DESIGN OF A VIRTUAL COMPUTER SECURITY LAB

Jörg Keller and Ralf Naues
LG Parallelität und VLSI
FernUniversität in Hagen
58084 Hagen, Germany
{joerg.keller,ralf.naues}@fernuni-hagen.de

ABSTRACT
We present the design and a prototype of a lab course on computer security, the necessity of which arises from the students' need to complement course work by hands-on experience. In order to guarantee maintainability of a number of Linux systems on which students change configurations, we decided to employ a virtual machine approach. This allows to reset configurations quickly without another costly operating system installation. We sketch the types of tasks the students are to perform, and our approach to check immediately whether students have completed a task. As students operate in larger groups, and the server hosting the virtual machines can only run a finite number of them simultaneously, a reservation scheme is employed to guarantee fair access for all participants.

KEY WORDS
Education, remote laboratory, resource sharing, IT-security, virtualization, semi-automatic assessment

1 Introduction

Computer security gains a growing importance in computer science and engineering curricula. Yet, courses on this subject have to be complemented by practical lab work in order to achieve full understanding of the subject matter. While this holds true for many branches of computer science, the value of training in computer security should not be underestimated: a subtle flaw in a firewall or a proxy server configuration can render useless all remaining efforts to secure a computer network!

For this reason we set out to define a set of tasks to be trained by computer science students in our computer science programme. As our institution is a distance teaching university, we specifically defined this lab course to be workable from a distance. Yet, as we focussed on network security, this was not really an issue, as most administrative tasks nowadays are not performed at a computer's console anyway, but via a remote shell or web-based via a browser.

The main issues to be considered were the following. First, a student and his supervisor need feedback on when a task is correctly performed. As many tasks consist of installing and/or configuring tools to protect a part of a network, this has to be done via a test. A hint whether the task is completed should be generated automatically, in order to relieve students from the unproductive waiting time until a human supervisor has checked their work. The latter happens still, but when the hint already indicates that there is still a hole in the firewall configuration, there is no need for the student to wait, and for the supervisor to be bothered. We therefore took care to implement a script for every task. Starting the script tests whether the main goals are achieved. This method also has the positive side effect that, in spite of a quite restricted manpower for guidance and supervision, a larger number of students can be admitted to this course. Self-study is further supported by providing students with task-specific links and hints.

Second, as a consequence of the large number of students in the lab course, many machines have to be provided. As network security often deals with regulating the communication between two machines, each student is provided two machines, one being the one to be protected, one being the "external" one that implements an attacker. As most of the tools in the lab course, such as IP filters, intrusion detection systems and the like, change machine configurations, machines cannot be shared between students although the machines run the multi-user operating system Linux.

Third, the large number of machines not only has to be provided but also actively maintained. If a student, working with administrator rights, misconfigures some tool, he may well have destroyed the whole machine configuration, so that it is necessary to get the system back in a stable state. This is difficult from a distance, and sometimes impossible, as the only solution may be to install the operating system anew. Besides the effort to do this, this would prevent the last task in a sequence of tasks being dependent on the completion of the earlier tasks, which is unrealistic in a normal network setting. Hence, what is needed is the possibility to save the machine state after the completion of each task, and the possibility to reset a machine to a saved state. This also simplifies maintenance of a large number of machines, as only one has to be configured as desired, and the others can be cloned by copying the state. We implement this by providing the students with virtual machines that all run on one physical server. The use of virtual machines has been reported recently for a lab course on operating system administration [1], yet the goals there were different. We are aware of two labs on IT-security employing virtualization [2, 3], yet the first

employs user mode Linux to realize virtual machines and VNC to avoid programs like ssh on the client side, and the second, while ZEN-based, focusses on modeling a separate network entity (like a router) with each virtual machine, and focusses on being able to configure different (virtual) network topologies. Also it does not support automatic test of task completion by scripts.

Fourth, although we employ a powerful server to host the virtual machines, the large number of participants prevents the students from all being active simultaneously, as the server can only run a restricted number of virtual machines simultaneously without being annoyingly slow. This is not really a problem, because the students, that normally work part-time and study part-time, have quite differing preferences with respect to their study hours. Yet, there is the need to coordinate their working hours by a reservation system, so that, when a student spends his scarce time on the lab course, the server is in fact available to host his virtual machine.

The working interface to the student is a remote shell, as is usual for system administration tasks in Linux systems. Yet, the students also need an interface to see their performance chart, the tasks alongside with links to further documentation, and hints towards the solution, and so on. Here we provide a web-based interface, that is coupled to a database that contains the participant data and their performance data.

After setting out the challenges in designing a network security lab for students in a distance, and pointing out the directions that our solutions take, the remainder of this article is organized as follows. In Section 2 we describe the architecture of the virtual network connecting the virtual machines of the students. In Section 3 we describe how we designed the tasks so that their completion can be efficiently checked. We also report on interface issues. In Section 4 we conclude and give an outlook on future work.

2 Lab Network Design

Figure 1 depicts the design of the network for the virtual lab course. The students connect via the internet to the lab server. The server's physical IP address *.*.*.100 is only used for maintenance purposes, and accordingly protected. The virtual network's connection to the internet is via *.*.*.101. This virtual server serves as a firewall and as a router to the network itself via a private IP 192.168.100.1. The network address translation (NAT) also performed in the virtual server. The virtual machines for the students are also in this network, they are depicted on the right. The server has a third virtual network interface (172.16.69.*) which only serves as a connection between physical server and virtual server, and hence will not play a role here. All virtual student machines have a second network interface towards a network 192.168.101.*, to connect them to the test server 192.168.101.1. This test server realizes a shared "external" machine, needed to test completion of some tasks, see next section.

The rationale behind this architecture is the following. The physical server itself shall only be accessible for administrative purposes such as configuring the VMware server that provides the virtual machines and the virtual network. Hence, this server can only be connected from a particular external computer in possession of the administrator. From this fact arises the need that the connection between the virtual server and the internet, which is used by the students to log in, has to be realized by a virtual network interface. As the virtual server also shields the virtual network against outside attacks, a fact that is sadly to be acknowledged these days, the connection to the virtual network requires another virtual network interface. A third virtual network interface is needed to communicate between virtual server and physical server. As three virtual network interfaces is the maximum number provided by VMware, there is only one network interface connecting to the virtual network of student machines. Those machines are the ones to be protected. However, there must also be an "external" machine that is used in the course of the tests at the end of the tasks. This machine has to be put in a different subnet. As it cannot be connected to the virtual server, there arises the need to connect this machine with the student machines by a second virtual subnetwork.

3 Task Design and Interface

3.1 Task and Test Design

The lab course start with the simple task that the students must acquire a certificate of our university, install a VPN client and a secure shell tool at their home computer, and connect to the lab server. Further tasks then detail installation and configuration of a simple firewall (iptables), of intrusion detection systems (tripwire and snort), and of network address translation.

With the example of the firewall we explain our philosophy for checking completion of tasks. A typical requirement for a firewall is that some services (identified via ports) are not accessible from the outside, or are only accessible from particular external computers. After the student has configured the firewall, he starts a port scan from an external virtual machine. This port scan will generate log entries in the firewall log for all accesses that were denied. If the firewall is correctly configured, then this port scan will also generate log entries for the service just restricted. The student completes his task by starting a script. This script will search the firewall log for the appropriate entry. If it finds the entry, then the task is considered completed, otherwise not. The script sends its results to the server where the results are entered into a database. When the student now look at his personal task performance web page, he sees whether the task is indeed completed.

This method also works for the other tasks, as all security measures typically either restrict access in some ways, or at least monitor whether an attempt is made to do something that is forbidden. As security tools typically

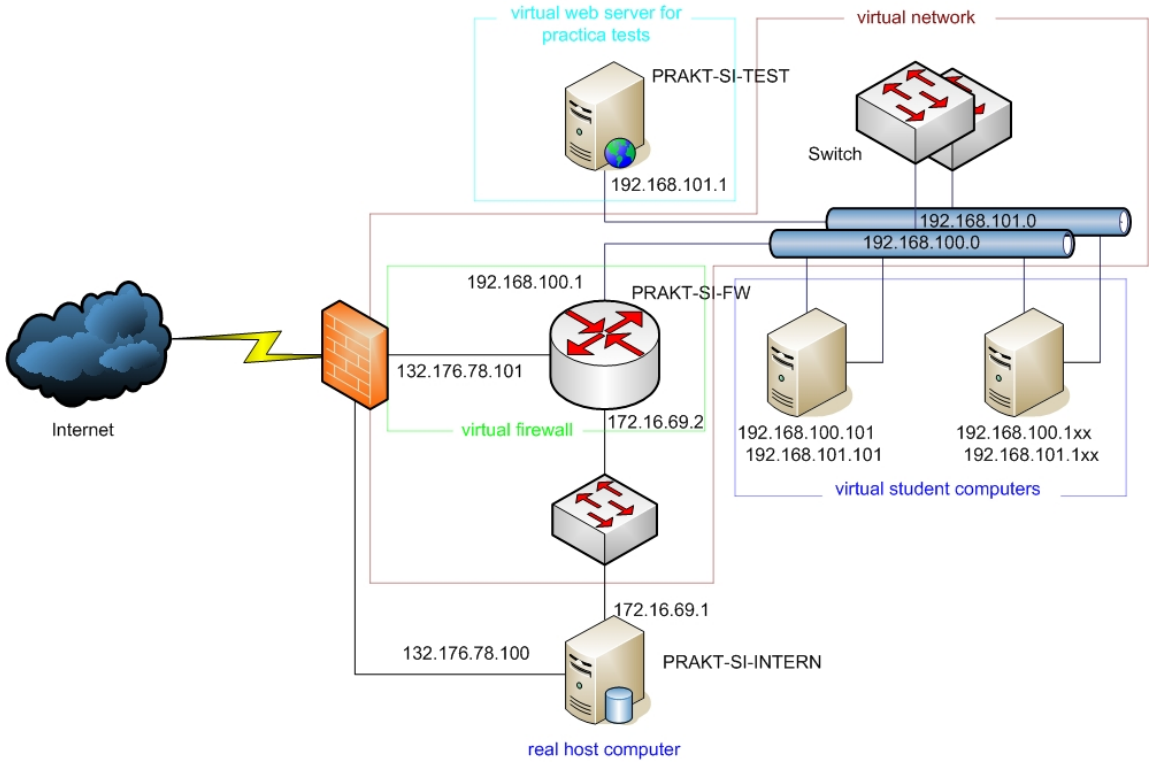

Figure 1. Design of virtual lab network

provide a log file in which they record all those violations, one can use this mechanism to generate tests. The test consists in some external actions that specifically do things that had to be forbidden in the task, and do some things that still should work. The task is completed if the log file records violations exactly for those actions that should be forbidden, i.e. it is also checked that there was not more restricted than demanded. While this seems a minor point, it is not to be underestimated in practice, because users are quite annoyed if after a firewall change, they are not able anymore to do their daily work. Also, a student may otherwise be able to complete tasks by simply forbidding everything. As the different tools use different log file formats, the test creation needs some skill and experience, but we believe that the fast feedback for the students is worth the effort. Obviously, the design of the tests also influences the design of the tasks themselves. While the domains of the tasks are fixed beforehand, the particular task has to be put in a way that allows to be tested efficiently. Fortunately enough, the security field seems to pose no difficulties in this respect, and have not encountered the situation so far, to be forced to change a task because it cannot be tested.

3.2 Student Interface, Reservation

The interface towards the student is web-based, see Figure 2. The students log into a web server that resides on the virtual lab server. There, they can access their personal performance chart together with the log excerpt, so that they get hints why they did not yet complete a task. Also, they can see all task formulations, together with links to tools and manuals, and with hints. In principle, a web-based secure shell frontend could have been integrated with those pages, so that students only would have to use a browser. However, as the secure shell tool is the standard tool for system administrators to work with, we thought it more realistic not to do this.

The students are supposed to use a second web-based interface, the CSCW tool CURE [4]. CURE provides the concept of rooms, where students can meet in groups, can post documents, comment on them, and perform similar tasks. CURE thus supports collaborative tasks. The rooms are entered with personalized keys. CURE has been extended to provide time-dependent keys [5] to control access to remote hardware lab. Hence, CURE can be used as a reservation scheme for the virtual server, to prevent overload. The maximum number of simultaneous reservations must be adapted to the server hardware in use.

4 Conclusions

Our virtual computer security lab has passed the first user tests and will be used in the coming fall semester for a lab course with about 50 students. Our current platform limits the number of concurrently working virtual machines to about 15 out of performance reasons. Therefore the reser-

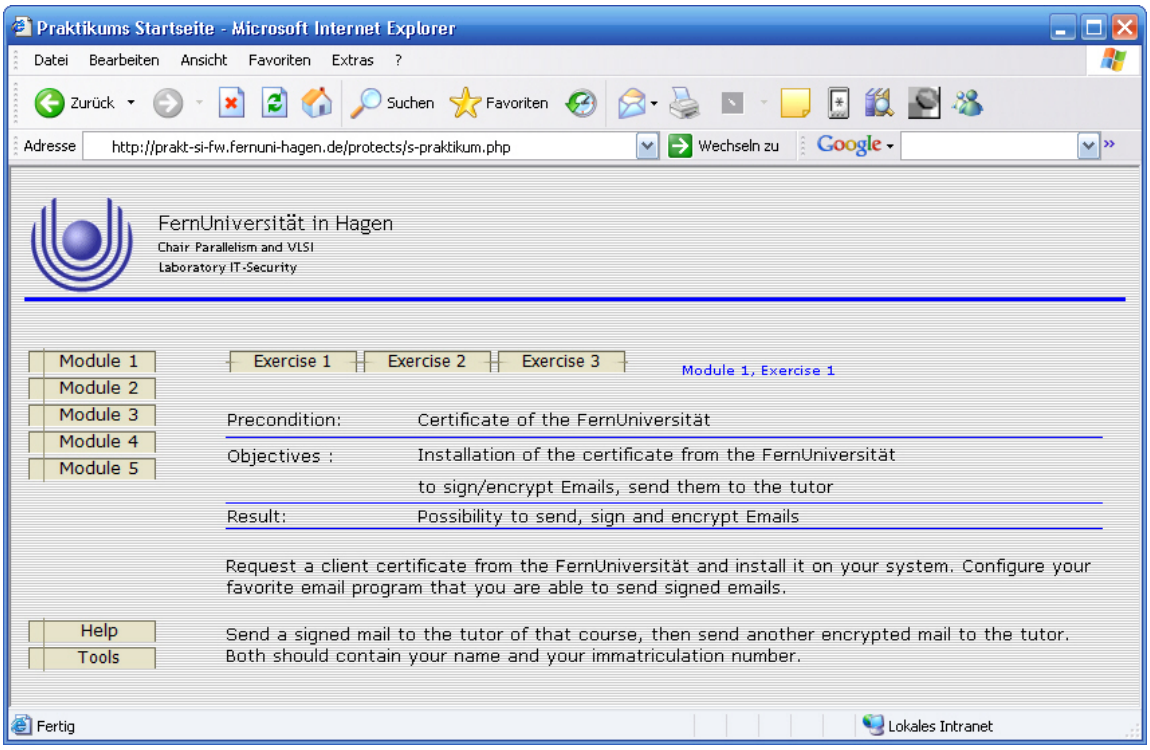

Figure 2. Student interface

vation scheme is indeed necessary. Currently, the CURE system is only used for reservation. In the future it shall also be used to implement collaborative tasks, i.e. groups of students commonly implementing and documenting a task such as setting up a virtual private network where each end of the VPN is managed under a different administrator. Also, a tighter integration (e.g. only one login) between the web-based interface for performance monitoring and CURE is intended.

The performance of servers providing virtual machines will be greatly accelerated in the very near future, as the leading processor manufacturers Intel and AMD both have announced hardware support for virtualization in their processors [6, 7]. This will allow to increase the number of concurrently running virtual machines on a given platform, thus enabling deployment in even larger courses. Some courses where lab excercises will be advantageous have enrolments of several hundreds, so that currently students have to use their own computers, which raises lots of questions because of differing operating systems and similar things.

More future work centers around the extended use of virtualization. For example, a compact disc containing a CD-bootable Linux version like Knoppix [8] and a ready-to-run virtual machine configuration could be provided to the students, so that they can use their own computers for part of the lab tasks, without the danger of changing their normal configurations. The ability to save a virtual machine state and couple it with a player to run it on another machine is now provided by VmWare [9]. The challenge here is to create a virtual network of these distributed virtual machines so that students still are provided with two machines.

References

[1] D. Hardway, M. J. Hogan, and R. G. Mathieu, Outsourcing the university computer lab, *IEEE Computer*, 38(9), 2005.

[2] J. Hu and C. Meinel, Tele-Lab IT-Security: A Means to Build Security Laboratories on the Web, *Proc. 18th International Conference on Advanced Information Networking and Applications (AINA 2004), Fukuoka (Japan)*, 2004, 285–288.

[3] M. Alexander and J.A. Lee, A Scalable Xen and Web-based Networking Course Delivery Platform, *Proc. International Conference on Education and Technology (ICET), Calgary, Canada*, 2006.

[4] J. M. Haake, A. Haake, T. Schümmer, M. Bourimi, and B. Landgraf, End-user controlled group formation and access rights management in a shared workspace system, *Proceedings of ACM CSCW04*, 2004.

[5] J. Haake and W. Schiffmann, GridLabs: Remote laboratories made accessible by an integrated collaborative web platform, submitted to *e-Science 2006*.

[6] R. Shiveley, Enhanced virtualization on Intel architecture-based servers, *Technology@Intel Magazine*, April 2005, 1–9. http://www.intel.com/technology/magazine/computing/intel-virtualization-0405.pdf

[7] Advanced Micro Devices, AMD Pacifica virtualization technology, March 2005. http://enterprise.amd.com/Downloads/Pacifica_en.pdf

[8] K. Knopper, *KNOPPIX Linux Live CD*, http://www.knoppix.org/

[9] VmWare, *Free VMware player*, http://www.vmware.com/products/player/