

Application Penetration Testing: Concepts and Taxonomy

John Melton, Gail-Joon Ahn

University of North Carolina at Charlotte

jtmelton@uncc.edu

Abstract

It seems there is a new bug or critical vulnerability announced daily in some piece of software. Application developers are creating new software without security in mind and are inevitably perpetuating the cycle. All the while, there are countless attackers devising new ways to break networks and applications. It sometimes seems like a hopeless situation for information security professionals.

How do these attackers gain access to systems with seemingly relative ease? While the answers may vary in technique, one common element in all is a gap or “hole” found in the application or network that the attacker slips through to gain partial or complete access. How do information security professionals combat these attacks? The answer often is as simple as “patching the holes” after using a technique known as penetration testing.

This paper identifies some of the techniques that can be used to perform penetration testing to help secure applications. A taxonomy is also provided to allow professionals to identify needed functionality when acquiring existing tools as well as developing new tools.

Keywords

Penetration testing, Application penetration testing, Zero-knowledge attack, Full-knowledge attack, Discovery, Enumeration, Vulnerability mapping, Exploitation

1. Introduction

Penetration testing, simply put, is finding system vulnerabilities through simulations of real-world attacks. Companies can hire outside attacking professionals or use their own professionals to attempt to attack various components, including applications and networks. Finding vulnerabilities is an essential step that can help developers to patch any necessary “holes”, which will result in a more secure product.

The software development community as a whole has not been able to create produce software that is safe and reliable in today’s environments. This research then aims to provide methods to secure applications once they have been released. The way this is done is through penetration testing an application. The results of the process should then be used to enhance the knowledge of the development

community. This is then simply a logical step in the evolution of software security.

This research will focus specifically on application penetration testing using open source [Open Source Initiative] applications. To perform the research, many of the open source tools currently available in the penetration testing field were reviewed to examine functionality. This was then used to further develop the ideas of what types of features should be considered when creating metrics and a taxonomy.

The rest of this paper is organized as follows. In section 2, we briefly overview penetration testing as well as relevant issues such as risks, operational steps and advantages. Section 3 discusses application penetration testing (APT) followed by the characterization of APT introducing metrics, taxonomy and a brief comparison between APT techniques in section 4. In section 5, we describe our preliminary usability study to demonstrate how well our taxonomy helps secure computerized organizations. Section 6 concludes this paper.

2. Penetration Testing: Overview and Issues

2.1 Penetration Testing Types

Penetration testing can be performed whether the tester has zero or total knowledge of the application.

Zero-knowledge attack is an attack where the attacker has little to no knowledge of the application. This type of attack tends to accurately simulate the most typical styles of attacks that would occur on a system since most attackers often know little about the internal design of a targeted application. A third-party group is typically hired to perform the test since they would know little about the application. (note: This type of attack correlates to what is known as “black-box testing”)

Full-knowledge attack is an attack where the attacker has as much information about the targeted application as possible. An example of this type of an attacker might be a disgruntled employee who knows the application’s design. In this simulation, an employee with knowledge of the application may be asked to attempt to attack the application. (note: This type of attack correlates to what is known as “white-box testing”) [Scambray, McClure, Kurtz 2003]

2.2 Penetration Testing Risks

While penetration testing may seem ideal and beneficial, there are some risks associated with it.

Exposure to the public – By using a third-party attacker, your application vulnerabilities could be exposed to the outside world. Leaking this valuable information to the public by that third-party could cause problems with public image and lower the trust level of customers. This risk can be mitigated through contractual agreements, but never is fully avoided.

False sense of security – Though application penetration testing can expose vulnerabilities, it is not likely to find all “holes” in an application. However, some may feel a level of comfort that the system has been tested, so it must be secure. This is simply false and can cause problems when assumed.

Unskilled testers – If the application is tested in-house or by a known third-party, there needs to be a proper level of testing knowledge. Amateur testers have often crashed networks, exposed confidential and critical information, and destroyed valuable digital assets in previous years.

Future attackers created – Penetration testing also provides a venue to train malicious attackers by teaching employees essentially “how to hack”. An employee supervising a penetration test could figure out basic attacking techniques and, in turn, attack another system when they practice those techniques at home. This is not of great concern, since much of this information is available freely on the Internet as well as in print media, but the risk does exist.

2.3 Penetration Testing Steps

To effectively expose weaknesses and therefore increase security, penetration testing must be performed in a structured manner using well-defined steps. These steps include the following: discovery, enumeration, vulnerability mapping, and exploitation. [Scambray, McClure, Kurtz 2003]

Discovery – In order to successfully identify vulnerabilities in applications, knowledge of that application is needed. Obtaining this knowledge is performed during this stage. Knowledge such as application descriptions, documentation on usage, changelogs, etc. can be found using the Internet, company employees, data files, etc. In attackers’ linguistics, this step is also known as “footprinting”.

Enumeration – As an extension of the Discovery step, the Enumeration step involves a further detailed study of not only the application, but also the domain names, network, and systems. An example might be determining the difference between one version of the software and the latest, updated version of the targeted software. Knowing why the update occurred will help in determining the actions needed to find the holes in the system.

Vulnerability Mapping – The next step of penetration testing is publicly mapping out all known or unknown vulnerabilities through the knowledge gained during the Enumeration and Discovery phases. This step tends to be a tedious and mundane process because inadequate or inaccurate information from the first two steps could cause problems when attempting to map out vulnerabilities. If only a few vulnerabilities are mapped, additional research may be needed and the first two steps would be performed again. This process would reoccur until the testers are comfortable moving on to the final step of penetration testing.

Exploitation – Once a map of all vulnerabilities has been created, the attacker will attempt to break the application. An example could be the attacker launching a password guessing attack on an application based on knowledge found in the previous steps revealing that the particular version of this application is vulnerable to an offline dictionary attack.

2.4 Penetration Testing Advantages

Penetration testing has its obvious benefits: determine the applications’ vulnerabilities, prevent production system attacks by repairing the “holes”, etc. However, there are additional advantages that penetration testing provides.

Information security as a field has historically had difficulties in providing concrete evidence of its usefulness. The common adage is “If it’s working, you won’t know it.” This simply means that security strives to keep malicious events from occurring. If those events do not occur, how do you prove that they would have even if you did not have the security in place? Application penetration testing gives concrete evidence of the usefulness of information security.

In addition, application penetration testing helps to provide a return on investment value for information security. This is important to ensure that

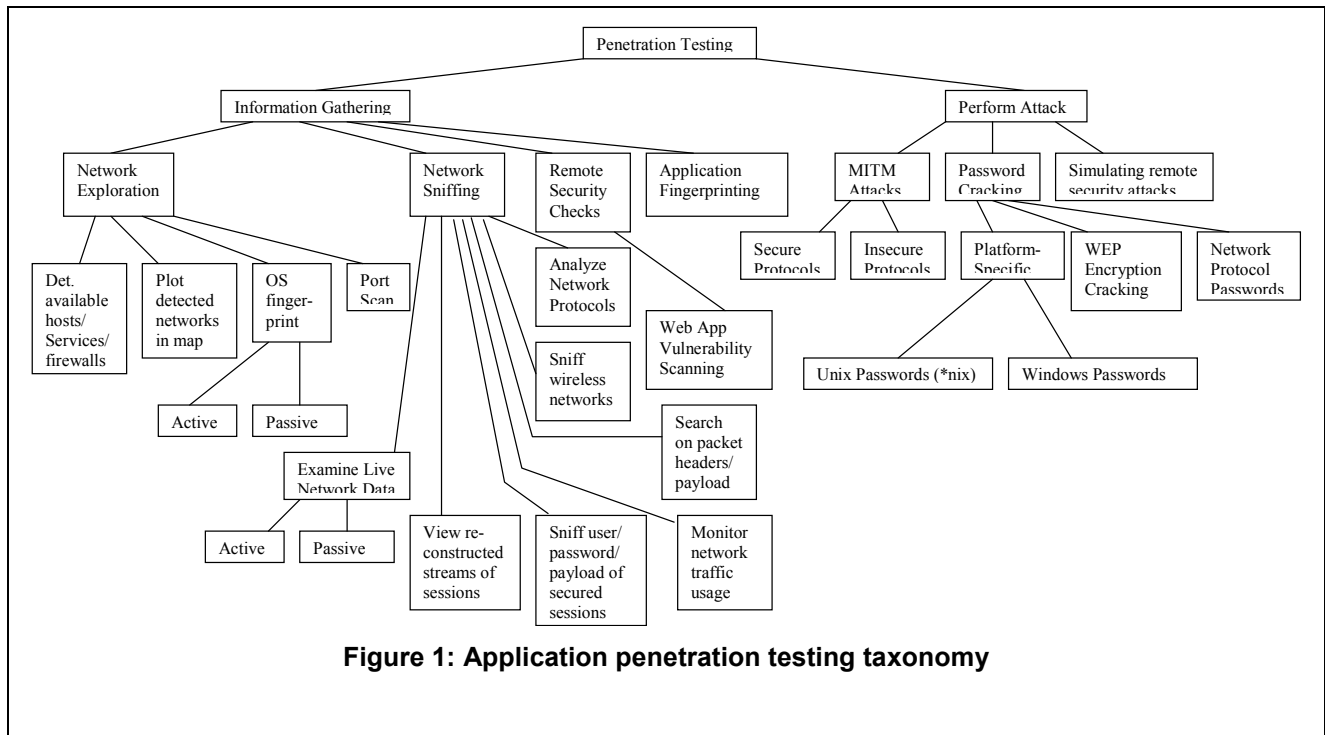


Figure 1: Application penetration testing taxonomy

corporations sustain focus on information security and that there is continued support of the field.

3. Application Penetration Testing (APT)

Application penetration testing is a subset of the penetration testing field focused specifically on applications. It can be described as the portion of security testing in which the evaluators attempt to circumvent the security features of an application. The evaluators may be assumed to use all system design and implementation documentation, that may include listings of system source code, manuals, etc. The evaluators work under the same constraints applied to ordinary users. [Digital Guards Glossary]

One of the main benefits of this research is the collaboration of information into a usable framework. There is a great amount of information available about application penetration testing, but in widely separated places. This research organized the disparate information into a usable resource. The goal is to develop a set of metrics and a usable taxonomy similar to what was done by Landwehr et. al with “A Taxonomy of Computer Program Security Flaws”. [Landwehr et. al] This requires detecting application flaws through the use of application penetration testing in an organized and logical manner.

4. APT Characterization

This section will define the major requirements of Application Penetration Testing tools before they could be included in this research.

The tools discussed in this paper and the taxonomy developed below form a dynamic framework that can be tailored for use in several contexts. The tools mentioned represent a sample of the more common and well-known application penetration testing tools available today. The taxonomy classifies tools based on several factors including active vs. inactive procedures, as well as the steps of the application penetration testing process. The framework then provides an easily understandable outline of the proper usage of application penetration testing. This framework can be applied in several contexts, including: red teaming, external security audits, and internal security audits. Many of the tools available offer enterprise-level performance, which means this framework can be used by individuals, small and medium businesses, or even large corporations.

4.1 Metrics

When considering what metrics should be used to classify application penetration testing tools, one must account for a wide array of functionality as there are many tools available and many possible combinations. This paper primarily bases the categorization on those tools that are used in *information gathering* versus those used in an actual *attack*.

The tools classified under *information gathering* are further broken down into those used for network exploration, network sniffing, performing remote security checks (including web application vulnerability scanning), and application fingerprinting.

Tools in the network exploration category offer functionality such as Operating System fingerprinting, the ability to determine available hosts/services/firewalls, etc., as well as port scanning. Operating system fingerprinting is the process of determining what operating system is being used by a particular system. Port scanning is a way of systematically scanning a computer to determine what ports are listening for connections. [Webopedia] Additional functionality in this section is offered by network exploration tools that map graphically what has been found in the exploration process. This gives the user a much more human readable visualization of the status of the network.

Tools in the network sniffing category offer functionality such as analyzing network protocols, examining live network data (both actively and passively), examining data from a capture file on disk, searching network data based on packet headers and payload, viewing reconstructed streams of network sessions, sniffing the username/password pair in a useful way, sniffing the data of a secured session, sniffing wireless networks, and monitoring network traffic usage. There are an array of tools in this category that range from the very basic functionality of viewing exactly what data is being passed across the network connection to intelligently identifying what that data means, up to and including analyzing the traffic for security breaches. The searching tools are useful to aggregate data quickly to perform analysis on a particular protocol or even data from a single machine.

The tools classified as performing an actual *attack* are further broken down into man-in-the-middle attacks, password-cracking attacks, and simulated remote security attacks. A man in the middle attack is a computer security breach in which a malicious user intercepts, and possibly alters, data traveling along a network. [Word Spy 2002] Man-in-the-middle attacks can be performed against secure as well as insecure protocols. Password-cracking can be categorized into platform-specific tools, primarily, Unix-based systems(*nix), and Windows-based systems. There are a plethora of password-cracking tools that perform attacks on various network protocols and against many standalone applications.

All of the functionality mentioned above is offered by open-source tools that are currently available.

4.2 Taxonomy

The taxonomy listed in Figure 1 classifies several tools based on the metrics listed above. (Note: These tools could be broken down further into platform-specific, environment-specific, or any other number of classifications. Also, many currently available tools can be located in multiple categories based on capabilities.) This taxonomy should act as a guide to what functionality to look for when attaining an existing tool. It can also serve as a starting point of ideas for development of new tools.

4.3 Comparison

There are some clear overlaps as well as differences in the tools classified as *information gathering* tools versus those classified as *actual attack* tools.

There is also a good amount of overlap between *information gathering* and *actual attack* tools simply because they are often performing similar tasks. For example, an attack tool might need to perform some information gathering activities to narrow the scope of the attack and make it more effective. Some currently available tools try to only focus on one specific area, whereas others may perform both *information gathering* and *actual attack* scenarios.

Inside the information gathering group, there are several similarities between the major subcategories: network exploration, network sniffing, remote security checks, and application fingerprinting. All of the tools in these categories are generally network-aware. The tools in the network sniffing category are the traditional passive tools. They would simply listen to what is being passed through the network, and possibly do some level of analysis depending upon the individual tool. The tools performing remote security checks and application fingerprinting tasks would usually need to have some of the network exploration functionality built-in or rely on the output of a network exploration tool to perform their tasks. This allows the tool to limit the focus to only relevant systems. The actual attack group carries some similarities with the information gathering attack group. The man-in-the-middle category of tools usually offers similar functionality to the network sniffing category of the information gathering group. This is because a man-in-the-middle attack must be able to capture the data crossing the network in order to alter it to perform the attack. The remote security attack simulation category also shares many similarities to the remote security checks category, only going a step further by

actually attempting to perform the attack. Both the man-in-the-middle attack and remote security attack simulation categories are network-aware which is also true for tools in the information gathering category.

The information gathering and attack categories also have several differences between them. First, the tools used for information gathering are more often, but not always, passive tools. This is contrasted with the *actual attack* tools which are active in nature and try to obtain information by performing some intrusive action. Also, *information gathering* tools often are able to perform several different tasks or perform one task in several ways, such as a network sniffer that can listen to and analyze various protocols. However, *actual attack* tools tend to, as they should, be more focused on one specific protocol or one vulnerability. A noted exception here would be a password-cracking tool that is able to perform attacks against various platforms, protocols, or applications. The password-cracking tools are often not network aware, that is unless of course, the tool is meant to crack a password for a network protocol.

Though the branches overlap and differ somewhat, they are essentially complements to one another where the *information gathering* tools offer their output as input to the *attack* tools. This simply means that the data captured through the information gathering phase is a direct source of useful information for the attack phase.

5. Preliminary Usability Study

Consider two companies: company A and company B. Company A is a small technology firm with 25 employees that does web development and are planning to host a couple of websites on the company network. Company B is a large financial institution struggling in a couple of their corporate offices with wireless networking issues.

Let us first consider company A. The issue here is that the company will be running new services and allowing new, but valid, traffic into the internal network. (or demilitarized zone[DMZ] depending upon configuration) The company would like to know the risks that it will face in starting this new service. After doing some initial research into the issue, and configuring the web servers for a production environment, company A may put the web server into its internal network. (or a separate network disconnected from the primary network for testing) Using the taxonomy provided above (figure

1), company A can then traverse the hierarchy to find that they need some of the functionality from the network exploration branch under *information gathering*. Since the sites that would be hosted would be general information sites not hosting any secure information, there is no need for network sniffing tools. There would, however, be a need for remote security checks, namely web application vulnerability scanning, which is of paramount importance in this context. Finally, company A might try some of the remote security attacks from the *actual attack* branch of the hierarchy. (View figure 1 to see classification of hierarchy)

Let us now consider company B. With the booming growth of wireless technology in today's marketplace, there are many who fear that the technology is moving too quickly for security to keep up. There are many stories concerning wireless security that keep information security professionals up at night. When thought of in the context of critical infrastructure, such as the financial sector, this is frightening. Wireless technology, however is a driving force today as it eases the task of many professional's day-to-day lives. What then can company B do when forced to deal with this situation? Again, using the taxonomy from Figure 1, we see that the network exploration tools under the *information gathering* branch would be very helpful. In particular, the tools that plot detected networks in a map form could be very helpful to provide a graphical view of the dynamic network layout at any given time. This could help company B notice when new wireless LANs are added to the network. From the network sniffing category, tools that analyze network protocols could be helpful, allowing company B to look for wireless-specific protocols if any exist. Also, those tools that sniff wireless networks would be helpful. Lastly, from the *actual attack* branch, under the password cracking category, the WEP encryption cracking tools could be used to show the insecurity of the network. Another possibility here is that company B could build an in-house tool that works well, and would like to release the tool to the public. Company B is then able to look through the taxonomy locating a place for their tool under plotting detected networks in a map, then under a subcategory there of plotting wireless networks. (View figure 1 to see classification of hierarchy)

There are some lessons that can be learned from these examples. First, existing security practices can help to secure a network or applications in a general way. However, it is useful to be able to see the environment from an attackers' point of view. This

is realistically how it will look from the outside. The taxonomy can be applied to show what classes of tools should be used in this process. The process can be used for a variety of situations, including testing new deployments and auditing existing systems.

Second, the taxonomy is fairly broad, which is a requirement for the desired extensibility feature. This means there is a plethora of categories that can be found in the taxonomy. If, however, the taxonomy does not offer a specific category for the particular functionality in question, it can easily be extended to add that category and improve the overall usefulness of the taxonomy in general.

Third, the ultimate goal of this research is for the taxonomy to be applied in a very practical way. These examples show that it is an effective method that can be used in several contexts and is easily extensible for new applications.

6. Conclusion and Future Work

This paper has provided a brief introduction to penetration testing and the concepts around it, and more specifically, to application penetration testing. The risks and benefits of penetration testing were also discussed, along with the steps of penetration testing.

Requirements were then discussed in the context of tools that could be classified into a taxonomy for application penetration testing. A set of metrics was offered describing how the taxonomy would be organized, then the taxonomy was developed and offered in Figure 1. A comparison based on the metrics was then performed.

Finally, a case study was performed using company A, a small technology firm, and company B, a large financial institution. The technology firm had to deal with offering a new service on their network for clients and ensuring that the service was secure. The financial institution had to deal with issues of wireless security.

Some future work that was considered lies in the taxonomy. The taxonomy could be developed further to account for more fields within application penetration testing while still being flexible and extensible. Also, this research could be incorporated into research in secure development methodologies. Considering what tools and processes are effective for attacking an application offers insight into what areas of development need to be improved upon. The output of this research could then be reviewed and

incorporated into a secure development curriculum in a manner similar to “lessons learned”.

This paper has shown that there is a useful service provided by application penetration testing. When done in an organized manner using a logical taxonomy, this type of testing can prove to be an invaluable tool in the goal of ensuring the protection and security of networks, applications and the information flowing through them.

7. References

- [1] Open Source Initiative. The Open Source Definition. [Online Document]. Available URL: <http://www.opensource.org/docs/definition.php>
- [2] Digital Guards Glossary. [Online Document]. Available URL: <http://www.digitalguards.com/glossary.htm>
- [3] Landwehr et al. Naval Research Labs. 1994. A Taxonomy of Computer Program Security Flaws, with Examples, [Online Document]. Available URL: <http://chacs.nrl.navy.mil/publications/CHACS/1993/1993landwehr-NRLFR9591.pdf>
- [4] Scambray, McClure, Kurtz. February 25, 2003. Hacking Exposed: Network Security Secrets and Solutions, Fourth Edition. McGraw-Hill Osborne Media.
- [5] Word Spy. March 22, 2002. Man in the middle attack, [Online document]. Available URL: <http://www.wordspy.com/words/maninthemiddleattack.asp>
- [6] Webopedia. Port scanning definition, [Online document]. Available URL: http://www.webopedia.com/TERM/p/port_scanning.html
- [7] Fyodor. May, 2003. Top 75 Network Security Tools, [Online document]. Available URL: <http://www.insecure.org/tools.html>
- [8] Naval Postgraduate School and Cebrowski Institute for Information Innovation and Superiority. 2003. The Nemesis project equipment list, [Online document]. Available URL: <http://ci.nps.navy.mil/Nemesis/equipment.html>