An Overview of Penetration Testing

Chiem Trieu Phong, Auckland University of Technology, Auckland, New Zealand

Wei Qi Yan, Auckland University of Technology, Auckland, New Zealand

ABSTRACT

Penetration testing is an effort to attack a system using similar techniques and tools adopted by real hackers. The ultimate goal of penetration testing is to call to light as many existing vulnerabilities as possible, then come up with practical solutions to remediate the problems; thus, enhance the system security as a whole. The paper introduces concepts and definitions related to penetration testing, together with different models and methodologies to conduct a penetration test. A wide range of penetration testing state-of-the-art, as well as related tools (both commercial and free open source available on the market) are also presented in relatively rich details.

Keywords: *Hackers, Information Security, Penetration Testing, Solutions, Vulnerability Assessment*

1. INTRODUCTION

As pointed out by Fahmida (2011), despite the fact that cyber attacks and malwares have been rocketing in this century of information, many companies and organizations today are not proactively testing their infrastructure to identity security vulnerability. Once connected to the Internet, companies' systems can be probed, scanned, and even attacked constantly with the proliferation of free hacking tools and inexpensive devices like key loggers and Radio Frequency scanners (Chan & Schaeffer, 2008, p.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 different 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 system 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. 569-595). In other words, the best way to determine how secure a system is to attempt to break into it. This is where the term *penetration testing* makes its appearance. The following section of this report introduce

DOI: 10.4018/ijdcf.2014100104

several definitions as well as concepts related to penetration testing, while section 3 covers different methodologies to perform a penetration test, together with various penetration testing models. Section 4 presents a wide range of penetration testing tools available (either free open source or commercial) for further reference. Finally, issues related to penetration testing are concluded in the last section.

2. PENETRATION TESTING DEFINITIONS AND CONCEPTS

2.1. Definitions and Concepts

With regard to penetration testing, there is a wide range of different definitions. As defined by Bacudio *et al.* (2011, p.19) and Ke *et al.* (2009), penetration testing is a series of undertaken activities to identify and exploit security vulnerabilities. It is security testing attempting to circumvent security features of a system (Wack *et al.*, 2003). The system here 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, a penetration testing can be defined as "an effort to penetrate a system in order to demonstrate that protection has weaknesses" (Cohen, 1997, p. 12-15).

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". A penetration test is an analysis of IT environment and search for exploitable vulnerabilities (Nicola, n.d.). It emphasizes on how deep one can get into the system.

Another interesting point to look at penetration testing is to reckon it as simulation of hacker's behaviors. "Penetration testing is properly defined as the simulation of attacks like real hacker against target systems to identify security vulnerabilities without false positive results," according to Tran and Dang (2010, p. 72-85). 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." To aid this, Hardy (1997, p. 80-86) 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."

To be more detailed, the term "hacker" refers to any person who illegally accesses the IT system of an organization without authorization (Naik *et al.*, 2009, p. 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 intention to take advantage of weaknesses of 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 easiest targets they can find with any available tools obtained from the Internet.

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 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 a system.

2.2. Goals of Penetration Testing

One point that any penetration testing practitioner should deeply bear in mind is that "ALL

SYSTEMS ARE VULNERABLE!" (SANS Institute, 2002). There is no system which is 100% secure either now or in the future.

Thus, one of the ultimate 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). Similarly, Wack *et al.* (2003) states that the aim of penetration testing is to determine methods of gaining access to a system by using common tools and techniques adopted by hackers.

Nevertheless, most 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, p. 15-17), as well as not to carry out some kind of "crime-watch" style reconstruction of real hacking attempts (Midian, 2002b, p. 10-12); but to provide countermeasures for found vulnerabilities, and meaningful advices to harden their security Ke *et al.* (2009).

As a result of this clarification, hackers and ex-hackers who are seemingly 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 that hiring hackers or ex-hackers to conduct a penetration test is definitely not wise decision for companies because they lack for 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 are proved incompatible (Schultz, 1997). 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).

2.3. Benefits and Drawbacks 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 demonstrating security weaknesses in the system (Budiarto *et al*., 2004). More importantly, penetration testing provides evidence to alert the management to the need of taking information security more seriously (Hardy, 1997, p. 80-86); 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, p.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 kind of 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 one will be developed to raise security awareness in not only education, but also in working environment.

Despite a great number of advantages mentioned above, the effectiveness of penetration testing is questionable due to several negative effects (Cohen, 1997, p. 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 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, p. 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. 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 only be conducted by highly-trained and knowledgeable security professionals with appropriate 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.

2.4. 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, p. 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. Particularly, with *social engineering* approach, there are a large number of tactical techniques, 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.

Generally, physical penetration testing is as important as technical penetration testing as 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 particular 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.

2.5. Vulnerability Assessment versus Penetration Testing

Vulnerabilities are defined as potential security weaknesses, or design flaws, bugs, or misconfigurations 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, p. 10-11). The most common way to conduct a penetration test is to attempt to penetrate 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 first 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.

2.6. 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 need serious protection 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 first 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*, *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 web application. However, in fact, black-box penetration testing is the mostly used approach for web developers to inspect a web application.

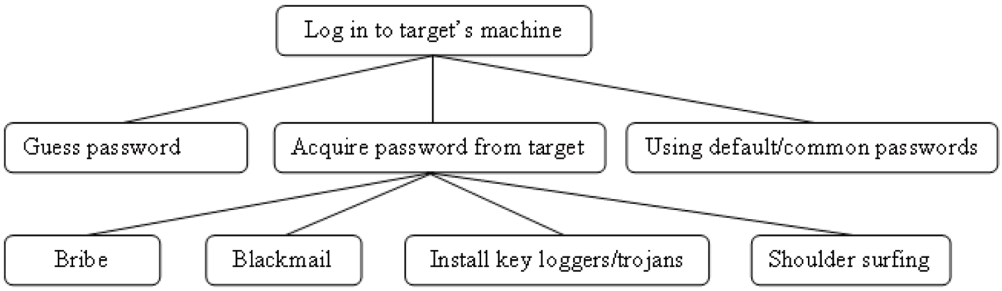
3. PENETRATION TESTING MODELS AND METHODOLOGIES

3.1. Penetration Testing Models

A wide range of different models are adapted to 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 (Weissman, 1973) is now in general use and remains the best current approach to penetration testing of new products at the end of development. 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 planned by setting the scope, establishing ground rules and objectives, as well as defining the purpose. Then, a back-

Figure 1. Example of an attack tree



ground 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 starts the brain storming sessions to generate hypothetical flaws. These hypothetical flaws 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, fixes to remediate the flaw 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.

Figure 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. 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 test-

ing 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.

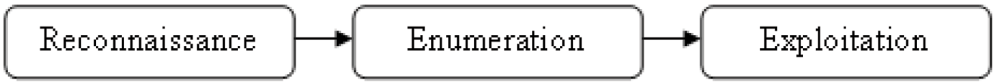
Despite the capability to avoid both false positive and false negative, *MBVT* still suffers the main drawback of the traditional MBT which is the significant effort needed to design test purposes, models, and adapters.

Last but not least, a powerful approach called *Topological Vulnerability Analysis (TVA)* for global network vulnerability analysis is proposed by Jajodia *et al.* (2005, p. 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.

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

Figure 2. A simple penetration testing methodology



penetration testing, software penetration testing, and social engineering penetration testing.

3.2.1. Network/System 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, p. 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* as shown in figure 2.

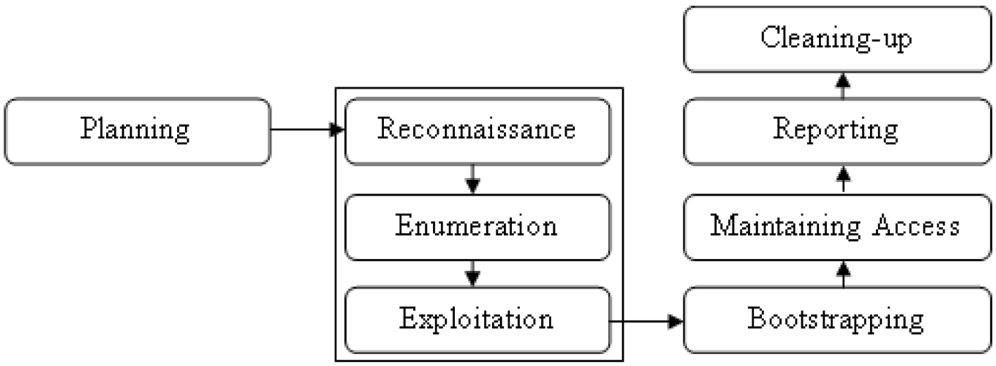
Reconnaissance, or *Recon*, occasionally referred to as *Information Gathering* step, is the process of searching for available information used in a penetration test (Tiller, 2011). Depending on the scope of a penetration test, reconnaissance activities can be ranged from ping sweeps to discover IP 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 infiltrate 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 3, a formal penetration test built around the three mentioned core steps, usually includes more sub-activities.

Figure 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 issue together with its potential impacts, security level rating of the issue, 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, p. 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, p. 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.

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, p. 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.

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, p. 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, p. 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, p. 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 in-

tegrated 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.

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, p. 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.

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.

4.1. *Google*: A Hacking Tool?

When mentioning *Google*, most people would regard 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.

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 name 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.

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, p. 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.

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, p.44-46). 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).

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 is *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.).

4.6. Popular 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 is Kali Linux which 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.

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 penetration testing, one of the better testing system 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*.

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 from 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 smartphones, 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.

6. CONCLUSION

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, p. 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 very few studies focusing on this issue, for example Austin *et al*. (2013, p. 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 choosing a penetration testing product.

Summing up, penetration testing is essentially significant for organizations to fortify their system security. With an appropriate approach coupled with suitable 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 academic 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.

REFERENCES

Alder, V., Burke, J., Keefer, C., Orebaugh, A., Pesce, L., & Seagren, E. S. (2007). *How to cheat at configuring open source security tools*. Elsevier.

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, . (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.

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

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. doi:10.1109/PRDC.2009.54

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: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), 19–38. doi:10.5121/ijnsa.2011.3602

Barrett, N. (2003). Penetration testing and social engineering: Hacking the weakest link. *Information Security Technical Report*, *8*(4), 56–64. doi: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*. CPNI.

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.

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/net-forensics/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*(3), 195–214. doi:10.1002/stvr.450

Hardy, G. (1997). The Relevance of Penetration Testing to Corporate Network Security. *Information Security Technical Report*, *2*(3), 80–86. doi:10.1016/S1363-4127(97)89713-0

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; 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

Midian, P. (2002a). Perspectives on Penetration Testing. *Computer Fraud & Security*, *2002*(6), 15–17. doi: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:10.1016/S1353-4858(02)11009-9

Midian, P. (2002c). Perspectives on Penetration Testing — What's the Deal with Web Security? *Network Security*, (8): 5–8. doi:10.1016/S1353-4858(02)07007-1

Moyer, P. R., & Schultz, E. E. (1996). A systematic methodology for firewall penetration testing. *Network Security*, *1996*(3), 11–18. doi: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.

Nicola, C. U. (n.d.). *Tools for penetration tests 1*. Retrieved from http%3A%2F%2Fweb.fhnw.ch%2Fplattformen%2Fns%2Fvorlesungsunterlagen-1%2Fnetwork-analysis-tools%2Fnetworktools.pdf&ei=4dO_VLmDNsmxggSS2YL4BQ&usg=AFQjCNGSCv_UOAxyEXwmAvHj4JcA4VcXSA&sig2=v-VaAt_8IhaUHqruQBoiRw&cad=rja

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

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? *New World (New Orleans, La.)*, *22*(48), 44.

Pfleeger, C. P., Pfleeger, S. L., & Theofanos, M. F. (1989). A Methodology for Penetration Testing. *Computers & Security*, *8*(7), 613–620. doi:10.1016/0167-4048(89)90054-0

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. doi:10.1109/MSP.2004.84

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*. SANS Institute.

Schultz, E. (1997). Hackers and penetration testing. *Network Security*, *10*(10). doi: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

Stiller, J. S. (2005). *The ethical hack: A framework for business value penetration testing*. Auerbach Publications.

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.

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

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*. Santa Monica, CA: System Development Corporation.

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