Practical Network Security Teaching in an Online Virtual Laboratory

Christian Willems and Christoph Meinel
Hasso-Plattner-Institute, University of Potsdam, Potsdam, Germany

Abstract— *The rapid burst of Internet usage and the corresponding growth of security risks and online attacks for the everyday user or enterprise employee have emerged the terms Awareness Creation and Information Security Culture. Nevertheless, security education has remained an academic issue mainly. Teaching system security or network security on the basis of practical experience inherits a great challenge for the teaching environment, which is traditionally solved using a computer laboratory at a university campus. The Tele-Lab project offers a system for hands-on IT security training in a remote virtual lab environment – on the web, accessible by everyone.*

An important part of security training focuses on network security: which attacks exist on the different network layers? What is the impact of those attacks? And, how can we secure a network through proper configuration or protective measures like firewalls?

The paper at hand briefly presents usage, management and operation of Tele-Lab as well as its architecture. Furthermore, this work introduces the integration of the Virtual Distributed Ethernet technology (VDE) into the Tele-Lab Server and the realization of learning units on network security with complex exercise scenarios such as eavesdropping on local network traffic or Man-in-the-Middle attacks by means of ARP spoofing.

Keywords: Web-based Training, Security Education, Virtual Laboratory, Virtual Machines

1. Introduction

Increasing propagation of complex IT systems and rapid growth of the Internet draws attention to the importance of IT security issues. Technical security solutions cannot completely overcome the lacking awareness of computer users, caused by laziness, inattentiveness, and missing education. In the context of awareness creation, IT security training has become a topic of strong interest – as well as for educational institutions as for companies or even individual Internet users.

Traditional techniques of teaching (i.e. lectures or literature) have turned out to be not suitable for security training, because the trainee cannot apply the principles from the academic approach to a realistic environment within the class. In security training, gaining practical experience through exercises is indispensable for consolidating the knowledge. Precisely the allocation of an environment for these practical exercises poses a challenge for research and development. That is, since students need privileged access rights (root/administrator-account) on the training system to perform most of the imaginable security exercises. With these privileges, students might easily destroy a training system or even use it for unintended, illegal attacks on other hosts within the campus network or the Internet world.

The classical approach is to provide a dedicated computer lab for security training. Such labs are exposed to a number of drawbacks: they are immobile, expensive to purchase and maintain, and must be isolated from all other networks on the site. Of course, students are not allowed to have Internet access on the lab computers. Hands-on exercises on network security topics even demand to provide more than one machine to each student, which have to be interconnected (i.e. a Man-in-the-Middle attack needs three computers: one for the attacker and two other machines as victims).

Tele-teaching for security education consists of multimedia courseware or demonstration software mostly, which does not offer practical exercises. In simulation systems users have a kind of hands-on experience, but a simulator doesn't behave like a realistic environment and the simulation of complex systems is very difficult – especially when it comes to interacting hosts on a network. The Tele-Lab project builds on a different approach for a Web-based teleteaching system (explained in detail in section 2).

The enhanced Tele-Lab architecture proposed in this paper makes this teleteaching platform even more equivalent to a physical dedicated computer security lab: integration of a virtual networking solution described in section 3 allows to provide training environments for complex exercise scenarios in a dynamic and flexible manner.

Section 4 introduces two learning units on network security – an eavesdropping scenario and the practical application of a Man-in-the-Middle attack – that show the feasibility of this solution. Section 5 summarizes and gives an outlook on future enhancements to the Tele-Lab platform as well as on additional use cases.

2. Tele-Lab: A Remote Virtual Security Laboratory

The Tele-Lab platform (accessible at `http://www.tele-lab.org/`, see Figure 1) was initially proposed as a

Christian Willems, Christoph Meinel:
"Practical Network Security Teaching in an Online Virtual Laboratory"
in Proceedings of the 2011 International Conference on Security & Management (SAM 2011),
CSREA Press, Las Vegas, Nevada, USA, 7, 2011, ISBN: 1-60132-198-8.

standalone system [1], later enhanced to a live DVD system introducing virtual machines for the hands-on training [4], and then emerged to the Tele-Lab server [3], [6]. The Tele-Lab server provides a novel e-learning system for practical security training in the WWW and inherits all positive characteristics from offline security labs. It basically consists of a web-based tutoring system and a training environment built of virtual machines. The tutoring system presents learning units that do not only offer information in form of text or multimedia, but also practical exercises. Students perform those exercises on virtual machines (VM) on the server, which they operate via remote desktop access. A virtual machine is a software system that provides a runtime environment for operating systems. Such software-emulated computer systems allow easy deployment and recovery in case of failure. Tele-Lab uses this feature to revert the virtual machines to the original state after each usage.

With the release of the current iteration of Tele-Lab, the platform introduced the dynamic assignment of several virtual machines to a single user at the same time. Those machines are connected within a virtual network (known as *team*, see also in [2]) providing the possibility to perform basic network attacks such as interaction with a virtual victim (i.e. port scanning). A victim is the combination of a suitably configured virtual machine running all needed services and applications and a collection of scripts that simulate user behavior or react to the attacker's actions (see also exemplary description of a learning unit below). A short overview of the architecture of the Tele-Lab platform is given later in this section.

2.1 Learning Units in Tele-Lab – an exemplary walkthrough

Learning units follow a straight-forward didactic path beginning with general information on a security issue, getting more concrete with the description of useful security tools (also for attacking and exploiting) and culminating in a hands-on exercise, where the student has to apply the learned concepts in practice. Every section concludes with hints on how to prevent the just conducted attacks.

An exemplary Tele-Lab learning unit on *malware* (described in more detail in [5]) starts off with academic knowledge such as definition, classification, and history of malware (worms, viruses, and Trojan horses). Methods to avoid becoming a victim and relevant software solutions against malware (scanners, firewalls) are presented as well. Afterwards, various existing malware kits and ways of distribution are described in order to prepare the hands-on exercise. Following an offensive teaching approach (see [7] for different teaching approaches), the user is asked to take the attacker's perspective – and hence is able to lively experience possible threats to his personal security objectives. The closing exercise for this learning unit on malware is to *plant a Trojan horse on a scripted victim's computer system* – in particular it is the outdated Back Orifice Trojan horse.

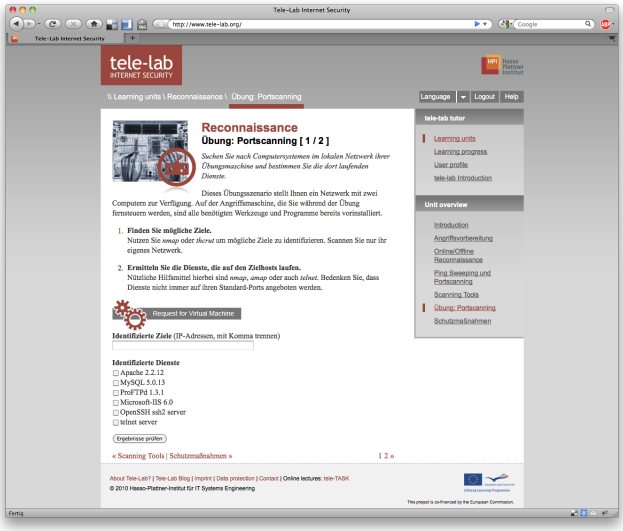

Fig. 1: Screenshot of the Tele-Lab Tutoring Interface

Back Orifice (BO) is a Remote Access Trojan Horse developed by the hacker group "Cult of the Dead Cow" (see [9]). In order to distribute the Trojan horse to the attacker, the student has to prepare a carrier for the BO server component and send it to the victim via e-mail. A carrier is usually a "gimmick" application that has actually no useful functionality but installs the Trojan horse server in the background. The script on the victim's virtual machine will answer the mail and indicate that the Trojan horse server has been installed (mail attachment has been opened).

The next step is the application of knowledge gained in a prior learning unit on *Reconnaissance*: in order to find the now vulnerable virtual machine, the network must be scanned for hosts that offer a service on the port used for the Back Orifice server. This can be done using a port scanner like the well-known *nmap* tool. The student can now use the BO client to take control of the victim's system and spy out some private information. The knowledge of that information is the user's proof to the Tele-Lab tutoring environment, that the exercise has been solved successfully.

Such an exercise implies the need for the Tele-Lab user to be provided with a team of interconnected virtual machines: one for attacking (all necessary tools installed), a mail server for e-mail exchange with the victim and a vulnerable victim system (unpatched Windows 95/98 in this case). Remote Desktop Access is only possible to the attackers VM.

Other learning units are also available on, e.g., authentication, wireless networks, secure e-mail, reconnaissance, firewalls, etc. The system can easily be enhanced with new content.

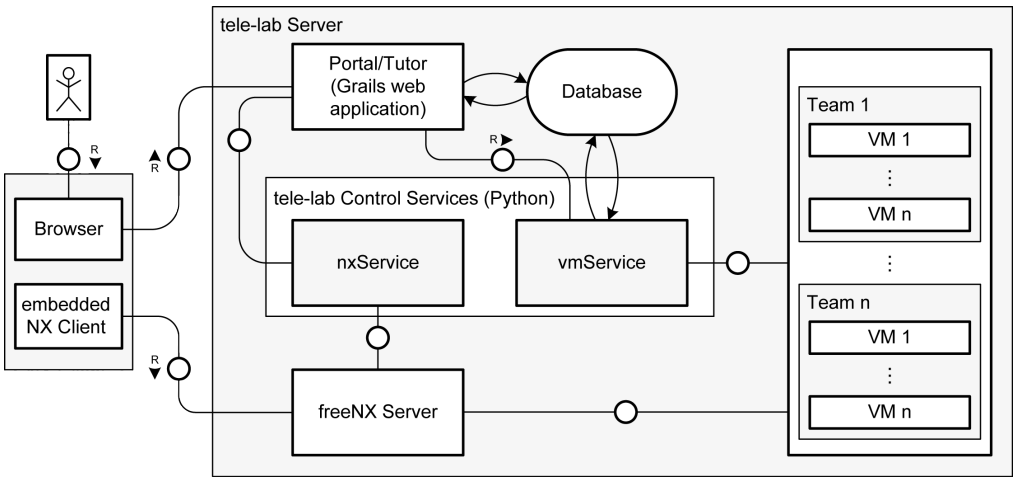

Fig. 2: Overview – Architecture of the Tele-Lab Platform

2.2 Architecture of the Tele-Lab Platform

The current architecture of the Tele-Lab server is a refactored enhancement to the infrastructure presented in [6]. Basically it consists of the following components (illustrated in Figure 2).

Portal and Tutoring Environment: The Web-based training system of Tele-Lab is a custom *Grails application* running in a *Tomcat application server*. This web application handles user authentication, allows navigation through learning units, delivers their content and keeps track of the students' progress. It also provides controls to request a team of virtual machines for performing an exercise.

Virtual Machine Pool: The server is charged with a set of different virtual machines which are needed for the exercise scenarios – the pool. The resources of the physical server limit the maximum total number of VMs in the pool. In practice, a few (3-5) machines of every kind are started up. If all teams for a certain exercise scenario are in use, new instances can be launched dynamically (again depending on the current load of the physical host). Those machines are dynamically connected to teams and bound to a user on request. The current hypervisor solution used for providing the virtual machines is *KVM/Qemu* [10], [11]. The *libvirt* package [16] is used as a wrapper for the virtual machine control. *LVM (Linux Logical Volume Management)* provides virtual hard discs that are capable of copy-on-write-like differential storage. Differential storage is important to save space on the physical hard disc, because the Tele-Lab server holds so called *VM templates* as master images and clones multiple instances of each template for use within the exercise environment. VM templates also contain configuration files defining hardware parameters like memory, number of CPUs, and network interfaces.

Database: The Tele-Lab database holds all user information, the content for web-based training and learning unit structure as well as the information on virtual machine and team templates. *Team templates* are models for connected VMs that allow performing specific exercise scenarios. The database also persists current virtual machine states.

Remote Desktop Access Proxy: The Tele-Lab server must handle concurrent remote desktop connections for different users performing exercises. Those connections are proxied using a free implementation of the NX server (*freeNX*, see [12]). The NX server forwards incoming connections to the respective assigned virtual machine accessing the Qemu framebuffer device via *VNC (Virtual Network Computing)*. The NX Client software launched from the student's browser connects to the NX Server using SSH-based authentication: client and server mutually certify each others identity using public-key authentication. Subsequently, the NX Client connects to a specific session with extra user credentials. For mandatory encryption of the remote sessions, NX offers *transport layer security (TLS)*.

Administration Interface: The Tele-Lab server comes with a sophisticated web-based administration interface that is also implemented as Grails application (not depicted in Figure 2). The main functionality of this interface is content management for the web-based training environment and user management for the whole platform. Additionally, the admin interface can be used for manual virtual machine control, monitoring and for registering new virtual machines or team templates.

Tele-Lab Control Services: Purpose of the central Tele-Lab control services is bringing all the above components together. To realize an abstraction layer for encapsulation of the virtual machine monitor (or hypervisor) and the remote desktop proxy, the system implements a number of *lightweight XML-RPC web services:* the *vmService* and the *remoteDesktopService*. The vmService is to control virtual machines – start, stop or recover them, grouping teams or as-

signing machines or teams to a user. The remoteDesktopService is used to initialize, start, monitor, and terminate remote desktop connections to machines, which are assigned to students when they perform exercises. The above-mentioned Grails applications (portal, tutoring environment, and web admin) let the user and administrators control the whole system using the web services.

On the *client side*, the user needs a web browser supporting SSL/TLS and the appropriate Java-plugin for the browser only. For the remote desktop connections, the *NX WebCompanion* is included in the tutoring web application. The WebCompanion is a launcher application for the NX Client implemented as Java applet.

3. Virtual Networking for Tele-Lab

As already stated, many scenarios for exercises in security training demand for a networked environment. Exercises on single host training systems are limited to very few tasks that could possibly also be performed on a physical local system without any harm. More interesting and complex exercises (like the malware learning unit described in section 2) and especially exercises on network security as introduced later in section 4 cannot be performed without a training environment providing machines that are connected within a local network.

Earlier implementations of Tele-Lab could connect virtual machines combined to a team using multicast groups: each team is provided with an individual multicast socket that is connected to each team member's virtual network. Routing, firewall, and virtual network devices on the physical host are dynamically configured to separate the network segments from each other. Each multicast group (VM team) can communicate internally only.

To understand this idea, we have to explain the networking concept of Qemu in detail: the virtualization suite sets up a *VLAN (virtual LAN)* for each Qemu process. Those VLANs can be understood as virtual hubs, where you can attach virtual network interfaces – such as the one of the virtual machine running in that process. All attached interfaces to a VLAN intercept all packages sent via that virtual hub. To connect the VLANs of a team of virtual machines, Tele-Lab connects a multicast socket to the virtual LAN of each machine belonging to the respective team, when it starts up. This technique for setting up a virtual network in a Tele-Lab team limits the resulting virtual Ethernet-based networks to:

a) LAN segments with a hub (no switched networks)
b) simple network structures: no routing, no internetworking (interconnection of networks)
c) static IP addresses for the VM templates: this limits the reusability of VM templates, i.e. if one wants to have more than one instance of the same virtual machine in one exercise scenario (respectively team template)

Since the paradigm for Tele-Lab is to provide a training environment being as realistic as possible, the integration of software-emulated networking devices to overcome the above limitations is a highly desirable enhancement.

3.1 Virtual Distributed Ethernet (VDE)

A suitable solution for more sophisticated networking within the VM teams in Tele-Lab exists with the *Virtual Distributed Ethernet (VDE)* project [8]. VDE is a system which consistently emulates all aspects of Ethernet networking on the data-link layer in a completely realistic manner. VDE maps hardware devices from the physical world – like switches, plugs and cables – on software running in user-mode. The main components of a VDE installation are:

VDE switch – a highly customizable software emulation of an Ethernet switch. It supports VLANs, different operation modes (switch/hub), cascading several VDE switches (including Spanning Tree Protocol), and extensive command line management. You can attach different kinds of network interfaces, such as TUN/TAP interfaces, QEMU/KVM-based virtual machines, and VDE plugs. *TUN* and *TAP* are virtual network devices provided by the Linux kernel. While TAP (as in network tap) simulates an Ethernet device and operates on ISO/OSI layer 2, TUN (as in network TUNnel) simulates a network layer device and operates with layer 3 packets (i.e. IP packets).

VDE plug – the virtual counterpart of an Ethernet plug can be connected to a VDE switch. It sends all data from the standard input to the VDE switch which is connected to and writes all data from the virtual switch to standard output. A tool named *dpipe* (a bi-directional pipe) can connect two VDE plugs to a virtual cable by diverting the standard output of one VDE plug to the standard input of the other one (and vice-versa). *wirefilter* is an enhanced version of dpipe, which also allows for simulating problems and limitations from the physical world like packet loss, duplicated packets, limited bandwidth or different MTUs.

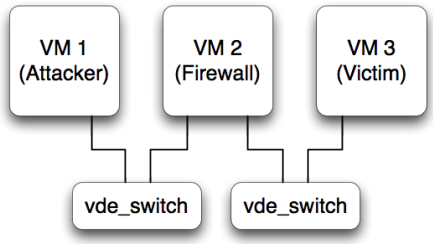

Fig. 3: Examplary Deployment for Virtual Distributed Ethernet in Tele-Lab

A possible VDE setup with virtual machines for a complex Tele-Lab learning unit may look like illustrated in Figure 3: let the task be a remote exploitation of VM 3, the attacker uses VM 1. While this would be an easy task if attacker and victim would be connected to the same local network, it gets much more challenging as soon as there is a firewall

between the attacker and target host. The use of two VDE switches, both connected to different network interfaces of the firewall host (VM 2) allows to compile such an exercise scenario.

VDE switches and VDE plugs can also be connected if they run on different physical hosts, which is also a useful feature for a further enhanced Tele-Lab architecture (see Outlook in section 5).

3.2 Integrating VDE into the Tele-Lab Architecture

When Tele-Lab creates a new team of virtual machines, the vmService (see Figure 2) is responsible for starting Qemu processes for each VM and for setting up the virtual network that connects the team members. It consumes a team configuration provided as XML file and transforms its elements to parameters for command line calls. Such an XML file for the example configuration from Figure 3 would look like depicted in Figure 4 (without attributes not relevant for virtual networking):

```
<tl:team name="Example Team" >
   <!-- virtual machine instances -->
   <tl:machine name="VM 1 (Attacker)">
      <tl:networkInterface
         mac="00:11:22:33:44:55"
         networkName="net0" />
   </tl:machine>
   <tl:machine name="VM 2 (Firewall)">
      <tl:networkInterface
         mac="11:22:33:44:55:66"
         networkName="net0" />
      <tl:networkInterface
         mac="22:33:44:55:66:77"
         networkName="net1" />
   </tl:machine>

   <tl:machine name="VM 3 (Victim)">
      <tl:networkInterface
         mac="33:44:55:66:77:88"
         networkName="net1" />
   </tl:machine>

   <!-- virtual network -->
   <tl:network name="net0" id="1"
      mode="switch" />
   <tl:network name="net1" id="2"
      mode="switch" />
</tl:team>
```

Fig. 4: Examplary XML Team Configuration

After parsing the XML data, the *vmService* starts up virtual machines from the VM templates identified by the `<tl:machine>` element and initiates the respective network interfaces specified with the enclosed `<tl:networkInterface>` items, i.e. two interfaces for the firewall machine (VM 2).

It also starts an instance of *VDE Switch* for each virtual network specified with `<tl:network>`, either as hub or as switch depending on the mode value. The network interfaces of the virtual machines are bound to the matching switch instances.

The assignment of IP addresses inside the virtual machines posed a challenge during implementation, since they had to be allocated dynamically. An obvious solution was to attach a *DHCP server* to each VDE switch after startup using TAP interfaces. This DHCP server assigns an IP address to each of the virtual machines connected to the virtual switch based on its MAC address. IP addresses for the virtual machines can also be defined in the team configuration file. If an administrator decides to do so, the DHCP server for the respective team is dynamically configured to issue those defined IP addresses to the network interface with the corresponding MAC address.

Due to security constraints, users of virtual machines in Tele-Lab should not be able to access any services running on the physical host. For this reason, the DHCP server and the TAP interface are shut down, after the DHCP leases have been issued.

The generation of the above described XML representations of virtual networks will be realized as a web based tool: Tele-Lab administrators can use a convenient interface to combine virtual machine templates to a team and define the network connections for the team members.

4. Network Security Exercise Scenarios

There are a lot of conceivable exercise scenarios in the area of network security, which require the provision of a networked training environment. Two such exercises have already been introduced earlier in this paper: the *malware* learning unit from section 2 needs three hosts on a network (attacker and victim machines, mail server). The exemplary scenario on *remote exploitation* outlined in section 3 requires three hosts on two different networks. In the following, two more learning units on network security are presented briefly.

4.1 Exercise Scenario: Eavesdropping of Network Traffic

Eavesdropping is basically about secret listening to some private communication of two (or more) communication partners without their consent. In the domain of computer networks, the common technique for eavesdropping is *packet sniffing*. There are a number of tools for packet sniffing – *packet analyzers* – freely available on the Internet, such as the well-known *tcpdump* or *Wireshark* [13] (used in this learning unit).

A learning unit on packet sniffing in a local network starts off with an introduction to communication on the data-link layer (Ethernet) and explains the difference between a network with hub and a switched environment. This is important for eavesdropping, because this kind of attack is

way easier when connected to a hub. The hub will forward every packet coming in to all its ports and hence to all connected computers. These hosts decide, if they accept and further compute the incoming data based on the MAC address put in the destination field of the Ethernet frame header: if the destination MAC is the own MAC address, the Ethernet frame is accepted, or dropped otherwise. If there is a packet analyzer running, also frames not intended for the respective host can be captured, stored and analyzed. This situation is different in a switched network: the switch does not broadcast incoming data to all ports but interprets the MAC destination to "switch" a dedicated line between source and destination ports. In consequence, the Ethernet frame is only delivered to the actual receiver.

After providing general information on Ethernet-based networking, the learning unit introduces the idea of packet sniffing and describes capabilities and usage of the packet analyzer *Wireshark*, especially how to capture data from the Ethernet device and how to filter and read the captured data.

The practical exercise presents the following task to the learner: *"Sniff and analyze network traffic on the local network. Identify login credentials and use them to obtain a private document."* The student is challenged to enter the content of this private document to proof, that she has solved the task.

When requesting access to a training environment, the user is assigned to a team of three virtual machines: the attacker machine equipped with the Wireshark tool, and two machines of (scripted) communication partners: Alice and Bob. In this scenario, Bob's machine hosts an FTP server and a Web server, while Alice's VM runs a script that generates traffic by initiating arbitrary connections to the services on Bob's host. Among those client/server connections are successful logins to Bob's FTP server. As this learning unit focuses on sniffing and the interpretation of the captured traffic, the machines are connected with a hub. There is no need for the attacker to get into a Man-in-the-Middle position in order to capture the traffic between Alice and Bob.

Since FTP does not encrypt credentials, the student can obtain username and password to log in to that service using the stolen credentials. On the server, the student finds a file called *private.txt* that contains the response to the challenge mentioned above.

The lesson concludes with hints on preventing eavesdropping attacks, such as the usage of services with secure authentication methods (i.e. SFTP or ftps instead of plain FTP) and data encryption.

4.2 Exercise Scenario: Man-in-the-Middle Attack with ARP Spoofing

The general idea of a Man-in-the-Middle attack (MITM) is to intercept communication between two communication partners (Alice and Bob) by initiating connections between the attacker and both victims and spoofing the identity of the respective communication partner (Fig. 5). More specific, the attacker pretends to be Bob and opens a connection to Alice (and vice versa). All traffic between Alice and Bob is being relayed via the attackers computer. While relaying, the messages can be captured and/or manipulated.

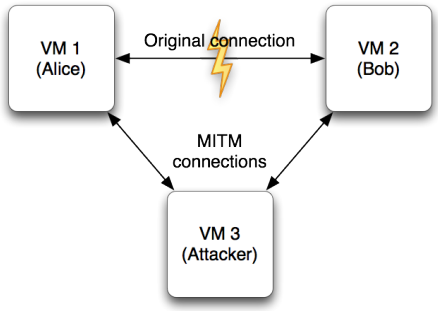

Fig. 5: General Idea of Man-in-the-Middle Attacks

MITM attacks can be implemented on different layers of the TCP/IP network stack, i.e. *DNS cache poisoning* on the application layer, *ICMP redirecting* on the Internet layer or *ARP spoofing* in the data-link layer. This learning unit focuses on the last-mentioned attack, which is also called *ARP cache poisoning*.

The Address Resolution Protocol (ARP) is responsible for resolving IP addresses to MAC addresses in a local network. When Alice's computer opens an IP-based connection to Bob's one in the local network, it has to determine Bob's MAC address at first, since all messages in the LAN are transmitted via the Ethernet protocol (which only knows about the MAC addresses). If the Alice only knows the IP address of Bob's host, (i.e. 192.168.0.10) she performs an *ARP request:* Alice sends a broadcast message to the local network and asks, *"Who has the IP address 192.168.0.10?"* Bob's computer answers with an *ARP reply* that contains its IP address and the corresponding MAC address. Alice stores that address mapping in her *ARP cache* for further communication.

ARP spoofing [14] is basically about sending forged ARP replies: referring to above example, the attacker repeatedly sends ARP replies to Alice with Bob's IP address and the own MAC address – the attacker pretends to be Bob. When Alice starts to communicate with Bob, she sends the ARP request and instantly receives one of the forged ARP replies from the attacker. She then thinks, the attackers MAC address belongs to Bob and stores the faked mapping in her ARP cache. Since the attacker performs the same operation for Alice's MAC address, he can also manage to imply Bob, that his MAC address is the one of Alice. In consequence, Alice sends all messages to Bob to the MAC address of the attacker (same for Bob's messages to Alice). The attacker just has to store the original MAC addresses of Alice and

Bob to be able to relay to the original receiver.

A learning unit on ARP spoofing begins with general information on communication in a local network. It explains the Internet Protocol (IP), ARP and Ethernet including the relationship between the two addressing schemes (IP and MAC addresses).

Subsequently, the above attack is described in detail and a tool, that implements ARP spoofing and a number of additional MITM attacks is presented: *Ettercap* [15]. At this point, the learning unit also explains what the attacker can do, if he becomes Man-in-the-Middle successfully, such as specifying *Ettercap filters* to manipulate the message stream.

The hands-on exercise of this chapter asks the student to perform two different tasks. The first one is the same as described in the exercise on packet sniffing above: *"monitor the network traffic, gain FTP credentials and steal a private file from Bob's FTP server"*. The training environment is also set up similar to the prior scenario. The difference is that the team of three virtual machines is connected through a virtual switch this time (instead of a hub), so that capturing the traffic with Wireshark would not reveal the messages between Alice and Bob. Again, the student has to proof the successful attack by putting in the content of the secret file in the tutoring interface.

The second (optional) task is to apply a filter on the traffic and replace all images in transmitted HTML content by an image from the attackers host (which would be displayed in Alice's browser). This attack is still working and dangerous in many currently deployed local network installations. The only way to protect oneself against ARP spoofing would be the usage of SSL with a careful verification of the hosts certificate, which is explained in conclusion of the learning unit.

A future enhancement of the practical exercise on ARP spoofing would be the interception of an SSL secured channel: Ettercap also allows a more sophisticated MITM attack including the on-the-fly generation of faked SSL certificates, which are presented to the victims instead of the original ones. The Man-in-the-Middle can then decrypt and re-encrypt the SSL traffic when relaying the messages

5. Conclusions and Outlook

The paper at hand presents a comprehensive infrastructure for a remote virtual computing lab for security education. The described enhancements with the Virtual Distributed Ethernet software suite allows the implementation of training environments for complex network security exercises, such as the learning units on packet sniffing and ARP spoofing.

Future work on the system includes the creation of more learning units in the network security domain as well as the implementation of technical enhancements. Additional learning units may cover topics like other Man-in-the-Middle attacks (i.e. the above mentioned DNS cache poisoning), firewall configuration, intrusion detection and prevention, etc.

Technical enhancements planned for the next iterations of the Tele-Lab server are

- integrating a convenient administration interface for the creation of team templates, precisely a graphical editor for virtual networks, where you can drag and drop virtual machine templates, switches and network cables,
- switching the Remote Desktop Access from NX to an HTML5/AJAX based VNC client (i.e. *noVNC*, see `http://kanaka.github.com/noVNC/`),
- the implementation of tools for remote collaborative learning and tutoring (e.g. Remote Desktop Assistance),
- and clustering on application level to provide larger virtual machine pools.

The clustering enhancement will allow users of interconnected Tele-Lab servers to use virtual machines running on other physical hosts than the one known to the user. The integration of VDE even allows having the virtual machines of one team running on different physical machines.

References

[1] J. Hu, M. Schmitt, C. Willems, and C. Meinel. "A tutoring system for IT-Security", in *Proceedings of the 3rd World Conference in Information Security Education*, p. 51–60, Monterey, USA, 2003.

[2] C. Border. "The development and deployment of a multi-user, remote access virtualization system for networking, security, and system administration classes", *SIGCSE Bulletin*, 39(1): p. 576–580, 2007.

[3] J. Hu, D. Cordel, and C. Meinel. "A Virtual Machine Architecture for Creating IT-Security Laboratories", Technical report, Hasso-Plattner-Insitut, 2006.

[4] J. Hu and C. Meinel. "Tele-Lab IT-Security on CD: Portable, reliable and safe IT security training", *Computers & Security*, 23:282–289, 2004.

[5] C. Willems and C. Meinel. "Awareness Creation mit Tele-Lab IT-Security: Praktisches Sicherheitstraining im virtuellen Labor am Beispiel Trojanischer Pferde", in *Proceedings of Sicherheit 2008*, p. 513–532, Saarbruecken, Germany, 2008.

[6] C. Willems and C. Meinel. "Tele-Lab IT-Security: an Architecture for an online virtual IT Security Lab", *International Journal of Online Engineering (iJOE)*, X, 2008.

[7] W. Yurcik and D. Doss. "Different approaches in the teaching of information systems security", in *Security, Proceedings of the Information Systems Education Conference*, p. 32–33, 2001.

[8] R.Davoli. (2011) Virtual Distributed Ethernet homepage. [Online]. Available: http://vde.sourceforge.net/

[9] Cult of the Dead Cow. (2011) Back Orifice – Windows Remote Administration Tool homepage. [Online]. Available: http://www.cultdeadcow.com/tools/bo.php

[10] Red Hat, Inc. (2011) Kernel-based Virtual Machine (KVM) homepage. [Online]. Available: http://www.linux-kvm.org/

[11] F. Bellard. (2011) QEMU – Open Source Processor Emulater homepage. [Online]. Available: http://www.qemu.org/

[12] F. Franz. (2011) FreeNX – the free NX project homepage. [Online]. Available: http://freenx.berlios.de/

[13] Wireshark Foundation. (2011) Wireshark homepage. [Online]. Available: http://www.wireshark.org/

[14] S. Whalen. (2011) An Introduction to ARP Spoofing. [Online]. Available: http://www.rootsecure.net/content/downloads/pdf/arp_spoofing_intro.pdf

[15] A. Ornaghi and M. Valleri. (2011) EttercapNG homepage. [Online]. Available: http://ettercap.sourceforge.net/

[16] The Libvirt Developers. (2011) libvirt – The virtualization API homepage. [Online]. Available: http://libvirt.org/