Hasso–Plattner–Institut für Softwaresystemtechnik
an der Universität Potsdam

A Virtual Machine Architecture for IT–Security Laboratories

Dissertation
zur Erlangung des akademischen Grades
"doctor rerum naturalium" (Dr. rer. nat.)
in dem Fachgebiet Internet-Technologien und -Systeme

von
Ji Hu

Potsdam 2006

Gutachter:
Prof. Dr. Christoph Meinel, Hasso-Plattner-Institut
Prof. Dr. Claudia Eckert, Technische Universität Darmstadt
Prof. Dr. Huaqing Wang, California State University

Vorsitzender:
Prof. Dr. Werner Zorn, Hasso-Plattner-Institut

Prüfungskommissionsmitglieder:
Prof. Dr. Andreas Polze, Hasso-Plattner-Institut
Prof. Dr. Mathias Weske, Hasso-Plattner-Institut
Prof. Dr. Andreas Schwill, Universität Potsdam
Prof. Dr. Michael Gössel, Universität Potsdam
Prof. Dr. Klaus Denecke, Universität Potsdam
Prof. Dr. Thomas Engel, University of Luxembourg

Datum der Disputation: 14. Juli 2006

Die Dissertation wurde am 26.01.2006 an der Mathematisch–Naturwissenschaftlichen Fakultät der Universität Potsdam eingereicht und am 14.07.2006 mit “magna cum laude” angenommen.

To my parents, Daoxian and Dewen

Zusammenfassung

Diese Dissertation beschreibt die Herausforderungen in der IT Sicherheitsausbildung und weist auf die noch vorhandene Lücke zwischen E-Learning und praktischer Ausbildung hin. Sie erklärt einen Ansatz sowie ein System, um diese Lücke zwischen Theorie und Praxis in der elektronischen Ausbildung zu schließen.

E-Learning ist eine flexible und personalisierte Alternative zu traditionellen Lernmethoden. Heutigen E-Learning Systemen mangelt es jedoch an der Fähigkeit, praktische Erfahrungen über große Distanzen zu ermöglichen. Labor- bzw. Testumgebungen sowie praktische Übungen sind jedoch unverzichtbar, wenn es um die Ausbildung von Sicherheitsfachkräften geht. Konventionelle Laborumgebungen besitzen allerdings einige Nachteile wie bspw. hoher Erstellungsaufwand, keine Mobilität, hohe Wartungskosten, etc. Die Herausforderung heutiger IT Sicherheitsausbildung ist es daher, praktische Sicherheitslaborumgebungen und Übungen effektiv mittels E-Learning zu unterstützen.

In dieser Dissertation wird die Architektur von Tele-Lab IT-Security vorgestellt, die Studenten nicht nur erlaubt theoretische Sicherheitskonzepte zu erlernen, sondern darüber hinaus Sicherheitsübungen in einer Online-Laborumgebung praktisch zu absolvieren. Die Teilnehmer können auf diese Weise wichtige praktische Erfahrungen im Umgang mit Sicherheitsprogrammen sammeln. Zur Realisierung einer sicheren Übungsumgebung, werden virtuelle Maschinen anstatt reale Rechner im Tele-Lab System verwendet. Mittels virtueller Maschinen können leicht Laborumgebungen geklont, verwaltet und über das Internet zugänglich gemacht werden. Im Vergleich zu herkömmlichen Offline-Laboren können somit erhebliche Investitions- und Wartungskosten gespart werden.

Das Tele-Lab System bietet eine Reihe von technischen Funktionen, die den effektiven, zuverlässigen und sicheren Betrieb dieses Trainingssystems gewährleistet. Unter

Beachtung angemessener Ressourcennutzung, Softwareinstallationen und Systemkonfigurationen wurden virtuelle Maschinen als Übungsstationen erstellt, die auf einem einzelnen Rechner betrieben werden. Für ihre Zuverlässigkeit und Verfügbarkeit ist das Managementsystem der virtuellen Maschinen verantwortlich. Diese Komponente besitzt die notwendigen Überwachungs- und Verwaltungsfunktionen, um kritische Fehler der virtuellen Maschinen während der Laufzeit zu erkennen und zu beheben. Damit die Übungsstationen nicht bspw. zur Kompromittierung von Produktivnetzwerken genutzt werden, beschreibt die Dissertation Sicherheits-Managementlösungen, die mittels Isolation auf System und Netzwerk Ebene genau dieses Risiko verhindern sollen.

Diese Arbeit ist der Versuch, die Lücke zwischen E-Learning/Tele-Teaching und praktischer Sicherheitsausbildung zu schließen. Sie verfolgt nicht das Ziel, konventionelle Ausbildung in Offline Laboren zu ersetzen, sondern auch praktische Erfahrungen via E-Learning zu unterstützen. Die Dissertation zeigt die Möglichkeit, praktische Erfahrungen mittels Sicherheitsübungsumgebungen über das Internet auf zuverlässige, sichere und wirtschaftliche Weise zu vermitteln.

Abstract

This thesis discusses challenges in IT security education, points out a gap between e-learning and practical education, and presents a work to fill the gap.

E-learning is a flexible and personalized alternative to traditional education. Nonetheless, existing e-learning systems for IT security education have difficulties in delivering hands-on experience because of the lack of proximity. Laboratory environments and practical exercises are indispensable instruction tools to IT security education, but security education in conventional computer laboratories poses particular problems such as immobility as well as high creation and maintenance costs. Hence, there is a need to effectively transform security laboratories and practical exercises into e-learning forms.

In this thesis, we introduce the Tele-Lab IT-Security architecture that allows students not only to learn IT security principles, but also to gain hands-on security experience by exercises in an online laboratory environment. In this architecture, virtual machines are used to provide safe user work environments instead of real computers. Thus, traditional laboratory environments can be cloned onto the Internet by software, which increases accessibility to laboratory resources and greatly reduces investment and maintenance costs.

Under the Tele-Lab IT-Security framework, a set of technical solutions is also proposed to provide effective functionalities, reliability, security, and performance. The virtual machines with appropriate resource allocation, software installation, and system configurations are used to build lightweight security laboratories on a hosting computer. Reliability and availability of laboratory platforms are covered by a virtual machine management framework. This management framework provides necessary monitoring and administration services to detect and recover critical failures of virtual machines at run time. Considering the risk that virtual machines can be misused for compromising production networks, we present a security management solution to prevent

the misuse of laboratory resources by security isolation at the system and network levels.

This work is an attempt to bridge the gap between e-learning/tele-teaching and practical IT security education. It is not to substitute conventional teaching in laboratories but to add practical features to e-learning. This thesis demonstrates the possibility to implement hands-on security laboratories on the Internet reliably, securely, and economically.

Acknowledgements

This thesis would not have been possible without encouragement, guidance, and advice of Prof. Dr. Christoph Meinel. I deeply thank him for providing the opportunity and financial support for me to pursue my PhD in Germany.

I am grateful to Dr. Michael Schmitt, Dirk Cordel, and Christian Willems for their friendly discussion and collaboration in this work. I very much enjoy working with my colleagues at the Institute for Telematics, University of Trier, and Hasso-Plattner-Institute. I particularly thank Viola Brehmer, Agnes Jacoby, Gennadij Umanskij, Mingchao Ma, and Volker Schillings. My thanks are also due to Prof. Zhiheng Lu, Prof. Dr. Huaqing Wang, and Dan Li for their helpful comments on the thesis.

I owe a great debt to my parents, Daoxian Zu and Dewen Hu for their constant love and support over the years. This thesis is dedicated to them.

Contents

List of Figures

List of Tables

1. Introduction

Computer has significantly changed our everyday life and also the way to acquire knowledge and skills. Thanks to information and communication technology, distance education and traditional classroom education are starting to be transformed to e-learning or online learning [Hiltz and Turoff, 2005]. E-learning increases accessibility to education, greatly reduces cost and time for offering and accessing lectures, and therefore is able to reflect rapid updating of knowledge [Neal and Miller, 2004]. E-learning also personalizes education. With access to computers or the Internet, students are free to learn at home or in the workplace and independent from lecture schedules. For those who wish to receive further or lifelong education, e-learning provides a mobile and flexible way for them to continue their studies [Meinel et al., 2003].

E-learning has been prevalently applied in computer science education for decades. Nevertheless, compared with classroom education, e-learning typically has difficulties in delivering hands-on experience because of the lack of proximity. Particularly, e-learning has not been a satisfactory means for teaching practical IT security because it has failed to deal with laboratory environments and real-life exercises. This thesis is intended to introduce practical features into e-learning and make it a practical teaching/learning tool for IT security.

1.1. The Challenge in Practical IT Security Education

With increasing emergence of security threats and system vulnerabilities, IT security has more and more impact on our daily work and life. In order to strengthen public awareness of IT security, many universities have integrated security courses into their curricula. Those courses are taught by traditional measures such as textbooks, slides, or papers. In most cases, however, only theoretical aspects of IT security are covered and one essential part of education, i.e. hands-on experience, has been neglected. In very few cases, students has a chance to learn practical skills or to experiment in

production environments. In fact, one of the most important purposes of security education is to prepare the skillful workforce in response to the future security challenges [Bishop, 2000]. Practical IT security education must meet the need of training students to apply security technologies for real environments. In this connection, traditional teaching measures have been proved to be insufficient. There is a need to provide hands-on security experiences by exercises in laboratory environments.

The problem in practical IT security education now is that known computer-aided instruction solutions (see Section 1.4) either have difficulties in offering hands-on experiences or are problematical because of immobility and high costs. Multimedia courseware only provides experience by video or animations, and no real-life experiences are offered. Demonstration software demonstrates cryptographic algorithms in detail, but neither production tools nor everyday environments are involved. Simulation systems are not intuitive since they are simulated in abstract environments where students have no chance to apply production tools. Although security exercises in dedicated computer laboratories are ideal for students to gain hands-on experience, laboratory environments are pitifully geographically limited and financially and labor-intensively expensive.

From a practical point of view, laboratory environments and hands-on exercises are indispensable instructional tools for IT security education. We believe that integration of security laboratories and exercises to e-learning is a promising and economical solution to the problem of practical security education. I.e. supported by e-learning, individuals will be able to access laboratory resources and practice security whenever and wherever they want. Nonetheless, transforming security exercises and laboratory environments into the e-learning form is a challenging task which poses the following problems:

- Security experiments imply an insecure environment. In most exercises, students need privilege rights on the computer to perform security operations, which introduces a risk that a user might compromise the computer system and related production networks by misusing his/her right.

- Laboratory platforms become unreliable because of security experiments. Errors or failures caused by users may easily result in system corruption and interruption of learning processes.

- Because of security concern, accessibility to the laboratories is limited only in

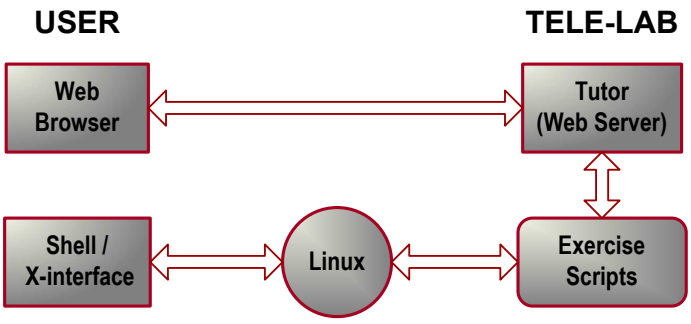

Figure 1.1.: Architecture of Tele-Lab IT-Security .

universities or colleges. The need of the learners outside the campus can not be met.

In response to the challenge, we need an architecture for automating administration and supervision of laboratory environments to reduce workloads and replace the roles of instructors and administrators with computers. This architecture should also be able to mobilize laboratory platforms and enable individuals to exercise security through more flexible tele-teaching or web tutoring systems. Developing such an architecture is the motivation of this thesis.

1.2. Tele-Lab IT-Security: The Concept and Architecture

Tele-Lab IT-Security ("Tele-Lab") is a novel and evolving concept for practical security education. Tele-Lab is intended to implement a web tutoring system which allows students to learn about IT security and to gain hands-on experiences in a lightweight and safe laboratory environment. The teaching subjects covered by Tele-Lab include various aspects of cryptography and network security such as encryption, certificates, secure email, authentication, etc. For each subject, Tele-Lab first presents theoretical basics about the subject field and then provides related security exercises which offer students a chance to experiment security in a lightweight Linux laboratory.

Figure 1.1 illustrates the concept of Tele-Lab IT-Security. Tele-Lab features a web user interface. The user end interacts with a *tutor* which is a web server by using a standard browser. The role of the tutor is to organize and present teaching materials, and to prepare and manage interactive security exercises. User work environment is

implemented on a native Linux machine. This machine is pre-configured and equipped with various opensource security tools so that it can be seen as a small security laboratory. Security exercises are managed automatically by scripts: they are invoked by the tutor for preparation of exercise. To perform exercises, the user can operate in the Linux system and apply system programs and security tools via a shell or X-Window interface. It is up to the tutor to evaluate the user's result. If necessary, the evaluation is done by scripts. In such a way, real-life learning/teaching scenarios are realized.

Tele-Lab was originally a standalone system intended to support security teaching in a laboratory [Schmitt et al., 2003b, Hu et al., 2003]. Its tutoring system, exercises, and user work environment are integrated on a single Linux machine. The problem of this version is that the computer could be easily corrupted by user errors during exercises. In case of failure, exercises are interrupted and the whole system has to be recovered using a partition backup.

The *Tele-Lab CD/DVD* was developed in order to avoid unnecessary recovery costs and to improve mobility [Hu and Meinel, 2004b]. The entire Tele-Lab system including the Linux system and teaching materials is integrated into a *Knoppix* CD/DVD [Knopper, 2000]. It is special bootable CD/DVD which runs a live Linux operating system on a computer without a hard-disk installation. By this means, a portable and reliable training CD/DVD has been realized. Students can bring it with themselves and easily use it on a general PC at home or any place they want. Failures or errors during learning would not affect hardware or software installations, and can be simply recovered by rebooting. Nevertheless, live CDs/DVDs have to reserve a large size of space for the operating system itself. We can not integrate many materials and exercises because of the space limit. With similarity to the standalone version, it is only a single-user system because of its local usability.

From the e-learning/tele-teaching point of view, it would be more desirable if students are able to learn about security and gain hands-on experience over the Internet. The Tele-Lab IT-Security server [Hu et al., 2004, Hu et al., 2005] is the next generation of Tele-Lab systems for e-learning. The motivation to develop the Tele-Lab server is to transform native Tele-Lab systems into a web-based frame and to make laboratory resource accessible for remote users. The Tele-Lab server inherits the same concept and has a similar architecture of the previous Tele-Lab systems. The distinct feature of the server is that it provides separate user work environments which are built with

virtual machines[1] instead of native Linux systems. Virtual machines are only user-level applications and can be created on a host and connected to a network. Thus, the entire laboratory network can be cloned on a computer. On each virtual machine, privilege rights can be freely assigned to users for finishing security tasks. The crash of the virtual machine can be easily recovered and will not affect the other parts of the Tele-Lab server. By introducing virtual machines into Tele-Lab, laboratory resources can be conveniently and economically simulated by software, which offers a great possibility to operate Tele-Lab on the Internet.

1.3. Solutions to Implementation Problems

The Tele-Lab IT-Security architecture is a virtual machine architecture for IT security laboratories. By applying this architecture in security education, conventional security laboratory environments and exercises become available over the Internet. Though Tele-Lab IT-Security is a novel and pragmatic idea, to realize such a real system, special implementation problems must be addressed:

- **Functionality**. In order to effectively clone those functions which are normally provided by real laboratory systems, virtual machines must be carefully installed and configured with appropriate virtual machine software. The user working on a virtual machine should not notice obvious functional differences from working on a real operating system.

- **Reliability**. Users would feel frustrated if services or virtual machines are frequently interrupted by failures or following recovery procedures are slow. Prompt detection of the virtual machine errors and fast recovery from failure are necessary to guarantee service availability.

- **Security**. With privileges allowed on virtual machines and Internet connections, the Tele-Lab virtual machine might be converted by its user to an attack workstation and endanger production networks. User activities on the virtual machine must be under control and misuses of laboratory resources should be prevented by security measures.

[1] Virtual machines ("VM") are software applications for simulating a physical computer (see Chapter 2).

- **Performance**. In order to run virtual machines as many as possible, the Tele-Lab server must provide enough system resources. On the other hand, performance is crucial for individual virtual machines. The Tele-Lab server has to manage simultaneous access and each virtual machine has to possess reasonable resources for its user.

The main contributions of this thesis include the proposal of the Tele-Lab IT-Security architectures [Schmitt et al., 2003b, Hu and Meinel, 2004b, Hu et al., 2004] and a set of solutions for effective implementation of functionality, reliability, security, and performance of security laboratories based on virtual machines [Hu and Meinel, 2004a, Cordel, 2004, Hu et al., 2005].

The virtual machine approach applied in Tele-Lab IT-Security has proved to be able to simulate laboratory platforms on a host machine [Hu and Meinel, 2004a]. System resources (processor, memory, and file-systems), software installation, system configuration, and user interface of virtual machines are well organized in Tele-Lab, so that virtual machines can be used as lightweight laboratories at a reasonable performance [Hu et al., 2005].

The virtual machine management framework presented in this thesis addresses reliability and availability of Tele-Lab IT-Security [Cordel, 2004, Hu et al., 2005]. In this framework, virtual machine administration and monitoring services are realized. With those monitoring and recovery services, critical failures of virtual machines during learning can be detected and recovered at run time.

This thesis specially addresses the security aspect of Tele-Lab IT-Security architecture [Hu et al., 2005]. Security requirements of the Tele-Lab architecture are analyzed firstly. Then system security isolation between virtual machines and the host is examined. Next, a network-level security solution is presented. This solution effectively enforces access control through the destination *Networking Address Translation* technique and firewalls.

Technical solutions described in this thesis offer the possibility to implement the Tele-Lab IT-Security as practical e-learning systems for security education, and also provide advanced features which distinguish it from other security teaching/learning approaches.

- Distinct from multimedia courseware and demonstration software, Tele-Lab IT-Security not only implements the features of conventional tutoring systems for

teaching theoretical facts, but also provides students with realistic laboratory experience.

- In Tele-Lab IT-Security, user's laboratory environments are simulated by lightweight virtual machines. Those virtual machines provide a real operating system environment with appropriate configurations and opensource security software installations. Therefore, differing from those simulation systems, Tele-Lab IT-Security offers students a real-life laboratory environment instead of limited simulation; compared with dedicated security laboratories, Tele-Lab IT-Security implements mobility of learning and economization of cost and effort.

- Tele-Lab IT-Security has a thin web user interface. With this interface, no additional software is required to install on the user-end; tools or programs needed in the security exercise are accessible via a VNC applet[2].

- Virtual machine management of Tele-Lab IT-Security provides high reliability and availability. Critical system errors or failures of the virtual machine can be detected and recovered automatically. Thus, even though dangerous security experiments are allowed on the virtual machine, the entire Tele-Lab system still remains a reliable manner.

- User activities in Tele-Lab are constricted in a safe scope by security management. Risks of compromising production networks by misuse of laboratory resources are eliminated by security isolation measures.

1.4. Related Work

Currently, four categories of computer-supported learning/teaching measures are applied in IT security education [Schmitt et al., 2003a]. They include multimedia courseware, demonstration software, simulation systems in specific problem areas of IT security, and dedicated computer laboratories for security experiments.

[2]Virtual Network Computing ("VNC") is a remote desktop access system based on the Remote Frame Buffer (RFB) protocol (See Section 4.3.1).

Figure 1.2.: The tele-TASK client end.

1.4.1. Multimedia Courseware

Supported by web and multimedia technologies, e-learning courseware has proved to be an intuitive way to digitize lectures or to demonstrate security processes. Today, multimedia courseware has been prevalently applied in web-based tele-teaching or e-learning. For example, tele-Task [Schillings and Meinel, 2002] is a state-of-the-art e-learning solution which has offered a variety of online security lectures and seminars[3]. All lectures are recorded and encoded as Real or MPEG-4 audio and video streams which can be viewed synchronously or asynchronously by using a web browser (see Figure 1.2). The Institute for Telematics at Trier [Institut für Telematik, Trier, 2001] developed a set of Flash animations to demonstrate security protocols such as encryption and firewalls. Figure 1.3 shows a demonstration for the Secure Socket Layer protocol (SSL) which visualizes interactions happening in each step of the protocol. In fact, security courseware for e-learning is a clone of traditional blackboard lectures by computer visualization or video technologies. Pitifully, no hands-on experiences are offered.

Parallel to academic e-lectures, there are a lot of web-based security training courses which are available in industry. E.g. Hill Associates is a big training service provider in the field of telecommunications and security[4]. Its virtual classroom training courses are offered based on voice-over-IP, video presentations, application sharing, and whiteboard featured technologies, by which professionals can learn, interact, and collaborate across the Internet. Similar training programs can be also found at RSA Security[5]

[3]Tele-TASK `http://www.tele-task.de/`.
[4]Hill Associates. "What training should be". Available at `http://www.hill.com/training/`.
[5]RSA Security. "RSA Product Training". Available at `http://www.rsasecurity.com/training/`.

Figure 1.3.: Visualization of the Secure Socket Layer (SSL) protocol.

and NetIQ[6]. Most industrial web courses address the training for particular products and very often they have to arrange hands-on workshops in the last phase of the course to offer trainees chances to apply techniques or tools. Therefore, in the practice phase, industrial online programs still have to resort to traditional face-to-face interaction which requires manual preparation of working environments and supervision through exercises.

1.4.2. Demonstration Software

Demonstration tools such as Cryptool [Esslinger, 2002] and CAP ("Cryptographic Analysis Program") [Spillman, 2002] are educational software suits which are used to teach or learn about cryptography and cryptanalysis. Cryptool is a free software program for demonstrating encryption algorithms and analysis procedures (see Figure 1.4). CAP is used to explore different implementations of the ciphers in a classical cryptology course. CAP also provides necessary tools for breaking ciphers.

The common feature of security demonstration software is that they provide an interactive user interface through which students are able to test real algorithms with various parameters. In this sense, demonstration software provides more practical features than multimedia courseware. This approach is mainly used to support teaching academic aspects of information security. But neither production tools nor everyday environments are involved in the learning or teaching processes.

[6]NetIQ. "NetIQ Training". Available at `http://www.netiq.com/training/`.

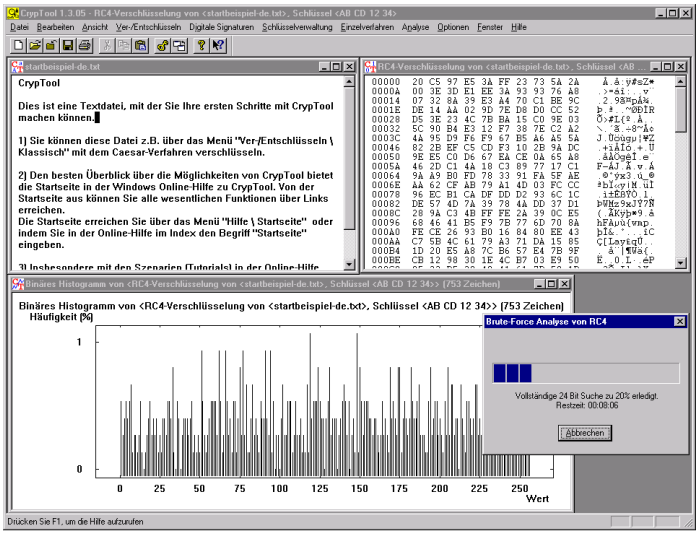

Figure 1.4.: Cryptool: an eLearning Program for Cryptology.

1.4.3. Simulation Systems

There are a few simulation systems for training students in specific problem areas of IT security. E.g. ID-Tutor [Rowe and Schiavo, 1998] and the intelligent tutoring system in [Woo et al., 2002] are intended to familiarize students with intrusion detection. ID-Tutor creates audit files with information on user logins and executed commands. A user has to make a decision about whether an intrusion has occurred, and, in case of an intrusion, he/she must resolve the problem. The tutoring system in [Woo et al., 2002] is very similar to ID-Tutor except that it generates security missions from a knowledge base. Both tools implement a simulation environment where students are able to apply commands to fulfill their tasks. CyberCIEGE [Irvine and Thompson, 2004] is a computer simulation game to teach computer security principles. As shown in Figure 1.5, players construct computer networks, and make choices affecting the ability of these networks and the virtual users to protect valuable assets from attacks.

Simulation systems offer students chances to perform operations for accomplishing "real" tasks such as identifying intrusion and recovering or cleaning systems. Nonetheless, those security operations are not really performed but simulated in an abstract environment. In fact, real computer systems can be modeled by such simulations only to a very limited degree. Simulation systems increase interactivity to some degree but still offer students no chance to apply real world tools and to see what's going on in practice.

Figure 1.5.: CyberCIEGE: an information assurance training tool.

1.4.4. Dedicated Computer Laboratories

As a part of security education, security laboratories and related exercises have been developed in many universities. Usually those security exercises are arranged on dedicated computer networks which are separated from production networks for security reasons. E.g. U.S. Military Academy [Ragsdale et al., 2003] developed an isolated laboratory for information assurance education. In this environment, students are able to familiarize themselves with various known computer exploits and exercise to employ technical measures to defend their network against such exploits. The George Washington University [Hoffman et al., 2003] has built a portable education network for assisting the study of computer security. This network comprises three distinct policy domains: attack, target, and administrative domains (see Figure 1.6). In the attack domain, students can use computers to attack the servers in the target domain. The administrative domain is concerned with configurations of firewalls for isolation of domains, and recovery of broken stations of the target domain with backup images. Other similar laboratory projects include intrusion detection experiments in [Lindskog et al., 1999] and live security exercises ("treasure hunt", "capture flag", etc.) on a testbed network in [Vigna, 2003].

Compared to other approaches, dedicated computer laboratories are ideal environments for practical security teaching because security exercises are performed by application of production software in real systems. However, practical education by laboratory measures normally results in high costs:

On the one hand, for financial reason, not every institution can afford dedicated net-

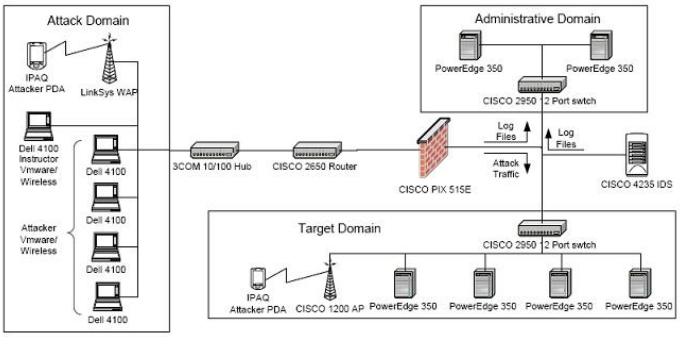

Figure 1.6.: Portable Electronic Network (PEN).

works which require expensive hardware/software investment and intensive administration effort to create, configure, and maintain laboratory environments. Besides administration, extra workforce such as instructors or teaching assistants is needed to prepare, supervise, and evaluate exercises.

On the other hand, most security exercises require system level access to the operating system. If students are allowed a privilege right on their computer, misusing their system becomes possible and easier. This introduces serious security risks and management inconvenience [Vigna, 2003]. For security reason, dedicated networks are normally separated from production networks and accessible only from fixed locations (e.g. instructional laboratories in universities). This implies that security laboratories pitifully fail to benefit a wider range of user groups such as those who cannot access university facilities.

1.5. Thesis Structure

The following chapters are organized as follows:

To help explaining the Tele-Lab IT-Security architecture, technical foundations are provided in **Chapter 2**. General concepts of virtual machines and the User-Mode Linux virtual machine monitor are introduced in this chapter. **Chapter 3** provides an overview of the Tele-Lab IT-Security architecture and highlights main functional components.

The next chapters including Chapter 4, 5 and 6 explain the design and implementation of the Tele-Lab IT-Security architecture in detail. **Chapter 4** introduces methods and procedures to create a virtual laboratory based on the User-Mode Linux virtual

machines and also addresses the user interface and performance issues. **Chapter 5** is devoted to virtual machine management. The management system framework, functionalities, and an administration web interface are described in this chapter. **Chapter 6** addresses security aspect of the Tele-Lab IT-Security architecture. Security isolation at the system and network levels are discussed and related security management solutions are proposed in this chapter.

Chapter 7 introduces applications of the Tele-Lab IT-Security architecture in practical security education and presents typical use processes and case studies. Finally, **Chapter 8** concludes the thesis and figures out future work.

2. Technical Foundations

In order to explain the Tele-Lab IT-Security architecture, necessary technical foundations are provided in this chapter. The most important technical foundation of this architecture is virtual machines. We create virtual machines for providing users with a virtual laboratory environment where they can apply production tools. This chapter first explains general concepts of virtual machines and then introduces User-Mode Linux which is a concrete virtual machine implementation Tele-Lab relies on.

2.1. Virtual Machines

In fact, virtual machine is not a new computing concept. It has been applied in many significant aspects from partitioning mainframe computing system, to implementing cross-platform high-level-language applications, and to designing operating systems for years [Smith, 2005].

2.1.1. Fundamental Concepts

Virtual machines were first developed and used in the 1960s. A famous example is IBM VM/370 (see Section 2.1.4). The function of the virtual machines at that time was to partition a large and expensive framework platform into one or more virtual machines which were similar to the physical machine. Thus, different user groups could run different operating systems on the shared hardware [Rosenblum and Garfinkel, 2005].

Goldberg defined a virtual machine as:

> A hardware-software duplicate of a real existing computer system in which a statistically dominant subset of the virtual processors instructions execute on the host processor in native mode. [Goldberg, 1972]

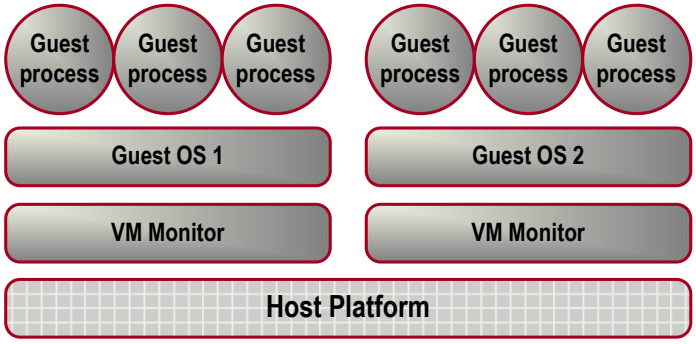

Figure 2.1.: Structure of a virtual machine system.

A more general definition is:

> A virtual machine is a protected and isolated copy of the underlying physical machine. By this means, each user on a virtual machine is given the illusion of having a dedicated physical machine. [Sugerman et al., 2001]

Figure 1 illustrates classic structure of a virtual machine. A virtual machine is simulated by a *virtual machine system (VMS)*. The *virtual machine monitor (VM Monitor)* is referred to the VMS software which creates a virtual machine [Goldberg, 1974]. The system that runs on a virtual machine called the *guest*. The underlying physical machine which supports the VM is called the *host* [Smith, 2005].

2.1.2. Classification of Virtual Machines

Virtual machines are classified into two categories: process virtual machines and system virtual machines in general [Smith, 2005]:

Process Virtual Machines

A *process virtual machine* is a virtual platform which executes an individual process. The virtual machine is created only if a process is created and terminates when the process terminates. The VM monitor that implements a process VM is also named *"runtime software"* (*runtime* for short).

Figure 2.2 depicts process virtual machines. The process VM monitor (runtime) lies on top of an operating system and supports user applications at the Application Binary Interface or Application Programming Interface level.

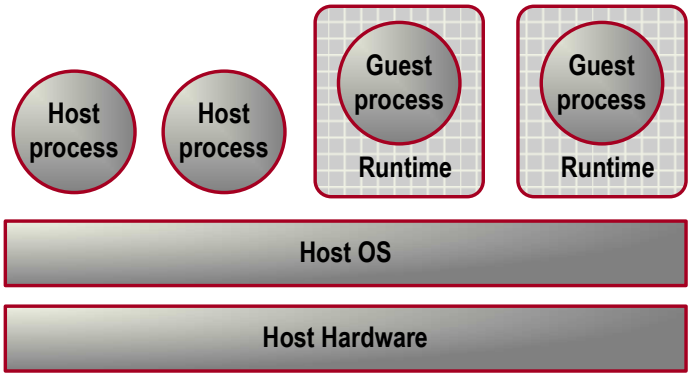

Figure 2.2.: Process virtual machines

In various contexts, process virtual machines are used for replication, emulation, and portability.

- Multiprogramming operating systems provide a replicated process-level virtual machine for each of the concurrently executing applications. By doing so, a process has its own address space, registers, and file structure. This gives each process the illusion of having a complete machine to itself.
- Another role of the process virtual machines is dynamic binary translators that emulate or interpret program binaries in a guest instruction set to host instructions at runtime. E.g. the Intel IA32-EL allows Intel IA-32 application binaries to run on Itanium hardware.
- The process VMs also provide full cross-platform portability. High-Level-Language (HLL) VMs like the Java VM architecture and the Microsoft .NET framework do not directly virtualize any real platform. Instead, HLL code is compiled into general abstract machine code. Each host platform implements a process virtual machine which is able to interpret abstract machine code into native code.

System Virtual Machines

A *system virtual machine* provides a complete, persistent system environment that supports an operating system along with its many user processes. It provides the guest operating system with access to virtual hardware resources, including networking, I/O, and perhaps a graphical user interface along with a processor and memory.

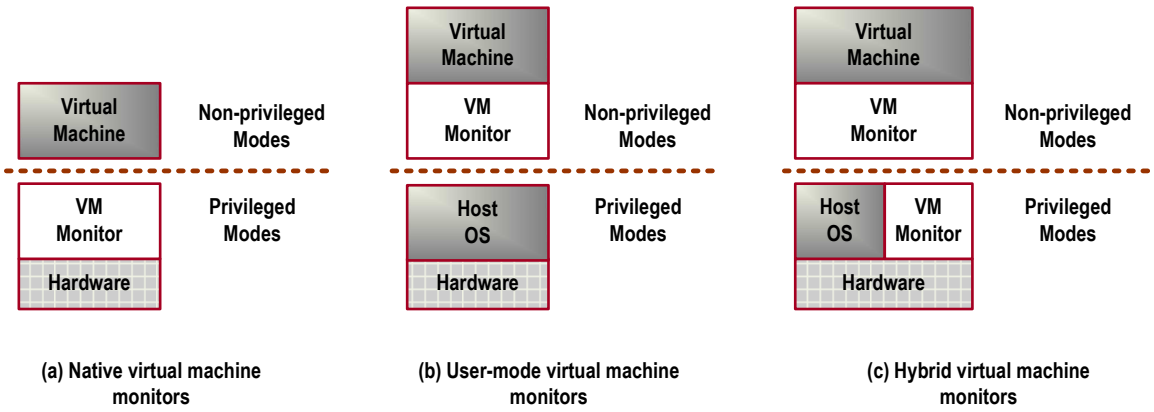

Figure 2.3.: Types of the system virtual machine monitors

The system VM monitor is between the host hardware machine and the guest software. The primary role of the VM monitor is to support multiple, isolated guest operating system environments simultaneously on a single-host hardware platform.

System virtual machine monitors can be further classified into three types: native VM monitors, User-Mode VM monitors, and hybrid VM monitors according to the lower-level platform they are built upon [King et al., 2003, Smith, 2005] (see Figure 2.3).

- *Native VM monitors,* e.g. IBM VM/370[1] and Disco[2], are implemented directly on the machine hardware. This type of the VM monitor is an operating system or kernel with virtualization capabilities to simulate machines. It performs scheduling and resource allocation for all virtual machines in the system and requires drivers for hardware peripherals. The VM monitor runs in the privileged mode, and the guest OS on a virtual machine runs in the user mode.

- *User-Mode VM monitors* such as SimOS[3], User-Mode Linux[4], and UMLinux[5] are built completely on top of a host operating system. This type of the VM monitor runs as a user mode application on a host OS and relies on the host for memory management, processor scheduling, resource allocation, and hardware drivers.

[1]See Section 2.1.4.
[2]Disco `http://www-flash.stanford.edu/Disco/`.
[3]SimOS `http://simos.stanford.edu/`.
[4]See Section 2.2.
[5]See Section 2.1.4.

- *Hybrid VM monitors* not only operate on the physical hardware, they also use the host OS to perform I/O. E.g. VMware workstation[6] and Microsoft virtual PC[7] use the host operating system to access some virtual I/O devices. Therefore, parts of the VM monitor run in the privileged mode and other parts run in the non-privileged mode. A hybrid VM monitor has the advantages from both the Native and User-Mode VM monitors and avoids the performance penalties of User-Mode VM monitors.

2.1.3. Benefits of Virtual Machines

Today, virtual machines have proved to be a successful technology which benefits many aspects of information technologies [King et al., 2003, Robin and Irvine, 2000, Chen and Noble, 2001].

- *Multiplex of Hardware Resource.* Virtual machines allow concurrent execution of different operating systems on a single host machine. Thus, users can conveniently run applications on various operating systems.

- *Supporting IT Education.* Virtual machines have the same properties of the real machines and could run various operation systems. This provides a simple and economical laboratory environment where students are allowed to experiment with different operating systems. Moreover, virtual machines can be used to simulate a network of independent computers on a single host. E.g. in order to support practical teaching of data communications, the Virtual Network Laboratory (VNL) was built by User-Mode Linux servers [McEwan, 2002]. Regular construction and reconfiguration of complex internets in VNL can be done easily with virtual machines. Damage or destruction of virtual machines in experiments would not bring any adverse effect on the underlying host system.

- *Improving Software Testing.* Virtual machines can provide a software environment for testing and debugging operating systems or privileged software, which is more convenient than using a physical machine. For testing particular applications, it is also possible to create virtual machines with desired configurations and resource without presence of real resource on the physical machine.

[6]See Section 2.1.4.
[7]See Section 2.1.4.

- *Security Isolation.* Isolation is the most dominating motivation of developing VM monitors. The VM monitor provides strong isolation among virtual machine instances which are running concurrently on the same hardware platform. If security on a virtual machine is breached or if a failure happens in a virtual machine, the software running on other virtual machines or the host system is not affected. Thus, a single server is allowed to run multiple, un-trusted applications safely. Also, with the isolation, it is possible to operate solid security services such as security logging and intrusion monitoring systems.

- *Performance Isolation.* This is another essential isolation offered by virtual machines. Machine resources, such as CPU time, memory, or disk space, can be allocated between competing users by VM monitors. A virtual machine can be given access to a certain amount of machine resources, and it will not exceed those resources. If a resource-intensive application runs inside a virtual machine, the performance inside that virtual machine would be poor, but it would not affect performance of the host or of other virtual machines.

There are many other benefits from virtual machines. Virtual machines have been applied in virtual hosting service[8] and creating virtual distributed or Grids computing environments with lower costs and reduced complexity [Ruth et al., 2005]. Virtual machine technology is also used to simulate old machines for historical reasons[9].

2.1.4. Virtual Machine Implementations

There are a large variety of commercial and open source virtual machine monitors. This section will introduce some important implementations.

IBM VM/370

The IBM System/370 mainframes were introduced in the 1960s. A research team of IBM built a program called Control Program (CP) which gave the illusion of several standard 370 machines [Canon et al., 1980]. CP could efficiently run a simple single user operating system called Conversational Monitor System (CMS) on the virtual machines. Later, CP was developed into VM/370 which can run CMS, DOS/VS, OS/VS, etc. Today, it has been renamed IBM z/VM[10] and runs Linux VMs on an

[8] *"What are people using it for"?* `http://user-mode-linux.sourceforge.net/uses.html`.
[9] The computer history simulation project `http://simh.trailing-edge.com/`.
[10] IBM z/VM `http://www.vm.ibm.com/`.

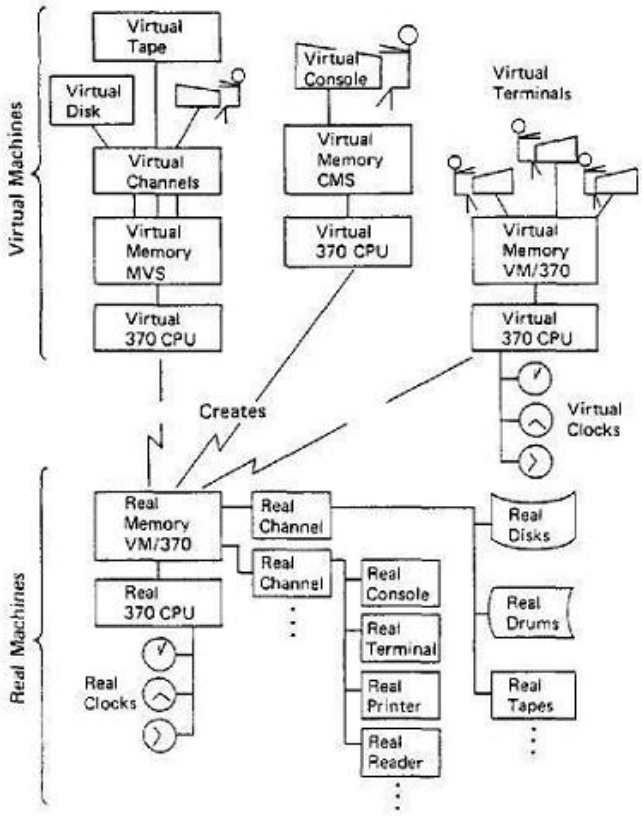

Figure 2.4.: System structure of IBM VM/370.

(picture source: [Canon et al., 1980])

IBM mainframe.

Figure 2.4 is a historical picture that depicted VM/370 in the 1970s. CP controls the real hardware resources, and users interface with the system through virtual machines. Each virtual machine is a logical replica of a 370 hardware system complete with virtual processor, memory, channels, devices, clocks, etc. CP is typically run an operating system in each virtual machine, e.g. CMS, other 370 operating system, and even a virtual CP, concurrently.

Xen

Xen [Barham et al., 2003] was originally developed at University of Cambridge. It aims to efficiently partition a single machine to enable multiple independent clients for providing protection, resource isolation, and accounting. Xen is a native system VM

monitor or 'hypervisor', for the x86 processor architecture. Xen has to partly modify the hosted guest operating systems instead of virtualization of the whole underlying hardware. Xen support is available for increasingly many operating systems including Linux 2.4/2.6, NetBSD, Plan 9, and FreeBSD 5.

An Xen system consists of multiple layers. The lowest and most privileged layer is Xen itself. Xen then runs multiple secure virtual machines, each of which is running a guest operating system. A created Xen virtual machine is called a domain. Domains are scheduled by Xen to make effective use of the available physical CPUs. Each guest OS manages its own applications, e.g. scheduling processes under the resource allocated to the virtual machine by Xen.

VMware

VMware is virtualization software from VMware, Inc.[11] VMware was originally developed at the Stanford University for partitioning x86-based workstations and servers into separate virtual machines, each of which containing its own copy of the OS. Operating systems currently supported by VMware as a guest include Windows /2000/NT/XP, FreeBSD, Solaris, Netware, DOS, and Linux. VMware workstation [Sugerman et al., 2001] is a hybrid VM monitor which allows itself to co-exist with a host operating system. VMware workstation consists of (1) a VM monitor to virtualize CPU, (2) a VM application that accesses I/O through the host OS, (3) a VM driver for transitioning between them. Thus, the VM monitor is allowed to deal with the diversity of PC hardware and to be compatible with legacy software. Currently, Windows NT/2000/XP and Linux can serve as hosts.

Virtual PC

Microsoft Virtual PC [Microsoft, 2004] was originally developed by Connectix Corp. It supports various guest operating systems such as Windows 9x/NT/2000/XP, MS-DOS, OS/2, and Linux. Virtual PC is a hybrid VM monitor and runs as a system service on the host OS (e.g. a Windows 2003 server). It is a virtualization layer which manages virtual machines and provides hardware emulation. Each virtual machine consists of a set of virtual hardware with a guest OS and applications. During virtual machine operations, the VM monitor kernel takes control over the CPU and hardware,

[11] VMware `http://www.vmware.com/`.

creates an isolated environment for the guest applications, and run them at the highest possible performance.

Opensource Virtual Machine Monitors

Following are some popular opensource virtual machine monitors:

- *UMLinux*[12] was developed at University of Erlangen-Nuernberg, Germany for use in fault-injection experiments. UMLinux is a User-Mode VM monitor. That is, the guest OS and applications run as a single process (the guest machine process) on a host Linux operating system. UMLinux provides a higher-level interface to the guest OS that is similar to the host hardware and relies on the host OS to simulate to virtual hardware.
- *Bochs*[13] is a complete emulation of an x86 CPU. It has been ported to several operating systems (including Win95, NT, and Linux) and various platforms such as MAC 68x and PPC). Bochs can also be attached to a network. However, it emulates every x86 instruction in software which makes it very slow.
- *Plex86*[14] is an extensible open source PC virtualization software program which allows users to run multiple operating systems concurrently on the same machine. Plex86 has the similar concept to VMware workstation.
- *Wine*[15] is an implementation of the Microsoft Win32 APIs on top of X-window and Unix. It is a process virtual machine which can run Windows binary applications on the x86 platforms.

2.2. User-Mode Linux (UML)

User-Mode Linux (UML) is an opensource VM monitor developed by Jeff Dike [Dike, 2000, Dike, 2001]. So far, UML has gained great popularity on the Internet. We have a particular interest in User-Mode Linux because it can create fast, secure and manageable virtual machines. UML provides a technical basis for this thesis.

[12]UMLinux is now renamed FAUmachine `http://www.faumachine.org/`.
[13]Bochs `http://bochs.sourceforge.net/`.
[14]Plex86 `http://www.plex86.org/`.
[15]Wine `http://www.winehg.com/`.

UML is a User-Mode VM monitor, because its virtual machines run in the user space of the host. Unlike other virtual machine software, UML is a port of the Linux kernel to Linux, i.e. implements a Linux virtual machine on a Linux host. The UML kernel is a modified version of the normal Linux kernel. It constructs virtual hardware from resources provided by the host kernel and is able to run nearly all of the applications and services that can run on the host.

2.2.1. The UML Principle

Normally, a Linux kernel provides an interface to the underlying hardware (video card, keyboard, hard drives, etc). User programs access hardware by placing a system call to the kernel.

The UML kernel plays a different role. It interfaces to a normal Linux kernel on the host hardware instead of directly talking to the hardware. User programs in UML run natively as if they were running on a normal kernel. If they need to access hardware, their system calls are interpreted and redirected by the UML to the host kernel. In this way, virtual hardware is cleverly simulated by the system call interface, and there is no need to emulate user space code. More exactly, UML does not simulate a generic machine architecture but a specific operating system.

The UML supports nearly all types of devices which are usual present in a normal host. Those virtualized devices include

- Consoles and serial lines. These devices can be attached to a number of various interfaces on the host, e.g. file descriptors, ptys, ttys, pts devices, and xterms.
- Block devices. UML block devices are normally associated with a file on the host. This file contains a filesystem and looks like a block device from inside the virtual machine. A block device file can also used as a swap or as a raw disk. Block devices are also used to access devices on the host, such as CD-ROMS, floppies, partitions, and memory devices (*/dev/mem*). UML block devices can be layered by the Copy-On-Write driver which allows multiple machines to share a filesystem[16].
- Network devices. They can be attached to most types of software network interfaces on the host, such as TUN/TAP, Ethertap, and slip devices. To exchange

[16]See also Section 2.2.3.

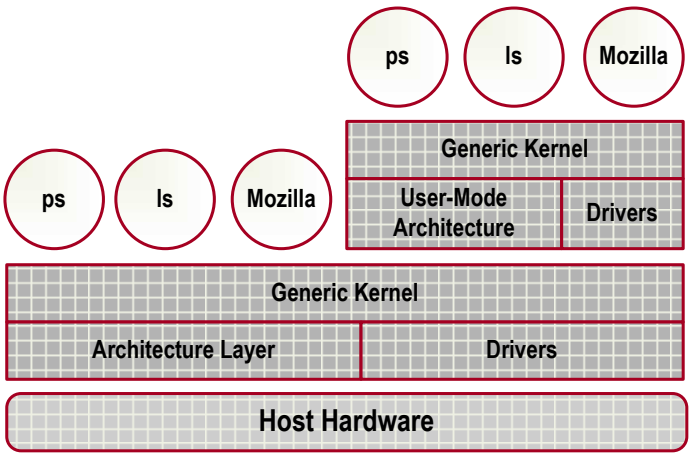

Figure 2.5.: System structure of the User-Mode Kernel.

(reproduced based on [Dike, 2000])

network packets within a virtual machine network, UML uses a switch daemon to forward packets, or use a multicast network to send packets directly to each other. To access the host and the external network, UML uses either ethertap or slip interfaces on the host.

UML has also implemented support for many of other devices such as SCSI, USB, audio, and PCI hardware devices.

2.2.2. System Structure

Figure 2.5 illustrates the system structure of the User-Mode kernel. The bottom layer is hardware with an operating system controlling it. The OS is separated into two layers. The higher layer is generic kernel which contains code independent of specific hardware architecture. The layer underneath the generic kernel consists of an architecture layer and drivers that implement the hardware-specific functions needed by the generic kernel. On the top of the OS are the system call layer and user-level processes like 'ls', 'ps', and 'mozilla'.

The User-Mode kernel is running on the top of the host kernel layer as user-level processes. It is structured exactly like the host kernel. UML kernel also has a generic kernel layer that implements nearly the same code as in the host kernel.

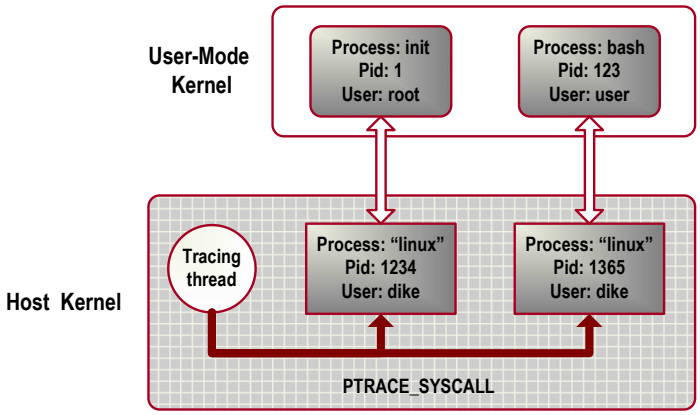

Figure 2.6.: Structure of the User-Mode processes

(reproduced based on [Dike, 2000])

The architecture-specific layer (virtual hardware for user processes inside UML) is implemented by a user-mode architecture layer and drivers.

The most essential part of the hardware that needs to be virtualized is the processor. The work to emulate a processor involves process management, memory management, and fault support.

Hardware Emulation

Normally, a process traps directly into the host kernel when it places a system call. The User-Mode kernel has to convert such switch to real kernel mode into a switch to virtual kernel mode and execute it in the virtual kernel.

The port of system calls is depicted in Figure 2.6. Every process in the virtual machine will get a process in the host kernel. That process is nothing but an execution context. E.g. the process *"init"* is owned by *"root"* and has *"pid 1"* in the virtual machine, but is mapped to a completely different process *"linux"* with *"pid 1234"* and *"uid dike"* in the host.

The tracing thread in the host virtualizes the system calls of the other processes by intercepting them with *ptrace*. This thread is notified when a thread is entering or leaving a system call, and has the ability to arbitrarily modify the system call and its return value. Thus, we can use it to read out the system call and its arguments, drop the system call, and divert the process into the user space kernel code to execute it.

Virtual Memory Emulation

The Linux kernel expects the hardware to provide access to physical memory which is mapped arbitrarily in the virtual memory area of a process or the kernel. The emulation of virtual memory is done by creating a file on the host which is the same size of the physical memory we allocate to the virtual machine. This file is mapped as a contiguous block into the physical memory area of the UML address space.

When a page is mapped into virtual memory, the page in the underlying file is mapped into the appropriate position in the virtual address space. By doing so, each page is treated as a page in the physical memory, and thereby can be mapped into process or kernel virtual address spaces.

Hardware Fault Emulation

Hardware faults are emulated with Linux signals, i.e. system calls. E.g. SIGSEGV is used to emulate a memory fault. It is delivered whenever a process accesses invalid memory. Hardware clock is realized by SIGALRM in the idle thread and SIGVTALRM in all other processes. Device interrupts are implemented with SIGIO. A device operates its IO through a file descriptor. If IO becomes available, SIGIO will be triggered.

2.2.3. Useful UML Features

Following are several useful features of User-Mode Linux, which have been applied in Tele-Lab:

- *The Copy-On-Write* ("*COW*") *Block Driver*. The Copy-On-Write layering capability is implemented to UML block devices. It divides a block device into two files: a read-only shared backing file and a read-write private COW file. Changes to blocks are stored in the COW file and unmodified blocks remain in the backing file. Writing is made to the COW file, and reading is made depending on whether the block to be read has been modified or not. The sparseness of the COW file helps to reduce the disk space requirement. If a number of virtual machines are started from the same filesystem and few changes are made, the overall space consumption is just little more than that of a single UML. Therefore, the capability of a host machine to run virtual machines is effectively

improved. This also brings an administration benefit. By using small COW files, it is much easier to create, copy, and remove filesystem images without touching the part of a large size. If the virtual machine crashes or important data missed in the COW file, this can be restored by rolling back the image to the backing file. In addition, the changes in a COW file can be simply merged to the backing file by a UML tool.

- *The Host File System* ("*hostfs*") is a virtual filesystem which provides access to the host filesystems. It turns the *"vfs"* calls into the equivalent system calls on the host. Mounted a host file system in the virtual machine is similar to mounting any usual filesystem. Either the whole host filesystem or individual directories can be exported to the virtual machine. Accessing a host directory as root inside UML does not necessarily mean a root access to the host files. If UML is started as a normal user, accesses by hostfs are under restriction of the permissions of that user.

- *The Management Console* (*mconsole*) allows access into the kernel from the host. The mconsole provides a simple utility to manage virtual machines. With mconsole, UML can be shutdown, halted, or re-started, and devices can be plugged and unplugged. A management client sends requests to the mconsole driver and receives replies from it through a Unix domain socket. The client is a simple text-mode command-line client and can be used to build more complicated virtual machine management applications. E.g. this feature has been integrated to Tele-Lab's web management interface.

- *The Separate Kernel Address Space Mode* (the "*skas*" mode). By default, a UML process runs under the *tracing thread* mode (the "*tt*" mode). That is, the UML process will get a process on the host. The tracing thread traces and amends the system calls of the UML processes[17]. Unfortunately, such a way to run UML raises security risks and a performance bottleneck. In the tt mode, the UML kernel space is writeable. By access kernel data, a process can break out to the host. Also, every system call results in four context switches (twice than a normal switch), which introduces a performance bottleneck. The "*jail*" mode of UML partly fixes this problem by making the kernel data area readonly. However, the kernel is still be visible and leads to more performance penalties. The "*skas*" mode makes the UML kernel run in an entirely different

[17]See Section 2.2.2.

host address space from its user processes. This solves the security problems because the UML kernel is inaccessible to UML processes. It also improves the performance of UML system calls.

3. Tele-Lab IT-Security Architecture

This chapter presents the design of the Tele-Lab IT-Security architecture. This architecture is based on the work described in [Hu et al., 2004] and [Hu et al., 2003]. In fact, virtual machines have already been applied in training for other subjects such as programming, operating systems, and network technologies. In most cases, however, those virtual machines are used in laboratories only as supplementary machine resource. They have not yet been integrated as a component of the online training/learning systems and included into an automated management frame as Tele-Lab IT-Security does.

Generally, Tele-Lab is a hierarchical architecture that consists of three layers: the host layer, the virtual machine layer, and a management layer (see Figure 3.1).

- The Host Layer. The host is the base of Tele-Lab: it is a hosting server for all Tele-Lab components. The Tele-Lab server provides system resources and environments for running virtual machines.

- The Virtual Machine Layer. This layer implements full-functional virtual machines. They are self-contained learning platforms which provide a security tutoring system and a well-configured work environment for the user (see Section 1.2).

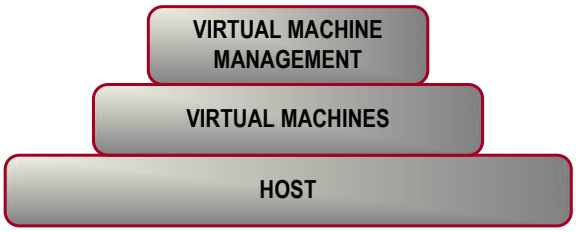

Figure 3.1.: Organization of Tele-Lab IT-Security.

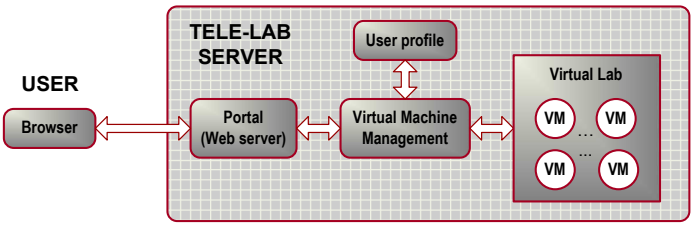

Figure 3.2.: An overview of Tele-Lab IT-Security architecture.

- The Virtual Machine Management Layer. The virtual machine management system is responsible for administration of virtual machines and users. The management system plays a critical role to make Tele-Lab reliable and secure.

This layered design can effective simplify Tele-Lab management and user interfaces, and benefit development and maintenance. E.g. virtual machines are self-contained and changes on them would not affect other layers. It is also possible to apply virtual machines in different types with appropriate management interfaces.

3.1. Architecture Overview

The Tele-Lab architecture is illustrated in Figure 3.2. There are following main components in this architecture:

- The Tele-Lab Portal is the front end of the Tele-Lab server. It provides a web interfaces through which users can access Tele-Lab and virtual machines via networks.
- The User Profile defines a user and records his/her performance. It is maintained by the management system and exported to a virtual machine when a user logs in.
- The Virtual Laboratory hosts a number of light weight virtual machines which are connected to a virtual network. Those virtual machines will be assigned to users as a dedicated learning platform.
- The Virtual Machine Management System manages users and the virtual machines in the virtual laboratory. Its tasks include assignment, monitoring, and maintenance of virtual machines.

3.2. The Tele-Lab Portal

The portal is the web interface of the Tele-Lab server. It provides a VNC applet by which a user can access his/her virtual machine via the browser. This portal is also the front end of the virtual machine management system. Each time when a user logs in/out, it contacts the management system to request/release a virtual machine for the user.

3.3. The Virtual Laboratory

The virtual laboratory is the place where virtual machines are hosted. All the virtual machines in the laboratory are connected to a virtual network and each of them is exclusively assigned to a user as a dedicated machine.

3.3.1. Virtual Machines

It is not realistic to apply heavy virtual machine implementations for creating virtual machines because they need intensive system and administration resources. Instead, we used those light weight virtual machines built with User-Mode Linux (UML)[1]. The virtual machine and host resources are carefully configured so that the Tele-Lab server is able to host more virtual machines at reasonable performance.

As shown in Figure 3.3, each virtual machine is a self-contained learning and work platform[2] which provides an IT security tutoring system and a well-configured user work environment. The IT security tutoring system is a local web server on the virtual machine, which familiarizes users with IT security concepts and techniques. The user work environment can be seen as a virtual security laboratory in which users are able to gain hands-on experience from practical exercises.

3.3.2. The Security Tutoring System

The security tutoring system consists of a tutor, a knowledge repository and exercise scripts, as well as the user profile (see Figure 3.3).

[1]User-Mode Linux is a software substitution of a real Linux operating system. See Section 2.2.
[2]Early mentioned in Section 1.2

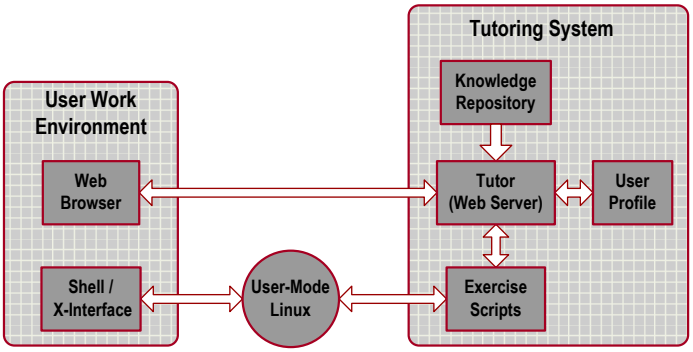

Figure 3.3.: Configuration of the virtual machine.

The Knowledge Repository

The knowledge repository is a database which stores teaching materials. Those materials are categorized into the following three types of sections: introduction to security concepts, tutorials of security tools, and exercises.

1. *Concept Sections.* Concept sections introduce declarative knowledge of an IT security subject, e.g. concepts of cryptography or authentication. They are presented web pages which contain text, figures or animation demonstrations.

2. *Tool Sections.* For each subject, we selected related security software tools. The tool sections present short and practically tutorials of those tools. Those tutorials consist of a series of instruction pages to teach users how to apply them in real situations. In most case, screenshots or dynamical demos of the tools are included.

3. *Exercise Sections.* The exercise sections present the knowledge about how a security task is performed. They are designed as scenarios in which users must take step-by-step interaction with the tutoring system. Normally, exercises are implemented by interactive scripts and users are required to apply the programs introduced in tools sections on the virtual machine.

Generally a security subject is organized as a chapter which consists of closely related sections of three types. E.g. as shown in Table 3.1, we have a chapter on password-based authentication. This chapter includes several sections which cover from basic password hashing concepts to the real password cracking exercises. The repository provides Meta information for teaching content organization. Each section has a

Table 3.1.: Structure of the Password-based Authentication chapter

Concepts	Password hashing (DES and MD5) The "passwd" file in Linux Password selection criteria
Tools	The *passwd* command in Linux The *John-the-Ripper* Password Cracker
Exercise	Cracking random-generated passwords with John-the-Ripper

description file which tells the profile of the section such as the title, type and pages it has, etc. Similarly, the description file of a chapter specifies the sections it has. Thus we can build a hierarchy of the content structure and generate appropriate links between pages or sections. This feature is helpful for assembling or rearranging contents without touching individual pages or sections.

The Tutor

The tutor is responsible for presenting teaching contents, navigating users in learning, and managing exercises execution.

- Content Presentation. The tutor presents materials in the repository. It searches the repository for the section or chapter description files. Based on those descriptions, the tutor links sections and pages in a structured manner and shows them as web pages.

- Learning Navigation. The tutor navigates a user in his/her learning. It tells a user where to start a chapter and where to continue at the end of a section, and provides help information where necessary. It also records user's results into his/her user profile and provides related statistics information about the performance.

- Exercise Execution. The tutor manages exercises on the virtual machine. It prepares security tasks and evaluates results of the user by running a series of exercise scripts.

Exercise Scripts

In implement step-by-step interaction with the user, security exercises contains a number of interactive scripts which are written in Perl, PHP, or Shell scripting languages. Those scripts normally apply tools or commands on the virtual machine to prepare tasks or check results. In an exercise, those scripts are invoked by the tutor in the following steps:

- The tutor invokes scripts to prepare the laboratory environment and to create questions or tasks for a user.
- The user applies related security tools and performs exercises on the virtual machine in order to resolve questions.
- The user creates desired results and submits them to the tutor. The tutor will use scripts to examine what the user has done and to notify the user of the evaluation results.

The User Profile

The user profile is used to record user performance, i.e. to keep track of a user's knowledge at every stage in a learning process. The user profile is maintained by the management system and imported into the virtual machine when a user logs in. The user profile contains two types of information.

- Personal Information. It includes user names, languages, account related information, etc[3]. The user profile is classified into one of three categories ("common user", "administrator", and "IT student"). Different sets of subjects are offered to those categories, e.g. IT students can take advanced subjects while common users are only given general subjects which focus on everyday skills or require less background knowledge.
- Performance Records. Information in performance record includes a list of the sections the user has finished, time spent on each section, etc. Those data are used to generate a summary of user performance and to present statistics on the current status.

[3]See Figure 7.1 in Chapter 7.

3.3.3. The User Work Environment

The user work environment is implemented by a virtual machine. The virtual machine is simulated by the User-Mode Linux and arranged to the baseline installation. Each virtual machine consists of a virtual operating system kernel and a basic file system (a Debian installation image), which provide a real and self-contained laboratory environment for exercises. On the virtual machine, basic system programs, graphical user interfaces and various open-source security tools are installed and appropriate configurations such as the user account and various software settings are predefined. Thus, the user can concentrate on the exercises conveniently and avoid unimportant setup steps. The local user interface on the virtual machine is simple: a user accesses the tutoring system by using a local browser; in order to perform exercises, the user can apply security programs or tools via a shell or an X-window interface. When necessary, a user is able to switch to a privileged mode and perform system-level operations.

3.4. Virtual Machine Management

The goal of the virtual machine management system is to ensure the Tele-Lab server continuously runs in a reliable way. The major management functions include:

- Virtual Machine Administration. The virtual machine management system is responsible for creation, assignment, and recovery of virtual machines. E.g. it assigns or reclaims a virtual machine when a user logs in or off.
- Virtual Machine Monitoring. It monitors the system status of each virtual machine and detect errors or failures. If critical errors are found on a virtual machine, it will be recovered. The interrupted learning process will be resumed on a new virtual machine.
- User Monitoring. User activities on a virtual machine are also monitored. If abnormal use of the virtual machine is found, it will be reclaimed to avoid unnecessary occupation of virtual machine resources.
- The Tele-Lab Administration Interface. It is a graphical web interface from which an administrator can operate the Tele-Lab server and monitor the system performance of the host and virtual machines.

More detailed description of virtual machine management is presented in Chapter 5.

3.5. Security Management

Security is an important factor which we must deal with in every aspect of the Tele-Lab architecture. Generally, each user is allowed a privilege right on his/her virtual machine. This situation introduces a serious security problem: users might convert their virtual machine into an attack station and corrupt the Tele-Lab system or compromise production networks. The mission of security management of Tele-Lab is to implement effective security isolation and to prevent misuse of virtual machines.

- System Security Isolation. Tele-Lab can run the UML virtual machine kernel in a safe mode. This mode enforces strict access control between a virtual machine and the host by isolating their processes spaces from each other. System security isolation can effectively prevent system level attacks from the virtual machine.
- Network Security Isolation. The function of the network security isolation is to constrain user actions within the scope of his/her virtual machine. Firewalls can be applied to provide access control and to prevent misuse of the virtual machines which is intended to attack production systems on the network level.
- Other General Security Protection. Besides hosting virtual machines, the Tele-Lab host is also a Linux web server. Web security issues such as encryption, authentication, access control, and availability should be also taken into account.

4. The Virtual Laboratory

This chapter describes installation, user interface, resource allocation, and benchmarks of virtual machines in Tele-Lab IT-Security. Each virtual machine is used as a learning and laboratory platform in Tele-Lab. The virtual laboratory is a central place where virtual machines are created, connected to the network, and distributed to users. As mentioned in Chapter 2, there are many various virtual machine implementations. All of them differ from each other in functionality, performance, resource usage, platforms, and so on. To find applicable virtual machine software and create effective virtual machine installation are challenging and effort-intensive. We need to investigate mainstream virtual machine software to find a functional and manageable one. We have to create virtual machines with appropriate configurations and user interfaces. We also need to benchmark those virtual machines and the Tele-Lab system to determine their performance and usability.

4.1. Requirements for Virtual Machines

The most important goal of the virtual laboratory is to provide substitution of the real machines by virtual machines. To create virtual machines, we must consider the following issues:

- Real environments. The user work environments (including operating systems, accounts, software, and configuration) provided by virtual machines must be as close to those of real machines as possible.

- Lightweight platforms. Resources of a physical host are fixed. Virtual machines must be compact and each of them should not demand a large amount of the CPU, memory, and disk space consumption. Thus, one host could support as many virtual machines as possible.

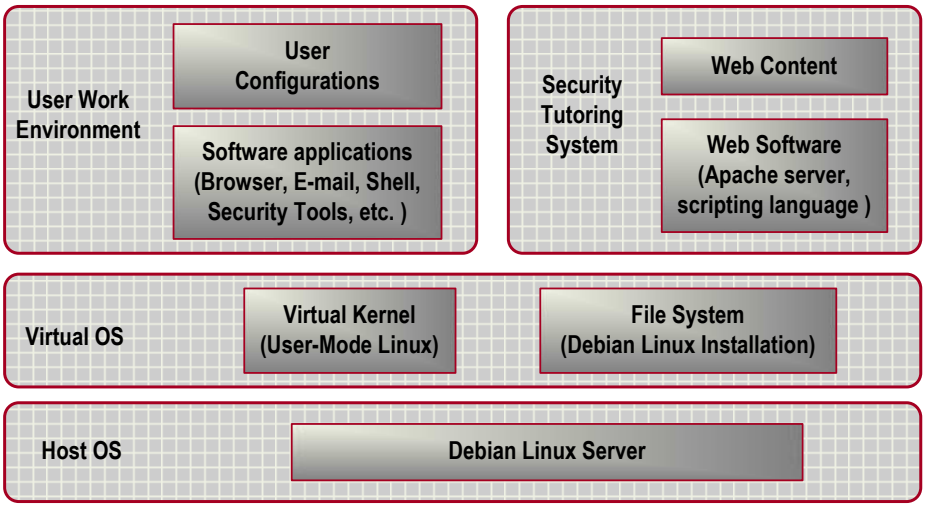

Figure 4.1.: Structure of the virtual machine installation.

- Reasonable performance. We have to make a trade-off between the number of the virtual machines we desired to run and the performance on each virtual machine. In general, a virtual machine user should not notice obvious performance difference from a real machine. Also, we need to examine performance of the user interface regarding its usability for different bandwidths.

4.2. Virtual Machine Installation

Figure 4.1 depicts the structure of a virtual machine installation which consists of three main components: a virtual operating system, a user work environment, and the tutoring system.

4.2.1. The Virtual Operating System

Basically, each of three types of system virtual machine monitors[1] can be used to create virtual machines. There are following reasons which persuade us to choose User-Mode Linux instead of other types of virtual machine monitors.

- As a user-mode virtual machine monitor, User-Mode Linux simulates a virtual Linux operating system on a native Linux operating system. This type of virtualization makes virtual machines resource-friendly.

[1]Native, user-mode, and hybrid system virtual machine monitors (see Section 2.1.2)

- User-Mode Linux virtual machines are created as normal processes on the native operating system. They can be managed and controlled as ordinary software applications.

- User-Mode Linux is a simple and open implementation. Its virtual machines can be easily integrated into a special-purpose application (e.g. Tele-Lab IT-Security) or customized for individual needs.

As shown in Figure 4.1, the host is a standard Debian Linux server which runs a User-Mode Linux (UML) monitor (version 2.4.26-3um). The virtual machine created by UML is a virtual operating system which consists of a virtual kernel and at least one virtual disk partition. We use the following code to start a UML virtual machine from the command-line:

```
$ /usr/bin/linux ubd0=/uml/uml_rfs_file con=xterm mem=
64M eth0=tuntap, , ,192.168.0.1 umid=uml_vm_1
/*Options:
ubd0: the file contains a UML root filesystem.
con: the virtual terminal of the UML machine.
mem: pseudo-physical memory allocated for the UML machine.
eth0: the virtual network interface.
umid: the unique identity of the UML machine.
*/
```

Figure 4.2 shows the booting screen of a UML virtual machine. The booting procedure of a virtual machine is very similar to that of a native Linux machine. The only difference between both is that the virtual machine is started from a user terminal instead of on real hardware.

UML only creates a virtual Linux 2.4.26-3um kernel (see Figure 4.2). Like any native Linux system which runs on disk partitions, the UML virtual machine needs at least one disk partition as a root filesystem. The virtual disk device is simulated with a file in the host OS. The following commands are used to generate a typical *"ext2"*[2] root filesystem in a file.

[2]The second extended high performance Linux disk filesystem.

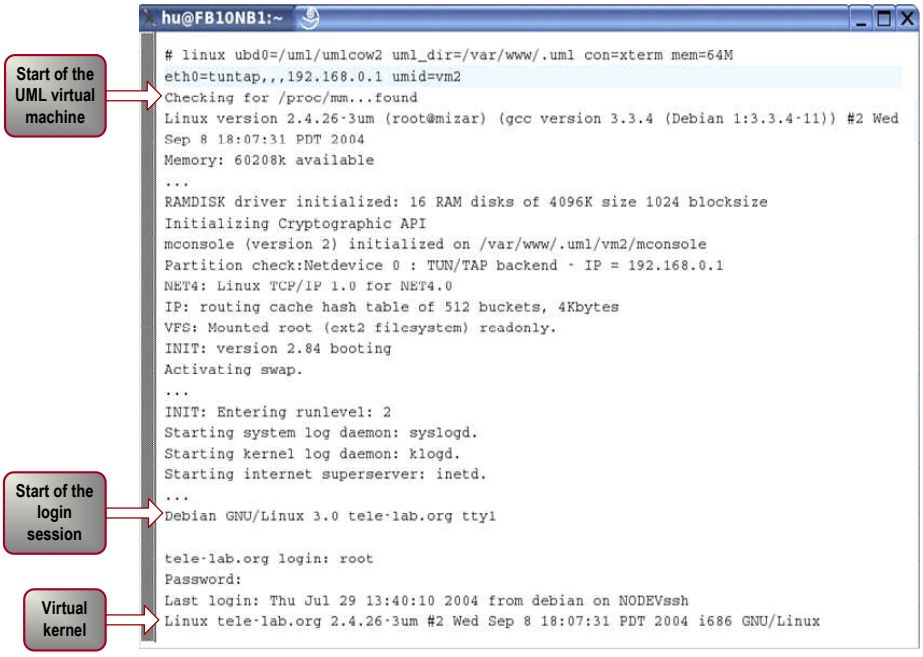

Figure 4.2.: The booting screen of a UML virtual machine.

```
/* create an empty root filesystem file(300MB) */
$ dd if=/dev/zero of=uml_rfs_file count=300 bs=1M
/* format it as ext2 filesystem */
$ /sbin/mke2fs -Fq uml_rfs_file
```

The generated filesystem is empty. We need to create a Debian installation on the filesystem so that the UML kernel is able to run a basic Debian Linux with necessary programs (including drivers, shell, essential system and graphical programs). The Debian installation for virtual machines is special compared with a normal installation because it must match a virtual kernel instead of a real kernel. The way to make the installation of this kind is to hack a normal Debian installation procedure:

The first step is to extract an installer, *root.bin* from the Debian rescue floppy[3]. The next step is to boot the UML machine from this installer:

```
linux mem=64M initrd=root.bin fakehd fake_ide
ubd0=uml_rfs_file con=xterm eth0=tuntap, , ,192.168.0.1
```

[3]The floppy image is available from any Debian mirror site, for example: `http://http.us.debian.org/debian/dists/woody/main/disks-i386/current/images-2.88/rescue.bin`.

Figure 4.3.: Directories of the virtual machine installation.

Then, a normal installation process starts. The special thing with this installer is we have to insert a UML virtual kernel directory into the kernel library directory before the installer configures device drivers[4]. After the installer sets up a basic system on the filesystem, additional application packages can be installed by an online package update tool, *APT*. Figure 4.3 is the screenshot of a virtual machine desktop. The file manager window shows the directory structure of a typical virtual machine installation.

Another way to install a root filesystem is to download an archive of the basic Debian packages[5], unpack it, and create a base filesystem installation in the root filesystem file.

4.2.2. The User Work Environment

The user work environment is the place where a user finishes his/her exercises. It requires necessary software installation and appropriate configuration.

1. **Support Software:** Basic system programs include a number of shell programs such as editors and administration tools. They are normally installed when we

[4]This is because the UML kernel version is unlikely to match the installer's standard kernel version, and the installer is unable to find the modules directory for the current kernel.

[5]Available from `ftp://ftp.debian.org/debian/dists/woody/main/disks-i386/base-images-current/basedebs.tar`.

create a UML virtual machine installation. For access to the local tutoring system, a simple graphical user interface (X-window and the *XFCE* window manager) and a standard web browser (*Mozilla-firefox*) are installed on the virtual machine, so that a user can access the local tutoring server. Other software applications are installed depending on the requirements of specific teaching topics and exercises. E.g.

- *GunPG*[6] is an opensource PGP program which is required when teaching encryption.
- *John-the-Ripper* is a Linux password cracker which is used in teaching Password-based Authentication.
- The *Exim* mail server, *OpenSSL*, and a *Mozilla-thunderbird* email client are applied for teaching Secure Email.

2. **Configuration:** Appropriate configuration makes the user interface friendly and convenient. When a virtual machine is assigned to the user, the user account and related configuration are automatically applied. E.g. "*bob*" is a default account on the virtual machine, and using this account is enough in most case. If privileged operations are needed, the user is able to switch to a root mode easily by clicking a shortcut icon. Another important configuration is the email account. Some exercises need it for exchanging email between the tutoring system and the user. Related account settings for a local mailing server and a Mozilla-thunderbird client are also prepared in advance, so that the user can concentrate on the exercise without taking care of unimportant setups.

4.2.3. The Security Tutoring System

The security tutoring system is a local web server on the virtual machine as mentioned in Section 3.3.2. The web contents of the tutoring system are a series of HTML or PHP pages and a number of CGI programs such as Perl scripts, shell scripts, and executable programs. So, web-related tools such as an *Apache* web server as well as the *PHP4* and *Perl* interpreters are installed on each virtual machine.

It is not necessary that each virtual machine has a copy of this web directory. Instead, web contents are hosted in a directory of the host and shared by each virtual

[6]The GNU Privacy Guard (GunPG) `http://www.gnupg.org/`.

machine through the host filesystem (see Section 2.2.3). Thus, web contents can be conveniently synchronized without touching the local installation of individual virtual machines. Also, this directory is used to handle user profiles which are shared between the portal server (on the host) and the tutoring system (on the virtual machine). The user profile can be copied to the virtual machine through this directory when a user session starts and will be copied back to the host when the user session is over.

4.3. User Interfaces

Each virtual machine must provide a local graphical user interface (see Section 4.2.2), so that the user can access the local tutoring server by using a Mozilla browser and perform exercises with shell or X-window programs. Meanwhile, the host needs to provide a user interface for remote access to the virtual machine. Through this interface, the user is able to run programs on the virtual machine. In general, there are two approaches to implement such a user interface.

1. Special-purpose command interfaces: it is possible to design a special interface for each security program which a user might apply. Thus, commands from the user will be sent to an agent on a remote machine via this interface. The agent interprets and executes commands on behalf of the user. Then the results will be sent back to the user and displayed in the browser.

2. A thin user interface: it allows a user to access the graphic desktop of a remote computer and avoids interpreting commands of specific applications. Only the user's keystrokes and mouse input will be sent to a server on the remote machine, and any change on the remote desktop will cause updates of the client view.

The first approach is a widely applied in software training. It can handle many command-line applications such as database or high-level language programs [Cao et al., 2002]. However, it can not deal with programs with a graphical interface, and it is not efficient to design command interpreters for all involved programs. The advantage of the second approach lies in the fact that it provides a universal interface for all applications (those of both command-line and graphical kinds) and there is no necessity to understand the commands of each application. Considering the situation that various security programs (including graphical ones) will be applied on the virtual machine, we chose to create a thin user interface based on the Virtual Network Computing technology (see Section 4.3.1).

4.3.1. Virtual Network Computing (VNC)

Virtual Network Computing (VNC) [Richardson et al., 1998] is a remote display system. VNC allows its users to view a computing desktop environment not only on the machine where it is running, but from anywhere on the Internet. VNC uses a dedicated remote desktop access protocol, the Remote Frame Buffer (RFB) protocol. Based on the RFB concept, input from a remote client is collected and sent to a VNC server. Updates of the server are encoded, compressed, and then sent back to the client. Since it works at the frame buffer level, it can be applicable to all operating systems, windowing systems, and applications e.g. X-window on Unix and MS Windows. The protocol operates over any reliable transport such as TCP/IP and SSH tunnels. VNC implements a thin-client interface because it makes very few requirements of the viewer. In this way, clients can run on the widest range of platforms.

TightVNC [7] is an enhanced version of the original VNC. It has a better image compression algorithm which can improve the VNC performance over WAN. It also provides security feature like SSH tunnelling. TightVNC has been used in Tele-Lab IT-Security for creating the thin interface of the virtual machine.

A TightVNC server is installed on each virtual machine. It exports the X-window desktop to a client via Port 5900. One important advantage of the thin user interfaces is that the client programs can be implemented as simple as possible. So the TightVNC client can be written in Java and run as an applet in the browser, and the user end does not need any extra client software. To support the applet client, the TightVNC server also listens on Port 5800 as a tiny web server. The default page of that web server contains an applet (See Table 4.1). By visiting this page from Port 5800, a browser can download the applet and run it automatically. Figure 4.4 shows the VNC interface on the portal page.

4.3.2. The VNC Performance

The VNC performance is crucial to user interfaces. It would not be accepted if the client cannot smoothly receive desktop updates over networks. Yang et al. [Yang et al., 2001] reported the benchmarks of main stream thin-client systems such as Microsoft Terminal service, Critrix MetaFram, Sun Ray, and VNC. Performances of those systems were measured and compared at ISDN, DSL/T1, and LAN network

[7] Tight VNC `http://www.tightvnc.com/`.

Table 4.1.: Code of the VNC applet page

```
<HTML>
<TITLE>
$USER's $DESKTOP desktop ($DISPLAY)
</TITLE>
<APPLET CODE=VncViewer.class ARCHIVE=VncViewer.jar
WIDTH=$APPLETWIDTH HEIGHT=$APPLETHEIGHT>
<param name=PORT value=$PORT>
<param name=ENCPASSWORD value=1Bffe8343876ae70>
<param name="Show controls" value=No>
<param name="Offer Relogin" value=No>
<param name=SERVER value="tele-lab.hpi-web.de">
$PARAMS
</APPLET>
</HTML>
```

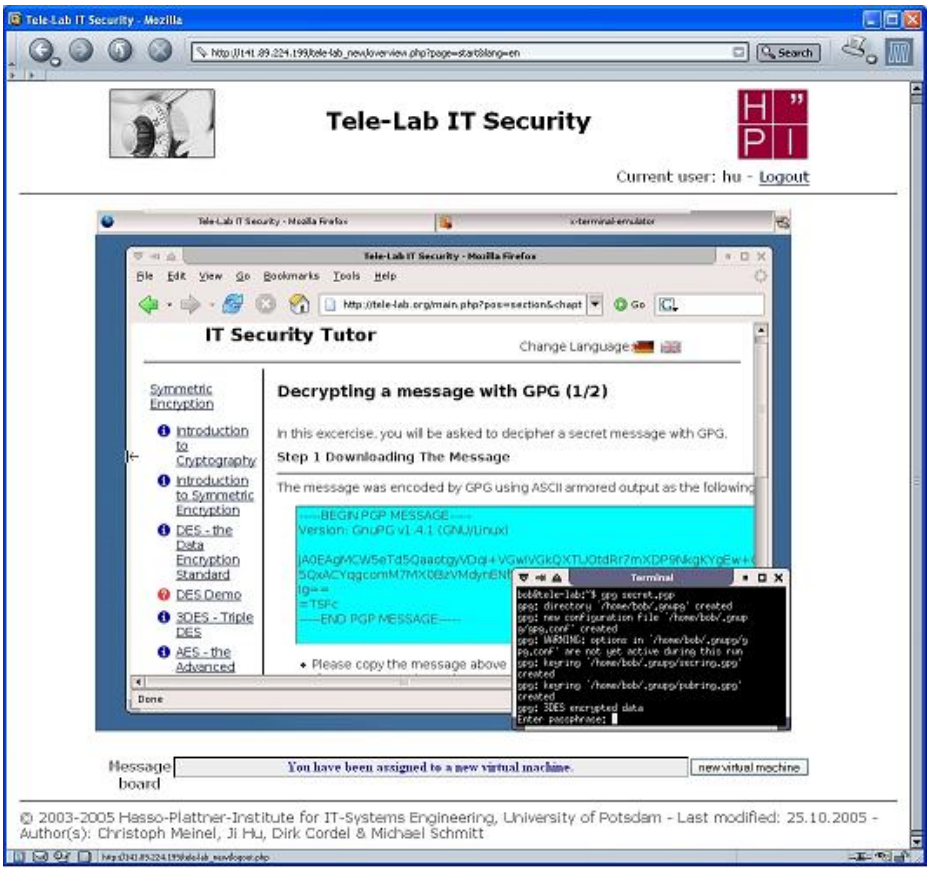

Figure 4.4.: The VNC interface to the virtual machine.

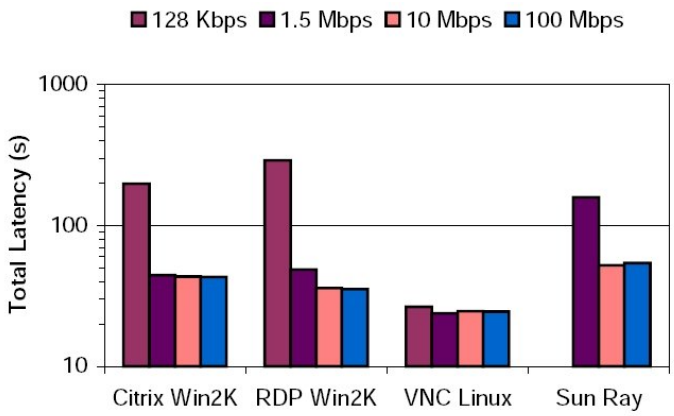

Figure 4.5.: Total latency of the thin-client systems.

(Picture source: [Yang et al., 2001])

bandwidths. Their result indicates that thin-client systems can smoothly reflect VNC updates of most programs, except for video applications, at each bandwidth. Figure 4.5 shows the total latency of those thin-client systems at various bandwidths. The total latency is the time required to download a certain number of pages from a web server. It was measured by forcing the browser to continuously load 109 pages. A per-page latency of less than one second is desirable to ensure that the user's browsing experience flow is not interrupted. This figure indicates that VNC performs extremely well. E.g. it maintains the same low latency across all bandwidths and outperforms the other systems. VNC is an effective user interface solution for Tele-Lab IT-Security because most applications running on the virtual machine are general graphical software such as consoles, web browser, and email programs.

4.4. System Resource Allocation

The purpose of the system resource allocation is to isolate resource and performance between competing virtual machines. This can be done by assigning each virtual machine access to a certain amount of processor, memory and disk resources of the host when it is started. So its resource consumption will not exceed those resources. Effective resource allocation is also useful to improve performance of virtual machines and save system resources.

4.4.1. Processor Resource Allocation

If a hostile user misuses his/her virtual machine by generating a large number of processes to exhaust CPU resource, it will cause a denial-of-service attack on the host and other virtual machines. Isolation of processor resources among virtual machines and other host processes is very necessary for preventing this kind of attacks. Processor resource allocation and isolation are supported by the following ways or their combinations:

1. Limiting the number of the virtual processors assigned to a virtual machine. That is, if a virtual machine is given two processors, it has no more than two processes running on the host no matter how many processes it is trying to generate.

2. Adjusting *nice*[8] value of virtual machine processes and degrade their priority. Then virtual machine processes will not be allocated more processor time than that for the host processes.

3. Enhancing the host processor capability. Powerful processors can effectively cope with the performance impact of virtual machines and improve the overall performance. E.g. the Tele-Lab host is equipped with a 2.8 GHz Intel IV hyperthreading processor and runs a *Linux 2.4-smp* kernel on it. The *SMP* (Symmetric Multiprocessor) kernel provides a high performance dual-processor computing environment by hyperthreading.

4.4.2. Virtual Memory Allocation

The size of "physical" memory of a virtual machine can be specified when it is booted. The "physical" memory is not the real size of physical memory space of the host. Instead, each virtual machine is allocated this amount of memory space from the virtual memory on the host. The virtual memory is emulated by a temporary file on the host. The User-Mode Linux kernel creates this file with the same size as the "physical" memory [Dike, 2000]. It is then mapped into the physical memory address space of the host. This virtual memory uses physical memory only when it needs,

[8] "*nice*" is a Linux command to execute a command with lower priority, i.e. be "nice" to other users. Niceness has a range of -20 (highest priority) to 19 (lowest priority). Normally, non-root users can only have lower priority (higher number)

otherwise it will be swapped out to the disk. Thus, we can provide virtual machines with enough "physical" memory space by creating a large virtual memory area instead of using the real size of physical memory of the host.

The directory of the virtual memory files is mounted as a dynamic RAM based filesystem, the temporary filesystem (TMPFS). TMPFS [Snyder, 1990] uses local memory for filesystem reads and writes, which is typically much faster than reads and writes in a normal *ext3* filesystem. Given enough RAM and swap space, TMPFS can effectively speed up the access of virtual memory files.

We need to run thirty virtual machines on the Tele-Lab host. Each of them is assigned 64 MB "physical" memory. Intuitively we need 1.8 GB (30 × 64 MB) RAM memory space for virtual machines. To use the virtual memory scheme, we reserved totally 6 GB virtual memory which consists of 2 GB RAM, 2 GB swap space for the same size of RAM, and 2 GB swap space for 30 virtual machines. Two separate 2 GB swap partitions were applied to provide 4 GB space. Then, the virtual memory file directory ("*/tmp*") was mounted as a TMPFS:

```
mount tmpfs /tmp -t tmpfs -o size=4096M
```

After starting all thirty virtual machines, we found 1.8 GB swapped virtual memory was created in "/tmp". It is exactly the amount of space allocated for virtual machines. For running those virtual machines, however, only about 809 MB RAM has been indeed consumed. This amount is only 42% of total space needed by virtual machines.

4.4.3. Virtual Disk Resource Allocation

In order to run virtual machines, we need to give each of them a disk image of an installed Linux filesystem[9]. We create this image by using a special file named "backing file". Changes to the backing file are stored in a small-sized Copy-On-Write (COW) file. Each virtual machine is booted from a COW file and all virtual machines share the same backing file[10]. COW files can be created, copied or deleted like any other ordinary file on the host. Therefore, virtual machines can be treated as normal applications, i.e. they can be easily started, killed, or recovered on demand. Moreover, sharing the backing file is helpful to reduce memory usage of virtual machines.

[9]See Section 4.2.1.
[10]See Section 2.2.3.

In Tele-Lab IT-Security, small images are assigned to virtual machines. This helps reduce the demand of system resources and improve performance. It will also benefit management procedures of virtual machines. E.g. it is fast to boot or halt a virtual machine from a small backing file. In the Tele-Lab implementation, the size of the backing file is only about 300 MB. It contains a typical Linux software installation. This backing file was also set as "read-only" to prevent crash of the filesystem.

If copy-on-write (COW) files are used, only one copy of the backing file is needed. Thus, a large number of disk space can be saved. E.g. the COW files used in Tele-Lab normally are very small (20 B - 200 KB). If we start thirty virtual machines on the host, we do not necessarily need 9000 MB (30 $\times$ 300 MB) disk space. Instead, about 306 MB real disk space (including space of thirty COW files and one backing file) is actually used. This is merely 3,4% of the total disk space of thirty virtual machines.

4.5. Virtual Machine Performance

Besides functionality, performance is important to usability of virtual machines. In general, performance difference between virtual machines and ordinary physical computers should be as little as possible. The following benchmark has been done to determine how effectively a virtual machine simulates a real computer in terms of system performance.

4.5.1. Performance Benchmark

We used *lmbench* [McVoy and Staelin, 1996] to measure virtual machine performance. Lmbench is an open-source software suite for operating system microbenchmarks. It provides a set of portable programs for use in cross-platform comparisons. Usually, performance issues are caused by latency problems, bandwidth problems, or their combinations. The idea of lmbench is to run a set of small microbenchmarks to measure system latency and bandwidth of data movement among the processor and memory, network, file system, and disk.

The machines to be benchmarked include the host, virtual machines, and other computers for comparison. Their system specifications are summarized in Table 4.2. Tele-Lab server ("P IV 2800") is the host. Its Linux kernel was specially patched to enable the User-Mode Linux "SKAS" mode, which improves virtual machine kernel

Table 4.2.: System specification of the performance benchmark.

System	Hardware	Kernel	Filesystem
P IV 2800	Pentium IV 2.8GHz (Hyper-threading) 2 GB RAM 50 GB IDE Hard-disk	Linux 2.4.26/i686-smp	EXT3
Virtual machine	Virtual processor 64 MB "physical" memory 300 MB virtual disk partition	User-Mode Linux 2.4.26	EXT2
P II 350	Pentium II 350MHz 64 MB RAM 10 GB Disk	Linux 2.4.18	EXT3
P Pro 167	Pentium Pro 167 MHz	Linux1.3/i686	EXT2
P Pro 133	Pentium Pro 133 MHz	SunOS 5.5.1	UFS

performance and security. We ran thirty virtual machines on the host. The 64 MB "physical" memory is simulated by the virtual memory of the host. The virtual disk partition is provided by a COW file which only stores differences from a shared root disk image. "P II 350" is a reference machine, i.e. a native PC with decent installation. The specifications and benchmark data of two native systems ("P Pro 167" and "P Pro 133") came from the lmbench database [McVoy and Staelin, 1996].

Lmbench version 3.0-a3 was applied in the benchmark. It was compiled and installed on each machine. We first started all virtual machines on the host, and then ran lmbench on the host and on the virtual machine respectively. Depending hardware conditions, it took lmbench about from 30 minutes to one hour to finish the benchmark on each machine. It then produced a long list of detailed benchmark results.

4.5.2. Benchmark Results

Among the benchmark output, we are particularly interested in those results related to the performance of processes, memory, and filesystems. Selected data which are presented in the following text include process creation time, memory read/write bandwidth, and file system latency.

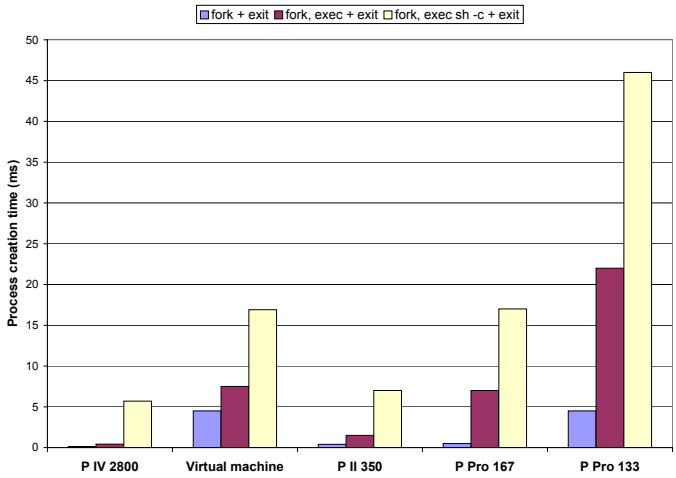

Figure 4.6.: Process creation time of the performance benchmark.

Figure 4.6 shows the time used to create processes on each platform. "*process fork+exit*" measures the time it took to split a process into two identical copies and have one of them exit. "*process fork+exec+exit*" measures the time it took to create a new process and have that new process run a new program. "*process fork+sh -c+exit*" measures the time it took to create a new process and have it run a new program by asking the shell to find that program and run it. The process creation time of the virtual machine is very close to the machine of "P Pro 167". I.e. the process performance of virtual machines is nearly at the level of that of an Intel Pentium Pro 167 computer. This performance looks not so satisfying compared with those of modern computers because the User-Mode virtual machine (including User-Mode Linux) has to spend extra time to make system calls for processor simulation. However, use of User-Mode virtual machines has its positive side. One of the User-Model Linux features is resource-friendliness. I.e. one host can effectively run more virtual machines at a relatively low resource expense. Although part of process performance is sacrificed, we gained efficiency.

Memory read and write rates are shown in Figure 4.7. The memory read bandwidth was measured by allocating 8 MB memory, filling it with zeros, and then measuring the times taken to read it as a series of integers. The memory write bandwidth was measured by allocating 8 MB memory, filling it with zeros, and then measuring time taken to write that memory as a series of 4 byte integers. The memory performance of the virtual machine is extremely impressive. Except for the host, it performs much better than other native machines. The memory bandwidths of virtual machine are

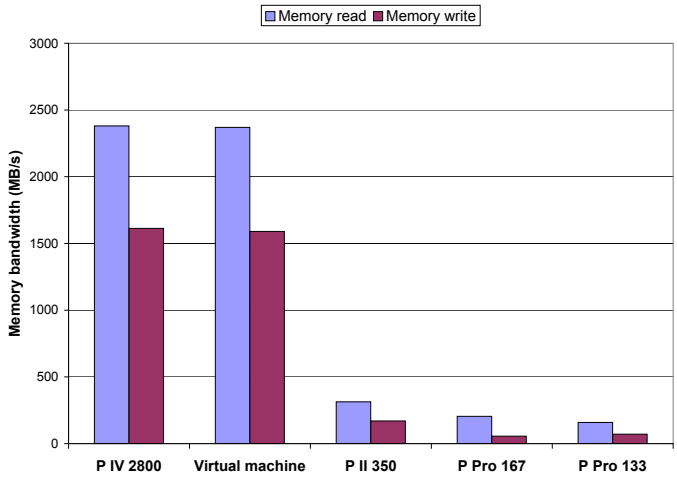

Figure 4.7.: Memory bandwidth of the performance benchmark.

about 6.6 times higher than those of "P II 350" and nearly equivalent to those of the host. This is because User-Mode Linux is able to utilize most resources of the host and therefore inherit a good performance.

Figure 4.8 shows the result of file system latency. Lmbench created 1,000 zero-sized small files in the current working directory and then removed those files. The time of the file creation and removal was measured respectively. Similar to the memory bandwidth, the filesystem performance of the virtual machines is also very satisfying and its file I/O is one time faster than that of the native computer of "P II 350".

The benchmark results of virtual machines indicate that high performance host helps improve performance of the virtual machines. With reasonable configurations, a virtual machine can be seen as a machine between a Pentium II 350 computer and a Pentium Pro 167 computer. Intuitive experience also shows that virtual machines can smoothly run most general applications from common system operations to web browsing. Therefore, virtual machines proved to be an effective and efficient computing-resource replacement of physical machines for Tele-Lab IT-Security.

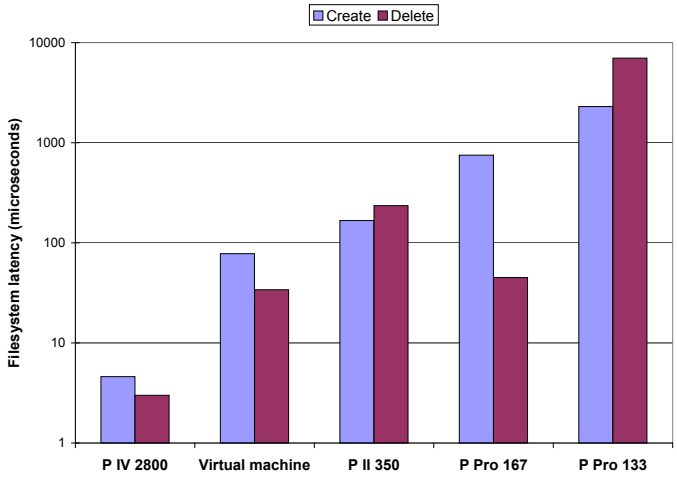

Figure 4.8.: Filesystem latency of the performance benchmark.

5. Virtual Machine Management

This chapter is based on the work described in [Cordel, 2004] and [Hu et al., 2005]. Virtual machine management is responsible for assignment and maintenance of the virtual machines. Virtual machine management has an important role in Tele-Lab IT-Security because it guarantees virtual machines work in a reliable manner. In fact, without effective virtual machine management, it is impossible to operate Tele-Lab IT-Security in practice. User-Mode Linux provides some virtual machine management functions of its own. But those functions are very limited, and only simple controls are implemented. On the other hand, existing sophisticated management tools from other virtual machine software are dedicated to their own products and cannot be adapted or extended for the virtual machines in Tele-Lab IT-Security. This is the reason why we need to develop our own management system.

5.1. Related Work on Virtual Machine Management

Mainstream virtual machine software programs such as Virtual PC and VMware have developed sophisticated management utilities. With those utilities, administrators can conveniently manage virtual machines through a graphical or web interface.

- Virtual PC provides an administration web site which is used to control a host system and virtual machines [Microsoft, 2004]. All management operations that create, configure, and control virtual machines are performed through that site (see Figure 5.1). The controls provided by the administration web site include commands to turn on, shutdown, remove, reset, configure, and backup virtual machines.

- VMware[1] also provides a series of management tools that allows an administrator to monitor the state of virtual machines and the host, control the virtual

[1] VMware GSX Server Administration Guide `http://www.vmware.com/support/gsx3/doc/`.

Figure 5.1.: A Virtual PC administration web site.

machines on that host, modify virtual machine configurations, create and delete virtual machines, and answer questions and acknowledge messages posed by virtual machines. Figure 5.2 shows a status monitor page. From this page, the administrator can get high level details about all the virtual machines or detailed information of each virtual machine such as virtual devices, configuration options, and a summary of recent events. The virtual machine menu provides control commands such as power on, suspend, resume, reset, power off, configure, and delete a virtual machine.

So far, management utilities provided by User-Mode Linux include *mconsole* and *UMLd*. They are relatively primitive compared with the management utilities above and only essential control functions are provided. Nevertheless, mconsole and UMLd are open and low-level interfaces and can be easily integrated into particular applications. Based on both utilities, we are able to develop dedicated and user-friendly management utilities for the Tele-Lab IT-Security virtual machines.

5.1.1. UML Management Utility: Mconsole

As early mentioned in Section 2.2.3, mconsole is a low-level management interface. It allows a client to access User-Mode Linux kernel from the host. Table 5.1 lists main

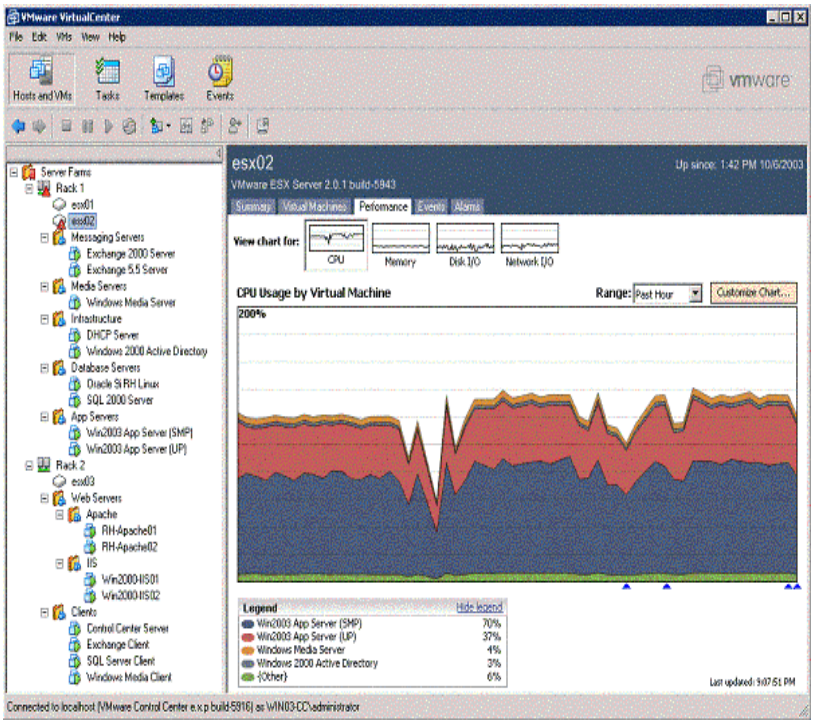

Figure 5.2.: VMware virtual machine management utility.

commands supported by mconsole and their respective management functions.

Mconsole also provides an event notification interface. Through this interface, programs inside a virtual machine can send messages to the host. The notification channel is implemented by Unix sockets. The mconsole client creates a socket on the host. Then the handle of that socket is passed to the virtual machine as an endpoint of the channel. The "*/proc/mconsole*" file acts as the other endpoint inside the virtual machine. The mconsole client on the host is listening for the message on the socket. Anything written to that socket will be forwarded to the the mconsole client. When a virtual machine has booted to a certain stage or a particular event inside happens, it can send a message out through this channel. The host can proceed depending on the message. The feature has been effectively used to monitor the state of virtual machines in Tele-Lab IT-Security.

Table 5.1.: Mconsole management functions.

Command	Function description
version	Read the UML version information. This command can be used to check that a UML is running.
halt and reboot	Shut the machine down by force, with no syncing of disks and no clean shutdown of user space.
cad	This invokes the *Ctl-Alt-Del* action on init. What exactly this ends up doing is up to /etc/inittab. Normally, it reboots the machine.
stop/go	"stop" puts the UML in a loop waiting for mconsole requests until a "go" mconsole command is received. "go"resumes a UML after being paused by a "stop" command.
config/remove	"config" adds a new device (e.g. a block or Ethernet device) to the virtual machine or queries the configuration of an existing device. "remove" deletes a device from the system.
proc	Monitor the runtime state of a virtual machine. It will return the contents of the corresponding */proc* file inside the UML.

5.1.2. UML Management Daemon: UMLd

UMLd[2] is a daemon to control and manage User-Mode Linux instances running on a host system. It provides an interface on a TCP socket and allows simple clients to manage individual virtual machines. In fact, UMLd is a TCP-wrapper of mconsole. By this daemon, a client can connect to the host server via networks or the Internet and control virtual machines remotely.

5.2. Requirements for Virtual Machine Management

To effectively manage virtual machines, we must consider following requirements:

- Performance monitoring. Mconsole only provides controls on virtual machine. It is necessary to collect the system states of virtual machines and the host in the real time. Based on their CPU, memory, disk, and network usage information, we can evaluate performance of each virtual machine and the host, and thus optimize the resource allocation scheme.

- Reliability. After a virtual machine is assigned to a user, he or she is allowed a *superuser* privilege for security exercises. This suggests that a virtual machine might be corrupted if errors are caused by an unskilled user, or the privilege is misused by a hostile user. Therefore, one of the requirements of virtual machine management is to detect and recover failed virtual machines as well as resume the interrupted learning process in time.

- Security. We must prevent either the attacks on the Tele-Lab system or misuse of virtual machines to compromise production networks. Chapter 6 will particularly discuss security isolation and protection of virtual machines.

- User interfaces. On the one hand, users of virtual machines should be informed of the latest system states of their virtual machines and be notified when special events happens. On the other hand, we need a friendly web management interface for administrators to monitor and control virtual machines.

[2]UMLd `http://uml.openconsultancy.com/umld/`.

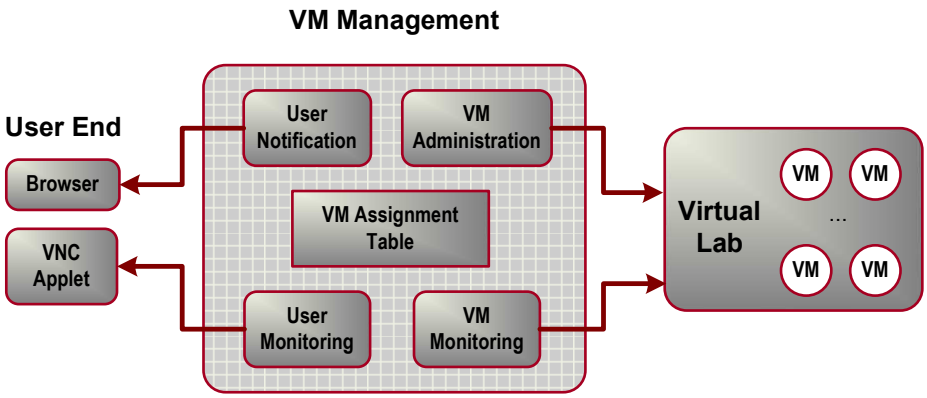

Figure 5.3.: Virtual machine management framework.

5.3. Virtual Machine Management Framework

In order to meet the management requirements above, we developed a virtual machine management framework and implemented it for Tele-Lab IT-Security. The management frame consists of five main components. They include a virtual machine assignment table and the modules for virtual machine administration, virtual machine monitoring, user monitoring, and user notification (see Figure 5.3). The virtual machine assignment table is a data structure which records actual states of each virtual machine. The virtual machine administration module is responsible for starting, stopping, or recovering virtual machines in specific circumstances. The virtual machine monitoring module detects and reports critical errors of the virtual machines. The user monitoring module monitors user activities on the virtual machine and detects abnormal events. The user notification module informs a user about events on his/her virtual machine.

5.3.1. The Virtual Machine Assignment Table

Information in the VM assignment table indicates which virtual machines have been assigned and to whom, which are free, and which are found failed and need recovery. The VM administration module updates this table each time when a virtual machine is assigned to or reclaimed from a user, or critical errors of the virtual machine are found. The VM assignment table is important to the management framework, because the information in the table is the base to assign and manage virtual machine resources.

The structure of the assignment table is simple. As shown in Table 5.2, each active

Table 5.2.: An example of the virtual machine assignment table.

VM Name	VM No.	Mode	Source IP	User
Tele-Lab_VM	1	assigned	141.89.224.219	schmitt
Tele-Lab_VM	2	recovered	non-IP	non-user
Tele-Lab_VM	3	assigned	136.199.55.8	cordel
Tele-Lab_VM	4	free	non-IP	non-user
...	...	...	...	...
Tele-Lab_VM	30	free	non-IP	non-user

virtual machine has an entry in this table. The data in the entry include virtual machine name and number, current mode, source IP, and current user. "VM Name" and "VM No." are used to identify a virtual machine. The private IP address of the virtual machine can be mapped from the virtual machine number. E.g. the third virtual machine will be assigned an IP address: "192.168.0.102" (see Section 6.4.2). "Source Address" is referred to the IP address where the user machine connects from. "User Name" specifies the user, to whom the virtual machine is assigned. "Mode" indicates the actual state of the virtual machine.

Each virtual machine is running in one of three possible modes: "*free*", "*assigned*", and "*recovered*". The mode transition of a virtual machine is illustrated in Figure 5.4. A virtual machine is "free" when it is started and has not assigned to any one. If none of free virtual machines is available, a user has to wait until free virtual machines are available. A virtual machine is "assigned" to a user when he or she logs in. Then it becomes an exclusive workstation for that user. The "recovered" mode is a transitional state and triggered in two cases: (1) if a virtual machine is found failed, it will be marked as "recovered" and restarted in the background; (2) if a user logs out, his/her virtual machine will be immediately reclaimed and recovered. Then, it will change to "free" mode until recovery finishes and a new virtual machine instance becomes ready.

5.3.2. Virtual Machine Administration

The VM administration module is implemented based on mconsole. Basic control functions of the VM administration module include assigning, reclaiming, and recovering virtual machines

- Assigning a virtual machine. when a user logs in, the administration module

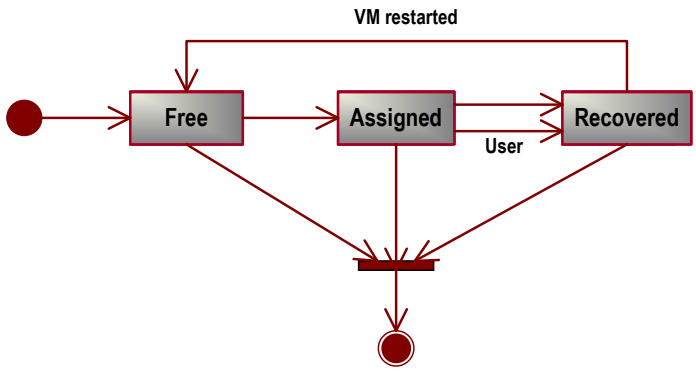

Figure 5.4.: Mode transition of a virtual machine

searches the assignment table for a free virtual machine and assigns one to this user if possible. Then the mode of the virtual machine changes to "assigned" and the corresponding entry in the assignment table is updated.

- Reclaiming a virtual machine. When a use logs out, his/her virtual machine is reclaimed and returns to "free" mode after a recovery procedure (see the next item).

- Recovering a virtual machine. A virtual machine is recovered to default settings by killing all processes it possesses and restoring its image with a default COW copy. Then it is booted again. Recovery is triggered when a user logs out or exceptional events happen. E.g. in case that the failure of a virtual machine is detected, it is reclaimed and recovered in order to prevent its user from working on a defected platform. A new virtual machine is then assigned to this user and he/she can continue exercises on it in a short time. Besides that, a user can request a new virtual machine if he/she notices a critical failure but no action is taken by the administration module.

The administration module is also responsible for preparation and maintenance of the virtual laboratory environment, e.g. initializing or terminating the virtual laboratory and setting up network connections, etc.

- Initializing virtual machines. When Tele-Lab IT-Security is started, the administration module first creates virtual machine image copies according to the number of virtual machines which Tele-Lab provides. Each image uses a COW file of an original backing filesystem. The administration module also determines the arguments of each virtual machine, such as memory size, network

interfaces, the mconsole socket, host directories etc. Then all virtual machines are started as "free" machines and connected to a virtual network in the batch mode.

- Setting up connections. Virtual machines are plugged in a virtual network. Each of them is assigned a private address, e.g. 192.168.0.100. By default, the access to the virtual network from outside is not allowed for security reason. If a virtual machine is assigned, it will be accessible by mapping its private address to a public access point by port forwarding[3]. This access point allows the user of the virtual machine to access the VNC desktop of the virtual machine.

- Terminating virtual machines. When Tele-Lab IT-Security is stopped, all virtual machines are forcefully shutdown at one time. The administration module kills all virtual machine instances, disables their connections, and cleans their COW files.

5.3.3. Virtual Machine Monitoring

The VM monitoring module monitors the state of each virtual machine and reports errors of the virtual machine to the administration module. There are two approaches to implement monitoring:

1. Installing a local monitor on each virtual machine. It sends state information of the virtual machine to the host in the real time. An advantage of this approach is that it can report precise and detailed state information, but the problem is that it can be easily interrupted by the virtual machine user.

2. Periodically polling virtual machines and testing essential services. Those services are necessary and any failure of them would prevent the user from continuing his/her work on the virtual machine. E.g. the VNC is a must to access the virtual machine and email service is a prerequisite for exercises. Hence, if any test of those services fails, the virtual machine is proved to be defective. This approach can effectively find out critical errors in general situations though some errors might be missed.

Compared with the first approach, the second one is more reliable and simpler to implement. Therefore, it has been applied in detecting virtual machine failures in

[3]See Section 6.4.1.

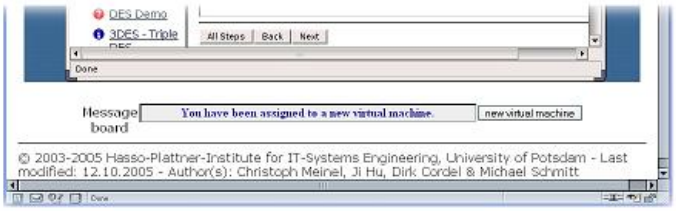

Figure 5.5.: The message frame and the "new virtual machine" button.

Tele-Lab IT-Security. Even in some cases, some of critical errors might be missed, a user can directly request a new virtual machine by clicking a button on his/her web page (see Figure 5.5).

After a virtual machine is started or recovered, it will notify the host of its readiness. Hence, another task of the VM monitoring module is to listen on the mconsole notification socket for readiness messages of the virtual machines. If one readiness message comes, the monitoring module will inform the administration module to update the mode of the matched virtual machine.

5.3.4. User Monitoring

The Tele-Lab IT-Security host can only run certain number of virtual machines because the system resources are limited. The user monitoring module is used to reduce unnecessary occupation of the virtual machine resources. The idea of user monitoring is to monitor user activities on the virtual machine and measure user's idle time. User activities are monitored by tracking user's keyboard and mouse input on the virtual machine. This can be implemented by either checking the system log of the virtual machine, or installing a monitor in the VNC client. It is problematic to check the system log because we need to extract user input events from the log file and we can not prevent the user from changing its content. We prefer monitoring the VNC client because it is straight and reliable: we can directly insert desired monitoring functions based on existing exception procedures of input events in VNC client.

As shown in Figure 5.6, we modified part of the original VNC client code and inserted a detection hook which records the time of latest keyboard or mouse input events on the VNC client window. In this way, the user's idle time on a virtual machine can be measured. If the value of the idle time exceeds a predefined threshold (e.g. half hour), we can conclude that the user won't continue his/her learning on the virtual

machine. The administration module will therefore reclaim his/her virtual machine and let him/her log out.

5.3.5. User Notification

The user notification module is necessary for informing a user of special events in real time. It reports the user on the latest status of his/her virtual machine. E.g. if any critical failure is found, the notification module will inform the user that his/her virtual machine needs recovery and he/she can continue exercises on a new virtual machine. However, the portal server (a web server) only passively responds to user requests because the HTTP does not support connection-persistent sessions like Telnet. Notification messages would never reach the browser if no requests are raised by the user. In order to capture messages in time, a small frame is embedded in the browser window as a message board (see Figure 5.5). It refreshes itself each 15 second to get the updates of the messages about virtual machine events.

5.4. Administration Web Interfaces

The administration web interfaces provide a friendly management interface to control virtual machines and monitor their system status. They are also used to test performance and determine the optimal number of the virtual machines the host resource can support. The administration web interfaces include an administration console and a system status monitor.

5.4.1. The Administration Console

The administration console provides control functions to start/stop the Tele-Lab system and to create/recover virtual machines. As shown in Figure 5.7, the "System Starten" page is used to initialize the Tele-Lab system. On this page, an administrator can create and start virtual machines with desired name and quantity. The "System Halten" button is used to stop the Tele-Lab system by halting and cleaning all virtual machines (see Figure 5.8).

The administration console has a page which shows updates of the virtual machine assignment table. From this page, we can get information of each virtual machine.

```
// Handle events.
public void keyPressed(KeyEvent evt) {
   processLocalKeyEvent(evt);
   // insert the detection hook.
    tellactivity();
}

public void mousePressed(MouseEvent evt) {
   processLocalMouseEvent(evt, false);
   // insert the detection hook.
    tellactivity();
}
. . .

// definition of the detection hook.
public void tellactivity() {
   try {
      // create a message
       Integer myInt = new Integer (VncViewer.port);
       String output = myInt.toString() + "\n";
       byte[ ] udpMsg = output.getBytes();
       // send the UDP message to the user monitoring
       // module of the host.
       InetAddress addr = InetAddress.getBy-
          Name(VncViewer.server);
       DatagramPacket packet = new Datagram-
          Packet(udpMsg, udpMsg.length, addr,
          VncViewer.udpport);
       DatagramSocket datagramSocket = new DatagramSocket();
       datagramSocket.send(packet);
       datagramSocket.close();
    }
. . .

}
. . .
```

Figure 5.6.: The detection hook in the VNC applet (in VncViewer.java).

Figure 5.7.: The "System Start" page.

Figure 5.8.: The "System Stop" button.

Figure 5.9.: The "Recover Virtual Machine" button.

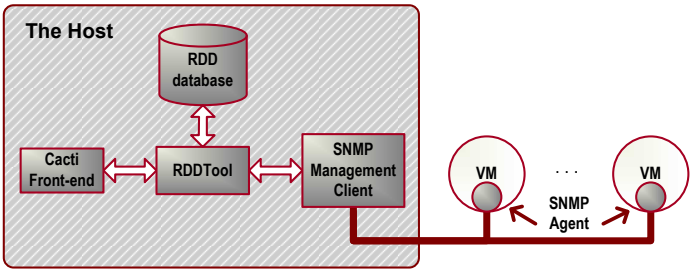

Figure 5.10.: Structure of the system status monitor .

It is also possible to reboot a questionable virtual machine by clicking the "recover" button on the right side of its entry (see Figure 5.9).

5.4.2. The System Status Monitor

The system status monitor provides system status information of the host and each virtual machine. Details about their processor, memory, disks, and network performance are presented by graphs. The idea the system status monitor is to periodically collect status data from the host and virtual machines. Then those status data are processed and performance-related statistics are generated. Finally a set of graphs is created from the performance data.

Figure 5.10 illustrates the system structure of the system status monitor. It integrated three opensource software packages: *Net-SNMP*, *RRDTool*, and *Cacti*.

- The Net-SNMP[4] is a program to manage network nodes by the *Simple Network Management Protocol* (SNMP).

[4]Net-SNMP `http://net-snmp.sourceforge.net/`.

- RRDTool[5] (Round Robin Database Tool). It is a tool to store, retrieve, and graph time-series data (e.g. SNMP statistics) in a Round Robin Database (RRD). RRD is a special database technique. It can efficiently store and process time-related data with a limited size of space[6].
- Cacti[7] is a web management front-end which works with RRDTool. Cacti provides a fast poller, advanced graph templating, multiple data acquisition methods, and user management features.

A SNMP agent is installed on each virtual machine. This agent collects status information about CPU, memory, and network traffic from the MIB (Management Information Base[8]) of the virtual machine. The SNMP management client runs on the host. It periodically queries MIB data from each agent and saves them into a RDD database. RDDTool generates statistics reports in both numerical and graphical forms based on the data stored in the RRD database. Cacti provides a web interface to configure monitored nodes and to present related statistics and graphs (see Figure 5.11 and 5.12).

[5]RRDTool `http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/`.

[6]Round robin works with a fixed amount of data and a pointer to the current element. The data space can be seen as a circle. When the current data is read or written, the pointer moves to the next element. After all the available space is used out, the process automatically reuses old locations. Thus, the dataset will not grow in size and therefore requires no maintenance.

[7]Cacti `http://www.cacti.net/`.

[8]The management information base stores objects which describe the state of a managed node.

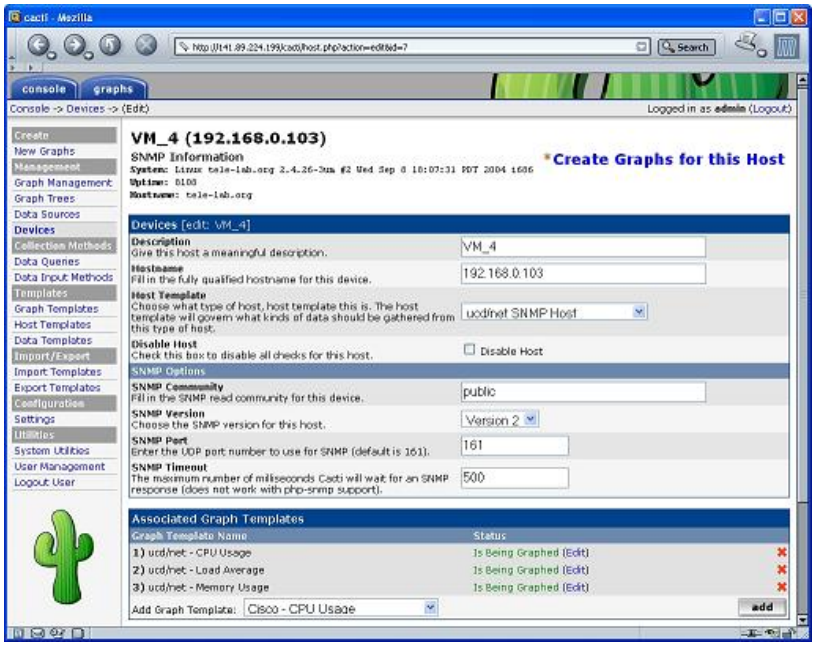

Figure 5.11.: Node configuration of the system status monitor.

Figure 5.12.: Graphical view of the system status monitor.

6. Security Management

Security is important consideration in the design and implementation of the Tele-Lab IT-Security architecture. Tele-Lab IT-Security has a special security requirements compared with other online learning or tutoring systems. In many security exercises, users are allowed a privilege right on the virtual machine. This introduces a security risk that users might convert their virtual machine into an attack station and corrupt the Tele-Lab system or compromise production networks. Therefore, on the one hand, we must prevent misuse of virtual machines and protect the Tele-Lab system; otherwise it will not be allowed online. On the other hand, we have to tolerate security risks on the virtual machine for those security exercises that typically need privileges. Security management must deal with such contradictory needs as well as guarantee both functionality and security of Tele-Lab IT-Security. In response to this challenge, we implemented security isolation for virtual machines, which allows necessary accessibility to virtual machines while constrains risks in a secure scope.

6.1. Security Policy on Virtual Machines

Each virtual machine is a laboratory platform where security experiments are performed. In order to finish security exercises, a user has to operate in a privileged mode. E.g. he/she needs to change system configurations in some exercises such as firewall configuration or security scanning, which is normal done by a privileged user. Some programs applied in the exercises must run in a privileged mode. E.g. in the *"Password Cracking"* exercise, the cracker program must be able to access the *"passwd"* or "*shadow"* file which is only accessible by a super user. Therefore, the security policy on the virtual machine can be stated as:

> A user is authorized to access any system resource and to perform any operation inside his/her virtual machine. The user can switch to a super-user mode if any privileged operation is needed.

This policy implies that the virtual machine is an insecure platform and subject to misuse or damages by its user. This is because given a privilege, unskilled users might corrupt virtual machines by mis-operation, or it is possible that Tele-Lab IT-Security or production networks are compromised by misuse of the virtual machines.

6.2. Security Requirements for Tele-Lab IT-Security

The corruption of the virtual machine is not a serious security problem because it can be handled by the virtual machine management system. What we concern are those risks which compromise the Tele-Lab system and production networks by misuse of the virtual machines. Security risk are often caused at the following three interfaces between a virtual machine and the outside environments.

1. *On the system level.* Each virtual machine is a software application which comprises a number of processes which are allocated the host resource and scheduled by the host operating system. Therefore, virtual machine processes run inside the same space of the host operating system. Communication between those processes and the host takes place at the system level. The security concern at this level is whether the virtual machine monitor (i.e. User-Mode Linux) is secure. If there is any vulnerability present in User-Mode Linux, it might be exploited by the virtual machine processes to break out of the virtual machine space. This is the most critical risk which compromises the security of the host.

2. *On the virtual network level.* The second communication interface is the virtual network. Each virtual machine is plugged in a virtual switch device [1] as a node of a virtual local area network (LAN). Thus, virtual machines and the host can access each other by any network communication protocol. The risks at this level include various network attacks such as sniffing, spoofing, or denial of service which are launched from one virtual machine on any of the others or the host.

3. *On the production network level.* Virtual machines are by default disallowed to access public networks. After a virtual machine is assigned, it becomes accessible over the production networks in order to provide a VNC interface for its user.

[1] See Section 2.2.1

If outgoing connection is misused by a malicious user, theoretically, any form of network attacks on production systems from a virtual machine could be possible.

To effectively control those security risks and protect the Tele-Lab system, security management must address the following requirements:

1. *System security isolation.* The host operating system should be protected from any intrusion from the virtual machine processes. Strict access control must be enforced between a virtual machine and the host to isolate their process space from each other.

2. *Network security isolation.* All user actions must be constrained within the scope of his/her virtual machine on the network level. We need effective mechanisms to prevent network attacks by access control between a virtual machine and outside nodes such as other virtual machines and production systems.

3. *General security protection.* The Tele-Lab host is also a Linux server which provides web service. Web security issues such as encryption, authentication, access control, and availability are important as well. However, we will not discuss those issues in this chapter because general web and Linux security have been profoundly discussed in various literatures such as [Garfinkel and Spafford, 2001], [Cheswik et al., 2003], and [Stallings, 2003]. The major focus of the security management is security isolation which is specially related to the Tele-Lab architecture and virtual machines.

6.3. Security Isolation at the System Level

The idea for security of User-Mode Linux is to use "root jail" [Dike, 2001]. Privilege right on a virtual machine can be safely assigned to the user if this virtual machine runs as an ordinary user application on the host. Thus, the root user of the virtual machine is jailed in the virtual machine and no privileged access from a virtual machine to the host operating system is allowed. Successful root jailing requisites that the system calls from the virtual machine on the host must be secure. This is because the hardware of the virtual machine is emulated by port of system calls and the most possible way to break out from a virtual machine to the host is to make use of system calls. Therefore, we must assure the virtual machine processes have no ability

to execute arbitrary system calls on the host even with a root right on the virtual machine.

User-Mode Linux virtual machine processes by default run in the tracing thread mode (the "*tt*" mode) [2]. This mode is problematic because the virtual machine kernel shares the same address space with its user processes and the kernel space is writable. By accessing kernel data, a process could possibly break out to the host by forging system calls. Security isolation requires that the virtual machine kernel memory is protected against modification by users pace and any critical information of the kernel must stay inside the kernel. In this way, user processes of the virtual machine have no possibility to forge system calls that they want and to escape from the jail.

This kind of security isolation can be implemented with the *Separate Kernel Address Space* mode (the "*skas*" mode)[3] which is a secure mode to run User-Mode Linux. It requires to apply a patch on the host kernel. This patch implements a separate address space scheme: the virtual machine kernel runs in an entirely different host address space from its user processes. The virtual machine kernel binary and data are totally invisible to its processes and to anyone logged in to it. This makes virtual machine kernel data secure from tampering by its processes. Therefore, "root jail" with support of the "skas" mode can effectively protect the host and implement system security isolation.

6.4. Security Isolation at the Network Level

The purpose of security isolation on the network level is to constrain all the user actions within the scope of the virtual machine. This can be achieved by applying control on the connections between the virtual machine and the outside. Effective access control must enforce the following policies:

1. *Local connections and local network services are allowed on a virtual machine.*
2. *A virtual machine is not allowed to initialize any network connection to the other virtual machines on the host.*
3. *A virtual machine is allowed to accept or respond to the connections for the VNC service only if it is an assigned machine.*

[2]See Section 2.2.3
[3]See also Section 2.2.3

4. *Except for those connections mentioned in Policy 3, a virtual machine is not allowed to launch any forms of connections for the Internet.*

The idea of to implement access control is to use a firewall called "*iptables*". This firewall is installed on the host and responsible for two major control functions: IP address reuse and packet filtering. With appropriate rules, iptables can enforce the desired control on network traffic of the virtual machines.

6.4.1. The iptables Packet Filter

Iptables is a firewall tool based on Netfilter[4]. Netfilter is the system compiled into the Linux kernel, which provides hooks into the IP stack. Iptables is one of loadable modules which use those hooks to perform operations on packets. In order to perform packet filtering, network address translation, and other packet mangling, iptables provides a generic table structure for defining control rule sets. Iptables manages three kinds of tables:

1. The "*filter*" table is specifically designed to filter packets.
2. The "*nat*" table is designed to perform network address translation for packets.
3. The "*mangle*" table is used mainly for mangling packets, that is, changing the contents of different packets and that of their headers.

Each table contains a number of chains, each of which defines a set of rules. Those rules are applied on the packets which traverse the chain. There are several different chains in iptables. The chains of the *nat* table and the filter table are the most important ones to define access control on the virtual machines.

The nat Table and Port Forwarding

Network address translation ("*NAT*") is a technique which allows several hosts to share the same IP address and traffics packets between the Internet and an internal LAN. The NAT rules are defined in the chains of the nat table. When a packet reaches a nat chain, the source/destination address of the packet will be changed to a

[4]Netfilter and iptables `http://www.netfilter.org/`.

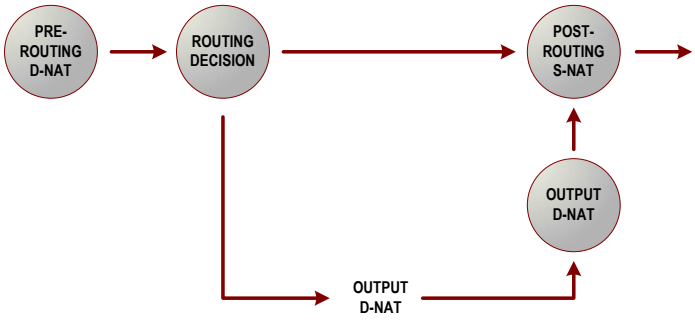

Figure 6.1.: A packet's journey through the nat chains.

different address by the NAT rules. NAT can remember how it altered a packet and will do the reverse translation on the reply packet when it passes through the other way [Russell, 2002].

There are three chains in the nat table: the "*PREROUTING*", "*POSTROUTING*" and "*OUTPUT*" chains. These three rules chains only apply to certain packets. As shown in Figure 6.1, packets pass through the PREROUTING chain when they enter the machine. Those packets can be destined for the local machine or for somewhere else, but the kernel does not know where they are going before any routing decision is taken. The OUTPUT chain corresponds to any packet originating from the local machine. After the routing decision is taken, those packets which leave the local machine but do not originate locally pass through the POSTROUTING chain.

There are two variants of NAT: source NAT or destination NAT.

- Source NAT works in the POSTROUTING chain and changes the source address or port of packets. Source NAT is used to hide internal IP addresses behind public IP addresses. E.g. we can insert the following rule to the POSTROUTING chain, which have all packets coming from an internal address (192.168.0.100) to correspond to a specific external IP (141.168.1.2)[5]:

[5]Common iptables options include:

-t specifies the name of the table to use.

-A, *-I* and *-D* followed by a chain name, to add, insert or delete rules respectively.

-p sets the IP protocol TCP, ICMP, or UDP.

– *dport* specifies the destination port of the packet.

-s and *-d* followed by an IP address to set the source and destination IP address.

-i and *-o* sets the interface where a packet will come in or go out depending on the routing.

-j is a target or jump to specify the action of the rule.

```
iptables -A POSTROUTING -s 192.168.0.100 -o ppp0 -j SNAT –to 141.168.1.2
```

IP Masquerading is a form of Source NAT, except that it uses the network interface instead of the IP address:

```
iptables -A POSTROUTING -o ppp0 -j MASQUERADE
```

- Destination NAT works in the PREROUTING or OUTPUT chains and changes the destination information of packets. E.g. the following rule will have the router or firewall forward all SMTP (Port 25) connections from the outside onto a mail server (192.168.0.1) on the internal LAN:

```
iptables -A PREROUTING -p tcp –dport 25 -i ppp0 -j DNAT –to 192.168.0.1:25
```

  Port forwarding is a form of the destination NAT. It examines the packet header and forwards it depending on the destination port number. Port forwarding allows several servers on the Internal LAN to share a public IP address by listening on different ports.

The filter Table and Packet Filtering

The filter table has three rule chains: the "*INPUT*", "*OUTPUT*", and "*FORWARD*" chains. The INPUT chain applies to all packets destined for the local machine. The OUTPUT chain responds to packets which originated locally. The FORWARD chain applies for packets which pass through the local but are destined for somewhere else. When a packet reaches a chain, it is examined by rules of the chain rules.

Each rule defines the criteria of packet features. If the coming packet matches the rule, a predefined action (target or jump) will be taken with the packet. E.g. to block all TCP connections to Port 23 of the local machine, we can define the following rule:

```
iptables -A INPUT -p tcp –dport 23 -j DROP
```

-m state followed by a *–state* for connection tracking or state matching.

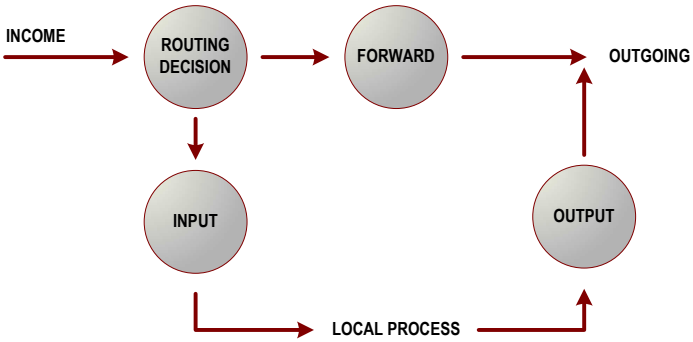

Figure 6.2.: Packet filtering chains.

If the rule does not match the packet, then the next rule in the chain is applied. The kernel will DROP the packet in case of no match.

Figure 6.2 illustrates the packet filtering processes. When a packet comes to the network interface of the local host, the kernel first takes routing decision to decide where to forward it:

- If this packet is destined for the local host, the packet reaches the INPUT chain. A local process will receive this packet if it matches any rule of the chain.
- If the coming packet is destined for another host, it will pass through the FORWARD chain. If this packet is accepted by the chain rules, it will be sent out.

If a packet is sent by a program on the local machine, it will immediately pass through the OUTPUT chain.This packet will continue out to the interface it is destined for if it matches the rules of the OUTPUT chain.

There are four types of connections which can by identified by iptables:

1. "*NEW*" corresponds to packets which are being used to create new connections.
2. "*ESTABLISHED*" relates to packets from a known connection.
3. "*RELATED*" applies to packets related to a active connection, such as an ICMP reply or active FTP sessions.
4. "*INVALID*" are malformed or unrecognized packets which should be dropped.

All this kind of matching is useful for connection tracking, or state matching. E.g. if we want to drop all NEW or INVALID packets, the following rule can be applied:

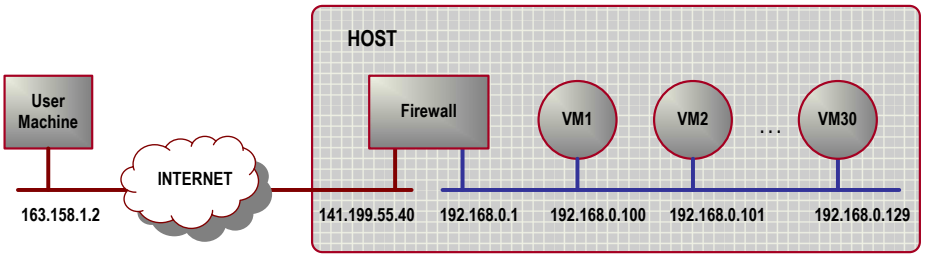

Figure 6.3.: Port forwarding for virtual machines.

```
iptables -A INPUT -m state –state NEW, INVALID -j DROP
```

6.4.2. IP-address Reuse for Virtual Machines

Each virtual machine is a node on a private network (the virtual LAN on the host). Normally, none of those virtual machines is accessible from a public network. However, if any virtual machine is assigned to a user, the VNC service on that machine must be allowed over the Internet. This requires several virtual machines must share the public address of the host. IP address reuse can be implemented by port forwarding (mentioned in Section 6.4.1). The idea of port forwarding is to assign the VNC server of each virtual machine unique port numbers. Thus, when VNC packets come to the public network interface, iptables checks the packet header and forwards them to right virtual machines depending on their destination port numbers. Access control can be enforced if we define the rules for port forwarding. E.g. we can define that only those packets from the user machine are allowed to be forwarded to the virtual machine which has exactly been assigned to this user.

Figure illustrates the port forwarding scheme. Each virtual machine on the virtual LAN is assigned a private address which is calculated from the number of the virtual machine, "**192.168.0.99 + number**". E.g. "VM1" has "192.168.0.100", "VM2" has "192.168.0.101", and so forth. The VNC server on each virtual machine is locally listening on two ports: "5800" and "5900". The internal access point of a VNC server can be expressed as: "**Private_address : (Port_1, Port_2)**". Table 6.1 lists the internal VNC access points of all virtual machines.

The port forwarding rules translate those internal access points to public access points. Then, those VNC services are mapped to different ports ("**5799 + number**" and "**5899 + number**") on the host network interface ("141.199.55.40"). The translation table is shown in Table 6.2.

Table 6.1.: Internal VNC access points.

Virtual Machine	VNC Access Point
VM1	192.168.0.100 : (5800, 5900)
VM2	192.168.0.101 : (5800, 5900)
VM3	192.168.0.102 : (5800, 5900)
...	...
VM30	192.168.0.129 : (5800, 5900)

Table 6.2.: The translation table of the VNC access points.

Virtual Machine	Internal Access Point	Public Access Point
VM1	192.168.0.100 : (5800, 5900)	⇒**141.199.55.40** : (580**0**, 590**0**)
VM2	192.168.0.101 : (5800, 5900)	⇒**141.199.55.40** : (580**1**, 590**1**)
VM3	192.168.0.102 : (5800, 5900)	⇒**141.199.55.40** : (580**2**, 590**2**)
...	...	...
VM30	192.168.0.129 : (5800, 5900)	⇒**141.199.55.40** : (58**29**, 59**29**)

Those translation rules in the nat table are dynamically managed depending on the assignment situation of the virtual machines. At the beginning, iptables disables any port forwarding for virtual machines. If a virtual machine is assigned, specific rules are inserted to the nat chains. When the user logs out, those rules for his/her virtual machine are immediately abandoned. For example, when a user from "163.158.1.2" logs in to the Tele-Lab host which publicly runs on the "141.199.55.40", a virtual machine must be assigned to this user. Here, we suppose VM3 on "192.168.0.102" is chosen by the virtual machine management system. Then two rules will be inserted into the PREROUTING chain in the nat table:

```
iptables -t nat -A PREROUTING -s 163.158.1.2 -p tcp dport 5802 -j DNAT to
192.168.0.103:5800
```

and

```
iptables -t nat -A PREROUTING -s 163.158.1.2 -p tcp dport 5902 -j DNAT to
192.168.0.103:5900
```

Those rules map the VNC service of VM3 to the public access point: "141.199.55.40 : (5803 and 5903)". The "***-s***", "***-p***" and "–***dport***" options are used for access control:

only VNC connections from the authenticated user (on "163.158.1.2") are allowed to be forwarded to that virtual machine. Finally, this public access point will be passed to the VNC client at the user end, so it knows where to connect the VNC server of the virtual machine. Table 6.3 shows the enforcement of the port forwarding rules on the packets in this example.

Table 6.3.: Enforcement of the port forwarding rules.

VNC Packet	Original	Altered
Client request	Source: 163.158.1.2/7890* Dest: 141.199.55.40/5802	Source: 163.158.1.2/7890 Dest: **192.168.0.102/5800**
Server reply	Source: 192.168.0.102/5800 Dest: 163.158.1.2/7890	Source: **141.199.55.40/5802** Dest: 163.158.1.2/7890
Client request	Source: 163.158.1.2/8335* Dest: 141.199.55.40/5902	Source: 163.158.1.2/8335 Dest: **192.168.0.102/5900**
Server reply	Source: 192.168.0.102/5900 Dest: 163.158.1.2/8335	Source: **141.199.55.40/5903** Dest: 163.158.1.2/8335

(* 7890 or 8335 is the local port number of the VNC client.)

6.4.3. Packet Filtering for Access Control

The virtual network of Tele-Lab is different from a conventional "*ether*" network. It is created by a virtual switch device from the User-Mode Linux package. As shown in Figure 6.4, "eth0" is the external interface of the host and "tap0" is the internal interface created by the virtual switch. This virtual network has a very specially feature: each virtual machine is attached to a central node (the virtual switch on the host), and any traffic of virtual machines must be relayed by the host. Therefore, it is possible to apply a firewall on the host and implement network security isolation by controlling traffic.

The idea to control traffic is packet filtering by iptables. In order to satisfy control policies specified at the beginning of Section 6.4, we defined and enforced two categories of filtering rules on iptables. Before describing those rules, we have to point out that all filtering rules match packets based on the internal addresses because the NAT

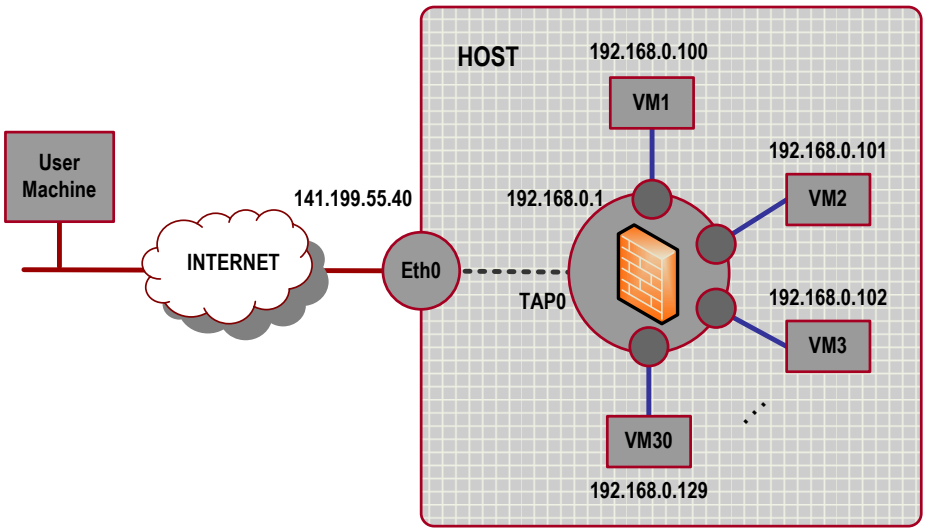

Figure 6.4.: Structure of the virtual network.

chains are processed before the route decision in the packet filtering chains. That is, the source or destination addresses seen by the packet filter are local addresses.

Packet Filtering between Virtual Machines and External Networks

The following rules are defined based on the network structure in Figure 6.4.

Rule 1: external hosts are only allowed to access the VNC service on the virtual network (on port 5800 and 5900).

```
iptables -A FORWARD -i eth0 -o tap0 -p tcp –dport 5800,5900 -j ACCEPT
```

Rule 2: only VNC related packets are allowed to go out of the virtual network to the outside.

```
iptables -A FORWARD -i tap0 -o eth0 -m state –state ESTABLISHED, RELATED
-j ACCEPT
```

Rule 3: any packet from the Tele-Lab host onto the virtual network is allowed (for virtual machine management).

```
iptables -A FORWARD -s 141.199.55.40 -o tap0 -j ACCEPT
```

Rule 4: only those packets related to the packets in Rule 3 are allowed to reach the Tele-Lab host from the virtual network.

```
iptables -A FORWARD -i tap0 -d 141.199.55.40 -m state –state ESTABLISHED,
RELATED -j ACCEPT
```

Packet Filtering between Virtual Machines

Iptables is able to filter any packets between virtual machines because the Tele-Lab host is a central node of the virtual network. In order to isolate internal connections between virtual machines, we can apply a simple rule on the INPUT chain.

Rule 5: a packet will be dropped if both the source and destination addresses of this packet fall in the same address range of the virtual network.

```
iptables -A INPUT -i tap0 -o tap0 -j DROP
```

6.5. Secure User Interfaces

The built-in security feature of the VNC is to authenticate users by one-time passwords[6]. This scheme is relatively safe because it avoids exposing real passwords on the Internet. However, the VNC session is unencrypted after authentication. Any input to the viewer is sent to the server in plain text. If the VNC connection is snooped by someone between the client and the server, it would be possible that this man-in-the-middle takes over the connection by forging packets which look like the right ones. Therefore, it is necessary to encrypt VNC sessions if they pass through untrusted networks. The idea of the VNC session encryption is to use VNC service through a secure channel, e.g. a VPN ("Virtual Private Network") [Yuan and Strayer, 2001]. In case of no VPNs available, we can tunnel the VNC protocol over SSH connections.

SSH ("Secure Shell") [Ylönen, 1996] is a protocol which allows encrypted and authenticated connections between any two ends. OpenSSH[7] is a free SSH suite which supports various encryption algorithms (such as AES-256 and 3DES), MAC algorithms, and public-key authentication. Figure 6.5 illustrates a VNC tunnelling scheme based

[6] One-time password method is also called the challenge-response system. The server sends a encrypted value. The user must decrypt this value, perform some computing, encrypt the result, and return it to the server [Haller, 1998].

[7] OpenSSH `http://www.openssh.org/`.

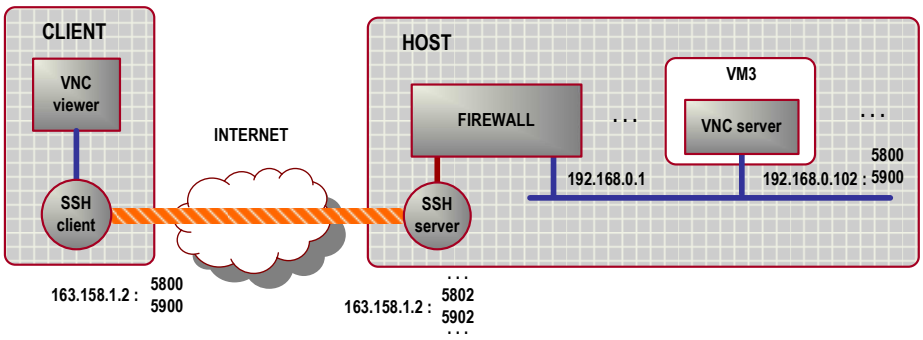

Figure 6.5.: VNC tunnelling over SSH.

on OpenSSH. A ssh client and ssh server are placed between the VNC client and server. The ssh client supports secure port forwarding and listens on a local port. Instead of directly connecting to the VNC server port, the VNC viewer connects to the ssh client on that local port. Then connections will securely reach to the ssh server which will forward them on to the destination ports of the VNC server.

For example, "VM3" is a virtual machine of Tele-Lab. Its internal and external access points are 141.199.55.40 : (5802, 5902) and 192.168.0.102 : (5800, 5900) respectively. The following commands will set up a ssh client on the user end:

```
# ssh -L 5800:localhost:5802 141.199.55.40
# ssh -L 5900:localhost:5902 141.199.55.40
```

The VNC client connects to the ssh client on local ports (Port 5800 and 5900). Then, the ssh client forwards the local VNC connections to the host port (5802 or 5902). After decryption by the ssh server, iptables forwards those connections to the VNC server of VM3.

The disadvantages of the VNC tunnelling include: (1) it degrades the VNC performance because of encryption processes; (2) users need an extra SSH client program (Putty[8] for Windows or OpenSSH for Linux). Therefore, tunnelling over SSH won't be used if VNC connections cross trusted networks.

[8]Putty `http://www.chiark.greenend.org.uk/~sgtatham/putty/`.

7. Application in E-learning

Tele-Lab IT-Security is a general architecture for teaching or training practical security and can be applied for different purposes. This chapter will point out the uses of Tele-Lab IT-Security and present case studies to demonstrate how it successfully supports learning and exercises.

7.1. Range of Application

Tele-Lab IT-Security can be used to teach various subjects of cryptography and network security. E.g. we have implemented a web-based tutoring system in the framework of the Tele-Lab architecture, which covers symmetric encryption, secure email, authentication, etc. Those teaching topics and the arrangement of the their content are listed in Table 7.1. Like other online training or tutoring systems, Tele-Lab IT-Security presents theoretical facts about subjects as well as relevant tutorials and demos. As shown in the "Exercise" column of Table 7.1, the distinction of Tele-Lab from other systems is that it offers students chances to perform security experiments in a lightweight and real-life laboratory environment.

In the winter semester 2005/2006, a student group at the Hasso Plattner Institute of the University of Potsdam is integrating more security topics into Tele-Lab IT-Security as their practicum work. Those topics will cover a broader range of subjects such as public key encryption, port scanning, packet sniffing, access control, intrusion detection, man-in-the-middle and firewalls (See Table 7.2).

With a modularized structure of tutoring contents, Tele-Lab IT-Security can be adapted for specific teaching purposes or user groups. In general, Tele-Lab IT-Security can be used in the following ways:

- *Supporting security courses.* Tele-Lab IT-Security only supplies laboratory exercises for a specific course. The contents are designed or re-organized according

Table 7.1.: Topics integrated into Tele-Lab IT-Security.

Topic	Introduction	Tutorial	Exercise
Security Foundations	IT Security components, threats and goals; Assurance.	Null.	Quizzes on basic security concepts.
Symmetric Encryption	Foundations of cryptography; Concepts of symmetric encryption; Classic and advanced algorithms (including a DES demos).	PGP: an encryption program; GunPG: an Opensource PGP program.	Encrypt/decrypt messages using GnuPG.
Secure Email	Security requirements for Email; Message encryption and digital signatures; Pretty Good Privacy (PGP); Secure Multipurpose Internet Mail Extensions (SMIME).	GunPG: an Opensource PGP program; The Mozilla Thunderbird client with both the SMIME security feature and Enigmail security module.	Apply for digital certificates or create PGP key pairs; Sign/encrypt messages and send/receive secure email via the Mozilla Thunderbird client; 1) Use SMIME functions, 2) Use Enigmail.
Authentication	Password-based authentication; Password hashing; Unix "passwd" file and "shadow" file; Criteria of strong passwords.	Unix commands related to Unix "passwd" and "shadow" ; Password cracker, John-the-Ripple.	User the John-the-Ripple to crack a randomly-generated "passwd" file.

Table 7.2.: Topics being developed in Tele-Lab IT-Security.

Topic	Practical Features
Access Control	Demonstrate how access control mechanisms in Linux are breached by Buffer Overflow.
Public key encryption	Exercise how to use GunPG and OpenSSL to create key pairs and certificates; Exercise encrypting and signing with both tools.
Port Scanning	Exercise how to find services on the target host with Nmap and close unnecessary services.
Firewalls	Exercise configuring an *iptable* packet filter and setting up a firewall in Linux.
Intrusion Detection	Exercise setting up the Snort IDS program; Detecting attacks from the Snort log files.
Security Scanning	Exercise to scan the target server to find system vulnerabilities; Patching up systems or close vulnerable services.
Man-in-the-Middle	Demonstrate how man-in-the-middle attacks compromise the SSL sessions.

to the course plan. In this circumstance, the Tele-Lab server can be deployed in a laboratory or accessible from the campus network.

- *Online-learning.* Tele-Lab is operated as an e-learning service. It is as a complete tutoring system: both theoretical basics and practical exercises are integrated. With the virtual machine architecture, Tele-Lab is a distinct teleteaching tool for delivering hands-on experience through the Internet.

- *Industrial training.* Tele-Lab is customized for security training in industry. Contents are developed for specific training topics or products which are interesting to companies or theirs customers.

7.2. Learning with Tele-Lab IT-Security

In this section, we describe main processes of the use of Tele-Lab IT-Security. These processes include user registration, user login, user logout, and the learning process on the virtual machine.

Figure 7.1.: A user registration page.

7.2.1. User Registration

Before a user starts learning, he/she must register an account from Tele-Lab IT-Security. As shown in Figure 7.1, on the registration page, the user can input personal data and select language settings ("English" or "German") and user categories ("admin", "general user", or "IT student"). Those data then are processed to create a user profile in a database.

7.2.2. User Login

As a user logs into Tele-Lab IT-Security from the portal page, the following actions are taken to set up the user's learning environment:

1. The portal sends the virtual machine management system a request for a virtual machine.

2. The virtual machine management system searches the virtual lab for virtual machines and assigns one to the user.

Figure 7.2.: The start page of the IT security tutor.

3. The iptables firewall sets up connections between the user and his/her virtual machine.

4. The user receives the VNC client (an applet) and the virtual machine's desktop will be shown in the browser as a frame embedded on the portal page (see Figure 7.2).

7.2.3. User Logout

A user logs out from Tele-Lab IT-Security by clicking the "*logout*" link on the portal page. Following steps will be taken to release system resource of this user.

1. The actual user profile is passed from the user's virtual machine to the host.

2. Connections between the virtual machine and the user are disabled by the iptables firewall.

3. The virtual machine is reclaimed by the virtual machine management system.

7.2.4. Learning Processes

The security tutor is by default started when the virtual machine is assigned to the user. When the user clicks the "Beginnen" on the tutor page, the profile of this user will be imported from the host and applied in the virtual machine (see Figure 7.2). Then the tutor presents a list of available chapters from which the user can choose one. A Tele-Lab chapter consists of several sections:

- In general, theoretical facts in a chapter are introduced first.
- Then tutorials of related security tools are presented.
- Finally practical exercises are prepared and assigned to the user.

Normally, an exercise is performed in three steps:

- *Exercise preparation.* To configure necessary system settings of the virtual machine the tutor will invoke particular Perl or shell scripts on the virtual machine. E.g. for the "Secure Email" exercise, a virtual role and related mail settings are created so that email can be exchanged in an effective circumstance.
- *Task generation.* The tutor activates scripts to generate data or materials (e.g. files or messages) needed in the exercise. Then questions are displayed on the web page. If possible, those questions are dynamically generated with different details each time. E.g. in the "Password Cracking" exercise, the "*passwd*" file to be decrypted is generated at run time.
- *Result evaluation.* The user completes tasks on the virtual machine. After he/she submits results, the tutor evaluates them by scripts. To simulate a real scenario, evaluation process is often done in an interactive way. For instance, to finish the Secure Email exercise, the user has to interact with the tutor and exchange signed or secret messages. In each step, the tutor checks messages and tells user what to do in the next step. In case of fail, the user can repeat the exercise until a correct solution is found.

7.3. Case Studies

We will demonstrate the concrete learning process with case studies. They include three example topics of Tele-Lab IT-Security, which are Password-Based Authentication, Symmetric Encryption, and Secure Email in Section 7.3.1, 7.3.2, and 7.3.3 respectively.

7.3.1. Password-Based Authentication

In this section, we will show an exercise which requires privileged operations or need to access to system files. The password-based authentication chapter is about how users are authenticated through passwords and how passwords are protected in the Linux systems. In its exercise, a Linux *passwd* file is generated. The user needs to crack it with an open-source cracker, "*John-the-Ripper*". The learning process of this chapter is illustrated below:

- In the introduction sections, concepts in password-based authentication are introduced first (see Figure 7.3). These concepts include password hashing (DES and MD5), the UNIX "passwd" or "shadow" files, and the password selection criteria.

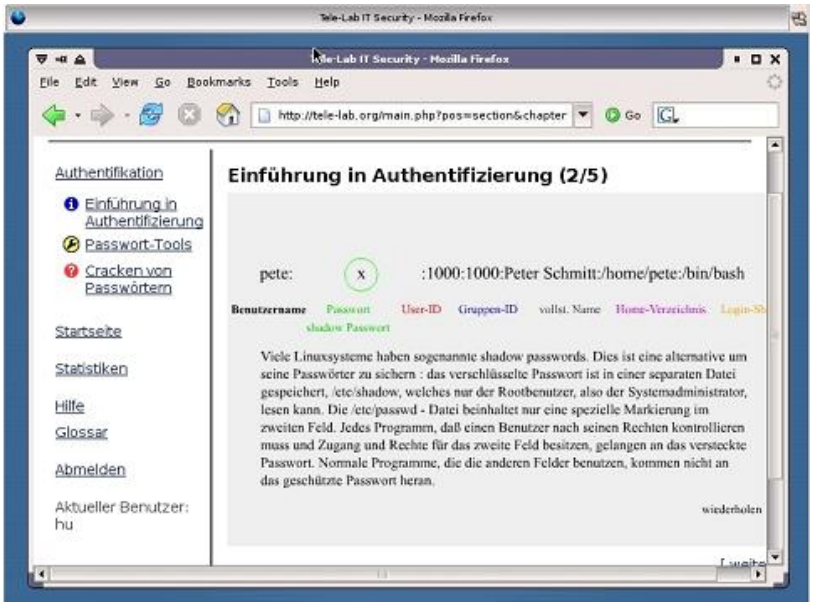

Figure 7.3.: Introduction to password-based authentication.

- Information of the relevant security tools is presented next. They include the *passwd* command[1], the PAM[2], and the John-the-Ripper password cracker. Some of tools are introduced with Flash animation clips (see Figure 7.4).

Figure 7.4.: The password cracker tutorial created by Flash.

- Tasks such as creating password hashes and cracking passwords are assigned to the user. Here we only demonstrate the completion of the password cracking task. The first step is work environment preparation: the password hashing configuration is initialized to use the default DES method on the virtual machine.

- The next step is to assign the user a task to crack several Linux passwords with John-the-Rippe (see Figure 7.5).

- Each time, a group of random passwords are generated and saved in a "passwd" file (see Figure 7.6).

- The user downloads the "passwd" file into the local directory of the virtual machine (see Figure 7.7).

[1] A Linux command which creates passwords by DES hashing.
[2] PAM is the "*Pluggable Authentication Modules*" in Linux, which can be applied to support MD5 password hashing.

Figure 7.5.: The password cracking exercise.

Figure 7.6.: Content of a Linux “passwd” file.

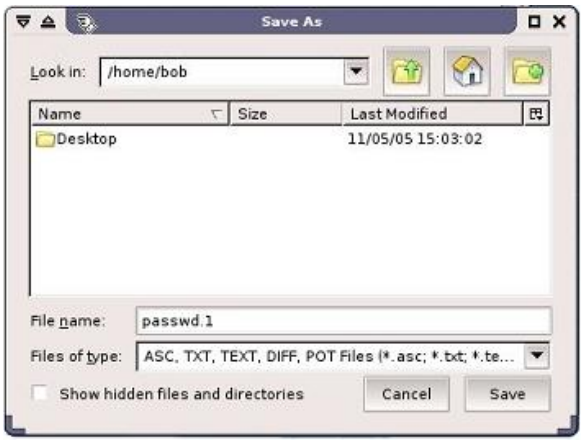

Figure 7.7.: Downloading the “passwd” file.

- Evaluating results. In order to apply John-the-Ripper, the user must have a privilege right to run this system program. Tele-Lab provides a "Root Terminal" button on the virtual machine. From that, the user can switch to the root mode and execute John-the-Ripper to crack passwords (shown in Figure 7.8).

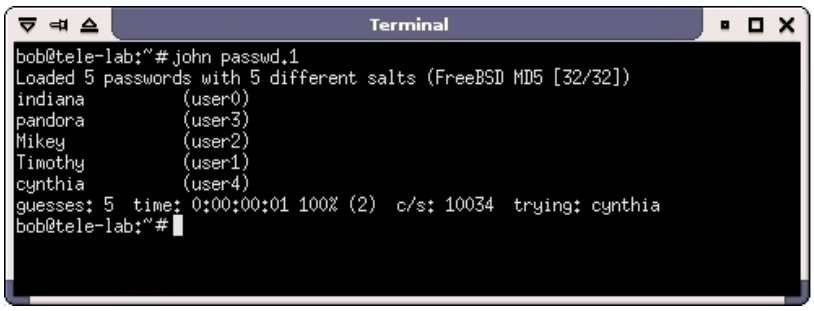

Figure 7.8.: Cracking the "passwd" file in a root shell.

- After passwords are found and the answers are submitted, the tutor compares them with correct ones (see Figure 7.9). The evaluation result is shown to the user and recorded in the user profile.

Figure 7.9.: Submitting the cracking result.

7.3.2. Symmetric Encryption

The Tele-Lab chapter, "Symmetric Encryption", is used to illustrate how Tele-Lab IT-Security and virtual machines help users understand and apply encryption technologies. It introduces important concepts on symmetric encryption, illustrates related algorithms, and provides practical exercises in which a user can apply a production encryption tool, GnuPG, to encrypt or decrypt messages. The exercises in this example do not require a user to have a privilege right on the virtual machine. Instead, the user is given an ordinary account by default when he/she is assigned a virtual machine. He/she can directly apply tools with that account. This is also applied to similar topics in which only ordinary operations are involved.

The detailed description can be found in Appendix A.

7.3.3. Secure Email

This Tele-Lab chapter is intended to familiarize a user with digital certificates so that he/she can sign and encrypt email. Exercises in this chapter simulate everyday email communications which need a complicated and interactive user work environment. The learning process is described as follows:

- The tutor presents concepts of secure email, which include security requirements for email, measures to protect email such as encryption and digital signatures, and secure email standards like *OpenPGP*[3] and *SMIME*[4].
- The *Thunderbird* email client is introduced. Thunderbird is a popular email tool used on Linux and Windows. It integrates a SMIME module and an OpenPGP module called *Enigmail*. The SMIME module is the default security feature of Thunderbird and Enigmail is a plug-in which encrypts or digitally signs email by GnuPG. The tutorials for both email security tools are presented to illustrate how to apply them to secure email.
- Two sets of tasks are designed for SMIME and Enigmail respectively. In both tasks, the user is required to securely exchange email with a virtual partner. Detailed processes can be found in Appendix B.

[3]"OpenPGP" is an encryption scheme based on PGP for interpreting and sending digitally signed and encrypted messages.

[4]"SMIME" (Secure Multipurpose Internet Mail Extensions) is the Internet standard for secure e-mail attachments based on RSA and MIME (Multipurpose Internet Mail Extensions).

8. Conclusions

In this chapter a summary of the work covered in this thesis will be presented. Contributions and advanced features of this work will be highlighted and directions of the future work will be proposed.

8.1. Summary and Contributions

This thesis has investigated the state-of-the-art in IT security education, pointed out the gap between e-learning and practical IT security education, as well as presented a work to fill the gap. E-learning is a flexible and personalized alternative to traditional education. Nonetheless, existing e-learning systems for IT security education have difficulties in delivering hands-on experience because of the lack of proximity. Laboratory environments and practical exercises are indispensable instruction tools to IT security education, but security education in conventional laboratories poses the problem of immobility as well as high creation and maintenance costs. Challenges in current IT security education suggest that there is a need to effectively transform security laboratories and exercises into e-learning forms.

In response to challenges, this thesis has presented the Tele-Lab IT-Security architecture. It is a novel virtual machine architecture for creating online IT security laboratories. Based on the virtual machines, Tele-Lab IT-Security allows students not only to learn IT security concepts and principles, but also to experiment security and gain hands-on experiences in a lightweight and safe laboratory environment.

Functionality, reliability, security and performance are realistic needs in the Tele-Lab implementation. In addition to the architecture itself, a set of technical solutions has been proposed in this thesis to address the needs above. Those solutions range from creating real and effective laboratory environments based on the virtual machines, to managing the virtual machines in a reliable way, and to preventing the misuse of the virtual machines by security isolation.

Virtual machines lay the foundations of this work. The User-Mode Linux virtual machines have been proved to be capable of simulating laboratory platforms on a single host. Through the appropriate resource allocation, software installation, system configurations, and user interfaces, virtual machines can be converted to lightweight security laboratories at a reasonable performance.

The reliability and availability of Tele-Lab IT-Security are covered by a virtual machine management framework. This framework provides the virtual machine administration and monitoring services. With these services, critical failures of the virtual machines can be detected and recovered at run time.

Running security experiments at a privileged level on the virtual machines poses security risks. This thesis has specially addressed the security aspects of Tele-Lab IT-Security. After identifying Tele-Lab's security requirements, the thesis has proposed a security management solution to prevent the misuse of the Tele-Lab virtual machines through security isolation at the system and network levels.

The applicability of the Tele-Lab IT-Security architecture has been demonstrated in this thesis. An example chapter on Password-based Authentication has shown that the security tasks running in the privileged mode can be successfully implemented based on the Tele-Lab architecture. The other two example chapters on Symmetric Encryption and Secure Email have further demonstrated the capability of the Tele-Lab IT-Security at supporting those security exercises which need complicated environments and interactions.

Tele-Lab IT-Security has the following advantages which distinguish it from other security teaching/learning approaches:

- As a better solution than multimedia courseware and demonstration software, Tele-Lab IT-Security not only implements a web tutoring systems for learning theoretical principles, but also offers students hands-on experiences.

- Tele-Lab IT-Security offers users a real-life laboratory environment instead of limited simulation. Compared with dedicated security laboratories, Tele-Lab IT-Security mobilizes the access to learning and reduces the costs and efforts for creation and maintenance of laboratory environments.

- Tele-Lab IT-Security has a thin web user interface through which security tools

and programs needed in the security exercise are accessible via a VNC applet[1]. This interface avoids installation of additional client software on the user-end.

- With the virtual machine management system, Tele-Lab remains reliable and available though security experiments may result in critical system failures.

- User activities in the virtual machine are constricted in a safe scope by security management. Risks of the misuse of laboratory resources are eliminated by security isolation measures.

8.2. Future Work

The future work will focus on the evaluation of the usability and educational effectiveness of Tele-Lab IT-Security as well as the improvements on the architecture to extend its application possibilities.

Evaluations

To demonstrate the applicability of the Tele-Lab IT-Security architecture, some of teaching chapters including cryptography, access control, authentication and secure email have been implemented on the Tele-Lab IT-Security CD and Server. Preliminary evaluation was done in the University of Trier in 2004. Two groups of students from the computer science department (IT) and in other disciplines (non-IT) respectively have joined in the evaluation. All students in both groups have not taken any network or information security courses before that evaluation. Feedbacks from the students showed both groups were able to well understand theoretical concepts and tool tutorials with the layout of the current chapter structure. Non-IT students were frustrated in security exercises by lack of essential knowledge and skills of Linux operating environment. For IT students, the situation was much better: since they already knew about Linux, they were able to follow chapters and finish exercises without support of an instructor. Performance data of the students were collected from the learning. From that, statistics about user performance and teaching content were generated, which helps the evaluation and future adjustment of the chapter contents.

[1] Virtual Network Computing ("VNC") is a remote desktop access system based on the Remote Frame Buffer (RFB) protocol (See Section 4.3.1).

The evaluation results indicate the use of open source software and Linux environment limits the application range of Tele-Lab IT-Security especially for most non-IT students, but Tele-Lab is still an effective approach for training IT students.

More thorough evaluations must be carried out to test the usability, performance and education effectiveness of the Tele-Lab architecture. To do this, we plan to develop a complete on-line course based on Tele-Lab and apply it for teaching the students at the Hasso-Plattner-Institute (HPI) of the University of Potsdam. In the Winter Semester 2005/06, a student group (about thirty IT students from HPI) has started to develop more course chapters as their practicum work.

Improvements on Architecture

The improvements on the Tele-Lab architecture will concentrate on the virtual machine management and laboratory platforms. Besides the User-Mode Linux virtual machines, we are considering to extend the management interface for other virtual machine software such as Xen, VMWare, or Virtual PC. By doing so, we can offer more possibilities to build virtual laboratory environment for students based on user's preference and experiences. Security on the Windows platforms has not been addressed by current development. The future work will attempt to integrate the Windows virtual machines into the Tele-Lab architecture. The concern about Windows virtual machines is that they are normally non-free and resource-intensive. Nevertheless, because of the importance of Windows security, it is still necessary to build the lightweight Windows virtual machines on a high-performance host platform.

8.3. Closing Remarks

Tele-Lab IT-Security is a research work attempting to bridge the gap between e-learning/tele-teaching and practical IT security education. Tele-Lab is not going to completely substitute the conventional laboratory teaching but add practical features to e-learning. The work in this thesis has demonstrated the possibility to implement hands-on security laboratories on the Internet reliably, securely, and economically.

A. Symmetric Encryption Demonstration

The Tele-Lab chapter, "Symmetric Encryption", introduces important concepts on symmetric encryption, illustrates related algorithms, and provides practical exercises in which a user can apply a production encryption tool, GnuPG, to encrypt or decrypt messages. Following contents and exercises are provided by this chapter:

- Essential cryptographic concepts and typical cipher algorithms are introduced in early sections (see Figure A.1). E.g. in order to explain the DES algorithm[1], we integrate a visual demonstration from which a user can run the real algorithm program and watch en-/decryption details .

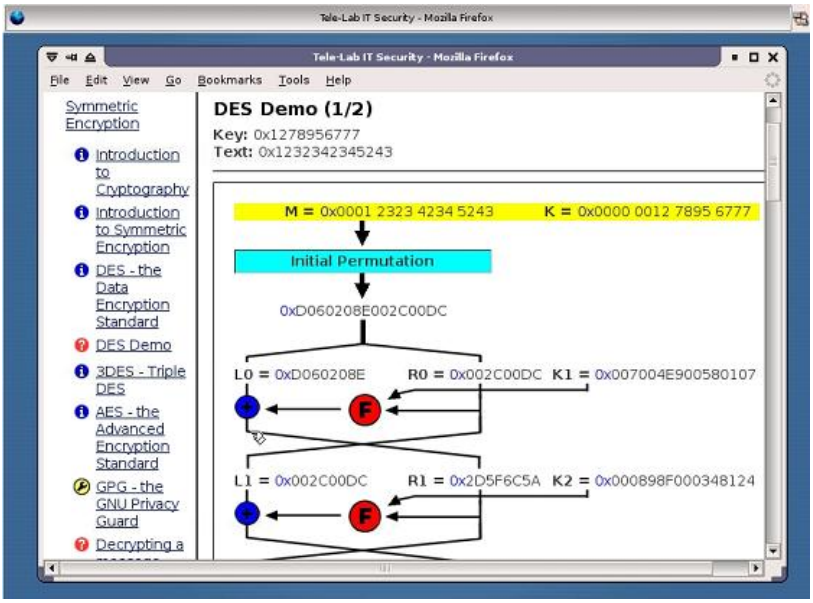

Figure A.1.: DES demonstration.

[1]DES is a significant symmetric algorithm which includes 16 encryption rounds.

Figure A.2.: The GnuPG tutorial.

- The tutorial section introduces an encryption tool, GPG ("*GNU Privacy Guard*") which is an opensource PGP ("*Pretty Good Privacy*") software program. From this tutorial, the user learns how to apply encryption technology and tools to meet everyday needs (see Figure A.2).

- In the exercise sections, encryption and decryption exercises are assigned to the user. The process of the decryption exercise includes the following steps. First, the tutor creates four texts and one of the four is randomly chosen. The chosen text is then encrypted by GPG in the background and the corresponding secret message is produced (see Figure A.3).

- The task for the user is to select the right plaintext of the list of four possible plaintexts. In order to finish this task, the user must download the secret message, find the key from email, and decrypt it with GPG. Then the user has to submit the plaintext he/she finds (see Figure A.4).

- After submission of the result, the tutor checks it and finds out whether it is the matched answer to the question. The evaluation result will be shown to the user and recorded into his/her user profile. In case of fail, the tutor will ask him or her to have another try.

- The encryption process is relatively simple. The tutor asks the user to encrypt

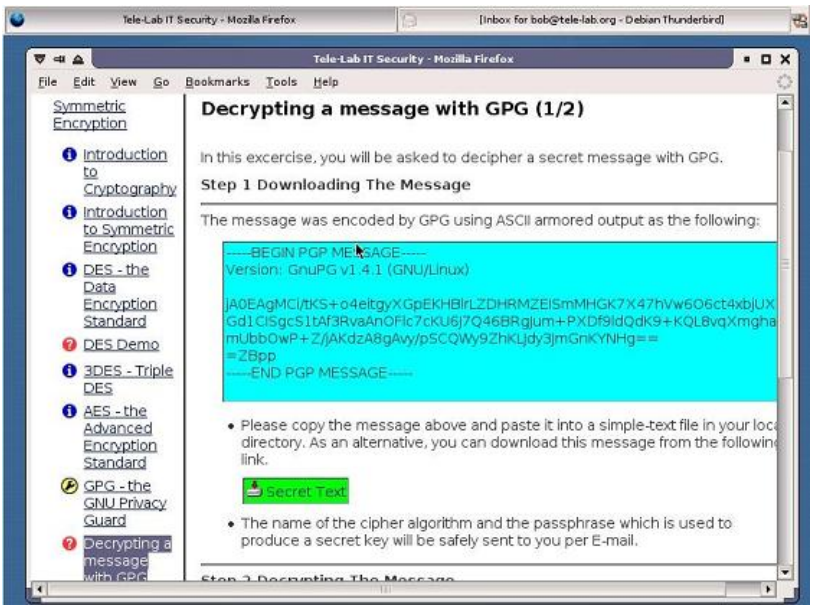

Figure A.3.: The task of the GPG decryption exercise.

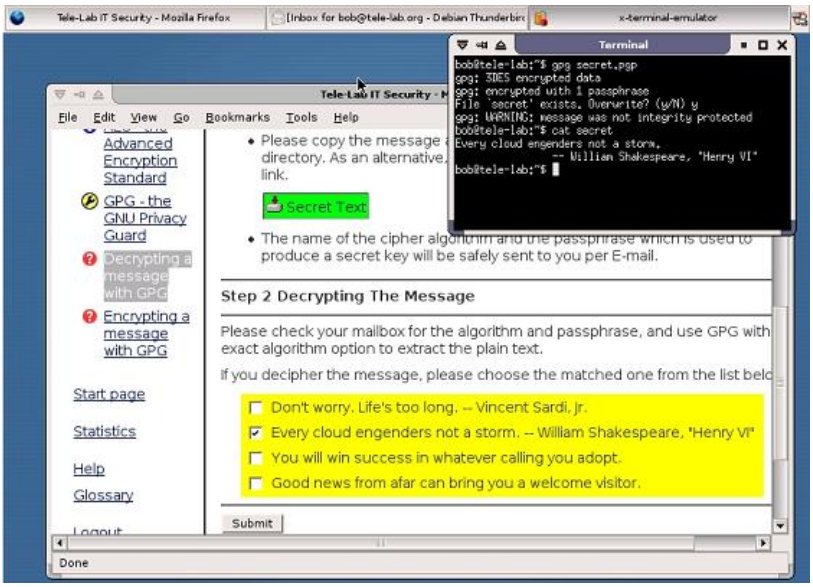

Figure A.4.: Completion of the GPG decryption exercise.

a text. The user has to do encryption and submit the secret text with a key which is used for encryption. Then the tutor tries decrypting the submission. If it can be successfully decrypted with that key, this means the user has done a job, otherwise he/she is asked to try again.

B. Secure Email Demonstration

B.1. The SMIME Exercise

As shown in Figure B.1, the SMIME exercise requires an interactive laboratory environment. In this environment, OpenSSL[1], a certificate authority (CA), and related mail services and accounts have to be provided. In order to provide effective interaction, a virtual email partner named "*Alice*" is also created. This partner account is necessary because, without it, a user does not know with whom to exchange messages. In this exercise, the user is asked to manage digital certificates, create/verify signature of messages, and encrypt/decrypt messages. The procedure that a user creates signature and sends signed mail to Alice is described below:

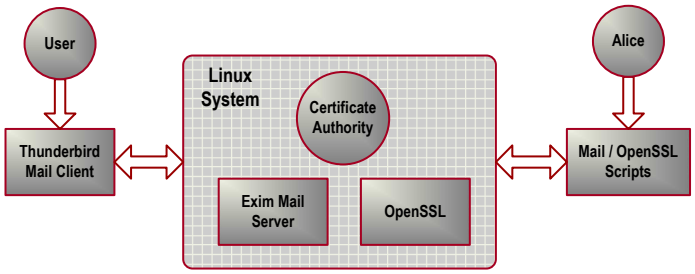

Figure B.1.: Working environment of the SMIME exercise

1. Work Environment Preparation.

   - The tutor generates a virtual user, Alice, i.e. creates and configures her Linux account and email settings.
   - The tutor creates a certificate authority by generating a root certificate and setting up a certificate service.

[1] OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) network protocols and related cryptography standards (`http://www.openssl.org/`).

- The tutor issues a certificate for Alice and configures her mail account settings with this certificate.

2. Exercise Execution.

   - Importing the certificate. ***(1)*** A certificate request page is shown to the user. He/she can fill out the certificate request form and apply for a certificate from the CA (see Figure B.2). ***(2)*** The user's private key is created and a corresponding certificate file (in *PKCS12*[2] format) is generated. The user is asked to download his/her certificate file and import it to his/her mail client. ***(3)*** The user opens the Thunderbird's SMIME certificate manager, imports the certificate, and trusts it for identifying email users (see Figure B.3). ***(4)*** The user also needs to configure his/her mail account to use the imported certificate.

Figure B.2.: Requesting a personal certificate.

   - Signing messages. The user is asked to write to Alice with a digital signature (shown in Figure B.4). The user must compose a message, sign it with his/her private key, and send it to Alice (see Figure B.5).

3. Result Evaluation

   - To evaluate the result, the tutor behaviors as Alice in the background. It first checks Alice's inbox and fetches the message sent by the user.

[2]PKCS 12 is the RSA personal information exchange syntax standard which describes a portable format for storage and transportation of user private keys, certificates etc.

Figure B.3.: Import and configuration of the personal certificate.

Figure B.4.: The task to digitally sign a message.

Figure B.5.: Signing a message.

- The tutor verifies its signature by OpenSSL scripts and confirms whether it has been sent from the student and matches with the origin (see Figure B.6).

Figure B.6.: Verifying the signature for evaluation.

- If verification succeeds, the user will receive a confirmation message and the completion of the task will be recorded into the user profile.

The message encryption/decryption task is arranged after the message signing task and follows a similar procedure:

- After the user receives the signed message of Alice and imports her certificate, he/she is asked to decode a message from Alice. The message is encrypted by Alice with the user's own public key.
- The user then encrypts a message with Alice's public key and sends it to Alice. The tutor will try to decrypt the message with Alice's private key corresponding to her certificate and record the (un-)successful completion of the task in the user profile.

B.2. The Enigmail Exercise

The Enigmail exercise includes the tasks to manipulate public-private key pairs, create/verify signature of messages, and encrypt/decrypt messages. In fact, the principle of the Enigmail exercise is very similar to that of SMIME (see Figure B.7). It also needs a virtual partner, Alice. The only deference between both is that Enigmail is based on open PGP which uses a trust web model and has no central certificate authority. Certificates (PGP key pairs) and trust have to be managed manually. The steps to prepare the key pairs include:

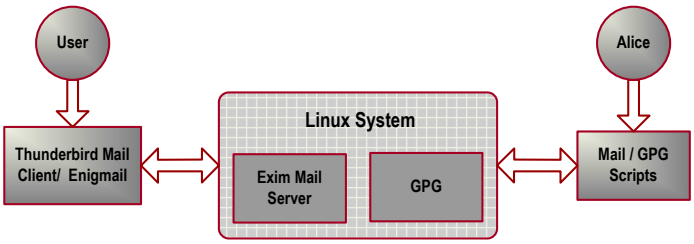

Figure B.7.: The working environment of the Enigmail exercise.

- First, the user must create a key pair of his/her own by using the GPG tool and then configure his/her Enigmail settings of Thunderbird with the created key pair (see Figure B.8).
- The user publishes his/her public key and obtains those of others. The user uploads the key to the tutor and the tutor will configure it in Alice's mail account. The user can download Alice's public key from the tutor. Thus, the user and Alice can exchange public keys with each other (see Figure B.9).

With key pairs available and public key exchanged, the rest tasks of email signing and encryption follow similar processes to those of the SMIME exercise. The difference is that the user will use the OpenPGP key pairs instead of certificates, and the tutor will use the GPG scripts in the background to run interactive tasks.

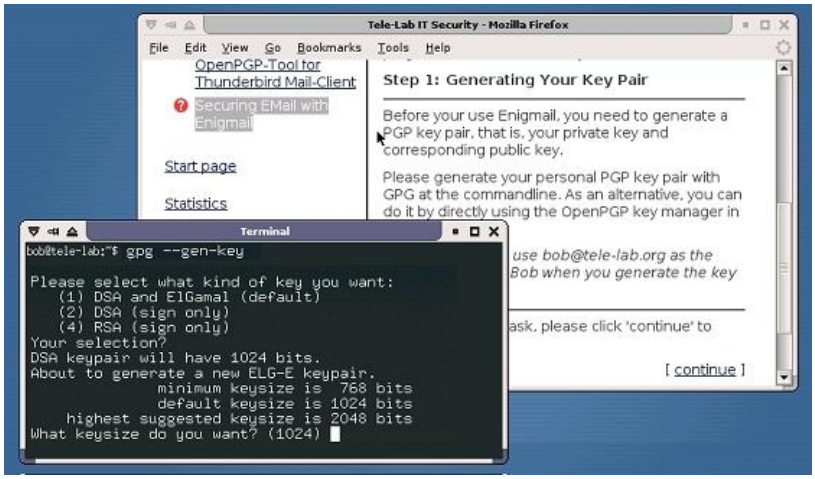

Figure B.8.: Creating a PGP keypair.

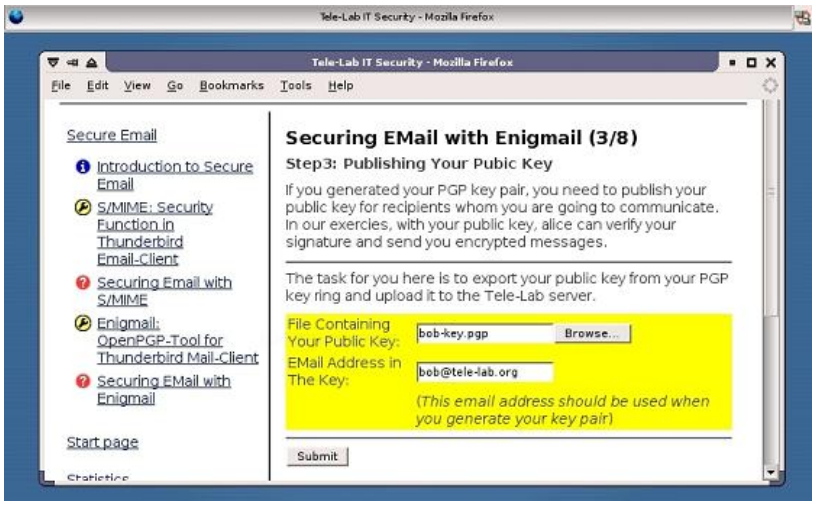

Figure B.9.: Publishing the public key.

Bibliography

[Barham et al., 2003] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*.

[Bishop, 2000] Bishop, M. (2000). Education in information security. *IEEE Concurrency*, 8(4):4–8.

[Canon et al., 1980] Canon, M. D., Fritz, D. H., Howard, J. H., Howell, T. D., Mitoma, M. F., and Rodriquez-Rosell, J. (1980). A virtual machine emulator for performance evaluation. *Communications of the ACM*, 23(2):71–80.

[Cao et al., 2002] Cao, J., Chan, A., Cao, W., and Yeung, C. (2002). Virtual programming lab for online distance learning. In *Proceedings of ICWL 2002*, pages 216–227, Hongkong, China.

[Chen and Noble, 2001] Chen, P. M. and Noble, B. D. (2001). When virtual is better than real. In *Proceedings of the 2001 Workshop on Hot Topics in Operating Systems (HotOS)*.

[Cheswik et al., 2003] Cheswik, W., Bellovin, S., and Rubin, A. (2003). *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, 2nd edition.

[Cordel, 2004] Cordel, D. (2004). *Management virtueller Maschinen für Tele-Lab IT-Security*. Diplomarbeit, Universität Trier.

[Dike, 2000] Dike, J. (2000). A user-mode port of the Linux kernel. In *Proceedings of the USENIX Annual Linux Showcase and Conference*, Atlanta, GA.

[Dike, 2001] Dike, J. (2001). User-mode Linux. In *Proceedings of the 5th Annual Linux Showcase & Conference*, Oakland, California, USA.

[Esslinger, 2002] Esslinger, B. (2002). Cryptool - spielerischer einstieg in klassische und moderne kryptographie: Neue version - fundierte awareness in deutsch und englisch. *Datenschutz und Datensicherheit*, 26(10).

[Garfinkel and Spafford, 2001] Garfinkel, S. and Spafford, G. (2001). *Web Security, Privacy & Commerce*. O'Reilly & Associates.

[Goldberg, 1972] Goldberg, R. P. (1972). *Architectural Principles for Virtual Computer Systems*. PhD thesis, Ph.D. thesis, Harvard University, Cambridge, MA.

[Goldberg, 1974] Goldberg, R. P. (1974). Survey of virtual machine research. *IEEE Computer*, pages 34–45.

[Haller, 1998] Haller, N. (1998). *A One-Time Password System*. RFC 2289.

[Hiltz and Turoff, 2005] Hiltz, S. R. and Turoff, M. (2005). Education goes digital: The evolution of online learning and the revolution in higher education. *Communications of the ACM*, 48(10):59–64.

[Hoffman et al., 2003] Hoffman, L., Dodge, R., Rosenberg, T., and Ragsdale, D. (2003). Information assurance laboratory innovations. In *Proceedings of the 7th Colloquium for Information Systems Security Education*, Washington, DC, USA.

[Hu et al., 2005] Hu, J., Cordel, D., and Meinel, C. (2005). Virtual machine management for Tele-Lab IT-Security server. In *Proceedings of IEEE ISCC 2005*, pages 448–453.

[Hu and Meinel, 2004a] Hu, J. and Meinel, C. (2004a). Tele-Lab IT-Security: A means to build security laboratories on the web. In *Proceedings of IEEE AINA 2004*, pages 285–288, Fukuoka, Japan.

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

[Hu et al., 2004] Hu, J., Meinel, C., and Schmitt, M. (2004). Tele-Lab IT Security: An architecture for interactive lessons for security education. In *Proceedings of ACM SIGCSE 2004*, pages 412–416, Norfolk, USA.

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

[Institut für Telematik, Trier, 2001] Institut für Telematik, Trier (2001). Visualization of IT-Security features. http://www.telematik-institut.org/erklarungen_und_definitionen/visualisierung.html.de.

[Irvine and Thompson, 2004] Irvine, C. E. and Thompson, M. F. (2004). Expressing an information security policy within a security simulation game. In *Proceedings of the Sixth Workshop on Education in Computer Security (WECS6)*, pages 43–49, Monterey, USA.

[King et al., 2003] King, S. T., Dunlap, G. W., and Chen, P. M. (2003). Operating system support for virtual machines. In *Proceedings of the 2003 Annual USENIX Technical Conference*.

[Knopper, 2000] Knopper, K. (2000). Building a self-contained auto-configuring Linux system on an Iso9660 filessystem. In *Proceedings of the 4th Annual USENIX Linux Showcase & Conference*, Atlanta, Georgia.

[Lindskog et al., 1999] Lindskog, S., Lindqvist, U., and Jonsson, E. (1999). IT security research and education in synergy. In *Proceedings of the 1st World Conference on Information Security Education*, Stockholm, Sweden.

[McEwan, 2002] McEwan, W. (2002). Virtual machine technologies and their application in the delivery of ICT. In *Proceedings of the 15th Annual NACCQ*, Hamilton, New Zealand.

[McVoy and Staelin, 1996] McVoy, L. and Staelin, C. (1996). Lmbench: Portable tools for performance analysis. In *Proceedings of the 1996 USENIX Annual Technical Conference*, pages 279–294, Berkeley.

[Meinel et al., 2003] Meinel, C., Schillings, V., and Walser, V. (2003). Overcoming technical frustrations in distance education: tele-task. In *Proceedings of e-Society 2003*, pages 34–41, Lisboa, Portugal.

[Microsoft, 2004] Microsoft (2004). Microsoft virtual server 2005 technical overview white paper. Technical report, Microsoft Corporation. http://www.microsoft.com/windowsserversystem/virtualserver/.

[Neal and Miller, 2004] Neal, L. and Miller, D. (2004). *Handbook of Human Factors in Web Design*, chapter The Basics of E-Learning. Lawrence Erlbaum Associates.

[Ragsdale et al., 2003] Ragsdale, D., Lathrop, S., and Dodge, R. (2003). Enhancing information warfare education through the use of virtual and isolated networks. *Journal of Information Warfare*, 2(3):53–65.

[Richardson et al., 1998] Richardson, T., Stafford-Fraser, Q., Wood, K. R., and Hopper, A. (1998). Virtual network computing. *IEEE Internet Computing*, 2(1):33–38.

[Robin and Irvine, 2000] Robin, J. S. and Irvine, C. E. (2000). Analysis of the Intel's Pentium ability to support a secure virtual machine monitor. In *Proceedings of the 9th USENIX Security Symposium*, Denver.

[Rosenblum and Garfinkel, 2005] Rosenblum, M. and Garfinkel, T. (2005). Virtual machine monitors: Current technology and future trends. *IEEE Computer*, 38(5):39–47.

[Rowe and Schiavo, 1998] Rowe, N. C. and Schiavo, S. (1998). An intelligent tutor for intrusion detection on computer systems. *Computers and Education*, 31:395–404.

[Russell, 2002] Russell, R. (2002). *Linux 2.4 NAT HOWTO*. http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html.

[Ruth et al., 2005] Ruth, P., Jiang, X., Xu, D., and Goasguen, S. (2005). Virtual distributed environments in a shared infrastructure. *IEEE Computer*, 38(5):63–69.

[Schillings and Meinel, 2002] Schillings, V. and Meinel, C. (2002). Tele-TASK - teleteaching anywhere solution kit. In *Proceedings of ACM SIGUCCS 2002*, Providence, USA.

[Schmitt et al., 2003a] Schmitt, M., Hu, J., and Meinel, C. (2003a). Design and implementation of a PHP-based web server for the Tele-Lab IT Security. Technical report, University of Trier.

[Schmitt et al., 2003b] Schmitt, M., Hu, J., and Meinel, C. (2003b). A tutoring system for IT security education. *Journal of Information Warfare*, 2(3):79–85.

[Smith, 2005] Smith, J. E. (2005). The architecture of virtual machines. *IEEE Computer*, 38(5):32–38.

[Snyder, 1990] Snyder, P. (1990). tmpfs: A virtual memory file system. In *Proceedings of the European UNIX Users Group Conference*, pages 241–248.

[Spillman, 2002] Spillman, R. (2002). CAP: A software tool for teaching classical cryptology. In *Proceedings of the 6th National Colloquium on Information System Security Education*, Redmond, Washington, USA.

[Stallings, 2003] Stallings, W. (2003). *Cryptography and Network Security: Principles and Practice*. Prentice Hall.

[Sugerman et al., 2001] Sugerman, J., Venkitachalam, G., and Lim, B.-H. (2001). Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor. In *Proceedings of the 2001 USENIX Annual Technical Conference, Boston, MA*.

[Vigna, 2003] Vigna, G. (2003). Teaching hands-on network security: Testbeds and live exercises. *Journal of Information Warfare*, 2(3):8–24.

[Woo et al., 2002] Woo, C., Choi, J., and Evens, M. (2002). Web-based ITS for training system managers on the computer intrusion. In *Proceedings of the 6th International conference on Intelligent Tutoring Systems*, pages 311–319, Biarritz, France and San Sebastian, Spain.

[Yang et al., 2001] Yang, S. J., Nieh, J., and Novik, N. (2001). Measuring thin-client performance using slow-motion benchmarking. In *Proceedings of the 2001 USENIX Annual Technical Conference*, Boston, Massachusetts, USA.

[Ylönen, 1996] Ylönen, T. (1996). SSH - secure login connections over the Internet. In *Proceedings of the 6th USENIX Security Symposium*, San Jose, CA.

[Yuan and Strayer, 2001] Yuan, R. and Strayer, W. T. (2001). *Virtual Private Networks: Technologies and Solutions*. Addison-Wesley Professional, 1st edition.