A Distributed Virtual Laboratory Architecture for Cybersecurity Training

Christian Willems*, Thomas Klingbeil*, Lukas Radvilavicius†‡, Antanas Cenys‡ and Christoph Meinel*

*Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
Email: {christian.willems, thomas.klingbeil, meinel}@hpi.uni-potsdam.de

†UAB nSoft
Vilnius, Lithuania
Email: lukas.radvilavicius@nsoft.lt

‡Vilniaus Gedimino Technikos Universitetas
Vilnius, Lithuania
Email: {lukas, ac}@fmf.vgtu.lt

***Abstract*—The rapid burst of Internet usage and the corresponding growth of security risks and online attacks for the everyday user or enterprise employee lead to the concepts of 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.**

The Tele-Lab platform provides individual learning environments for each student, that may consist of up to three virtual machines per learning environment. Besides in explorative learning, where students use the laboratory whenever they like, the Tele-Lab is used in a blended learning approach: a lecturer introduces a security topic in class using e.g. Powerpoint slides. Subsequently, the students perform a supervised practical exercise in the virtual laboratory. A typically sized course with 15 students can in consequence request up to 45 virtual machines from the Tele-Lab server.

The paper at hand briefly presents usage, management and operation of Tele-Lab as well as its architecture. Furthermore, this work introduces an architecture for clustering of the virtual lab on application level and the necessary prerequisites for the implementation. The paper also presents an existing distributed usage scenario.

I. 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, cybersecurity 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 cybersecurity 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 cybersecurity 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 (e.g. a Man-in-the-Middle attack needs three computers: one for the attacker and two other machines as victims).

Tele-teaching for cybersecurity 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 tele-teaching system (explained in detail in section II).

Christian Willems, Thomas Klingbeil, Lukas Radvilavicius, Antanas Cenys, Christoph Meinel:
"A Distributed Virtual Laboratory Architecture for Cybersecurity Training"
in Proceedings of the 6th International Conference for Internet Technology and Secured Transactions (ICITST 2011),
IEEE Press, Abu Dhabi, UAE, 12, 2011 (to be published).

Section III presents an agile and comprehensive distributed setup for Tele-Lab's virtual laboratory that allows on-site enhancement of the physical resources for the lab environment as well clustering of independent installations of Tele-Lab on globally dispersed locations.

Section IV summarizes and gives an outlook on future enhancements to the Tele-Lab platform.

II. Tele-Lab: A Remote Virtual Security Laboratory

The Tele-Lab platform (accessible at http://www.tele-lab.org/, see fig. 1) was initially proposed as a standalone system [4], later enhanced to a live DVD system introducing virtual machines for the hands-on training [5], and then emerged to the Tele-Lab server [6], [8]. The Tele-Lab server provides a novel e-learning system for practical security training in the WWW while meeting the requirements of traditional 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 (e.g. 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.

A. Learning Units in Tele-Lab – an exemplary walkthrough

Learning units follow a straightforward 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 learning unit concludes with hints on how to prevent the just conducted attacks.

An exemplary Tele-Lab learning unit on *eavesdropping* (described in more detail in [10]) starts off with academic knowledge such as information on technologies for local area networks (LAN), the difference between switches and hubs or wireless networking. After that, various existing tools for packet sniffing are presented, such as *tcpdump* or the well known *Wireshark* network protocol analyzer.

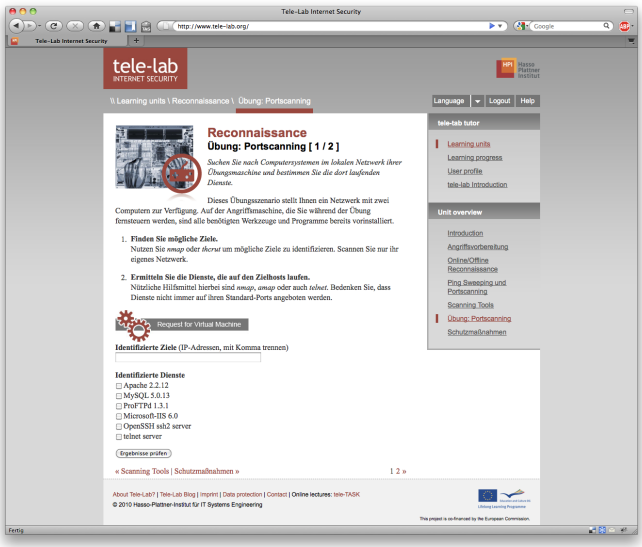

Fig. 1. Screenshot of the Tele-Lab Tutoring Interface

Following an offensive teaching approach (see [11] 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 is to *eavesdrop on network traffic* between two virtual communication partners, reveal credentials (username and password) for services from the captured messages and use these to steal private data from an FTP server.

Since the laboratory machines are connected on a virtual hub-like device, the student is able to capture all messages on the network – including the traffic between the two virtual victims Alice and Bob. Bob runs a server with HTTP and FTP services, Alice uses those services. The student has to use wireshark and inspect the captured packets for the login data. After that, he can log into Bobs servers using Alices username and password.

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), one machine for Bobs services and a third one for the client (Alice) running a set of scripts that access Bobs server. 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, malware, Man-in-the-Middle attacks etc. The system can easily be enhanced with new content.

B. Architecture of the Tele-Lab Platform

The current architecture of the Tele-Lab server is a refactored enhancement to the infrastructure presented in [8]. Basically it consists of the components illustrated in fig. 2.

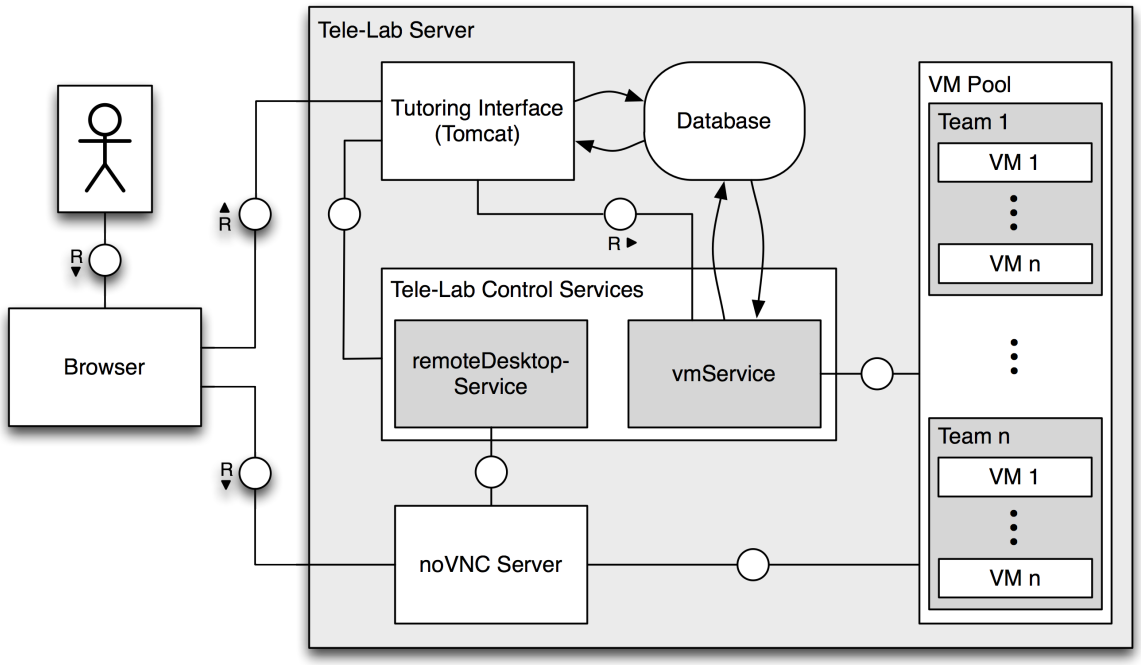

Fig. 2. Overview – Architecture of the Tele-Lab Platform

The following overview just explains the components that have to be enhanced for a distributed setup of the learning environment.

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* [12], [1]. The *libvirt* package [13] 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.

For the *network connections within the teams*, Tele-Lab uses the Virtual Distributed Ethernet (VDE) package [3]. VDE emulates all physical aspects of Ethernet LANs in software. The Tele-Lab Control Services launch virtual switches or hubs for each virtual network defined for a team of VMs and connect the machines to the appropriate network infrastructure. For the distribution of IP addresses in the virtual networks, a DHCP server is attached to every network. After sending out all leases, the DHCP server is killed due to security constraints.

Remote Desktop Access Proxy: The Tele-Lab server must handle concurrent remote desktop connections for users performing exercises. This is realized using the open-source project noVNC, a client for the Virtual Network Computing protocol based on HTML5 Canvas and WebSockets [7]. The noVNC package comes with the HTML5 client and a WebSockets proxy which connects the clients to the VNC servers provided by QEMU. Ensuring a protected environment for both the Tele-Lab users and system is a challenge that is important to thoroughly implement at all levels, as the issue of network security for virtual machines in a Cloud Computing setting (such as the case of Tele-Lab) poses special requirements. The system uses a token-based authentication system: an access token for a remote desktop connection is generated, whenever a user requests a virtual machine team for performing an exercise. Using TLS ensures the confidentiality of the token.

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 the encapsulation of the virtual machine monitor (or hypervisor) and the remote desktop proxy, the system implements a suite 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 assigning 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 only needs a web browser supporting SSL/TLS. The current implementation of the noVNC client does not even need an HTML5-capable browser: for older browsers, HTML5 Canvas and/or the WebSockets are emulated using Adobe Flash.

III. Enhancing the Tele-Lab Architecture for a Distributed Setup

As already mentioned, the Tele-Lab platform is intentionally also used to provide learning environments for hands-on practical exercises in a blended learning approach – where a high number of students pose a challenge to the scalability of the physical host running the virtual laboratory. The obvious solution is to integrate additional physical hosts in the data center to be able to run more virtual machines and balance the load between those servers.

During a project funded by the Leonardo da Vinci program for life-long learning of the European Commission, the Tele-Lab platform has been transferred to the Vilniaus Gedimino Technikos Universitetas, the technical university of Vilnius, Lithuania. The cooperation led to the existence of two completely independent instances of Tele-Lab. Both sites do not just run the same platform but also share most of the content – learning units as well as virtual machines for learning environments are the same or at least very similar. Considering the differences in lab scheduling between the sites in Germany and Lithuania, it became obvious, that "borrowing" resources from the respective other site could help both universities providing more virtual machines for lab classes without the need to purchase additional hardware.

The first task – load balancing between physical hosts at the same site – is fairly easy to solve, since Tele-Labs *vmControlService* is implemented with a 2-layer architecture: the encapsulation for the management of the VM lifecycle consists of an agent (KVM Wrapper), that controls the hypervisor on a physical host and a middleware layer to address several agents on different host machines. All on-site agents register with a single instance of the middleware. A Tele-Lab administrator can control all the virtual machines using a unified web-based administration interface.

The second task – borrowing virtual machines from another independent system – poses a number of challenges:

1) The Tele-Lab instances must be able to negotiate about the provision of virtual machines to users of the other system.
2) The VM teams on each server must be comparable: when requesting a specific team from another Tele-Lab server, the team template must be identified properly.
3) Virtual machines on a server must be assigned to users from another server. The remote desktop access to the virtual machines must be able to transparently forward a user of site A to a VM on site B.
4) In the original system, the Tele-Lab control services can only be accessed from localhost due to security constraints. The exposure of the *vmControlService* to a remote host must also be realized in a secure way.

A. VM and Team Identification

Virtual machines and teams of VMs in Tele-Lab consist of the VMs hard disk images and XML-based descriptions. A VM template description configures network interfaces (number, MAC addresses) and physical resources (CPUs, memory, hard disk image), a team template description determines which virtual machines are aggregated into a virtual network and the network structure. Fig. 3 shows a basic VM team template with three virtual hosts in two different virtual network. The XML file describes, to what network each network interface of all the virtual machines belongs to, e.g. the firewall machine having two network adapters connected to different switches.

```
<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. 3. Exemplary XML Team Configuration

To address the second challenge presented at the end of the preceding section, the equality of virtual machines and team templates has to be defined as follows. We define two *VM templates* A and B as *identical (VM A = VM B)*, if 1) the hard disk images are identical, and 2) the number of network interfaces is the same. Therefore, we calculate the md5 hash value for the hard disk image, concatenate the number of network interfaces to the md5 hash and hash that value again.

We call this value the $vmid_i$ for a virtual machine i.

$$vmid_i = md5(md5(hd_i) + \sharp nic_i) \quad (1)$$

$$A = B \Leftrightarrow vmid_A = vmid_B \quad (2)$$

This definition still classifies two machines as identical, if they differ in the amount of allocated memory or the number of CPUs, since those parameters just influence on the performance of the machines, but not on the basic functionality.

We define two *team templates* as *identical* if 1) all VM templates are identical, and 2) the network structure is identical. Let a team of VMs as in (3) be a 2-tupel consisting of a vector of VMs v and a vector of network representations n:

$$team_i = (v_i, n_i) \quad (3)$$

$$teamid_i = md5(v_i + n_i) \quad (4)$$

$$A = B \Leftrightarrow teamid_A = teamid_B \quad (5)$$

The elements of v are the $vmid$s of the team members in the lexicographic order. n is a vector representing the network structure as a normalized form of the network connections between the team members. A network is expressed as strings of the format *type(sorted_indices_of_vms)*, where *type* is either "switch" or "hub" and the *sorted_indices_of_vms* is a vector built from the indices of the elements of v, e.g. $switch(1, 3, 4)$. The elements of n must also be in lexicographic order.

The id of a team (4) is then defined as the md5 value of the vector v concatenated with the vector n (both represented as string without whitespace). Two teams A and B are *identical*, if their $teamid$s are equal (5).

In the above example (see fig. 3, there are three virtual machines in two networks. Let the calculated $vmid$s be as follows:

- $vmid_{vm1}$ = `7e716d0e702df0505...`
- $vmid_{vm2}$ = `d41d8cd98f00b204e...`
- $vmid_{vm3}$ = `a3cca2b2aa1e3b5b4...`

Vector v in the team tupel $team = (v, n)$ would be $v = (vmid_{vm1}, vmid_{vm3}, vmid_{vm2})$, while the networks in n would be constructed as $n = (switch(1, 3), switch(2, 3))$. The id for the example team template would be calculated as $teamid = md5(v, n)$.

The defined identifiers must be calculated, when a new virtual machine or VM team is created. The identifiers must also be updated, whenever a virtual machine changes (for example after installing a new tool, changing configuration, ...) or the team template is reconfigured. A new $vmid$ for a team member must be propagated to any team that uses the respective VM template.

B. Enhancing the vmControlServices

When a user requests for a VM team in order to perform an exercise, the server runs through a sequence of steps to check, whether the request can be approved. The sequence described as follows is the situation before the implementation of the off-site distribution workflow:

1) The tutoring frontend requests a VM team (more precisely a remote desktop connection to a team) from the *remoteDesktopService*. That service forwards the request to the middleware. The desired template for the specific learning situation that is requested from the Tele-Lab's control services has to be specified by the author of a learning unit.
2) The middleware checks, if there is an unoccupied team that has been cloned from the corresponding template on one of the on-site servers connected through the agents (servers A or B at location A in fig. 6).
3) If this is not the cause, the server checks its physical resources and the current load. If the load allows to launch additional virtual machines, the *vmControlServices* are triggered to fire an additional instance of the requested team template. It can also be the cause, that there are active virtual machines in the VM pool, that are no longer needed. These can be shut down to free additional resources.
4) If the current load is to high to provide additional VMs, the user request is added to a waiting queue.
5) Otherwise – if a suitable virtual machine team is available – the requested VMs are assigned to the user. The *remoteDesktopService* issues a token-secured URL for the VNC connection to the assigned machine.

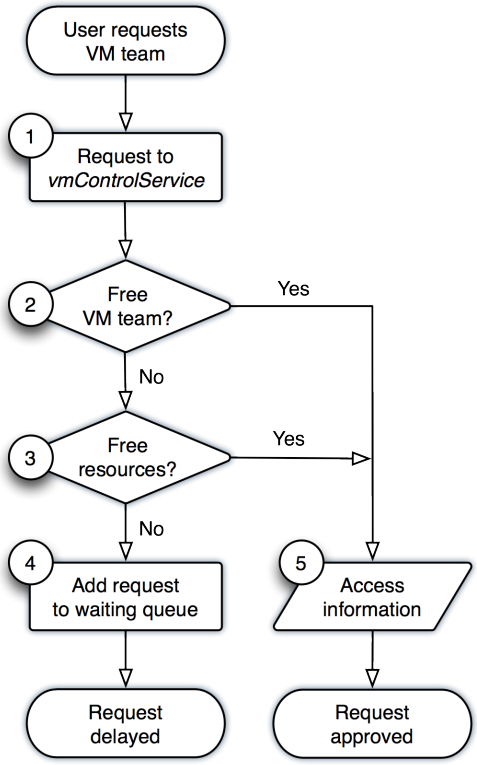

Fig. 4. Workflow: Requesting a Team of Virtual Machines

The situation changes, if there is another independent Tele-Lab server. If the on-site physical resources do not allow the immediate assignment of a requested VM team (step 4), the local Tele-Lab server can ask the other server for an instance of

that team template. Such functionality has to be implemented in the middleware part of the *vmControlService* (see fig. 6): a middleware instance of multiple connected Tele-Lab servers has to be aware of other middleware nodes, must be able to request virtual machine teams from other instances and needs an extensive ruleset to decide the response to a resource request. In particular, the latter aspect holds several issues that have to be considered when implementing the middleware extensions.

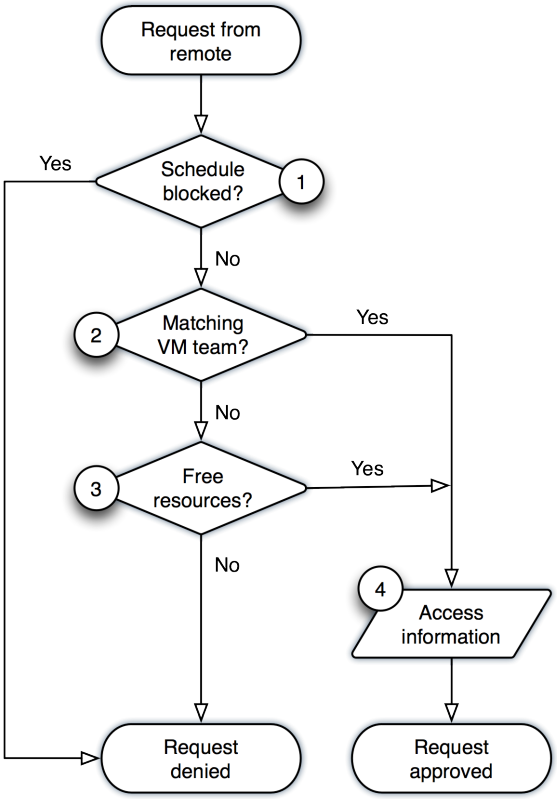

Fig. 5. Workflow: Decision on Remote VM Team Requests

Connecting multiple middleware nodes is realized in a straightforward way: system administrators add the IP address of other known and cooperating servers to a configuration file. There is no auto-discovery or any other automatic features (see section III-D for details on mutual authentication).

For requesting a virtual machine team from another server, the the XML-RPC API of the middleware is amended with a function *requestRemoteTeam()*. The only parameter for this function is the *teamid* for the requested team template calculated as in section III-A. On success – an instance of the requested VM team is available and can be borrowed – the function returns a socket for access, precisely an IP address and port number for the VNC connection. Otherwise, the function returns `false` and an error message for local logging. *requestRemoteTeam* is called after the local Tele-Lab server determined too high load value for providing a VM from its own resources in step 4 of the sequence in fig. 4.

More complex is the decision policy, whether a request for a remote virtual machine can be approved or must be denied (illustrated in fig. 5). The influencing factors for this decision are not only of technical nature: the first restriction to be checked is the lab schedule at the remote site. A Tele-Lab administrator can define time slots where no resources can be requested from remote. This can happen in order to guarantee, that on-site classroom sessions can not run out of resources due to load caused by lent virtual machines. If a request happens during such an "exclusive" time slot, it is instantly denied.

The next limiting factor is the availability of VM team instances matching to the requested team template. The request and the existing templates can be matched using the transmitted *teamid*. To determine a free instance, the middleware uses the procedures described in step 2 and 3 of the original workflow (fig. 4). If there are no free resources to provide the requested machine, the request is not enqueued in a waiting list, but simply denied.

C. Assigning and Providing Virtual Machines for Remote Users

Assigning a virtual machine team involves two activities in Tele-Lab: the VM team is flagged as *in use* in the database by the vmControlService, and the remoteDesktopService stores a mapping of the requesting user to the VNC socket provided by the middleware. This procedure must not be changed at all for the implementation of distributed setup.

A user at location A requests a VM team that can be provided by the server at location B (see fig. 6). The middleware at location A has received the access information (IP address and port number for the VNC connection) from its counterpart at location B and hands it back to the *remoteDesktopService*. This service works as follows:

1) a data structure containing the user id, the connection information and a timestamp is created
2) the service generates a hash value (access token) from the information in this data structure
3) a URL built of the address of the noVNC proxy server and the access token is transmitted to the user
4) the browser opens the web-based VNC client, the token is used for authentication
5) the noVNC proxy forwards the client to the VNC server socket stored in the data structure and flags the connection as established
6) the *remoteDesktopService* waits for the user to terminate the remote desktop session (i.e. close the client window)

When the *remoteDesktopService* recognizes, that a session is terminated, it informs the *vmControlService* to release the VM team and roll it back to the original state. This procedure has to be extended for the distributed setup: the middleware can easily determine, if a service call to terminate a session is meant for a local VM team or a remote on (comparison of the IP address in the connection information). If the request involves a remote team, the middleware passes the call to the server being responsible for that team.

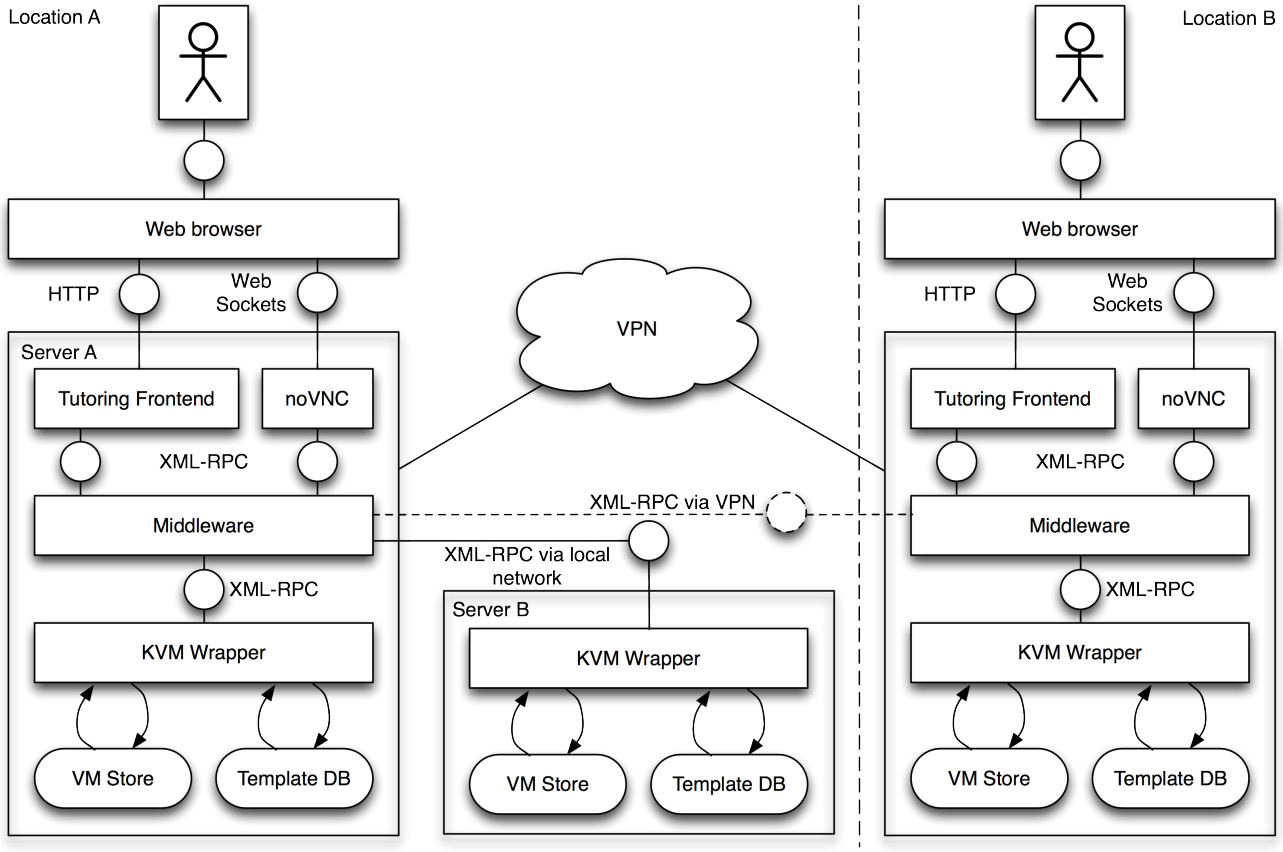

Fig. 6. Clustering of Tele-Lab Servers: On-Site Clustering (Location A) and Clustering of Independent Instances

The process of "borrowing" a virtual machine team from a remote location is completely transparent for the user, there is no difference in the user experience at all.

D. Security Considerations

As Tele-Lab provides the users with full-privileged access on the virtual machines they should use for training, the security of the system is a major issue for the implementation (described in detail in [9]). In general, the implementation aims on minimizing the attack surface and allowing as few attack vectors as possible: the only services accessible from the Internet shall be the web server for the tutoring interface (ports 80 and 443) and the noVNC proxy server for remote desktop connections (port 10099). Even the web interface for administration is only available from the internal network, XML-RPC based services are actually only accessible from the localhost or (in case of on-premise clustering, illustrated with location A in fig. 6) from the respective other Tele-Lab servers in the local network.

Ports for the VNC remote desktop connections are not accessible from the Internet – the policy for the XML-RPC services also applies for this service. All remote desktop connections must be initiated through the noVNC proxy server, that checks the transmitted access token from a request against the database and thus determines if the access can be authorized.

To prevent attacks with IP spoofing (external attacker modifies source IP address to belong to the address range of the internal network), there is an additional physical network interface for the communication with the local network. All packets sent from an internal IP addresses and received on the network interface for Internet connection are dropped by the firewall.

The enhanced system described in the paper at hand requires to permit access to the XML-RPC services provided by the middleware to the off-site instances of the Tele-Lab system, i.e. the Tele-Lab server in Lithuania must be allowed to access the middleware services provided by the Tele-Lab server in Germany. Access to the middleware services allows various attacks, such as starting up a large number of virtual machines and creating an overload situation or shutting down VM teams that are in use. Strong mutual authentication and authorization mechanisms are therefore essential.

A sufficient solution can be provided with the use of virtual private networking. The Tele-Lab hosts are connected through a VPN tunnel based on IPsec and certificates. In practice, OpenSWAN is configured to create the VPN tunnel. OpenSWAN creates new network interfaces on both machines for the endpoints of the VPN tunnel. The firewall must be adjusted, to also accept requests to the XML-RPC port

of the middleware from these network interfaces (analogous to the additional dedicated network interface for the local network). Since the VPN also encrypts all traffic, there is an supplementary protection against any replay-style attacks.

For a future increase of scalability of this solution, the introduction of security tokens for the access to the middleware services is considered.

IV. Conclusion and Future Work

The paper at hand presents a comprehensive architecture for a distributed virtual computer security laboratory with geographically dispersed hosting sites. The solution preserves the organizational independence of the institutions providing the local lab infrastructures, while it allows sharing physical computing resources for all participating lab providers. The paper also introduces a straightforward method to compare virtual machines and VM teams in order to assure the equality of two exercise scenarios on different lab servers.

The proposed architecture has been implemented for the Tele-Lab platform. A distributed infrastructure has been established for two independent instances of Tele-Lab in Germany and Lithuania.

A known limitation of this architecture is the dependence on the existence of a suitable (equal) virtual machine team on a remote server. If there is no such equal team (or team template) available, the request for the remote resources must be denied. A valuable supplement for the proposed architecture is the possibility to instantly exchange virtual machines or VM team descriptions. While the exchange of team descriptions in XML format do not pose a real challenge, the immediate transport of VM hard disk images over the VPN tunnel consumes CPU resources for the encryption and takes some time due to bandwidth limitations: 2 Gigabyte of binary data (reasonable size for a VM hard disk) would take about 6.5 minutes to be transferred with a T3 connection (45 Mbps). The transfer of a team of three virtual machines would take more than 15 minutes, even with such a fast connection. It is quite likely to assume, that there will be free resources on the local server within this time frame and the initiated transfer becomes useless. Furthermore, the cost for data transfer indicates the dynamic transfer of virtual machines or VM teams to be infeasible.

For the future, there are two more sophisticated approaches to this problem to be investigated. The first approach bases on the observation, that independent Tele-Lab instances usually start with an equal set of virtual machines. Local administrators apply minor adjustments to the VMs to fit the needs of their students (i.e. change the default language and key mapping). The idea is to introduce a version control for virtual machines: the base disk image will be kept in a repository, changes to a VM are stored as differential files, and a history of *vmids* allows to determine the base image of a virtual machine. In case of a remote request, that would have to be denied with the current architecture, the system can look for a suitable base system and just transfer the differential files. Future work will evaluate the performance of tools for differential synchronization of binary files and so thus determine the feasibility of this approach.

A second approach could be the description of the specifications of the system running in a virtual machine in a computable manner. Operating systems and versions, tools running, services, vulnerabilities, and all other parameters influencing the typical application possibilities of a machine must be described in a formal way. Semantic technologies (e.g. the Resource Description Framework RDF or ontologies) could be useful components of this approach. While this idea allows the definition of a universal comparison operator for systems running inside the VMs (respectively exercise scenarios), it poses a lot of challenges for the data collection: this should at least be realized in a semi-automatic manner, since human input tends to be prone to error.

Acknowledgment

This work was partially funded by the Leonardo da Vinci program for Lifelong Learning of the European Commission (project number: LLP-LdV-TOI-2009-LT-0037).

References

[1] F. Bellard. (2011) QEMU – Open Source Processor Emulater homepage. [Online]. Available: http://www.qemu.org/, accessed: 2011-10-17

[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] R. Davoli. (2011) Virtual Distributed Ethernet homepage. [Online]. Available: http://vde.sourceforge.com/, accessed: 2011-10-17

[4] 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.

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

[7] J. Martin. (2011) noVNC project website. [Online]. Available: http://kanaka.github.com/noVNC/, accessed: 2011-10-17

[8] 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.

[9] C. Willems, W. Dawoud, T. Klingbeil, and C. Meinel. "Protecting Tele-Lab – Attack Vectors and Countermeasures for a Remote Virtual IT Security Lab", in *International Journal of Digital Society (IJDS)*, Volume 1, Issue 2, p. 113–122, 2010.

[10] C. Willems and C. Meinel. "Practical Network Security Teaching in an Online Virtual Laboratory", in *Proceedings of Security and Management 2011*, p. 65–71, Las Vegas, USA, 2011.

[11] 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.

[12] Red Hat, Inc. (2011) Kernel-based Virtual Machine (KVM) homepage. [Online]. Available: http://www.linux-kvm.org/, accessed: 2011-10-17

[13] The Libvirt Developers. (2011) libvirt – The virtualization API homepage. [Online]. Available: http://libvirt.org/, accessed: 2011-10-17