Protecting Tele-Lab – attack vectors and countermeasures for a remote virtual IT security lab

Christian Willems, Wesam Dawoud, Thomas Klingbeil, and Christoph Meinel
Hasso Plattner Institute
Potsdam, Germany

Abstract

The rapid burst of Internet usage and the corresponding growth of security risks and online attacks for the everyday user or the enterprise employee have emerged the terms Awareness Creation *and* Information Security Culture. *Nevertheless, security education widely has remained an academic issue. Teaching system 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 within a remote virtual lab environment – over the web, accessible by everyone.*

Such a system is inherently exposed to various security threats, since it has to provide full access to virtual machines running attack tools for potentially malicious users. The paper at hand introduces usage, management and operation of Tele-Lab as well as its architecture. Furthermore, this work focuses on possible attacks, the challenges when securing such a system, and shows how to set up an infrastructure that ensures the main security objectives identified as authentication, authorisation and availability.

1. Introduction

The increasing propagation of complex IT systems and rapid growth of the Internet more and more attracts notice 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 companies as for individuals.

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, because students need privileged access rights (root/administrator-account) on the training system to perform most of the imaginable security exercises. With these privileges, students can easily destroy a training system or even use it for unintended, illegal attacks (see section 2.2).

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 have Internet access on the lab computers. Teleteaching approaches for security education mostly consist of multimedia courseware or demonstration software, which do 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. Those approaches to security education will be presented more precisely in section 3.

2. Tele-Lab: a remote virtual security lab

The Tele-Lab[1] project was first proposed as a standalone system [6], later enhanced to a live DVD system introducing virtual machines for the hands-on training [5], then emerged to the Tele-Lab server [4, 13]. 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 ma-

[1] http://www.tele-lab.org

chines. Students can perform practical exercises on those virtual machines (VM) on the server, which they operate via remote desktop access (see Figure 1). 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 Tele-Lab 2.0, the system introduces 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*[2]) providing the possibility to perform complex network attacks such as man-in-the-middle or interaction with a virtual (scripted) victim (see example in section 2.1).

A short overview of the Tele-Lab architecture is given in section 4.

2.1. Walkthrough: A learning unit in Tele-Lab

As Tele-Lab is a web-based e-learning portal, a student has to log on to access his set of available learning units. Users can track their learning progress or continue from the last session. All learning units consist of information sections, where concepts and academic knowledge is presented. The information parts are not just pure textual content but also embedded multimedia clips, e.g. lecture recordings from the tele-TASK[3] archive as user friendly podcast clips [14]. Usually, these sections are closed with a quiz-like exercise. More concrete information has been collected for the tool-sections: here, the student is introduced to relevant operating system components, defensive software as well as tools for auditing and attacking. Each learning unit concludes with a practical hands-on experience.

For example, a Tele-Lab learning unit on *malware* (described in more detail in [12]) starts off with definition, classification, and history of malware (worms, viruses, and trojan horses). Also methods to avoid becoming a victim and relevant software solutions against malware (scanners, firewalls) are introduced. Afterwards, various existing malware kits and ways for distribution are presented in order to prepare the hands-on exercise.

Following an offensive teaching approach[4], the user is asked to take the attackers perspective – and hence is able to lively experience possible threats to his personal security objectives. So the closing exercise for this learning unit on malware is to plant a trojan horse on a virtual victim called Alice. This virtual victim is actually a script running on a virtual machine connected to the same virtual network as the students machine. In particular the trojan horse in this exercise scenario is the outdated Back Orifice[5].

To start the hands-on experience, the student uses a HTML control in the browser to request a virtual machine for exercising. When clicking the link, the Tele-Lab server checks its virtual machine pool for an appropriate virtual machine and assigns it to the user. The attacker's machine pops up as a graphical remote desktop window.

On his machine, the student will find all necessary items (trojan horse itself, fake file, preconfigured e-mail client) to prepare a carrier for the Back Orifice server component and send it to Alice via e-mail. The script on the victim VM periodically checks the e-mail account for the mail and answers it to indicate that the trojan horse server has been installed (mail attachment has been opened). From the mail headers, the student can determine the victims IP address and then use the Back Orifice client to take control of the victim's system and spy out some private information stored on Alice's hard disk. The knowledge of that information is the user's proof to the Tele-Lab tutoring environment, that the exercise has been solved successfully. The student can now close the remote desktop window, enter the gained secret into the tutoring frontend and finish the learning unit.

Such an exercise implies the need for the Tele-Lab user to be provided with a team of three 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 2000 in this case). Remote Desktop Access is only possible to the attacker's VM.

Learning units are also available on e. g. reconnaissance and port scanning, authentication and password security, wireless networks, secure e-mail, encryption and digital signatures, attacks on browsers and websites, buffer overflows and remote exploiting, etc. The system can easily be enhanced with new content.

2.2. Security objectives for Tele-Lab

As already stated, the necessity to give Tele-Lab users superuser privileges on the virtual machines in the training environment requires a concept to prevent unintended usage. Moreover, the fact that users are deliberately equipped with hacker tools (port scanners, malware, tools for perfoming man-in-the-middle or DoS attacks) even reinforces the need for strong isolation of the virtual lab environment – while the Tele-Lab server still must provide remote desktop access. The system must ensure several, diverse security objectives

[2] see also in [1]

[3] Tele Teaching Anywhere Solution Kit `http://www.tele-task.de`

[4] see [15] for different teaching approaches

[5] Back Orifice (BO) is a Remote Access Trojan Horse run by the hacker group "Cult of the Dead Cow", see `http://www.cultdeadcow.com/tools/bo.php`

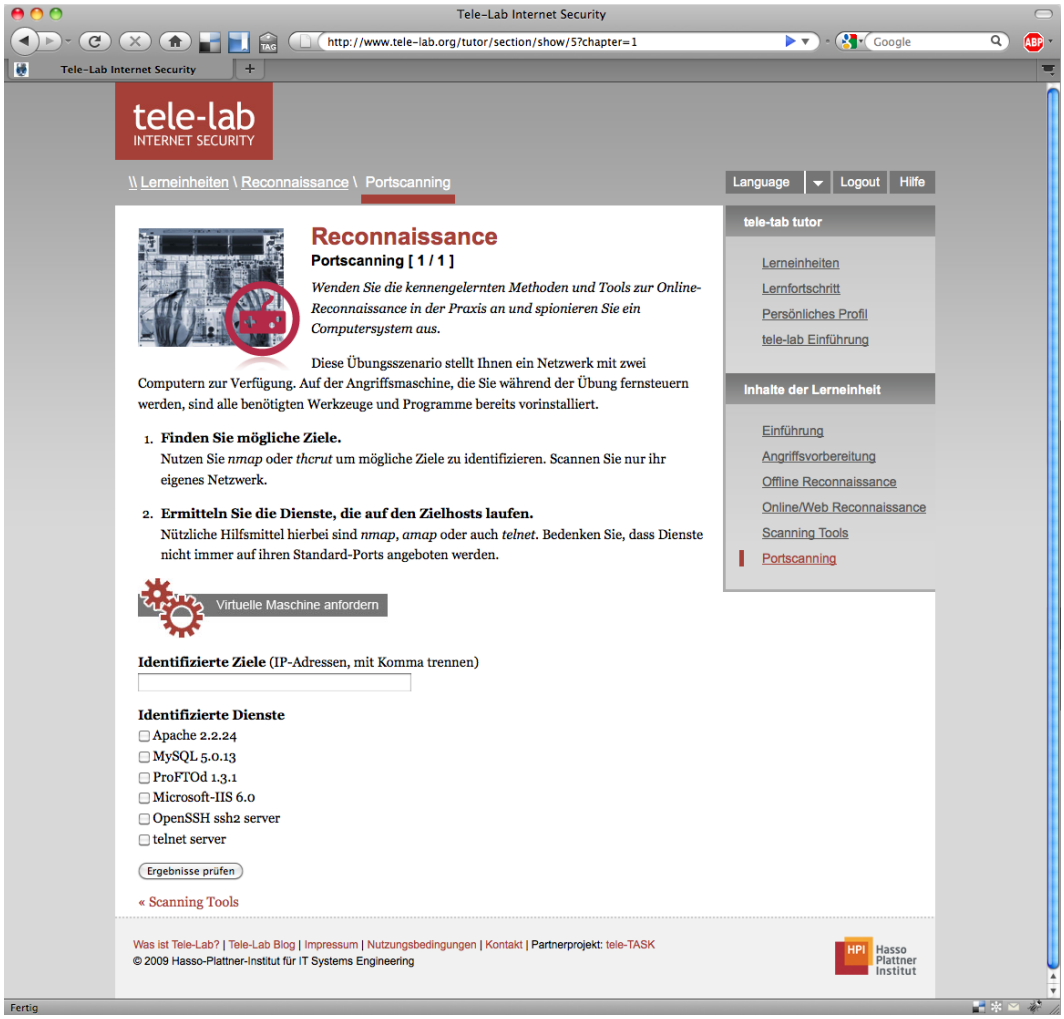

Figure 1. Screenshot: a hands-on experience section in the Tele-Lab tutoring interface – exercise assignments and solution form

- protection of the Tele-Lab host system against attacks from malicious VM users,
- protection of the campus network and arbitrary Internet hosts against attacks from VM users,
- separation of the VM teams (no user may attack another user's VM),
- and of course the protection of the Tele-Lab host against attacks from the Internet.

An additional threat for the overall system are the above mentioned victim machines. The victim VMs must be explicitly vulnerable in most cases to allow the respective attack or attack sequence that shall be exercised. Those vulnerable systems must be inaccessible from the outside (Internet) world in any case. If this can't be guaranteed, the machines could easily be hijacked and hence used for either attacking other Internet users or e.g. for providing a site for illegal downloads – because of the vulnerability.

These objectives do not only apply for Tele-Lab but for all remote laboratories offering privileged access for the students.

Section 4 explains security relevant components of the Tele-Lab infrastructure. In-depth considerations on certain possible attacks and solutions implemented in Tele-Lab 2.0 are stated in section 5.

3. Related Work

Related work in security education mainly includes web-based training and multimedia courseware, demonstration software, simulation systems, and dedicated computer lab-

oratories for security experiments. Recently, efforts in compiling practical computer science courses using virtual machine technology share basic technologies and concepts with Tele-Lab. Also similar efforts on virtual and/or remote labs will be considered. This section will focus on the security aspects of the related work.

3.1. Web-based training and e-Lecturing

Web-based training (WBT) classifies e-learning applications that present learning units via WWW in the user's browser. Those learning units usually consist of text, images and custom made animations. Extended to multimedia courseware, WBT also offers real multimedia content, i.e. audio and video. WBT courses usually are digitized lectures and demonstrations. A modern e-lecturing system is e.g. tele-TASK [10]. It is a state-of-the-art streaming system that allows to create online lectures and seminars for teaching e.g. IT security. WBT and e-lectures in nature support e-learning on the web, but they offer no hands-on experiences to learners at all. WBT applications are exposed to all attacks that apply to web applications in general – such as SQL injection, XSS, Request Forgery, Account Stealing, Session Hijacking, and others.

3.2. Dedicated computer laboratories

Dedicated computer laboratories for IT security have been created in many universities, e.g. described in [8] or [11]. Security experiments or exercises are usually arranged using small, isolated computer networks (three computers). Compared to other approaches, dedicated computer laboratories are ideal environments for practical security teaching because security exercises are performed using productive security software in real systems. However, practical education by laboratory measures normally results in high costs. Dedicated networks require expensive hardware/software investments and intensive efforts to create, configure, and maintain laboratory environments as well as to prepare, supervise, and evaluate exercises. On the other hand, most security exercises require system level access to the operating system. This introduces the risk of misuse and inconvenience of administration. For security reasons, dedicated laboratories are usually operated in isolated networks, which implies that such security labs pitifully fail to benefit a wider range of learners outside campus.

Those labs overcome any critical security issue when offering practical security training – for the cost of the above mentioned disadvantages.

3.3. Virtual machine based computer courses

Instead of dedicated laboratories, at some universities virtual machines have been established for courses with special requirements to hard- or software configuration. In [3], the author describes the utilization of UML[6]-based virtual machines for a course in operating systems maintenance, [2] proposes the use of VMware Workstation[7] for courses on network administration, security and database maintenance. The virtual machines are mainly used for providing an environment, where students are allowed to arbitrarily configure an operating system and different applications. The virtual machines must be compiled individually for every course and distributed to the students in some way. Both authors confirm the opinion of the Tele-Lab research group, referring to the virtual machine technology as a cost efficient replacement for physical dedicated labs.

The VMs in the above described scenarios are issued to the students for either using them on their own computer (no security issues at all) or within a campus computer lab, where users have to authenticate against the physical system they are using.

3.4. Other (virtual) remote labs

In parallel to Tele-Lab, other approaches have been chosen to implement remote labs for computer science courses. Some of them are introduced here, highlighting potential security issues to fit the scope of this paper.

The authors of [7] built a remote lab to educate information security. The lab supports individual and collaborative experiments, and allows the students to setup own interactive experiments. The problem with this lab is its dependency on the physical machines; this makes it difficult to maintain or to reset the lab setup after usage or failure. In addition to that, the lab does not isolate the machines from each other. This is not a security issue but a usage restriction: the lab can only be used exclusively by one user or group per timeslot anyway. The lab architecture connects one of the two network interfaces of the host machine to the SPAN port of the switch; this allows the clients to sniff all the traffic including the traffic between the instructor machine and the file server and VPN concentrator.

In [1] the author describes a multipurpose remote lab for networking, security, and system administration classes. To isolate the students work from each other, the virtual machines are divided into teams (similar to Tele-Lab); only the virtual machines in the same team can communicate with each other. The problem with this system is its complete dependence on commercial applications e.g. MS Remote Desktop Access and VMware GSX – resulting in high cost and the dependency on a Windows system on client side. In Tele-Lab, the dependency on open source packages makes the system expansion cheaper, and allows a wide range of

[6] http://user-mode-linux.sourceforge.net
[7] http://www.vmware.com/de/products/ws

users to use our system (client side must only have a web browser supporting Java).

Another multipurpose remote lab is proposed in [9]. The solution schedules VPN access to a limited number of terminal servers. After each session termination, the system restored to the initial state by reverting back to the predefined images. The architecture described may allow users to attack the lab servers and the other terminal servers (besides the own) since they are on the same network segment. Also the Console Server (system for managing network equipment like switches or routing appliances) can be used to affect the other users' concurrent experiments.

4. Architecture of the Tele-Lab server

The current infrastructure of the Tele-Lab 2.0 server is a refactored enhancement to the architecture presented in [13]. 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 run in a Glassfish 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 When the server starts up, it is charged with a set of different virtual machines needed for the exercise scenarios – the pool. The maximum total number of VMs in the pool is limited by the resources of the physical host. In practise, a few (3-5) machines of every kind are started up. Those machines are dynamically connected to teams and bound to a user on request. The current hypervisor solution used to provide the virtual machines is KVM/QEMU[8] and the libvirt virtualization API[9].

Database The Tele-Lab database holds all user information, the WBT content and learning unit structure and, the information on virtual machine and team templates[10]. It also persists current virtual machine states and information on running sessions or remote desktop connections. The database is actually a data abstraction layer that encapsulates a set of XML files and a relational database for transparent access.

Remote Desktop Access Proxy The Tele-Lab server must handle concurrent remote desktop connections for users performing exercises. Those connections are proxied using a free implementation of the NX server[11]. The NX server forwards incoming connections to the respective assigned virtual machine accessing the QEMU framebuffer device via VNC (Virtual Network Computing).

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). On the one hand, this interface is made for content management in the web-based training environment and for user management. Additionally, the admin interface can be used for manual virtual machine control and monitoring or for registering new virtual machines. The Tele-Lab administrator can also easily build new team templates for exercise scenarios. Registered virtual machines can be grouped and enhanced with virtual networking configuration. The administration interface also allows monitoring of the NX server and running sessions as well as manual termination of remote connections.

Tele-Lab Control Services Bringing all the above components together is the purpose of the central Tele-Lab control services. To realise an abstraction layer for encapsulation of the virtual machine monitor (or hypervisor) and the remote desktop proxy, the system implements those as lightweight XML-RPC webservices. The *vmService* is for controlling virtual machines – start, stop or recover them, grouping teams or assigning machines or teams to a user. The *nxService* is used to initialize, start, control and end remote desktop connections to assigned machines. The above Grails applications (portal, tutoring environment, and web admin) control the whole system using the webservices.

On the client side, the user only needs a web browser supporting SSL/TLS and the appropriate Java-Plugin for the browser. For the remote desktop connections, the NX WebCompanion is included in the tutoring web application. The WebCompanion is a variant of the NX Client wrapped in a Java applet.

NX Clients connect to the NX Server using SSH-based authentication: client and server mutually certify each others identity using public-key authenticaton. Subsequently, the NX Client connects to a specific session with extra user credentials (session management described in detail later on). For mandatory encryption of the remote sessions, NX offers transport layer security (TLS).

[8] see `http://www.linux-kvm.org` and `http://www.nongnu.org/qemu`

[9] `http://libvirt.org`

[10] A VM template is the description of a VM disk image, that can be cloned in order to get more VMs of that type. Team templates are models for connecting VMs in order to perform certain exercises.

[11] freeNX, see `http://freenx.berlios.de`

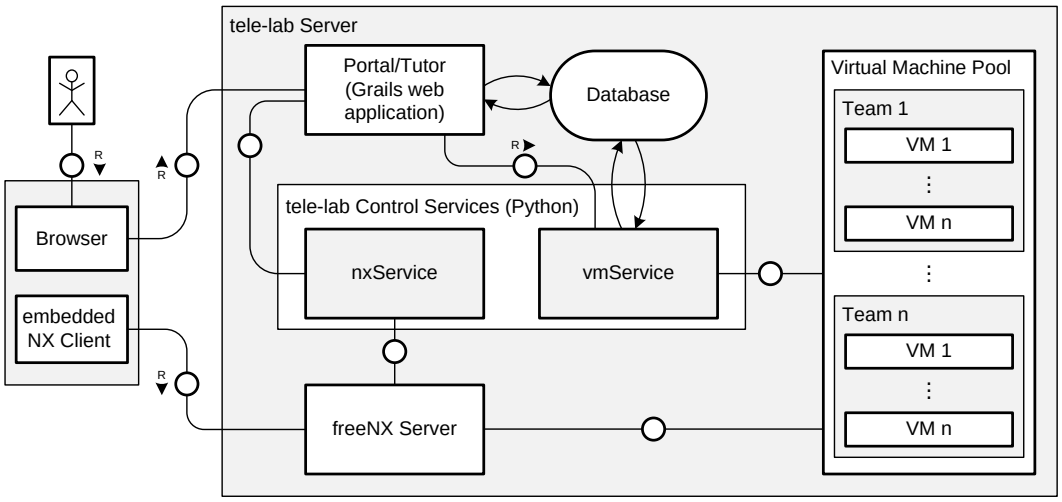

Figure 2. Overview: architecture of the Tele-Lab server

5. Securing the Tele-Lab server

As explained, the nature of the Tele-Lab architecture poses the server-side infrastructure to various security threats. This section will discuss general security objectives and possible attack classes and also introduce specific security solutions to avoid certain attacks.

5.1. Objectives and Attacks

The main security objectives for the Tele-Lab are

- *authentication* (users have to prove their identity against the system)
- *authorisation* (only qualified users may access the virtual lab)
- *availability* (the systems services should be reliably accessible)

Privacy is a minor issue for Tele-Lab. Of course, also in the field of e-learning it is necessary to protect users' personal data from unauthorised access and prevent eavesdropping on user sessions. But since the above mentioned objectives are more important, this is considered to be beyond the main scope of this paper.

Since Tele-Lab offers its services on the Internet, there is no possibility to reduce the attack surface below a certain point: at least the services for *http/https* (used for the web application) and the *ssh* port (22, used for the NX remote desktop) must listen for connections from the outside world.

Possible attacks include but are not limited to:

- Denial of Service (DoS),
- session hijacking (as well as for sessions in the web application as for the remote desktop session),
- and privilege elevation in order to gain unauthorized (administrative) access using e.g. replay attacks.

Also any kind of method compromising web applications in general could be suitable for attacking portal and tutoring interface of Tele-Lab.

5.2. Attack Vector: Web Application

As already mentioned, the WBT application of Tele-Lab has been implemented using Grails[12] – a modern web framework based on Groovy and the Java world. Grails incorporates numerous well-known Java (web) frameworks such as Hibernate for persistence, Spring and in particular Spring Security. In general, Java-based (web) applications are considered to be quite secure, since Java is largely immune against common *buffer overrun* or *malformed URL* exploits.

Also lots of the common web attacks are prevented or at least made more difficult just by the choice of the framework. *SQL injections* are hardly possible since the Hibernate-powered ORM (object relational mapping) automatically escapes data before persisting to the database. *XSS attacks* (cross-site scripting) mainly rely on unescaped user input from form fields or URLs. Due to typification of user input and the usage of carefully implemented tag libraries, this cannot happen within Tele-Lab.

To prevent *eavesdropping on the authentication* procedure in order to gain a valid username/password combination, submission of that kind of data (any personal user data)

[12] see `http://www.grails.org`

is strictly encrypted using TLS/SSL. Optionally, the Tele-Lab server can be switched to a mode where it only delivers https requests. In that case, also privacy for the current user session can be ensured.

Of course there may be possible future security issues arising from the *famous framework problem*: the more people use a web framework (or an application server), the more people attack such applications, the more general exploits will be published. The system administrators must be aware of this problem and watch out for security relevant updates at any time.

5.3. Attack Vector: Remote Desktop Access

The remote desktop access is a crucial factor for the overall security of the Tele-Lab system, since the most dangerous attacks on the server (or other Internet hosts) may be performed using one of the virtual training machines. To gain unauthorized access to a virtual machine, an attacker has to either *bypass the authentication* mechanism or to *hijack a remote desktop session*. As already described in section 4, the design of the NX technology makes attacks on the authentication mechanism very difficult for an attacker without valid user credentials. The NX server is as secure as the underlying implementation of the SSH server used for authentication. Hijacking a running session can be hindered by forcing the NX server to encrypt all connections using SSL. An exception to that would be an attack based on an intercepted NX session file (covered in the following paragraph). In conclusion, the Tele-Lab administrator also must scan for security bulletins concerning ssh- and openssl-packages.

This situation is different for a legitimate user – being in possession of valid user credentials. To understand this, the Tele-Lab authentication mechanism must be explained in more detail: basically, a user logs on to the web application (portal/tutor) submitting his username and password or OpenID[13] through an SSL tunnel.

When it comes to performing an exercise and requesting a team of VMs, the authentication against the NX server is managed user-transparent by the system: a so called NX session file is created containing all necessary credentials an the internal IP address of the respective VM. The file is transferred to the user's NX client browser plugin in a human-readable XML format. So the user has full access to the NX session file and might use the contained information – especially the VM user credentials and internal IP address – for a *replay attack* in order to log in to a VM that has not been assigned to that student.

To prevent that kind of replay attack, the *nxService* (see section 2) carefully restricts the temporal validity of the credentials and the availability of NX session files:

1. NX user accounts are disabled by default. On a valid VM request, the respective account is activated.
2. The NX session files are created dynamically for every connection. The password transmitted in that session file is randomly generated and only valid for the current session.
3. The filename of the session file is obfuscated using a hashed (MD5) concatenation of username and a timestamp.
4. This obfuscated session filename is transferred to the user (SSL encrypted). The NX browser plugin will download the session file – also through the SSL tunnel.
5. When a session terminates (timeout, user quits, connection is interrupted), the session file is deleted and the NX user account is disabled again.

This procedure ensures, that an authenticated user can't login unauthorized by replaying the NX session data. The encrypted transmission of all session data and the masquerade of session file names circumvents outside attackers from stealing valid credentials.

Additionally, the *vmService* generates a new random VNC password on every virtual machine recovery. This prevents users to replay the VNC credentials on another than their own VM or at some later time.

The remote desktop connection is also a channel for malicious authorized users to perform unwanted actions or attacks against the virtual machines or the physical host. Unintended behavior could be e.g. copying tools from the virtual machine to the users machine or vice versa. Since the virtual machines can be equipped with attack tools, exploits, and malware which may not be offered for download, the NX client must prevent any copy-paste activities between the users physical machine and the virtual machine.

As another protective measure, there can be no compilers provided on the virtual machines, that are remotely accessible by users. Malicious users shall be hindered to implement and run own attack code that could compromise the hypervisor implementation and allow the user to execute shellcode in the context of a process on the physical host.

5.4. Attack Vector: Admin Web Interface and Control Services

Since the administration panel for Tele-Lab is a Grails web application, it would be posed to all above mentioned web-based attacks – and inherits all benefits and flaws from the framework. While the tutoring interface must be accessible from the Internet and implicitly poses an attack surface, the administration interface does not. There is no need

[13] web-based single-sign-on technology, see `http://openid.net`

for configuring Tele-Lab over (the public part of) the Internet. The basic setup only allows connections to the administration panel itself only from the local network, requests to the Tele-Lab control services (XML-RPC webservices) are actually only permitted from localhost.

Access to the administrative interface or the webservices would enable an attacker to perform not only massive DoS attacks on the whole system (deleting and shutting down VMs, etc.), but would also allow creation or modification of user accounts and the illegal acquisition of the restricted user data.

To boost the access control for the admin interface even more, the Tele-Lab server even can be equipped with a second network interface for the local control/configuration network making prevention of spoofing-based attacks easier. Finally, access to the admin panel from outside the local network can be realized using a sufficient VPN solution, where the second NIC would only accept incoming connections from the respective VPN.

5.5. Attack Vector: Virtual Machine Pool

The main security issue for the virtual machine pool is – as already mentioned – the separation of the VMs. Though they share the same hardware, those common resources (memory, CPU registers, networking) must be strictly detached. For the separation of memory and CPU, Tele-Lab must rely on the underlying virtualization technology. Since the hypervisor component in Tele-Lab can be easily replaced (by implementing an interface), the most secure hypervisor solution can be used at any time.

Currently, the implementation of Tele-Lab depends on QEMU/KVM as virtualization platform. Each guest operating system (hosted VM) is considered as a single process of the host operating system (or hypervisor). Guest mode is used to execute guest operating system code for non-I/O processes. Any I/O requests are intercepted and routed to the user mode (non-privileged mode) to be emulated by the QEMU process. The KVM function is to provide each new initiated virtual machine a virtualized memory and a mapping to a determined address space. Actually, the physical memory mapped for each VM is the virtual memory mapped for the process. This is common practice for virtual machine separation on kernel-level.

Another dimension of separation is implemented in the virtual networking. Within a team of virtual machines network communication must be enabled, while communication with other teams or the physical host cannot be allowed. Tele-Lab connects the members of a VM team using multicast groups – each team is provided with an individual multicast socket that is connected to each team members 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 only communicate internally.

To understand this idea, we have to explain the networking concept of QEMU a little more 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 – as the (at least) 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 the described multicast socket to each machine's virtual LAN when the team starts up. The *vmService* consumes a team configuration in XML syntax as illustrated below (only items relevant for networking):

```
<tl:team name="Example Team" >
   <!--  first machine -->
   <tl:machine name="example_server">
      <tl:networkInterface
         mac="00:22:33:44:55:66"
         networkName="net0" />
   </tl:machine>
   <!-- second machine -->
   <tl:machine name="example_client">
      <tl:networkInterface
         mac="00:22:33:44:55:77"
         networkName="net0" />
   </tl:machine>
   <!-- virtual network -->
   <tl:network name="net0" id="1" />
</tl:team>
```

Here, we have a team of two virtual machines within the same virtual network *net0*.

As the *vmService* uses the libvirt hypervisor API to access QEMU/KVM, it transforms the configuration directives (enriched with some information from the database) to appropriate configuration file for each of the teams virtual machines. The networking part in that file would look like this for the second machine of the above team:

```
<devices>
   <interface type="mcast">
      <mac address="00:22:33:44:55:77">
      <source address="230.230.230.1"
         port="5001" />
   </interface>
   ...
</devices>
```

The Tele-Lab server has chosen the source IP address 230.230.230.1 from a pool of multicast addresses. All team members configuration files are set up with the same address. The next instance generated from that team template

would have another multicast IP from the address pool. This multicast address is from the physical hosts address space – and has nothing to do with the internal IP addresses of the guest systems. Those can be obtained via DHCP or configured statically.

Ongoing and future work on the Tele-Lab server targets on implementing a user-space network stack that allows connecting virtual machines without the need for multicasting – and to allow more complex network setups by connecting different instances of that user-space virtual switch to each other. Having all virtual networking handled as non-privileged operations also enhances security.

One known vulnerability of Tele-Lab was the possibility of repeated requesting of VM teams, which may end up with Denial of Service (DoS). If a user requests a team and immediately quits the session, he could have requested another team while the VMs just released were still recovering. Doing this repeatedly, the user could bring all VMs to the *recovering*-state causing a very high load on the physical server and hindering other users from performing any exercises.

The solution is to keep records of the currently running team of the virtual machines, the owner of each team, and of course a timestamp generated when the VM team is assigned to the user. This record of *user_id, vm_team_id, timestamp* is inserted into the database by the *Tele-Lab Control Service* and is kept until a) the user finished his work and logged out, *or* b) until the connection has terminated accidentally but the virtual machine recovers successfully *and* a time period named *safe_recovery_period* has been exceeded.

The *safe_recovery_period* is used to guarantee that the user cannot automatically (i.e. by running scripts) log in to the system and requests virtual machines repeatedly. The period should exceed the time, a virtual machines takes for recovery after usage. We suppose 3 to 5 minutes as suitable values for the *safe_recovery_period*. It is selected to be not too long for users with bad connection and repeated disconnects, but enough for the virtual machine to shutdown and start up again.

Each time the user asks for new virtual machines team, the *Tele-Lab Control Service* checks for this record. If found, the request will be denied and the user is asked to try again later – which must be considered as a disadvantage causing inconvenience for the user. Another disadvantage of this technique is that it prevents the user from running more than one experiment in parallel which is not (yet) a common case in our system.

The countermeasure must fail in the case, where a sufficiently large group of malicious users performs a joint attack and blocks all virtual machines on the server, or an attacker can gain enough user accounts. For the latter, we could store the remote IP address for each VM team assignment in the described record and prevent more than one connection for each remote IP within the *safe_recovery_period*. Due to the problems caused by this solution for user groups behind routers with network address translation, this option has not been implemented.

6. Conclusions and Future Work

The paper at hand describes a comprehensive infrastructure for a remote virtual computing lab and its secure operation at the example of "Tele-Lab Internet Security". This work focusses on the complex of problems arising from the need to provide privileged rights in an untrusted lab environment (the Internet) – which is inavoidable when teaching security with an open remote system. The carefully chosen components for the architecture (e.g. NX server with public key authentication and TLS) contribute to the servers overall security as well as the design-time decision to detach the system from the underlying virtualization technology. Custom solutions developed for known security problems in Tele-Lab may also apply for other virtual and/or remote computing laboratories.

Continuous work on Tele-Lab includes the creation of new content and exercise scenarios on the one hand and evaluation of the learners experience on the other. For evaluation, the system is being used for tutoring students who attend the lecture on "Internet Security – Weaknesses and Targets" at Hasso Plattner Institute. Future work shall show, if the offensive teaching approach in conjunction with practical hands-on experience in realistic environments generates a significantly better learning result and greater awareness boost. The ongoing installation of a second instance of the Tele-Lab server at the Vilnius Gedimino Technical University will allow to collect more user data in shorter time.

Technical enhancements planned for the next iterations of the Tele-Lab server are tools for remote collaborative learning and tutoring (e.g. Remote Desktop Assistance), clustering on application level to provide larger virtual machine pools, and the implementation of sophisticated virtual networking devices to allow more complex networking scenarios (several network segments in a team, i.e. DMZ). 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 above mentioned Tele-Lab server in Vilnius will be connected to the original one in at the Hasso Plattner Institute in Potsdam allowing students of both universities to use free resources on the whole cluster. Security has to be a central issue for this part of the future work, since the Tele-Lab server will need an additional networking channel causing an augmented attack surface. The installation of a VPN tunnel between the cluster nodes will provide a secure solution.

Another future activity is the functional detachment of

the virtual lab management system from the tutoring environment. Tele-Lab will be one use case for that virtual lab management, others are upcoming (i.e. the "SOA Security Lab"[14]).

References

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

[2] W. I. Bullers, S. Burd, and A. F. Seazzu. Virtual machines – an idea whose time has returned: application to network, security, and database courses. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 102–106, New York, NY, USA, 2006. ACM.

[3] R. Davoli. Teaching operating systems administration with user mode linux. In *ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pages 112–116, New York, NY, USA, 2004. ACM.

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

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

[6] 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*, pages 51–60, Monterey, USA, 2003.

[7] H. A. Lahoud and X. Tang. Information security labs in ids/ips for distance education. In *SIGITE '06: Proceedings of the 7th conference on Information technology education*, pages 47–52, New York, NY, USA, 2006. ACM.

[8] D. Ragsdale, S. Lathorp, and R. Dodge. Enhancing information warfare education through the use of virtual and isolated networks. *Journal of Information Warfare*, 2:53–65, 2003.

[9] S. Rigby and M. Dark. Designing a flexible, multipurpose remote lab for the it curriculum. In *Proceedings of the SIGITE conference*, pages 161–164. ACM, 2006.

[10] V. Schillings and C. Meinel. Tele-TASK – tele-teaching anywhere solution kit. In *Proceedings of ACM SIGUCCS*, Providence, USA, 2002.

[11] G. Vigna. Teaching hands-on network security: Testbeds and live exercises. *Journal of Information Warfare*, 2:8–24, 2003.

[12] 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*, pages 513–532, Saarbrücken, Germany, 2008.

[13] 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.

[14] K. Wolf, S. Linckels, and C. Meinel. Teleteaching anywhere solution kit (tele-task) goes mobile. In *SIGUCCS '07: Proceedings of the 35th annual ACM SIGUCCS conference on User services*, pages 366–371, New York, NY, USA, 2007. ACM.

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

[14] see `http://www.hpi.uni-potsdam.de/meinel/web_lab/soasecurity1/soa_security_lab.html`