This tab is similar to the lobby chat, except that messages will only be readable by the group members.

Fig. 25. Manual group forming

Despite a group already exists or not, the student can access the exercise selection area (see Fig. 24). In Fig. 26, one exercise titled *Basic Networking A1* is included with a desired group size of 3. After selecting the exercise, the student has to decide to work alone or to search for a group. If the student is already a member of a group, which does not have an exercise selected yet, a third option will be available to assign the selected exercise to this pre-existing group.

Work alone means that the exercise will start immediately without waiting for group members. By selecting *Search a group*, the DVCL environment will look for an already existing group, where at least one group member is still missing and the group decided to solve the same exercise. If such a group is found, the student will be assigned to this group. If no suitable group can be found, the student will be assigned to a waiting queue (see Fig. 27). In case that the queue already holds a suitable student (who has selected the same exercise), a new group will be created for them. Finally, the group size will be checked. If the group is already complete, which means that the group has the predefined group size, the exercise will be started. If the group is still incomplete, the group members can vote to start the exercise anyway (see Fig. 28), but this action will requires the approval of all other group members (see Fig. 29).

Fig. 26. Exercise selection

Fig. 27. Student is assigned to waiting queue

Fig. 28. Overwrite a predefined group size

Fig. 29. Confirmation dialogue

5 Implementation and Evaluation

5.1 Implementation

The VCL in Sect. 2.1 make use of Netkit¹, User Mode Linux², Knoppix³, and the virtualisation layer VMware Player⁴ or VirtualBox⁵.

The DVCL in Sect. 2.2 is based on a software component, consisting of a Ghost Host and a Remote Bridge. The Ghost Host provides an interface to the virtual network, where Ethernet frames can be extracted and injected. The Ethernet frames that are extracted by the Ghost Host, are first encapsulated in the IP/UDP protocol (acting as transport protocol), and next sent to the Remote Bridge endpoint of a fixed distant destination. For incoming data, the IP/UDP protocol is removed by the Ghost Host and the Ethernet frames are injected into the local network. We developed both components using the programming language *C*.

The DVCL with Central Authority in Sect. 2.3 makes use of our developed components Ghost Host and Remote Bridge, extended by a new software component called Central Authority. We developed the Central Authority also using the programming language *C* using threads, shared memory, and TCP/UDP sockets.

Our Intelligent Tutoring System in Sect. 3 is also a software program, written in the programming language *C*. It makes use of Ghost Hosts to connect to Netkit's virtual networks, and a file reader interface to load and parse the exercise configuration. The reasoning engine is included by the software package SWI-Prolog⁶.

The Security Measures in Sect. 4.1 improve the DVCL with Central Authority. Security measure 1 and 2 are based on source code extensions of the pre-existing *C* code, measures 3 to 7 additionally make use of the software package OpenSSL⁷.

The Graphical User Interface (GUI) in Sect. 4.2 and also the Virtual Classroom Prototype in Sect. 4.3 are developed using the programming language *C*. We use GTK⁸, a multi-platform toolkit for creating graphical user interfaces.

5.2 Evaluation

Before we started the development of our DVCL with CA system, we evaluated [29, 35] an on-campus practical networking course at the Cologne University

¹ Netkit, <http://wiki.netkit.org>, Online, accessed April 2018.

² The User-mode Linux Kernel Home Page, <http://user-mode-linux.sourceforge.net>, Online, accessed April 2018.

³ KNOPPIX, <http://www.knopper.net/knoppix>, Online, accessed April 2018.

⁴ VMware Player, <http://www.vmware.com>, Online, accessed April 2018.

⁵ VirtualBox, <http://www.virtualbox.org>, Online, accessed April 2018.

⁶ SWI Prolog, <http://www.swi-prolog.org>, Online, accessed April 2018.

⁷ OpenSSL - Cryptography and SSL/TLS Toolkit, <https://www.openssl.org>, Online, accessed April 2018.

⁸ GTK+, <https://www.gtk.org/>, Online, accessed April 2018.

of Applied Sciences, where students have to work out and solve assignments. They also have to defend their solutions. A special property of this course is that the students are given a high degree of flexibility during their assignment preparation time. We wanted to discover the learning behaviour of students who are free to choose their learning environment in order to successfully complete the course. In addition, we wanted to determine the success of the course. The results were used as a justification and also as motivation for our work with respect to an optimised alignment of our technical implementation and practical course concepts in place.

249 students signed up for the practical course “Communication technology and networks”. 191 students passed, i.e. they worked out the assignments, and demonstrated and defended their solutions successfully. Most of the 23% unsuccessful students registered but did not participate at all; some seemed to have other shortcomings; a few tried but could not defend their solution properly. 178 of them (71%) participated in our evaluation process.

While the preferences for learning environments and behaviour were fairly distributed, a predominant majority of the students thought of *working in groups* as well as *receiving guidance and feedback* as crucial to their learning success. Students are also interested in new and innovative learning environments. It is important that these new environments do not replace more traditional ones, but rather add on to them.

One way of modernising the practical course would be the introduction of an e-learning system, which would be explicitly welcomed by 49% of the students. In addition to that, nearly half of the students said that they would like to work independently from the lab at least partially, which they would be enabled to do by the introduction of such a system.

Given the students’ preference for group working and guided learning, one should take these two key factors into account when introducing an e-learning system. This means that, in order to gain the students’ acceptance, an e-learning system should enable collaboration as well as guidance, and shouldn’t be limited to simply providing an environment for solving assignments online.

6 Related Work

The idea of using an isolated network as an environment to perform IT related tasks for the purpose of research or education is widely recognised [36–40]. There are two general approaches to create such an environment:

The first approach is to create or use an isolated, physical network with physical hosts that is separated from an operational network such as a campus network [36,41]. This isolation may be achieved by physical separation of the networks or by using components like firewalls to restrict data flow between network areas [42]. Within this isolated network the students can perform exercises and work with a real-world like network setup. Remote access to such a network may be granted by using remote access technologies such as Virtual Private Network (VPN) [43]. Administration and maintenance of such a lab however is

labour-intensive. Students work in the lab with super-user rights and can modify system configurations at will. After a session, it is necessary to clean up system configurations, which may even require reinstalling operating systems.

The second approach makes use of virtualisation technologies to create an isolated, virtual network with virtual hosts. Literature refers to such an environment in the context of education or e-learning usually as a virtual lab [47, 50–54]. This approach significantly reduces the amount of physical hardware resources (e.g., switches, routers, hosts), since the required resources are created by virtualisation. Cleaning up or reinstalling a virtual lab simply means reloading the virtual environments, which can even be an automated task.

Literature also reports two main approaches to provide an isolated network. In the first one, the environment is located at a central place, usually at the university [44–49] and students can get physical or remote access by using a secured network connection. A central place could also be a cloud [55–57] or a federated lab [58, 59]. Although such labs may be accessed remotely at any time from any place, they are generally not easily scalable. Allowing an arbitrary number of students to participate at the same time requires students to reserve timeslots in advance for working in the lab. This may impose restrictions for students in distance education, who usually study in evening hours and weekends. Provisioning a remote lab for peak access outside office hours, may result in a largely over-dimensioned lab with a low average degree of utilisation and hence a waste of resources.

Second, the environment is provided as a preconfigured, stand-alone software package which can be installed and used by students on any computer, usually their private computer [18, 60–62]. This gives the students the opportunity to safely carry out assignments wherever and whenever they want to. We used the Virtual Computer Security Lab [18] as base for our research work.

Literature also reports many scopes where intelligent tutoring systems (ITS) were developed or applied, e.g. in the scope of teaching mathematics [8, 9, 63, 64], databases [10–12], programming languages [13] like JAVA [14, 15] or Haskell [16], IT security [1, 3, 5], and physics [2, 4]. We added an ITS to model and also to verify networking exercises.

7 Conclusion

Step by step we developed a virtual classroom for use in different educational contexts for cybersecurity network education. The classroom has been realised both in a distance learning context and in an on-campus context. This was made possible by extending a basic virtual security lab (VCL) with distributivity, a central authority, tutoring by an exercise assistant, and finally security and educational enhancements.

Starting from the VCL as a basis, which we had designed as an isolated system, we first enabled group work for students by connecting two VCLs on the network level. In the resulting Distributed Virtual Computer Lab (DVCL), remote students are able to work synchronously together, using an intermediate connection, e.g. the internet. The DVCL system ensures that the isolated

network stays isolated and cannot intercommunicate with the internet. We next extended and improved the DVCL by providing a central authority (CA). The CA simplifies the usage of our DVCL for the students (and also for academic staff) and - in addition to it - avoids administrative configuration errors while connecting remote labs.

When working on practical exercises in a lab environment, students need guidance and feedback. It is challenging to provide such feedback and guidance in a virtual lab when human course advisors are not available. This is e.g. the case when students use the DVCL at home outside office hours. We showed that network traffic captured in a lab can give some indication of what a student has already configured according to a certain exercise. We used this insight to develop an Electronic Exercise Assistant. This software program is able to recognise the progress of students in an exercise and can provide appropriate feedback and support, based on preloaded rules and conditions. This significantly improves the learning situation for students. Besides this automatic support, the exercise assistant can also verify intermediate and complete solutions of an exercise.

In order to turn our DVCL into a virtual classroom, we made several enhancements. A first enhancement is to resolve security issues and prevent misuse of the DVCL. Such security requirements are necessary when using the DVCL in an actual, real-life learning environment. A second enhancement is to add a GUI which supports students in connecting to the lab and joining sessions with other students. Our final enhancement was to extend our virtual classroom with facilities to support social interactions, such that students e.g. can meet, form learning groups, talk, and discuss. These enhancements allow students to communicate, organise, and carry out educational activities in our virtual classroom in a similar way as in an on-campus classroom setting.

We described in this paper the technology behind our virtual classroom, including both the technical implementation details and the educational principles on which it is based. It is now up to the educators to use the virtual classroom and to evaluate its impact on the learning of students. We intend to use the virtual classroom in practice at the Cologne University of Applied Sciences and the Open University in the Netherlands, and to continue our research on the application of DVCLs in distance education.

References

1. Hu, J., Meinel, C., Schmitt, M.: Tele-lab IT security: An architecture for interactive lessons for security education. In: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2004, pp. 412–416. ACM, New York (2004)
2. Albacete, P.L., VanLehn, K.: The conceptual helper: an intelligent tutoring system for teaching fundamental physics concepts. In: Gauthier, G., Frasson, C., VanLehn, K. (eds.) ITS 2000. LNCS, vol. 1839, pp. 564–573. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45108-0_60
3. Mahdi, A.O., Alhabbash, M.I., Naser, S.S.A.: An Intelligent Tutoring System for Teaching Advanced Topics in Information Security (2016)

4. Vanlehn, K., et al.: The andes physics tutoring system: lessons learned. *Int. J. Artif. Intell. Educ.* **15**(3), 147–204 (2005)
5. Hu, J., Schmitt, M., Willems, C., Meinel, C.: A tutoring system for IT security. In: Irvine, C., Armstrong, H. (eds.) *Security Education and Critical Infrastructures. ITIFIP*, vol. 125, pp. 51–60. Springer, Boston, MA (2003). https://doi.org/10.1007/978-0-387-35694-5_5
6. Corbett, A.T., Koedinger, K.R., Anderson, J.R.: Intelligent tutoring systems. *Handb. Hum. Comput. Interact.* **5**, 849–874 (1997)
7. Psotka, J., Massey, L.D., Mutter, S.A.: *Intelligent Tutoring Systems: Lessons Learned*. Psychology Press, Hillsdale (1988)
8. Melis, E., Siekmann, J.: ACTIVEMATH: an intelligent tutoring system for mathematics. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) *ICAISC 2004. LNCS (LNAI)*, vol. 3070, pp. 91–101. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24844-6_12
9. Canfield, W.: ALEKS: a web-based intelligent tutoring system. *Math. Comput. Educ.* **35**(2), 152 (2001)
10. Yang, F.-J.: A virtual tutor for relational schema normalization. *ACM Inroads* **2**(3), 38–42 (2011)
11. Kenny, C., Pahl, C.: Automated tutoring for a database skills training environment, vol. 37. *ACM* (2005)
12. Suraweera, P., Mitrovic, A.: An intelligent tutoring system for entity relationship modelling. *Int. J. Artif. Intell. Educ.* **14**(3, 4), 375–417 (2004)
13. Queirós, R.A.P., Leal, J.P.: PETCHA: a programming exercises teaching assistant. In: *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE 2012*, pp. 192–197. ACM, New York (2012)
14. Sykes, E.R., Franek, F.: A prototype for an intelligent tutoring system for students learning to program in Java (TM). In: *Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education*, pp. 78–83 (2003)
15. Vesin, B., Ivanović, M., Klačnja-Milićević, A., Budimac, Z.: Ontology-based architecture with recommendation strategy in java tutoring system. *Comput. Sci. Inf. Syst.* **10**(1), 237–261 (2013)
16. Jeuring, J., van Binsbergen, L.T., Gerdes, A., Heeren, B.: Model solutions and properties for diagnosing student programs in ask-elle. In: *Proceedings of the Computer Science Education Research Conference, CSERC 2014*, pp. 31–40. ACM, New York (2014)
17. Pizzonia, M., Rimondini, M.: Netkit: easy emulation of complex networks on inexpensive hardware. In: *Proceedings of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, TridentCom 2008*, pp. 7:1–7:10, ICST, Brussels, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering) (2008)
18. Vranken, H., Koppelman, H.: A virtual computer security lab for distance education. In: *Proceedings of the 5th IASTED European Conference on Internet and Multimedia Systems and Applications, EuroIMS A 2009*, pp. 21–27. Acta Press (2009)
19. Colmerauer, A., Roussel, P.: The birth of prolog. In: *The Second ACM SIGPLAN Conference on History of Programming Languages, HOPL-II*, pp. 37–52. ACM, New York (1993)
20. Rescorla, E., Modadugu, N.: Datagram transport layer security (2006). <http://www.ietf.org/rfc/rfc4347.txt>. Accessed 22 July 2014

21. Rescorla, E., Modadugu, N.: Datagram transport layer security version 1.2 (2012). <http://www.ietf.org/rfc/rfc6347.txt>. Accessed 22 July 2014
22. Dierks, T., Allen, C.: The TLS protocol version 1.0 (1999). <http://www.ietf.org/rfc/rfc2246.txt>. Accessed 22 July 2014
23. Dierks, T., Rescorla, E.: The transport layer security (TLS) protocol version 1.1 (2006). <http://www.ietf.org/rfc/rfc4346.txt>. Accessed 22 July 2014
24. Dierks, T., Rescorla, E.: The transport layer security (TLS) protocol version 1.2 (2008). <http://www.ietf.org/rfc/rfc5246.txt>. Accessed 22 July 2014
25. Wielemaker, J.: Logic programming for knowledge-intensive interactive applications. PhD thesis, University of Amsterdam (2009)
26. Vranken, H., Haag, J., Horsmann, T., Karsch, S.: A distributed virtual computer security lab. In: Proceedings of the 3rd International Conference on Computer Supported Education, CSEDU 2011, pp. 110–119. SciTePress (2011)
27. Haag, J., Horsmann, T., Karsch, S., Vranken, H.: A distributed virtual computer security lab with central authority. In: Proceedings of the Computer Science Education Research Conference, CSERC 2011, pp. 89–95. Open Universiteit, Heerlen (2011)
28. Haag, J., Karsch, S., Vranken, H., Van Eekelen, M.: A virtual computer security lab as learning environment for networking and security courses. In: Proceedings of the 3rd Annual International Conference on Computer Science Education: Innovation and Technology, CSEIT 2012, pp. 61–68. Global Science & Technology Forum (2012)
29. Haag, J., Witte, C., Karsch, S., Vranken, H., Van Eekelen, M.: Evaluation of students' learning behaviour and success in a practical computer networking course. In: Proceedings of the 2nd International Conference on E-Learning and E-Technologies in Education, ICEEE 2013, pages 201–206. IEEE (2013)
30. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York (2003)
31. Tanenbaum, A.S.: Computer Networks. Prentice Hall PTR, Upper Saddle River (1985)
32. Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., Marshall, M.S.: GraphML progress report structural layer proposal. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) GD 2001. LNCS, vol. 2265, pp. 501–512. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45848-4_59
33. Dike, J.: User Mode Linux. Prentice Hall, Englewood Cliffs (2006)
34. Dike, J.: A user-mode port of the linux kernel. In: Proceedings of the 4th Annual Linux Showcase & Conference, ALS 2000, vol. 4, p. 7. USENIX Association, Berkeley (2000)
35. Haag, J., Witte, C., Karsch, S., Vranken, H., Van Eekelen, M.: Evaluation of students' learning behaviour and success as a prerequisite for modernizing practical on-campus networking courses in higher education. *Yükseköğretim Dergisi/J. High. Educ.* **4**(2), 83–90 (2014a)
36. Bishop, M., Heberlein, L.: An isolated network for research. In: Proceedings of the Nineteenth National Information Systems Security Conference, pp. 349–360 (1996)
37. Agarwal, K.K., Critcher, A., Foley, D., Sanati, R., Sigle, J.: Setting up a classroom lab. *J. Comput. Sci. Coll.* **16**(3), 281–286 (2001)
38. Taylor, K.D., Honchell, J.W., DeWitt, W.E.: Distance learning in courses with a laboratory. In: Proceedings of the 26th Annual Frontiers in Education, FIE 1996, vol. 01, pp. 44–46. IEEE Computer Society, Washington (1996)

39. Lo, D.C.-T., Qian, K., Chen, W., Shahriar, H., Clincy, V.: Authentic learning in network and security with portable labs. In: 2014 IEEE Frontiers in Education Conference (FIE), vol. 00, pp. 1–5 (2014)
40. Bardas, A.G., Ou, X.: Setting up and using a cyber security lab for education purposes. *J. Comput. Sci. Coll.* **28**(5), 191–197 (2013)
41. Jakab, F., Janitor, J., Nagy, M.: Virtual lab in a distributed international environment - SVC EDINET. In: Fifth International Conference on Networking and Services, ICNS 2009, pp. 576–580 (2009)
42. Yang, T.A., et al.: Design of a distributed computer security lab. *J. Comput. Sci. Coll.* **20**(1), 332–346 (2004)
43. Yoo, S., Hovis, S.: Remote access internetworking laboratory. In: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2004, pp. 311–314. ACM, New York (2004)
44. Drigas, A.S., Vrettaros, J., Koukianakis, L.G., Glentzes, J.G.: A virtual lab and e-learning system for renewable energy sources. In: Proceedings of the 1st WSEAS/I-ASME Conference on Educational Technologies, EDUTE 2005, pp. 149–153 (2005)
45. Border, C.: The development and deployment of a multi-user, remote access virtualization system for networking, security, and system administration classes. In: Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2007, pp. 576–580. ACM, New York (2007)
46. Hu, J., Cordel, D., Meinel, C.: Virtual machine management for tele-lab “IT-security” server. In: Proceedings of the 10th IEEE Symposium on Computers and Communications, ISCC 2005, pp. 448–453. IEEE Computer Society (2005)
47. Keller, J., Naues, R.: Design of a virtual computer security lab. In: Proceedings of the IASTED International Conference on Communication, Network, and Information Security, CNIS 2006, pp. 211–215. Acta Press (2006)
48. Krishna, K., Sun, W., Rana, P., Li, T., Sekar, R.: V-NetLab: a cost-effective platform to support course projects in computer security. In: Proceedings of 9th Colloquium for Information Systems Security Education. The Printing House Inc. (2005)
49. Lahoud PhD (ABD), H.A., Tang PhD, X.: Information security labs in IDS/IPS for distance education. In: Proceedings of the 7th Conference on Information Technology Education, SIGITE 2006, pp. 47–52. ACM, New York (2006)
50. Brian Hay, K.L.N.: Evolution of the ASSERT computer security lab. In: Proceedings of the 10th Colloquium for Information Systems Security Education (2006)
51. Bullers Jr., W.I., Burd, S., Seazzu, A.F.: Virtual machines - an idea whose time has returned: application to network, security, and database courses. In: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2006, pp. 102–106. ACM, New York (2006)
52. Damiani, E.: The open source virtual lab: a case study. In: Proceedings of the Workshop on Free and Open Source Learning Environments and Tools, FOSLET 2006 (2006)
53. O’Leary, M.: A laboratory based capstone course in computer security for undergraduates. In: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2006, pp. 2–6. ACM, New York (2006)
54. Li, P.: Exploring virtual environments in a decentralized lab. *SIGITERes. IT* **6**(1), 4–10 (2009a)
55. Ellabidy, M., Russo, J.P.: Using the cloud to replace traditional physical networking laboratories. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education, SIGCSE 2014, p. 729. ACM, New York (2014)

56. Mhd Wael Bazzaza, K.S.: Using the cloud to teach computer networks. In: Proceedings of the 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), UCC 2015, pp. 310–314 (2015)
57. Salah, K.: Harnessing the cloud for teaching cybersecurity. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education, SIGCSE 2014, pp. 529–534. ACM, New York (2014)
58. Peterson, L., Anderson, T., Culler, D., Roscoe, T.: A blueprint for introducing disruptive technology into the internet. *SIGCOMM Comput. Commun. Rev.* **33**(1), 59–64 (2003)
59. Berman, M., Chase, J.S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R., Seskar, I.: GENI: a federated testbed for innovative network experiments. *Comput. Netw.* **61**, 5–23 (2014)
60. Li, P.: Exploring virtual environments in a decentralized lab. *SIGITERes. IT* **6**(1), 4–10 (2009b)
61. Li, P.: Centralized and decentralized lab approaches based on different virtualization models. *J. Comput. Sci. Coll.* **26**(2), 263–269 (2010)
62. Seeling, P.: Labs@home. *SIGCSE Bull.* **40**(4), 75–77 (2008)
63. Heeren, B., Jeuring, J., van Leeuwen, A., Gerdes, A.: Specifying strategies for exercises. In: Autexier, S., Campbell, J., Rubio, J., Sorge, V., Suzuki, M., Wiedijk, F. (eds.) *CICM 2008. LNCS (LNAI)*, vol. 5144, pp. 430–445. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85110-3_36
64. Gerdes, A., Heeren, B., Jeuring, J.: Properties of exercise strategies. In: Proceedings International Workshop on Strategies in Rewriting, Proving, and Programming, IWS 2010, Edinburgh, UK, 9th July 2010, pp. 21–34 (2010)
65. Haag, J.: DVCL: A Distributed Virtual Computer Lab for security and network education. Ph.D. thesis, Open Universiteit (2018)