EDUCATING SYSTEM TESTERS IN VULNERABILITY ANALYSIS

Laboratory Development and Deployment

Leonardo A. Martucci, Hans Hedbom, Stefan Lindskog and Simone Fischer-Hübner

Department of Computer Science, Karlstad University – Sweden

Abstract: This paper presents a vulnerability analysis course developed for system testers and the experiences gained from it. The aim of this course is to educate testers in the process of finding security weaknesses in products. It covers the four steps of a vulnerability analysis: reconnaissance, research and planning, mounting attacks, and assessment. The paper describes in detail ten different laboratory assignments conducted within the course. For each experiment, an overview and a description on how to run the assignment together with the expected knowledge obtained are presented. In addition, a course evaluation and lessons learned are also provided.

Key words: IT security education, vulnerability analysis, assurance, laboratory assignments.

1. INTRODUCTION

A constantly growing number of security vulnerabilities, threats and incidents has increased the efforts of IT and Telecom industry to invest in the development of more secure systems. A lack of security functionality and security assurance bears high risks for IT and Telecom vendors as it can, for instance, result in high costs for patching running platforms on the clients' sites, dissatisfied clients due to security breaches and system downtime, as well as reputation damage for the vendors' clients.

Vulnerability Analysis (VA) is an important means for improving security assurance of IT systems during test and integration phases. The approach that VA takes is to find weaknesses of the security of a system or parts of the system. These potential vulnerabilities are assessed through penetration testing to determine whether they could, in practice, be exploitable to compromise the security of the system. The Common Criteria have requirements on VA to be performed for the evaluation of systems with an Evaluation Assurance Level 2 (EAL2) or higher [1].

For improving security of their platforms before deployment, a major telecom company decided to increase their efforts in VA performed in their test labs, which requires well-educated software testing personnel. This company decided to outsource the education and training of its employees in the area of VA to an academic institution with experience in both VA and education. For that purpose, a compact VA course was

developed at our department to be held during three working days for an international and heterogeneous, in terms of knowledge in the security area, group of software testers from industry. The emphasis of this course was put on practical, hands-on experiments in VA. The course was first held in spring 2005, with 16 participants, and two times more in fall 2005 with 15 and 14 participants, respectively.

Penetration testing had been employed by our group in the past to evaluate network-based computer systems. Our intent with the test performing was primarily to find quantitative measures of operational security. Students from an applied computer security course were engaged to attack a target system and evaluate its security [2]. Experiences and lessons learned from these supervised student experiments provided us with some inputs for the preparation of this VA course.

This paper is organized as follows. First, the course content is briefly presented in section 2. Second, we provide a detailed description of the hands-on experiments, explaining the purpose and goal of each laboratory assignment and the knowledge that participants should obtain from those experiments in section 3. We present the lessons learned from the teachers' point of view and a summary of the course evaluation in section 4. Finally, we present our concluding remarks in section 5.

2. COURSE CONTENTS

As mentioned in section 1, the VA course described in this paper was developed primarily on request from the industry. The requested course was aimed for software testers with no or little knowledge about security in general and vulnerability analysis in particular, but with an extensive knowledge in software testing. Since the target group was practitioners, a practical course was requested.

The list of topics included in the VA course was, naturally, agreed with the contractor. The contractor had presented us a preliminary list of topics that should be covered by the course. This list was later transformed into a content list and a laboratory set that was discussed with a contractor's group composed of software testers, software designers, managers, industrial researchers and security experts. A three-day course (24 hours course) was selected as the best choice for the course length, since the testers should not be absent from their tasks for a long period. The course also includes a follow-up one-day recycling class after one year to update the participants about newly found vulnerabilities and VA tools. Approximately 30 to 40% of the course covers theoretical aspects and 60 to 70% is used for practical assignments. The latter was intended to give the attendees hands-on experience on how to conduct a VA of either a software component or a complete system. The course is structured in four blocks covering the theoretical part with hands-on assignments providing support to the theory. The course is divided into the four following blocks:

a) Introduction to Computer and Network Security. This block includes motivation, terminology, evaluation criteria, an overview of security standards, risk analysis, ethics and course rules.

b) Computer and Network Security Protocols and Tools. It covers cryptography, pseudo-random number generation and testing, public key infrastructure (PKI), firewalls, intrusion detection systems (IDS), virtual private networks (VPN), IPSec, SSH, and SSL/TLS. This block also includes the assignments presented in sections 3.3 to 3.5.

c) Vulnerability Analysis. An in-depth description of VA four distinct steps, i.e.: reconnaissance, research and planning, attack mounting and assessment.

d) Known Vulnerabilities, Reconnaissance Tools and Information Gathering. This block includes common host attacks, a short section on viruses, worms and anti-viruses, system hardening strategies and several practical assignments presented in sections 3.6 to 3.10.

The last assignment is a full-day final project final practical assignment that concludes the course, a "putting it all together" experiment that summarizes the full vulnerability analysis process.

3. HANDS-ON ASSIGNMENTS

In this section, we justify the selection of this list of experiments, present the learning goals and also describe how the laboratories were deployed, i.e. network topologies used, applications needed and operational systems employed. In addition, we describe how the knowledge obtained can be applied in the students' working environment in order to improve the security of a delivered product. However, we first introduce the ethical rules, the laboratory environment and the preparations that had to be done before running the course. The laboratories are described in this paper in the same order as they were presented during the course, in order to follow the theory presented in parallel.

3.1 Ethical Rules

VA course teachers clearly instructed the participants that VA strategies and penetration tools must be used for testing purposes and controlled environments only, in order to avoid intentional or unintentional harm to others. Hence, for this VA course we set up the following ethical rules that participants were requested to adhere to:

– Do not experiment with VA-tools without explicit permission of an authorized party.

– Do not pass on/publish material, tools, and vulnerabilities to unauthorized parties.

– Do not use your technical skills in criminal or ethically questionable activities.

– Always report flaws to vendors/developers first.

– Software tools provided in this course must only be used in a laboratory environment and on laboratory computers.

3.2 The Laboratory Environment

The laboratory is prepared for up to 20 students working in pairs. The workstations are dual boot - Windows XP/Linux and Fedora Core 3 Linux distribution - Intel Pentium 3 900MHz with 256MB of RAM, with one auto-sense Ethernet/Fast Ethernet network interface card (NIC). Two additional workstations were configured as servers – one running Windows Server 2000 and one running Fedora Core 3 Linux – running various services. Servers are used in some, but not all, laboratories as target systems. Moreover, the server machines are also used to store image files for the other workstations, in order to fast redeploy images if necessary.

3.3 Password Cracking

Passwords are a very basic and common authentication method. Unfortunately, passwords are not the most secure way to implement authentication, but, on the other hand, passwords are the most practical and also less expensive way to implement it. Password policies are out of scope of this assignment, even though they should be mentioned during the theoretical part of a VA course. During the implementation phase, it is common that developers use some default passwords for debugging and testing their applications before being delivered to integration and test teams in order to speedup the development phase. However, it is not uncommon that those debugging and test passwords are left in the application after the development phase is over. Those development and debugging backdoors are a serious threat to deployed platforms, as they can even grant full privileges. Detecting those built-in passwords is a must activity for testers. In this experiment, we go one step beyond detection, as students are encouraged to crack password files, in order to evaluate how easy it can be achieved with open-source tools.

Running the Assignment

In this experiment, students run a local password cracker application. There are several open source tools available for this purpose – our choice was John the Ripper v.1.6, a password cracker tool which main purpose is to detect weak UNIX passwords [3], since it is an easy to use and easy to deploy. We also provide synthetic (artificially populated) *passwd* and a *shadow* files from a Linux box. Before starting to use the tool, we provide students with some well-known rules that are used to choose good passwords, such as substituting letters for numbers, how to identify if passwords are encrypted with *crypt*( ) or hashed with MD5 and also the structure of password files. Finally, the students just have to *unshadow* the *passwd* file and run the John the Ripper tool. Since running the password cracking tool can take long time to run, it is advisable to have easy passwords added in the password file. Therefore, students can extract some passwords from the password file quickly.

Knowledge Obtained

Even though the experiment is more focused on cracking passwords, developers should first be able to detect if there are backdoors in the system, and, if possible, correct the problem and document it.

3.4 Testing for Randomness

Pseudo-random number generators (PRNG) are used in several applications with a very broad scope, from simulation and numerical analysis to gaming applications. PRNG are also a crucial cryptographic primitive and a flawed PRNG implementation can compromise the security of a whole system. A classic example was a PRNG flaw in earlier versions of the Netscape browser that could compromise the security of SSL connections [4]. Well-known PRNG have, in most of the cases, public algorithms and are submitted to several batteries of tests. In addition, the U.S. National Institute of Standards and Technology (NIST) also has a list of approved random number generators applicable to FIPS PUB 140-2 standard [5]. However, it is possible that a programmer might use a flawed implementation of a given PRNG. Therefore, it is crucial for testers to identify weak binary sequences produced by a PRNG. This is the main goal of this assignment.

Running the Assignment

In this experiment, the NIST Statistical Test Suite [6] is used to evaluate outputs from different PRNG. It runs in Windows 32bit systems. It implements 15 statistical tests and also has embedded implementations of well-known bad PRNG, such as the XOR RNG, and also NIST approved RNG, such as the ANSI X9.31. Sample data files are also provided as companion part of this test suite. Even though other test suites, such as DIEHARD [7] and ENT [8] could be used, the NIST test suite is the latest and most complete test for randomness. Moreover, installing and running the NIST Statistical Test Suite installation is straightforward and it is also freely available.

Since generating sufficiently large files of random numbers is a time-consuming activity, the sample files available with the NIST Test Suite are used, since outputs from good and also bad PRNG are provided. However, our recommendation is to produce files generated using Java classes *java.util.Random* and *java.security.SecureRandom* beforehand and also with the *rand* function from C. Therefore, students can evaluate the output quality of those PRNG, which are popular among developers.

Finally, since analyzing the outputs of the NIST Test Suite demands basic knowledge of statistics, it is crucial to provide to the students a short background on hypothesis testing before running the test suite. It is also important to briefly explain the tests and the test requirements as well, since tests have different requirements regarding the minimum number of samples and the minimum length (in bits) of each sample.

Knowledge Obtained

In order to evaluate if a given PRNG provide weak binary sequences, developers should inform and provide methods to call random number generator objects or functions in order to test them (if any are used) or, at least, document that a given PRNG is being used for security reasons. In addition, it is possible to verify if a given implementation of a given PRNG class produces weak binary sequences or not, and, eventually, validate a PRNG class. Furthermore, testing for randomness can be automated.

3.5 Firewall

Firewalls can be briefly described as filters that follow some screening policies defined by the network administrator. There are several firewall applications and boxes available in the market and even open source firewalls, such as *ipTables*. It is of ultimate importance for system integration teams to know how firewalls work, where they should be placed in a network topology and how their rules are defined and verified. The goal of assignment is to provide hands-on experience on such aspects.

Running the Assignment

This exercise is based on an assignment originally published in [12]. Students are divided into groups of two. Each group is given a description of a setup as the one described in Figure 1 and are asked to write firewall rules in Linux using *ipTables* implementing a given policy defined in the problem statement. Rules are written as if all the necessary NIC were present. Note, however, that the students are only working with virtual interfaces and not with two separate NIC.

Knowledge Obtained

When deploying systems that have a firewall, testers should be able not only to test the system from a black box point of view, but also read, understand, verify and evaluate firewall rules, in order to look for inconsistencies in the access control list. Therefore, hands-on experience is fundamental for testers in order to perform such tasks.

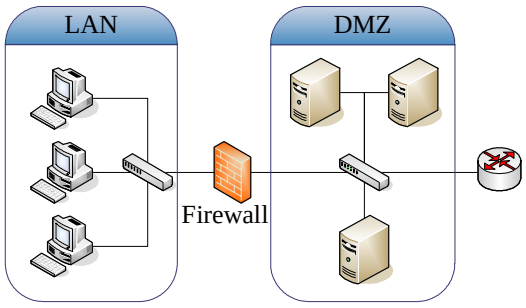

Figure 1. Network topology for the firewall assignment.

3.6 Black Box Test Method

Communication protocols are basically defined in open and proprietary standards. Those standards are usually well-known and were evaluated by a committee and also by the users' community. Therefore, finding out shortcomings in standards is not an everyday event. However, finding out vulnerabilities in implementations of communication standard is not uncommon since standards provide guidelines for implementing, but not for the coding. Thus, careless programming and also coding errors can lead to a series of vulnerabilities that can compromise a running system. Testers should therefore be able to detect such flaws in the communication protocols during test and integration phases. There are basically two ways to test the implementation of a

communication protocol. The first is to extensively review the source code in order to find out implementation flaws, but this method is cost and time inefficient. The other alternative is the black box testing approach, i.e. a functional test method that evaluates a system from the input and output point of view in order to identify system vulnerabilities. It is important to testers not only to be familiar with those tools but also how to interpret results and draw conclusions from them. In this experiment, the students are encouraged to work in larger groups and execute a black box testing against a running system.

Running the Assignment

In this experiment the PROTOS tool [9] is used as a black box testing tool against the SNMP protocol implementation of Cisco 1005 router running IOS 11.1(3). The PROTOS Test Suite c06-snmpv1 is used here to perform a Denial of Service (DoS) attack against the Cisco router. PROTOS c06-snmpv1 test suite [10] is divided into four separate test material packages: two test packages regarding packets with encoding errors and other two test packages regarding packets with application exceptions. Since PROTOS is a Java-based application, any OS can be selected, as long as a Java Virtual Machine is installed. Students were divided in two groups - each group with four workstations. The network topology used in this assignment (for each group) is shown below in Figure 2.

The goal of this experiment is to check if the SNMP implementation of this router is vulnerable to a malformed packet generated using the PROTOS test suite and also identify the vulnerability type (i.e. test material, test group and test case), if it exists. PROTOS c06-snmpv1 test suite documentation [10] is used to identify test groups. It is useful to provide a short introduction of the SNMP protocol PDU in order to provide some background information on this protocol.

In this assignment students are encouraged to cooperate in order to find out if the router is vulnerable or not to SNMP malformed packets. All workstations continuously ping the router in order to check if it alive or not. No console access to the router is provided until the test case is identified. After the test case is identified – in the particular case of this router running the IOS 11.1(3), it is a basic encoding rule (BER) error in the length field of a Get-Request SNMP PDU – students are allowed to connect to the Router using the console port and witness the router crashing after receiving just one malformed packet. Basically, PROTOS just send one test case after another, but the running process doesn't stop if the target system misbehaves. Therefore, students have to keep track of the echo reply messages, and follow PROTOS execution to finish this assignment.

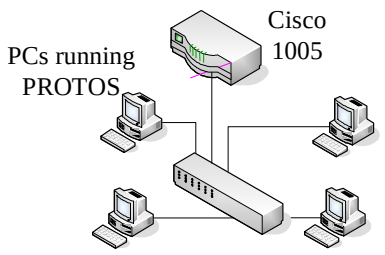

Figure 2. Network topology for the vulnerability test using the black box test method assignment.

Knowledge Obtained

Black box test strategy is very successful method to identify vulnerabilities in implementations of communication protocols. Automated testing tools are especially useful in the security evaluation of a given system because an entire test battery can be executed with a single command. The PROTOS test suite is a freely available tool, but few communication protocols are available – WAP, HTTP Reply, LDAPv3, SNMP, SIP and H.225.0. Additional test suites are available within Codenomicom [11], the commercial version of PROTOS. Using such tools, testers can quickly evaluate the robustness of protocol implementations and report vulnerabilities found back to the development team in order to fix them.

3.7 Network Analyzers and ARP Spoofing

Network analyzers are tools used to capture and analyze network traffic being transmitted in a given collision domain. These tools simply set the network interface card to promiscuous mode in order to bypass the MAC filtering rules. They are also very common tools for testers to evaluate the network traffic received and transmitted by a given system. However, testers also need to understand that some weaknesses are inherent to the protocol design. It is well-known that some protocols were designed with very little concern about security, such as FTP, TELNET and etc. Although the decision of using such protocols is not up to test and integration teams, they should be able to identify and report the existence of such protocols. A second part of this experiment is prepared in order to prove to students that knowledge of the networking device internals and network topology can impact security analysis. In the same assignment we also present an ARP spoofing experiment. The reason for this experiment is to prove to the students that some general affirmations about security are not true, and that skepticism and hands-on knowledge (in this case networking) are very important points in the security area. In this experiment we prove that the following assertion is incorrect: “Switched-based networks are obviously protected against network analyzing because each switch port is one collision domain”. Although this assignment is basically related to network security, its main goal is to prove that knowledge in network environment (i.e. topology and protocols) is definitively important to assess system vulnerabilities.

Running the Assignment

The assignment is divided in three parts, each part with a different network topology. In the first part, students verify that a very popular network protocol, the TELNET, is very weak regarding security using a network analyzer tool. The main goal of this part is to introduce a network analyzer tool to the students. In the second part, a rather insignificant change in the network topology modify test results – and it is up to the students to figure out why changes occurred – and, finally, in the last part of this assignment students test an ARP spoofing tool on a switched network, verify the achieved results and explain those results according. In this assignment we deployed in all workstations Ethereal v.0.10.11 [13], an open source (license under the GNU GPL) network analyzer. Ethereal source is available for download, and installers are available for Linux Fedora, Unix Solaris and Windows 32bits systems.

Ettercap [14] and Cain & Abel [15] are two flavors of ARP spoofing tools. Ettercap is an open source tool (source code is available, but no binaries) and have more features than Cain & Abel. On the other hand, Cain & Abel runs only in Windows 32bits platforms, but has a nicer GUI and it is easy to install, in addition, Cain & Abel is a freeware, but no source code is available. Therefore, the OS choice is up to the instructor. We considered that a better GUI is more important than a more powerful tool, at least from the didactic point of view, and, therefore, we picked Ethereal, Cain & Abel in a Windows 32bits environment. The three different scenarios created for this assignment are detailed next.

1st Part: Network Analysis I

The first part of the assignment has two goals: introduce the network analyzer (Ethereal) through a very simple scenario and also demonstrate that the design of some protocols have not taken security as a requirement – TELNET, a popular terminal emulation application is used to prove the assertion. All workstations and a Cisco 1005 router are connected to a 10Mbps hub. The instructor task is to access the router via TELNET. The network topology is depicted below in Figure 3.

The experiment goal is to use the network analyzer to capture the access password and the privileged user password. Even though this is a very simple exercise, instructors should teach beforehand how to set and write filter rules in Ethereal. In order to demonstrate the importance of filtering, the instructor should keep a regular background network traffic – that can be achieved with a continuous ping from the instructor workstation to the router, for instance.

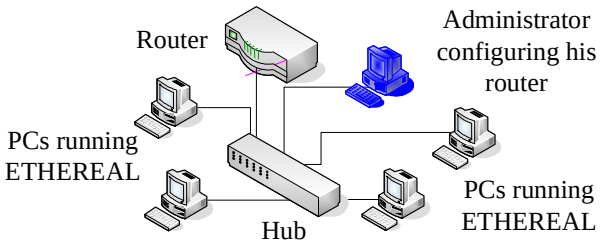

Figure 3. Network topology for the network analysis assignment - Part I and Part II.

2nd Part: Network Analysis II

The goal of the second part of this assignment is to demonstrate that knowledge of network device and its internals can make difference on a VA test. The scenario is basically the same of the first part of this assignment, presented in Figure 3: a system administrator is configuring a router using TELNET. However, the network topology has slightly changed: the 10Mbps hub was upgraded by the system administrator to a 10/100Mbps auto-sense hub.

Before starting to access the router via TELNET, the instructor must force his network interface card to 10Mbps (without students noticing it). Therefore, TELNET traffic from the instructor workstation to the router will be send in the 10Mbps backplane. Since the

NIC of the student workstations will be automatically set to 100Mbps, no TELNET traffic will be transported in the Fast Ethernet backplane of the hub, and, therefore, they will not be able to capture the traffic exchanged between the instructor's workstation and the router, but they will be able to verify that the router is up and running using ping or TELNET. Students are then invited to think and explain why such subtle change in the network topology had dramatically changed the achieved results. The answer lays in the internals of the auto-sense 10/100Mbps hub.

Since the auto-sense 10/100Mbps hub has two independent backplanes, one running at 10MHz and the other at 100MHz, it also has two independent collision domains connected by a layer 2 bridge. Thus, the traffic between the router and the administrator's workstation is transmitted only on the 10MHz backplane, but this traffic is not forward to the 100MHz backplane, where all student workstations are connected.

3rd Part: ARP Spoofing

The goal of the third part of this assignment is to prove that some common assertions on security can be proven wrong with a clear understanding of the subject – ARP (Address Resolution Protocol) protocol in the case of this assignment. A simple star topology is used, with student workstations interconnected to a switch. Since one MAC address shall be spoofed by only one machine per time period in order to capture the all traffic flow, and in order to allow all students to work in parallel, student workstations will spoof other student machines, in a logical ring topology. In other to create a constant traffic flow, students were instructed to continuously ping the workstation to their right or to their front.

The target assigned for the ARP spoofing experiment is the ICMP echo request traffic flowing from the next workstation to the right to the one next to it. Therefore, students send to the next workstation to the right ARP reply PDU in order to update its ARP table with the attacker's MAC address instead of the MAC address of the recipient. Figure 4 presents the ICMP echo-request PDU traffic flow chain, the ICMP echo-request target flow, the attacker and target workstation.

The logical chain topology depicted in Figure 4 allows all students to work in parallel in order to optimize the time to run the assignment. With such kind of topology, students play the role of attackers and victims at the same time, since while they are poisoning the ARP table of their neighbor to the right, the other neighbor is poisoning their ARP table. Cain & Abel was used as ARP poisoning tool in parallel with Ethereal, which was used to capture traffic in the network segment. However, the main goal of the experiment is not just the understanding of how such attack is performed, since it is basically straightforward as soon as the students were told that ARP is a stateless protocol.

Knowledge Obtained

Network analyzers are powerful tools that testers should be able to master, since they can provide a good insight of the data being transferred from and to a running system. However, only mastering such tools is not enough without knowledge about the test environment, as proved in the second past of this assignment, and about the protocols involved, as demonstrated in the ARP spoofing experiment. In addition, ARP spoofing is an easy to detect attack. The goal of this assignment is to demonstrate that a lot of myths

in the security area are not true, and those myths can only be proven wrong with a good understanding of the target system and its running protocols. In conclusion, it is fundamental for testers to have a deep understanding and knowledge of the test environment and of the running system especially when testing for security.

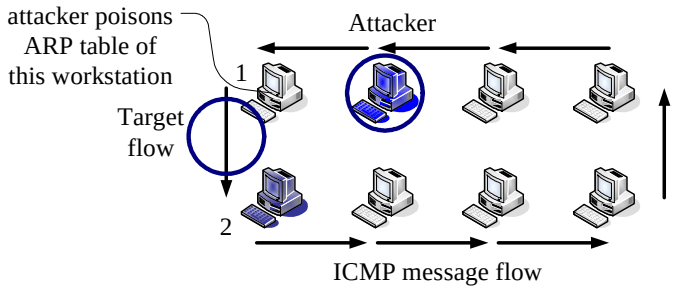

Figure 4. The ICMP echo-request traffic flow.

3.8 Port Scanning

Port scanners are used in the network reconnaissance phase of VA. Such tools basically scan one or more devices in order to find open ports that could be exploited for an attack. It is a fundamental task for testers to evaluate if unexpected open ports can be found in a system, especially during the integration phase of a system, just before deployment.

Running the Assignment

The goal of this experiment is gathering information about two running systems. Two workstations configured as servers (one Linux and one Windows server) are used as target systems. Students run Network Mapper (NMAP) [16], an open source port scanning tool, under Linux (however, it can be also compiled for Windows 32bit, since the source code is available). Servers run several network services, such as FTP, HTTP, NetBIOS (Windows server), etc. Students have to find and identify the servers in a given IP network range, since their IP addresses are not provided, find out the operational system running and identify the open ports in each server. The network topology employed is a simple star topology with all workstations and servers belonging to the same broadcast domain and in the same IP subnet.

Knowledge Obtained

Port scanners are reconnaissance tools usually employed by attackers in order to gather information about a system and identify open ports that can lead to successful break-ins. Testers should use such tools is order to identify unexpected open ports (i.e. ports that are open, but should not be), report them, and, eventually, terminate a process in order to close those ports during system integration phase.

3.9 Node Hardening

Part of the testing process is to test systems that are to be delivered and to do system tests of product. In that phase it is desirable that the testers make sure that the products are as secure as possible. One step in this securing process is to be sure that no unnecessary services are running and that the services needed run with minimal rights only. It is also vital to be sure that all the software used is patched to the latest patch level. The whole process is known as host hardening. Host hardening can be conducted in several ways, such as using checklists, experts in the area or software tools.

Running the Assignment

Since the students have a heterogeneous background regarding in-depth knowledge of different operating systems, an open source tool for the hardening process was selected for this assignment. The chosen tool was Bastille [17]. It is a node hardening software that lets the user answer a number of questions on how they want to be configured and then configures the system according to the answers. In essence it works as a high level automated checklist. It can also be used to analyze the current configuration and give suggestions on detected suspicious or dangerous configurations. Students were asked to make there own computers as secure as possible given that it still should be functional as a networked computer. They were also told that, later on, they will use a security scanner to test how secure their systems were (section 3.10). The goal of this exercise is to provide to the students an understanding of the host hardening process and the phases that are part of this process.

Knowledge Obtained

Even though there are tools that can be used in the hardening process they still rely on in-depth knowledge of the system that is to be hardened. It is also essential that these tools are up to date with the latest configuration files. Even though they can be a great help in the hardening process and also in the later verification of the hardening there are still a large manual portion of both these steps that require practical system knowledge. A system tester needs to know that in order not to be overly confident in the reports that these tools generate.

3.10 Security Scanning

A security scanner is a tool that attacks, or simulates attacks, on a system. The attacks it will perform are kept in a configuration file. The more up-to-date this configuration file is, the more accurate the scan. This means that an up-to-date security scanner will scan the system for all known vulnerabilities from the outside. Some scanners also have components that make it possible to perform attacks both from the inside and from the outside. A number of flavors exits but the most known ones are IS [18], Retina [19] and Nessus [20]. These tools are often a good starting point when doing a VA of a system since they automate the testing of the known weaknesses. The problem with these tools are generally not running them but rather how to configure the attacks and how to interpret the results, since false negatives and false positives in the attack process is highly influenced on the setup of the tool.

Running the Assignment

In this exercise, two target servers were set up. One Windows 2000 (set up as a domain server) and one running Fedora Core 3 Linux. Both not patched, running all standard services and acting as target systems for the security scanner. The Windows server ran IS and the Linux server ran Nessus. Neither the configuration nor the IP addresses of the servers were given to the students. The students were divided into groups of two. All the groups were asked to find the servers IP addresses and to find out which operating systems they were running. Students were also told that the servers were the only two network devices running an HTTP server in the network. After finding the servers, the students had to scan them and report all the vulnerabilities found. Finally, they were also requested to scan their own workstations in order to verify the hardening process that was performed earlier (section 3.8).

Knowledge Obtained

These tools are a good help to the testers in the verification process. However, they cannot be fully trusted and will inevitably generate both false positives and false negatives. A good knowledge of the system is needed in order to be able to correctly interpret the tool outputs. A tester also needs a good understanding of the tool in order to configure it correctly and customize its behavior if it is to be used on a non-standard system or with non-standard services.

3.11 Putting it all Together

Finally, we let the students put all their knowledge together in a final “grain to bread” exercise. Our goal was to make this assignment as close to their reality as we could, in a controlled environment, with limited resources and time.

Running the Assignment

In this exercise, the students were divided into groups of four. Each group was given a Fedora Core 3 Linux server with all services running. Every group also got a requirement document describing the role of the server and the security requirements on it. They were asked to find out what needed to be done in order to fulfill the requirements and also to perform the changes and verify the results. In order to do this they had access to all the tools used in the previous exercises and a list of useful internet links. They also had access to the Internet and were told that they could use it freely. Moreover, they could also use any other freely available tool found on Internet. The students had to report all the miss-configured parameters, vulnerabilities and also had to suggest changes in the system in order to adhere to the specification. The results were discussed in a summing-up session just after the exercise.

Knowledge Obtained

This assignment provided to the students a wider insight of the highly creative art of VA and it further reinforced their conclusions from previous assignments that the combination of good system knowledge, good understanding of the tools internals and their shortcomings are essential in order to correctly interpret and test a system.

4. EVALUATION AND LESSONS LEARNED

In this section, we present the course evaluation according to our feedback from the students. We also present the lessons learned from the teachers and university staff's point of view.

Just after the course conclusion, the students received a questionnaire used to evaluate the course and the usefulness of the laboratories. The students were invited to individually answer the questions following a scale from 1 (lowest grade) to 7 (highest grade). They were also invited to comment their choice. No identification field was included in the questionnaire in order to provide anonymity to the students. A summary of the achieved results is provided in Table 1, regarding 36 returned questionnaires.

Table 1: Questionnaire summary.

Questions:	Average Grade	Standard Deviation
Evaluation and usefulness of the assignment:		
Security scanning	5.8	0.9
Port scanning	5.8	0.9
Node hardening	5.6	1.1
Black box test method	5.4	1.1
Password cracking	5.4	0.7
Network analysis (Parts I and II)	5.4	0.9
ARP spoofing	5.2	1.2
Putting it all together	4.8	1.2
Firewall	4.4	1.2
Testing for randomness	4.0	1.7
Did the course fulfill your expectation?	5.1	1,1

According to Table 1, it is possible to verify that seven out of the ten assignments of the course were highly classified by the students, being graded in the interval 5.2 and 5.8. They were: security scanning, port scanning, node hardening, vulnerability analysis using the black-box method, password cracking, network analysis and ARP spoofing. The reasons reported by the students included the usefulness of these assignments in real test environments and the easy and fast deployment in the tests. However, some participants were already familiar with some security tools and others reported that some of these assignments were not particularly useful in their specific tasks.

Firewall, testing for randomness and the final assignment were considered the least interesting assignments, being graded between 4.0 and 4.8. Some considered that the time reserved for the final experiment was short, while others thought that it was no more than a sum-up of the previous exercises. However, part of the students considered it a good concluding exercise. The firewall assignment was considered not that useful because testers hardly evaluate or write access-list rules in general, so this experiment was down rated by the participants compared to other assignments. Finally, testing for randomness was considered hard to be implemented and most of the students were not comfortable with their background in statistics. Indeed, testing for randomness is a test were a security primitive is being tested, which is usually underrated by testers, that are more concerned in securing systems in an upper abstraction layer.

To the question "did the course fulfill your expectations?" the average obtained grade was 5.1 (with a standard deviation of 1.1), what was considered very good result from the teachers and the contractor's point of view, regarding the heterogeneity of the audience.

From the course backstage side, we noticed that having a system administrator available is essential during the course in order to provide assistance for the laboratories setup. This may alleviate the burden over the teachers, since a system administrator can redeploy images in the workstations or rebuild the network topology while teachers are explaining the assignments.

5. CONCLUDING REMARKS

This paper presented the outline of a practical vulnerability analysis course developed for software testers with emphasis on the descriptions of its laboratory assignments. The course evaluation clearly shows that participants were very satisfied in general with the course, and we are convinced that the course has significantly raised their awareness concerning network and computer security in general and of vulnerability analysis in particular.

However, we are not yet sure to which extent they use their new knowledge in their daily work with software and system testing. In spring 2006, during the one-day follow-up recycling class, we intend to investigate how the participants from the first course used their knowledge in vulnerability analysis when testing software products. In conclusion, we expect that the guidelines presented in this paper and the laboratory assignments are a very good reference for other academic institutions and industry to start a vulnerability analysis course in their premises in order to increase security awareness of their students and personnel.

REFERENCES

1. Common Criteria. Common Criteria for Information Technology Security Evaluation. Version 2.2. Jan. 2004. Available at: <www.commoncriteriaportal.org/>. Accessed in: Aug. 29th, 2005.

2. Lindskog S., Lindqvist, U. and Jonsson E. IT Security Research and Education in Synergy. In: L. Yngström, S. Fischer-Hübner (Eds). Proceedings of the IFIP TC11 WG11.8 First World Conference on Information Security Education - WISE'1, p.145-162. Kista, Sweden, Jun. 1999.

3. Openwall Project. John the Ripper Password Cracker v.1.6. Available at: <www.openwall. com/john/>. Accessed in: Jul. 25th, 2005.

4. Goldberg, I. and Wagner D. Randomness and the Netscape Browser. Dr. Dobb's Journal, n.260, Jan.1996.

5. National Institute of Standards and Technology. Annex C: Approved Random Number Generators for FIPS PUB 140-2, Security Requirements for Cryptographic Modules. Federal Information Processing Standards Publication 186-2. Draft. Gaithersburg, MD, USA: NIST. Jan. 2000.

6. National Institute of Standards and Technology. NIST Statistical Test Suite version 1.8. Available at: <http:// csrc.nist.gov/rng/rng2.html>. USA. Accessed in: Jul. 21st, 2005.

7. Marsaglia, G. DIEHARD Battery Test for Randomness. Available at: <www.stat.fsu.edu /pub/diehard/>. USA. Accessed in: Jul. 21st, 2005.

8. Fourmilab Switzerland. ENT – Pseudo-Random Number Sequence Test Program. Available at: <www. fourmilab.to/random/>. Switzerland. Accessed in: Jul. 21st, 2005.

9. University of Oulu. PROTOS Security Testing of Protocols Implementations. Available at: <www.ee.oulu.fi/ research/ouspg/protos/index.html>. Finland. Accessed in: Jul. 25th, 2005.

10. University of Oulu. PROTOS Test Suite: c06-snmpv1. Available at: <www.ee.oulu.fi/research/ouspg/protos/ testing/c06/snmpv1/>. Finland. Accessed in: Jul. 26th, 2005.

11. Codenomicon Ltd. Codenomicon. Available at: <www.codenomicon.com>. Finland. Accessed in: Jul. 26th, 2005.

12. Mixter. Gut Behütet. C'T - Magazin für Computer Technik. v.4, p.202-207. Heise, Germany, 2002.

13. Ethereal. Ethereal: A Network Protocol Analyzer v.0.10.11. Available at: <www.ethereal. com>. Accessed in: Jul. 27th, 2005.

14. Ettercap. Ettercap NG-0.7.3. Available at: <http://ettercap.sourceforge.net/>. Accessed in: Jul. 27th, 2005.

15. Oxid.it. Cain & Abel v.2.7.3. Available at: <www.oxid.it/>. Italy. Accessed in: Jul. 27th, 2005.

16. Insecure.org. NMAP - Free Security Scanner for Network Exploration & Security Audits. Available at: <www.insecure.org/nmap/>. Accessed in: Jul. 28th, 2005.

17. Bastille-Linux. The Bastille Hardening Program. Available at: <www.bastille-linux.org/>. Accessed in: Aug 31st, 2005.

18. Internet Security Systems. Internet Scanner. Available at: <www.iss.net/products_services/enterprise_ protection/vulnerability_assessment/scanner_internet.php>. Accessed in: Aug. 31st, 2005.

19. eEye Digital Security, Retina Network Security Scanner. Available at: <www.eeye.com/html/Products/ Retina/index.html>. Accessed in: Aug. 31st, 2005.

20. Nessus Open Source Vulnerability Scanner Project. Nessus. Available at: <www.nessus.org/>. Accessed in: Aug. 31st, 2005.