1. Introduction

1.1 Vulnerabilities in software components

Computers have become part of our lives, but we have to wonder if the software we use is secure or not, as well as how we can test it and what the effects of software faults are.

1.1.1 What is a vulnerability?

According to NIST Special Publication (SP) 800-30, vulnerability is defined as "[a] flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system's security policy."

1.1.2 Discovering vulnerabilities

One possible classification of techniques to analyze vulnerabilities is the performance of a black box, a white box or a gray box analysis.

Black box analysis: the system is analyzed making used of its interfaces.
White box analysis: this analysis involves owning the software code. This process is similar to auditing.
Gray box analysis: the software is partially known. This technique bases on having partial knowledge of the system. It facilitates the identification of more errors than a black box analysis.

Other possible classification of techniques to analyze vulnerabilities is the performance of a **static** or a **dynamic** analysis.

1.1.3 Types of vulnerabilities according to software life cycle

Design: they occur in the system design. Most of cases programs satisfy client requirements but functions that they should not do are not tested.
Implementation: again, programs do expected operations/ functionalities but not all are appropriately tested to verify that they do not do what it is not expected.

Operation: this type of vulnerabilities appears in operation and maintenance processes. These vulnerabilities are not in the code but in how the software interacts with the environment.

1.1.4 Types of vulnerabilities according to the system

Software: this type of vulnerabilities is related to any kind of system.

Web systems: this type of vulnerabilities affects web systems, requiring vulnerability analysis at the server side, at the client side and at the communication channel.

1.2 Exploitation mechanisms

Mechanisms can be divided in two types:

- **Automatic:** lots of tools can be used to exploit vulnerabilities. However, the most appropriate tool should be chosen.
- **Manual:** a pair of different scenarios can be noticed. The former refers to the possible performance of an automatic analysis but specific tools do not exit. The latter refers to scenarios in which just trial and error is available.

1.3 Exploitation laboratory and tools.

There are multiple vulnerability analysis and synthesis tools. In this Section the following tools are presented:

1.3.1 Scanners

They facilitate the performance of network analysis to identify the network structure, identify available hosts or perform a network traffic analysis. One of the most well-known applications is Wireshark. On the other hand, Nmap is the best-known port scanner.

1.3.2 Hacking and cracking

Hacking can be defined as the permanent quest for new computer-related knowledge, its security mechanisms, its vulnerabilities and its possible exploitation techniques and countermeasures. On the other hand, cracking can be defined as the permanent attempt of exploiting software vulnerabilities.

Hacking and cracking tools should be used for educational purposes. Between the most noteworthy are *password crackers* like John the Ripper, based on the use of *crypt()* function in Unix and the comparison with a predefined dictionary. From all existing tools, Metasploit is a well-known one. It performs penetration tests which help identify vulnerabilities.

1.3.3 Reverse engineering

Reverse Engineering is a term that comes originally from the field of mechanical engineering. It refers to the process of analyzing an existing object or system by laying out its construction plan to then rebuild it in every detail.

Decompilers like JD (http://jd.benow.ca/) for Java or Boomerang (http://boomerang.sourceforge.net/) for C are noteworthy.

On the other hand, between the most challenging disassemblers, Interactive Disassembler (IDA, https://www.hex-rays.com/products/ida/index.shtml) should be highlighted. It is characterized by its versatility of processors and operating systems.

Recalling concepts

CPU registers, Computer memory……all you need is on "Cyber Security Basics: A Hands-on Approach" edX MOOC, Lecture 3

https://www.edx.org/es/course/cyber-security-basics-hands-approach-uc3mx-inf-2x

References

1. Software exploitation, Inteco: http://www.incibe.es/extfrontinteco/img/File/intecocert/EstudiosInformes/cert_inf_software_exploitation.pdf
2. Hacking: The Art of Exploitation, 2nd Edition: http://proquest.safaribooksonline.com/9781593271442
3. Assembly programming: http://www.tutorialspoint.com/assembly_programming/index.htm
4. Assembly tutorial: http://www.tutorialspoint.com/assembly_programming/assembly_tutorial.pdf
5. Reversing practice: https://www.reversinghero.com/
6. CPU registers: http://www.cs.virginia.edu/~evans/cs216/guides/x86.html