Effective penetration testing with Metasploit framework and methodologies

Filip Holik, Josef Horalek, Ondrej Marik, Sona Neradova, Stanislav Zitta
Faculty of electrical engineering and informatics
University of Pardubice
Pardubice, Czech Republic
josef.horalek@upce.cz, ondrej.marik@student.upce.cz, sona.neradova@upce.cz, stanislav.zitta@student.upce.cz

Abstract — **Nowadays, information security is very important, because more and more confidential information, like medical reports, is being stored electronically on computer systems and those systems are often connected to computer networks. This represents new challenges for people working in information technology. They have to ensure that those systems are as much secure as possible and confidential information will not be revealed. One possible way how to prove system security is to conduct regular penetration tests – e.g. simulate attacker's malicious activity. This article briefly introduces the basics of penetration testing and shows how to deploy and use Metasploit framework when conducting penetration testing. Finally, a case study in production environment is shown. Software tools and techniques described in this work are also valid and applicable for SCADA systems and moreover in any other field where computer networks are used.**

Keywords — penetration testing, vulnerability, exploit, Nmap, Nessus, OpenVAS, Metasploit

I. INTRODUCTION

Today's society is often called information society. That is because people who have the right information have power, and value of some information is immeasurable. If such information is stored in a computer system, it somehow must be proven that this system is safe and not vulnerable. One way how to do so is called penetration testing. Although penetration testing is an approach to how to increase and strengthen information system security, it definitely does not prove that the system is completely safe and not prone to hacker attacks. Penetration testing is able to detect publicly known security issues [1] that have been previously revealed and published.

In the quite short history of computers and computer networks, there was plethora of incidents related to security and computers in general. Although these incidents differ in impact and level of danger, none of them was insignificant. In publication dealing with basics of hacking and penetration testing [2] one such example of harmful security incident is noted. Administrator of a medium sized serverfarm was suffering from random server restarts. Problem he was not aware of was that the restart is only collateral damage and is caused by the attacker ending his remote shell session he should not have access to, of course.

On the other hand, danger can be hidden in environments, where it is not expected at first sight. Consequences of such situations can be therefore more fatal – e.g. industry production lines.

This article outlines what tools and aids can be adopted for better and more effective workflow of penetration test procedures. At the end of this article, a case study, which demonstrates a penetration test conducted against a network in pharmaceutical company where automated lines used for production of medicaments reside, is presented.

II. METHODOLOGIES AND SOFTWARE TOOLS

One of the aids which can be used is penetration testing methodology. Two common methodologies used for penetration testing are introduced in part A of this chapter. In part B of this chapter the most common tools for penetration testing are described and the most attention is given to Metasploit framework.

A. Methodologies and benefits stemming from their usage

As there are many penetration testing methodologies available, it could be sometimes difficult to choose the right one. The more experienced penetration tester is and the better knowledge about the tested environment he has, the more accurate choice of appropriate methodology he makes. There are plenty of penetration testing methodologies penetration tester can choose from. Possible representatives of current modern penetration testing methodologies are Open Source Security Testing Methodology Manual created and maintained by ISECOM and Open Web Application Security Project. Each of abovementioned methodologies is suitable for different kinds of penetration tests.

OWASP project is created and maintained by OWASP foundation – it is a non-profit organization and its members are inter alia experts from web application development industry or software experts. Numerous details about OWASP target audience and project targets can be found on OWASP project webpage [4].

OWASP foundation released numerous publications and one of them is *A Guide to Building Secure Web Applications and Web Services* [5], where what should be done in each phase of software development if resulting product targets to be as much secure as possible can be found. OWASP project adopted Microsoft's threat modeling because of its simplicity

978-1-4799-5338-7/14/$31.00 ©2014 IEEE

and efficiency. More detailed description of OWASP is out of scope of this article and readers who are interested in this topic can find more information on OWASP project web page [4] in OWASP testing guide [7].

OSSTMM is another penetration testing methodology. It was first introduced in 2000 and is maintained in current version 3.0 by Institute for Security and Open Methodologies. This methodology is heavily reputable in penetration testers community. This methodology aims to be up to date with modern software trends and technologies. Therefore it is updated half-yearly. Although this methodology introduces a lot of new terminology and theory, its practical outcomes are highly valuable. Philosophy of this methodology is as follows: OSSTMM claims that infrastructure elements interact. Approach to how to increase security is seen as controlling and restricting those interactions. This methodology has the broadest area of activity. Security is discussed in five areas here:

- Physical security,
- human factor,
- wireless communication,
- telecommunication,
- data networks and operating systems.

Like the other two methodologies, this methodology also establishes penetration test workflow that helps to better organize penetration test procedures. Phases of penetration test, in accordance with OSSTMM, are:

- Induction phase,
- interaction phase,
- inquest phase,
- intervention phase.

Main tasks of each phase can be easily deducted from its name, however brief introduction of each phase follows. The induction phase should clarify time range for penetration test and types of tests to be used. The interaction phase determines which targets are in scope of particular penetration test. The inquest phase searches and gathers as much data as possible about the target systems. And last but not least – the intervention phase verifies the functionality of security and alerting mechanisms.

After the penetration test has been finished, the reporting phase, where the results are processed, begins. Compared to the other methodologies, OSSTMM has one immense advantage. It has a toolset to process results effectively – Security Test Audit and Reporting. Usage of this toolset is simple – it is a predefined spreadsheet where inputs are identified interactions and controls of said interactions, and output is a numerical value called rav. Rav expresses if there is need for additional controls (value below 100) or everything is properly balanced (value equal 100), or if there are more controls of interactions than necessary (value above 100). Therefore, if one environment is tested twice, it can be easily seen if security of this environment has been improved or has deteriorated.

The truth is, OSSTMM is quite comprehensive. It takes considerable amount of time to get familiar with all the details and new terms defined there. Despite that fact, OSSTMM offers complete guideline for penetration testing to non-experienced penetration testers. More information about OSSTMM can be found on [9].

Both abovementioned methodologies are suitable for different purposes and penetration tests. Although there has to be some initial effort to understand terms and procedures defined in methodologies, this effort is often worth the result, especially for testers who have little or no experience with commercial penetration testing. What needs to be done before starting penetration test is shown, what a contract should contain, and what steps need to be taken during the penetration test. Some methodologies even give some tips regarding final reporting. There are often different audiences, thus different kinds of reports have to be created – more comprehensive for technical staff and less detailed for managers. Methodologies often contain hints of what each kind of report should contain and how it should be structured. Paper by authors Pradini and Ramili [1] analyses diverse penetration testing methodologies more thoroughly and more information can be found there.

B. Software tools for penetration testing

Some of the methodologies outlined in the preceding paragraphs often give hints of how to structure a penetration test. Moreover, they often suggest to divide the test into phases, where the output from each phase servers as an input for the next phase. Consequences of such approach are that different kinds of tools are suitable for different phases of the penetration test. The following paragraphs outline and introduce these tools briefly.

Different approaches can be taken when choosing what toolsets can be used for performing individual phases of the penetration test. There are plenty of tools and toolsets for penetration testing and can be used for testing various types of products and conducting diverse types of attacks. The following paragraphs introduce briefly some of these toolsets and discuss what benefits they bring into penetration testing.

Backtrack is a Linux Debian-based distro. Nowadays, its development has been discontinued and is superseded by Kali Linux, which has the same purpose and way of use. Kali's or Backtrack's feature is that it can run on tester's computer without installation, in so-called live mode. On the other hand, changes in operating system made in live session are lost after reboot. Above mentioned distros are pre-loaded with large amount of software for different areas of penetration testing. Kali contains numerous tools and frameworks for penetration testing. These tools are usually used for:

- Database security audit,
- SQL injection techniques,
- network traffic eavesdropping or tampering,
- network infrastructure attack,

- network stress testing,
- DoS attacks,
- manipulating user data,
- web application penetration testing,
- and many more tasks.

Further info about Kali linux can be found at Kali project webpage [10].

The first goal of each penetration test is to find systems on network that reply to network communication and thus are interesting from penetration tester's perspective in further phases of the penetration test, because there could be vulnerable software. This can be done using a network scanner. The most known network scanner is Nmap.

The most valuable information Nmap provides the penetration tester with is usually the version and type of operating system running on the target system, state of network ports, types and versions of services operated on the particular system. After finding a list of applications and a list of versions, the penetration tester has to determine whether any vulnerability of either operating system itself or vulnerability of any installed application is present. Vulnerabilities can be found with e.g. Nmap or OpenVAS (these tools will be briefly introduced in further paragraphs of this chapter), or fulltext search engines can be used and last but not least, Metasploit framework itself contains a database of vulnerabilites and associated exploits.

It is obvious that the scanning procedure and initial analysis of open ports reveal lot of valuable data to the penetration tester. The article by authors Kalia and Singh [11] deals with introduction of network scanning techniques and deployment of countermeasures against revealing info about operating systems with these network scanning techniques.

When all the relevant systems have been scanned, the penetration tester gets some basic idea about services (including their version) used in the network. The next step is to determine whether these services are somehow vulnerable. This can be done with assistance of tools like Nessus or OpenVas. Both tools are briefly introduced in the following paragraph.

Nessus can be controlled via web interface. There are default policies that can be used for scanning right after install or one can implement and fine-tune their own policies. If the second option is chosen, it must be defined which techniques will be used for port scanning or what plugins will be used for site vulnerability survey. Once the test is finished, a summary of results follows. Individual vulnerabilities are shown and each vulnerability is categorized depending on its severity. Obviously, the highest attention should be paid to the vulnerabilities in category labeled critical. Nessus can be extended with plugins that are written in NASL - Nessus Attack Scripting Language. There is also a possibility to buy professional edition which has some extended capabilities like SCADA systems scanning compared to free edition. Accurate comparison between Nessus versions can be found at official Nessus website [12]. Professional license costs are in thousands of dollars per year. Free alternative to Nessus is OpenVAS. Years ago, Nessus was free and its source code was available. OpenVAS developers got inspired by Nessus' source code, thus OpenVAS and Nessus share some similarities.

The responsibility of Nessus and OpenVAS is to determine if there are any exploitable vulnerabilities in the tested system. The next step when performing penetration test is to attempt to exploit the vulnerabilities. For these purposes, snippet of source code - so-called exploit - is needed. After successful exploitation, a sequence of steps is usually performed. As a result of vulnerability exploitation, tester often gets permission to execute arbitrary commands on target system.

Tasks like vulnerability exploitation or post-exploitation exploration of the target system can be automated to a certain extent. For these purposes Metasploit project toolkit is available. Its development began in 2003. Since 2003, Metasploit went through rapid development and in 2009 was acquired by Rapid7 company. Nowadays, Rapid7 is in charge of funding and development of Metasploit. Metasploit is designed with an emphasis on scalability. Metasploit's architecture aims to be modular and it's depicted on figure 1.

Libraries provide basic services like networking and files manipulation to neighboring components, thus the developer does not need to take care of routine tasks like communicating via network - it is already programmed in libraries. Via interfaces, Metasploit core is controlled by users. There are many interfaces and the following paragraphs introduce them briefly.

Msfcli is used directly from command line. It is suitable for beginners and newcomers to Metasploit, because it helps to understand how Metasploit internals work. Msfcli is controlled via command line parameters and arguments. There are detailed tutorials for Msfcli usage on Metasploit unleashed website [13].

Msfconsole is another interface available for Metasploit interaction. Compared to Msfcli, Msfconsole is more robust, scalable, and easier to use. It allows defining global

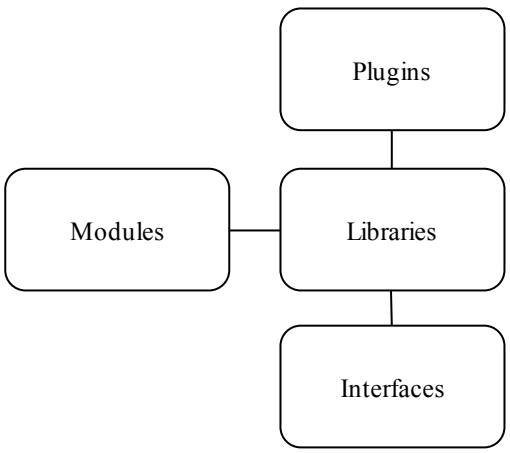

Figure 1. Metasploit architecture overview

variables, perform lookups in exploit database, and more. Meterpreter sessions can be maintained in a single Msfconsole - practical usage is shown in Metasploit unleashed tutorial.

There is also GUI for Metasploit - it is called Armitage. It's suitable for users who do not have so much experience with command-line usage and displays connection topology of tested systems.

One can benefit from using Metasploit in several ways. There is an effective exploit management (lookup, update, documentation) or plethora of payloads (tasks that are performed after successful exploitation of target system). Payloads can either perform one specific task (e.g. user creation) or can be more complex and offer more advanced functionality (Meterpreter, which is described later, is one such example). Figure 2 shows sequence of steps that are necessary to establish two-way communication channel between tester‘s and tested system in order to be able to control the tested system remotely.

The most crucial part of figure two is the transmission of special DLL library with Meterpreter shell, which will be described in the next paragraph. Few conditions have to be met in order to succeed when control of the target is needed and that control has to be achieved with taking running antivirus software into consideration. Creating a new process has to be avoided (it has to run in context of the exploited process) and creating a new file has to be avoided too. Both can be considered a red flag for antivirus software. These conditions are met through DLL injection technique. Meterpreter is also designed with extensibility in mind, it allows to load different modules which perform different tasks (e.g. network traffic sniffing).

When the above mentioned steps were completed and Meterpreter DLL is successfully loaded via DLL injection technique, the person who has access to the Meterpreter shell has literally unlimited control over the target system. List of all the possibilities of Meterpreter would be too long, therefore only the most important and the most interesting possibilities are listed:

- Directory listing,
- Upload or download of files,
- File attributes manipulation,
- Editing files,
- Hash dumps,
- Event logs deletion,
- Routing table alteration,
- Command execution,
- Taking screenshots,
- Migrate between various processes,
- Keyboard/mouse control,
- Launch WMI query,
- Control of clipboard,
- Control of services,
- Stealing tokens,
- User creation,
- …and much more.

Even though a detailed description of each possible way how Meterpreter and target machine can interact is out of the scope of this article, let’s go briefly through the most important abilities of Metasploit. One can upload, download or manipulate files with Metasploit. Depending on the target system, this can have serious consequences. One can manipulate processes and services on the target machine. Killing antivirus software and installing backdoor can be done. One can manipulate routing tables of hosts with Metasploit. This can result in the man in the middle attack. One can delete event logs. This is useful when diagnostics and investigation of a security breach should be made more difficult. One can steal tokens – either local or domain. Tokens can be considered to be keys to resources (e.g. folder). When a token is stolen, the thief usually has access to all the resources which can be accessed by the legitimate owner of the token. Fileservers can usually be considered as storehouses of tokens, because various users usually access files stored in there. Metasploit offers much more possibilities of manipulation with the target system. More detailed info can be found at Metasploit Unleashed webpage [13].

Communication protocol between target and tester's machine is built on type-length-value (TLV) model. TLV approach is chosen because Meterpreter and Metasploit are developed with scalability in mind. There are other applications of type-length-value model, too. One such example is EIGRP routing protocol developed by Cisco company. If a protocol based on type-length-value model needs to be extended, a new type is defined and the current source code implementing the current protocol behavior does not need to be changed, only extended. Traffic between Meterpreter the client (tester) and the server (target) is encrypted, thus, privacy is ensured.

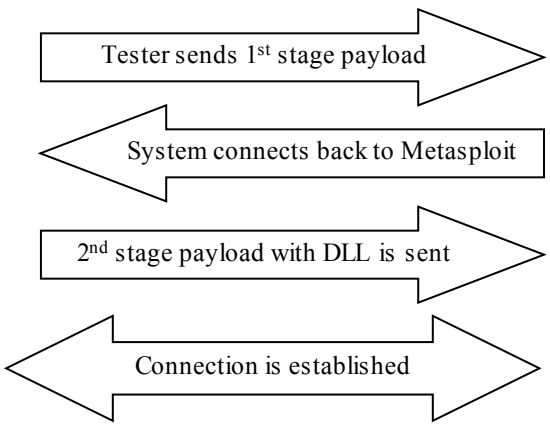

Figure 2. Metasploit connection establishment

Metasploit Framework is also updated on regular basis – new malicious code used for exploiting system vulnerabilities is added when an update is triggered. Despite its initial complexity for newcomers to penetration testing, tools like Metasploit Project offer many valuable tools to conduct penetration testing and simplify some tasks that would have to be solved programmatically without framework.

III. CASE STUDY

This case study aims to demonstrate the usage of above mentioned tools and utilities. This particular case study takes place in a production environment of a pharmaceutical company and its outcome should be verification whether machines at production subnet are safe and the production can run smoothly. It can be stated that computers and information technology is not always in the state of art, particularly in cases, when core business of corporation has nothing to do with information technology as in case of mentioned pharmaceutical company. The network topology is depicted on figure 3. The cloud represents unexplored production subnet. It is not known which systems are present and whether they are somehow vulnerable.

The only thing known is the subnet range, where the production network resides. Let's look at this subnet from the penetration tester's point of view. His laptop is connected to the production subnet, has a valid IP and booted KALI Linux. Let's begin with the discovery of hosts which are alive, their OS versions and open ports (list of hosts and output were shortened).

```
root@kali:~# nmap -PS 192.168.1.96/27 -O

Starting Nmap 6.46 ( http://nmap.org )
Nmap scan report for 192.168.1.107
Host is up (0.00010s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
1025/tcp open  NFS-or-IIS
5000/tcp open  upnp
MAC Address: 00:0C:29:6F:41:69 (VMware)
Device type: general purpose
Running: Microsoft Windows 2000|XP
OS details: Microsoft Windows
2000 SP0 - SP4 or
Windows XP SP0 - SP1
Network Distance: 1 hop
```

It can be clearly seen, that the host with IP address 192.168.1.107 is up and running. It is also observable, that

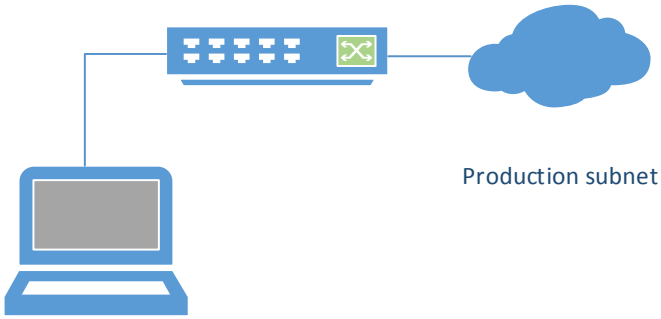

Figure 3. Case study – network topology

operating system of the target machine seems to be Windows XP-SP1.

Thus, assumption that this system is vulnerable to MS03-026 DCOM exploit is made and this assumption is not investigated further (with tools like Nessus or OpenVAS). Consequently, msfconsole is launched and exploit for DCOM vulnerability is searched.

```
msf > search dcom

Matching Modules
================
Name: exploit/windows/dcerpc/ms03_026_dcom
Disclosure Date:2003-07-16
Rank:great
Description: MS03-026 Microsoft RPC DCOM
Interface Overflow
```

Lookup was successful, exploit has been found. Next steps are quite straightforward. This particular exploit is chosen, the target IP address is set with appropriate commands and finally the exploit is launched. After few seconds of data processing, prompt changes from `msf>` to `meterpreter>`. This indicates that the exploitation was successful and the penetration tester is now able to control the remote system via set of advanced commands or launch the command line on that target system. This is also a very simple task:

```
meterpreter > shell
Process 132 created.
Channel 1 created.
Microsoft Windows XP [Verze 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

From now on, behavior of the target system can be influenced in several ways – e.g. files can be deleted or replaced, additional processes can be launched, and so on. It is up to imagination of individual penetration tester.

This case study has shown that penetration testing is important also in production environment, where systems like XP embedded can still be present nowadays. These systems can control large production lines and can be connected to the network for easier management. When exploitation of these systems is successful, serious issues can occur. If this particular system would be a part of the production line used for creating mixture for medicaments, the penetration tester or a malicious hacker could easily alter the formula for that mixture. Consequence of that would be a situation in which the production runs and medicaments are being produced, but using a different formula. If these medicaments with different formulas would get outside the factory and would be sold to patients, the consequences could be lethal.

IV. CONCLUSION

In the modern world, there is a need for a proactive approach to information security in order to avoid potential security breaches. It is so, because a security breach and a consequent data loss or data tampering usually cost huge amounts of money and cause loss of reputation for the company. There are often increased costs for security measures, but advantages brought by these measures are worth it. Level of security can be determined by a discipline called penetration testing. There are many toolsets and utilities which can be used – from methodologies to software utilities introduced in this article. Usually, it is desirable to spend some time learning and getting familiar with methodologies and software tools, because cumbersome tasks are automated and done in a few clicks or with a few commands. Thoughts about complex and rigorous preparation for a penetration test can be emphasized by quoting the former US president, Abraham Lincoln, who said:"If I had eight hours to chop down a tree, I'd spend six hours sharpening my axe".

ACKNOWLEDGMENT

This work and contribution is supported by the project of the student grant competition University of Pardubice, Faculty of Electrical Engineering and Informatics No. 60140 / 20 / SG640009.

REFERENCES

[1] Towards a practical and effective security testing methodology. Proceedings - IEEE Symposium on Computers and Communications [online]. 2010, č. 1, s. 320-325 [cit. 2013-04-03]. DOI: 10.1109/ISCC.2010.5546813. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5546813f49b

[2] Mati Aharoni ([2011]). Penetration testing with Backtrack . 3rd ed. Cornelius, NC: Offensive Security. 12. SP800-115. Technical Guide to Information Security Testing and Assessment.

[3] Gaithersburg: [NIST Computer Security Division], 2008. Available: http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf

[4] Owasp foundation. (2014). Owasp project web page. Available: https://www.owasp.org/index.php/Main_Page. Last accessed 14th Jan 2014. OWASP foundation. (2011). A Guide to Building Secure Web Applications and Web Services [Online]. Available: http://sourceforge.net/projects/owasp/postdownload?source=dlp

[5] Owasp foundation. (2005). A Guide to Building Secure Web Applications and Web Services. Available: prdownloads.sourceforge.net/owasp/OWASPGuide2.0.1.pdf?download. Last accessed 14th Jan 2014.

[6] Offensive Security. (2013). A Guide to Building Secure Web Applications and Web Services. Available: https://www.owasp.org/images/6/6b/OWASP_Blue_Book-Educational_Institutions.pdf. Last accessed 14th Jan 2014.

[7] Owasp foundation. (2008). OWASP Testing Guide. Available: https://www.owasp.org/index.php/OWASP_Testing_Guide_v3_Table_of_Contents. Last accessed 14th Jan 2014.

[8] Karen Scarfone, Murugiah Souppaya, Amanda Cody, Angela Orebaugh (2008). Technical Guide to Information Security Testing and Assessment. Gaithersburg: NIST. 1-80.

[9] ISECOMM. 2010. Open Source Security Testing Methodology Manual (OSSTMM). [ONLINE] Available at: http://www.isecom.org/mirror/OSSTMM.3.pdf. [Accessed 26 January 14].

[10] Kali Linux. 2014. Kali Linux | Rebirth of BackTrack. [ONLINE] Available at: http://www.kali.org/. [Accessed 26 January 14].

[11] S. Kalia & M. Singh, "Masking approach to secure systems from Operating system Fingerprinting," *Tencon 2005 2005 Ieee Region 10*, Dec. 2005.

[12] Tenable network security. 2014. Nessus editions | Tenable network security. [ONLINE] Available at: http://www.tenable.com/products/nessus/editions. [Accessed 26 January 14].

[13] Metasploit Unleashed. 2014. Metasploit Unleashed. [ONLINE] Available at: http://www.offensive-security.com/metasploit-unleashed/Main_Page. [Accessed 26 January 14].