Web Application Penetration Testing

Disclaimer

- Tutorial and demo provided on Hackers Terminal is for informational and educational purpose only, and for those who're willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing. Any time the word "Hacking" that is used on this site shall be regarded as Ethical Hacking.
- Do not attempt to violate the law with anything contained here. If you planned to use the content for illegal purpose, then please leave this site immediately! We will not be responsible for your any illegal actions. Neither administration of this website, the authors of this material, or anyone else affiliated in any way, is going to accept responsibility for your actions.
- You shall not misuse the information to gain unauthorised access. However you may try out these hacks on your own computer at your own risk. Performing hack attempts (without permission) on computers that you do not own is illegal.
- The misuse of the information in this course can result in criminal charges brought against the persons in question. The authors and lian group will not be held responsible in the event any criminal charges be brought against any individuals misusing the information in this website to break the law.

WHOAMI!

Penetration Testing

Malware Analyst

Purple Teaming

liangroup

“If you have any question … Please keep it for yourself … I’m not Google”

— Honest Person

“We need Theories!”

— Honest Person

TABLE OF CONTENTS

TABLE OF CONTENTS

“Why Programming?!”

— Noob Person

Auditing Tips

- PHP
  - Type Juggling
  - Phar Deserialization
- Java
  - JNDI Injection
  - EL Injection
- Ruby
  - Regex (^ and $) are multiline by default!
  - Mass Assignment
- JavaScript
  - Type Juggling
  - Prototype Pollution

01. JavaScript for Hackers

We will going to learn JavaScript for Hackers

JavaScript

JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved.

JS Files and HTML

```
<html>
 <head></head>
 <body>
  <script>
   JS Codes
  </script>
 </body>
</html>
```

Or hackerfile.js

Define Variables

```
var variable_name = value;

const variable_name = value;

let variable_name = value;

console.log(value);
```

Operations

```
x_var = 1337

x_var = x_var +   2;      x_var +=   2
x_var = x_var -   2;      x_var -=   2
x_var = x_var /   2;      x_var /=   2
x_var = x_var %   2;      x_var %=   2
x_var = x_var *   2;      x_var *=   2
x_var = x_var **  2;      x_var **=  2
```

Operands

42 > 10	true
42 < 10	false
42 >= 42	true
42 > 42	false
42 === 42	true
42 === “42”	false
42 == 42	true
42 == “42”	true
42 != 42	false

If Conditions

```
if (true) {
    console.log("I'll Always Execute");
}
if (false) {
    console.log("I'll not Execute at All");
}
```

If Conditions Part 2

```
if (false) {
    console.log("I'll not Execute");
} else if (true) {
    console.log("I'll Execute");
}
```

```
if (false && false) {
    console.log("I'll not Execute");
} else if (false || false) {
    console.log("I'll not Execute too");
} else {
    console.log("I'll Execute");
}
```

Strings

```
let username = “@INVOXES”;

username.length

console.log(“Hello” + username);

const username = `@INVOXES`;

const  wlcmsg = `Hello ${username}`;

const s3 = `Calculate ${1 + 2 + 4}`;
```

Arrays and Multidimensional Arrays

```
const my_array = [1, 2, 3]
const my_array_2 = Array(1, 2, 3)
const my_array_3 = Array.of(1, 2, 3)
```

```
const matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9],
]
console.log(matrix[0][0]);
console.log(matrix[2][0]);
console.log(matrix.length);
```

Arrays Features

```
const my_arr = [1, 2, 3]

my_arr.push(4)      // [1, 2, 3, 4]
my_arr.pop()        // [1, 2, 3]
my_arr.unshift(0)   // [0, 1, 2, 3]
my_arr.shift()      // [1, 2, 3]
```

Loops (While)

```
my_arr = ['a', 'b', 'c']
let i = 0
while (i < my_arr.length)
{
    console.log(my_arr[i]);
    console.log(i);
    i += 1;
}
```

```
while (true) {
    if (somethingisTrue) break
}
```

```
while (true) {
    if (somethingisTrue) continue
}
```

Loops (For)

```
const my_list = ['a', 'b', 'c']

for (let i = 0; i < my_list.length; i++) {
    console.log(my_list[i]);
    console.log(i);
}
```

Loops (For of)

```
const my_list = [‘a’, ‘b’, ‘c’]

for (const value of my_list) {

    console.log(value);

}
```

Functions 0x01

```
function getData() {
  // do something
}
getData();
```

```
function getData(color, age, name) {
 // do something
}
getData('Black', 24, 'John Doe');
```

Functions 0x02

```
function getData(age, name) {
  if (typeof age !== 'undefined') {
    // do something
  }
}
```

```
function getData(age, name) {
  if (age) {
    // do something
  }
}
```

Functions 0x03

```
function getData() {
  // do something
  return 'Hello Friend!'
}
```

```
function getData() {
  return ['LianGroup', 1337]
}
let [name, yrs] = getData()
```

```
function getData(user='admin', age=24) {
  // do something
}
```

immediately invoked function expression (IIFE)

```
(function() {})();
```

Context in JavaScript

Every function in JavaScript has its own set of properties and data attached to it. We call these the function's *context*. Context is not set in stone and can be modified during runtime. Objects stored in a function's context can be referenced using the keyword this:

```
const func = function() {
    this.age = 25;
    // will return 25
    console.log(this.age);
};
// will return undefined
console.log(this.age);
```

Context in JavaScript

As you can imagine, many annoying programming bugs are a result of context being hard to debug—especially when some object's context has to be passed to another function. JavaScript introduced a few solutions to this problem to aid developers in sharing context between functions:

```
// create a new getAge() function clone with the context from
ageData
// then call it with the param 'joe'
const getBoundAge = getAge.bind(ageData)('joe');
// call getAge() with ageData context and param joe
const boundAge = getAge.call(ageData, 'joe');
// call getAge() with ageData context and param joe
const boundAge = getAge.apply(ageData, ['joe']);
```

API

Application Programming Interfaces (APIs) are constructs made available in programming languages to allow developers to create complex functionality more easily. They abstract more complex code away from you, providing some easier syntax to use in its place.

Client-side JavaScript, in particular, has many APIs available to it — these are not part of the JavaScript language itself, rather they are built on top of the core JavaScript language, providing you with extra superpowers to use in your JavaScript code.

Browser API

Browser APIs are built into your web browser and are able to expose data from the browser and surrounding computer environment and do useful complex things with it. For example, the Web Audio API provides JavaScript constructs for manipulating audio in the browser — taking an audio track, altering its volume, applying effects to it, etc. In the background, the browser is actually using some complex lower-level code (e.g. C++ or Rust) to do the actual audio processing. But again, this complexity is abstracted away from you by the API.

Browser API

- APIs for manipulating documents
- APIs that fetch data from the server
- APIs for Drawing and manipulating graphics
- Audio and Videos API
- Device APIs
- Client-side storage APIs

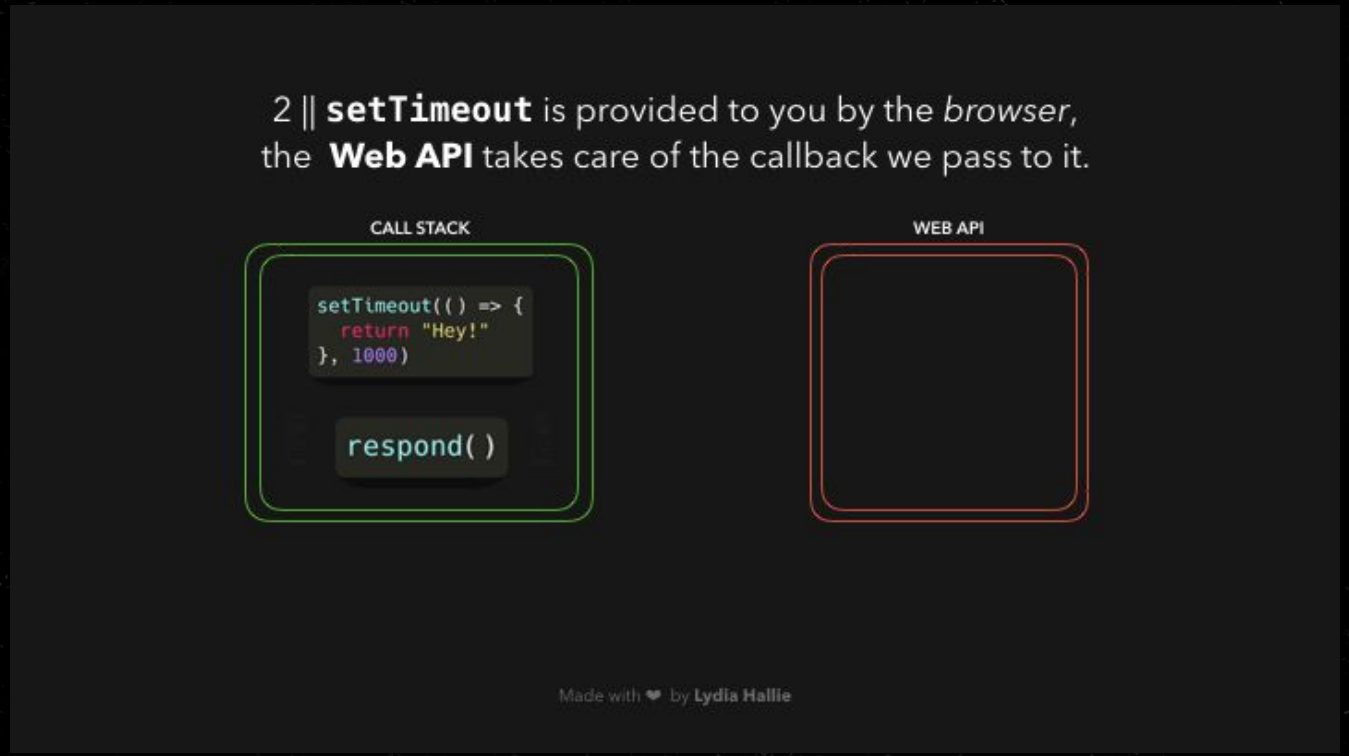
Asynchronous Programming

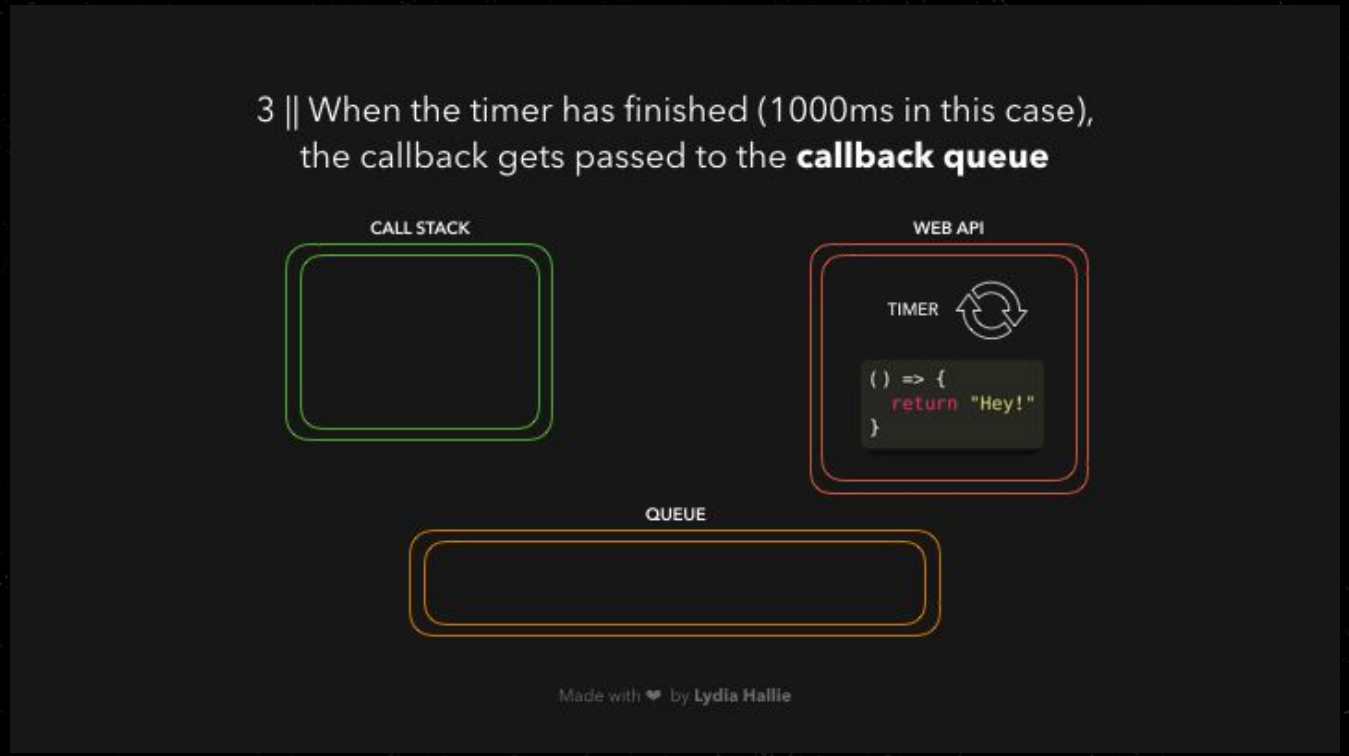
```
console.log('before');

setTimeout(() => {

    console.log('inside the function');

}, 2000);

console.log('after');
```

1 || Functions get **pushed to** the call stack when they're **invoked** and **popped off** when they **return a value**

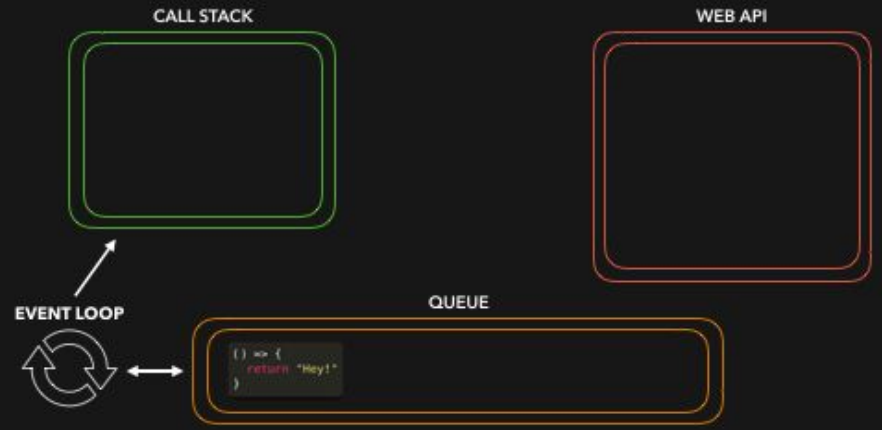

```
function greet() {
  return "Hello!"
}

function respond() {
  return setTimeout(() => {
    return "Hey!"
  }, 1000))
}

greet()
respond()
```

Made with ❤ by **Lydia Hallie**


2 || setTimeout is provided to you by the browser,
the Web API takes care of the callback we pass to it.
CALL STACK
setTimeout(() => {
return "Hey!"
}, 1000)
respond()
WEB API
Made with ❤ by Lydia Hallie


3 || When the timer has finished (1000ms in this case), the callback gets passed to the **callback queue**
CALL STACK
WEB API
TIMER
() => {
return "Hey!"
}
QUEUE
Made with ❤ by Lydia Hallie

4 || The **event loop** looks at the **callback queue** and the **call stack**.
If the call stack is empty, it pushes the first item in the queue onto the stack.



Made with ❤ by **Lydia Hallie**

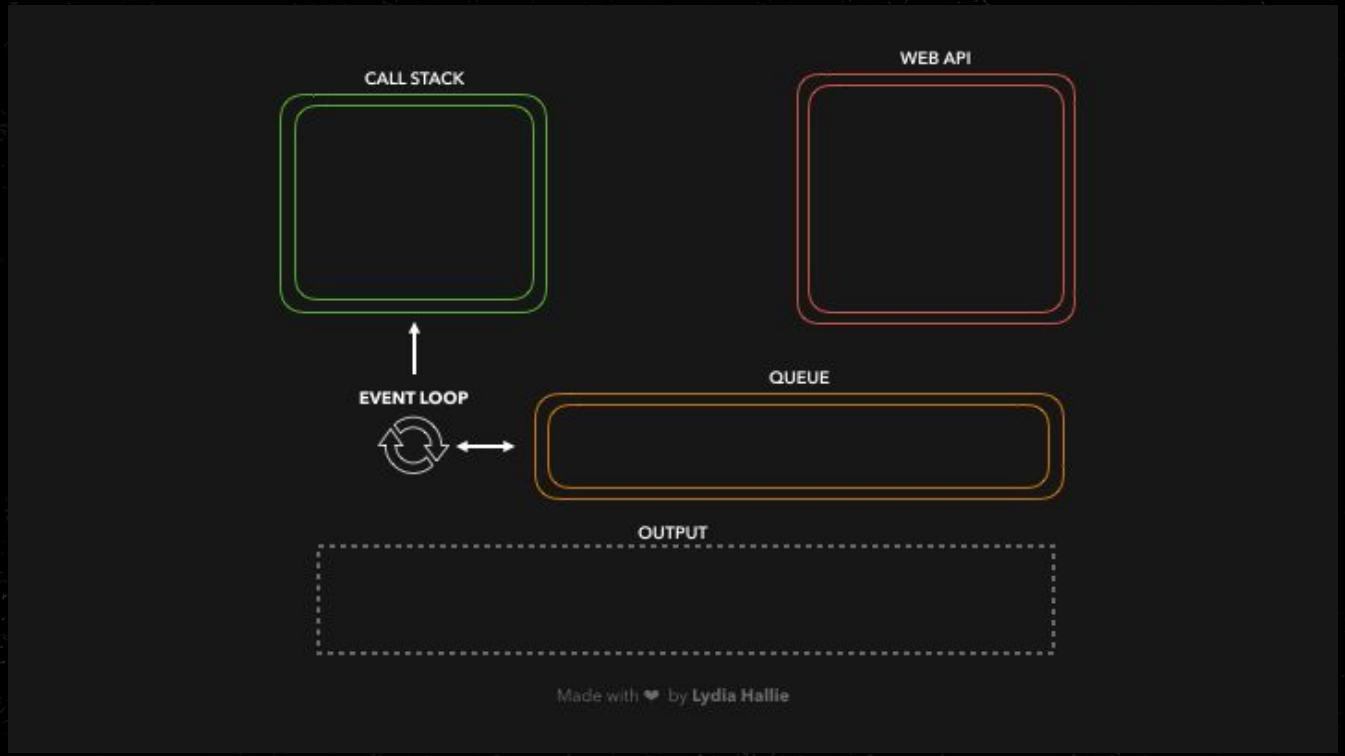
5 || The callback is added to the call stack and executed.
Once it returned a value, it gets popped off the call stack.

OUTPUT

```
function greet() {
  return "Hello!"
}

function respond() {
  return setTimeout(() => {
    return "Hey!"
  }, 1000))
}

greet()
respond()
```

Asynchronous Programming

```
const foo = () => console.log("First");

const bar = () => setTimeout(() => console.log("Second"), 500);

const baz = () => console.log("Third");

bar();

foo();

baz();
```


CALL STACK
WEB API
EVENT LOOP
QUEUE
OUTPUT
Made with ❤ by Lydia Hallie

Question?!

```
function f(){
      for (const v in (Array.from(Array(1000000).keys()))){
      a = v*v*v;
      }
      return a;
}
const foo = () => console.log(f());
const bar = () => setTimeout(() => console.log("Second"), 500);
const baz = () => console.log("Third");
bar();
foo();
baz();
```

Dynamic Function

```
var some_fixed_value = 1;

var f = new Function("return " + some_fixed_value);
```

Web API

document.getElementById()

doc*.getEle*.textContent

document.cookie

document['cookie']

document.domain

document.URL

document.location

window.open()

Window.opener

localStorage.setItem('x', '1')

localStorage.getItem('x')

localStorage.removeItem('x')

localStorage.clear()

document.write()

02. Basics of MySQL

Introduction to Databases and MySQL

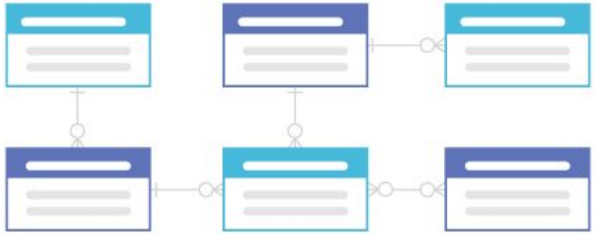
Database

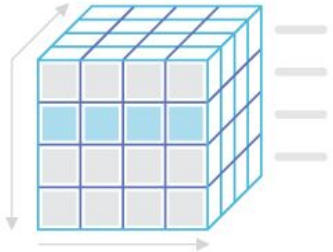
A database is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling techniques.

liangroup

SQL

Relational Database Management Systems (RDBMS)

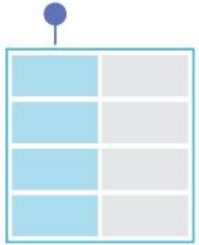




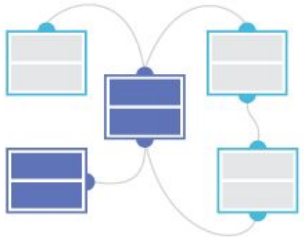
Online Analytical Processing (OLAP) Cube

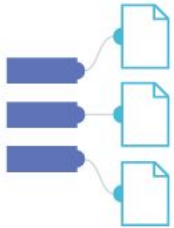
NoSQL

Key-Value



Graph





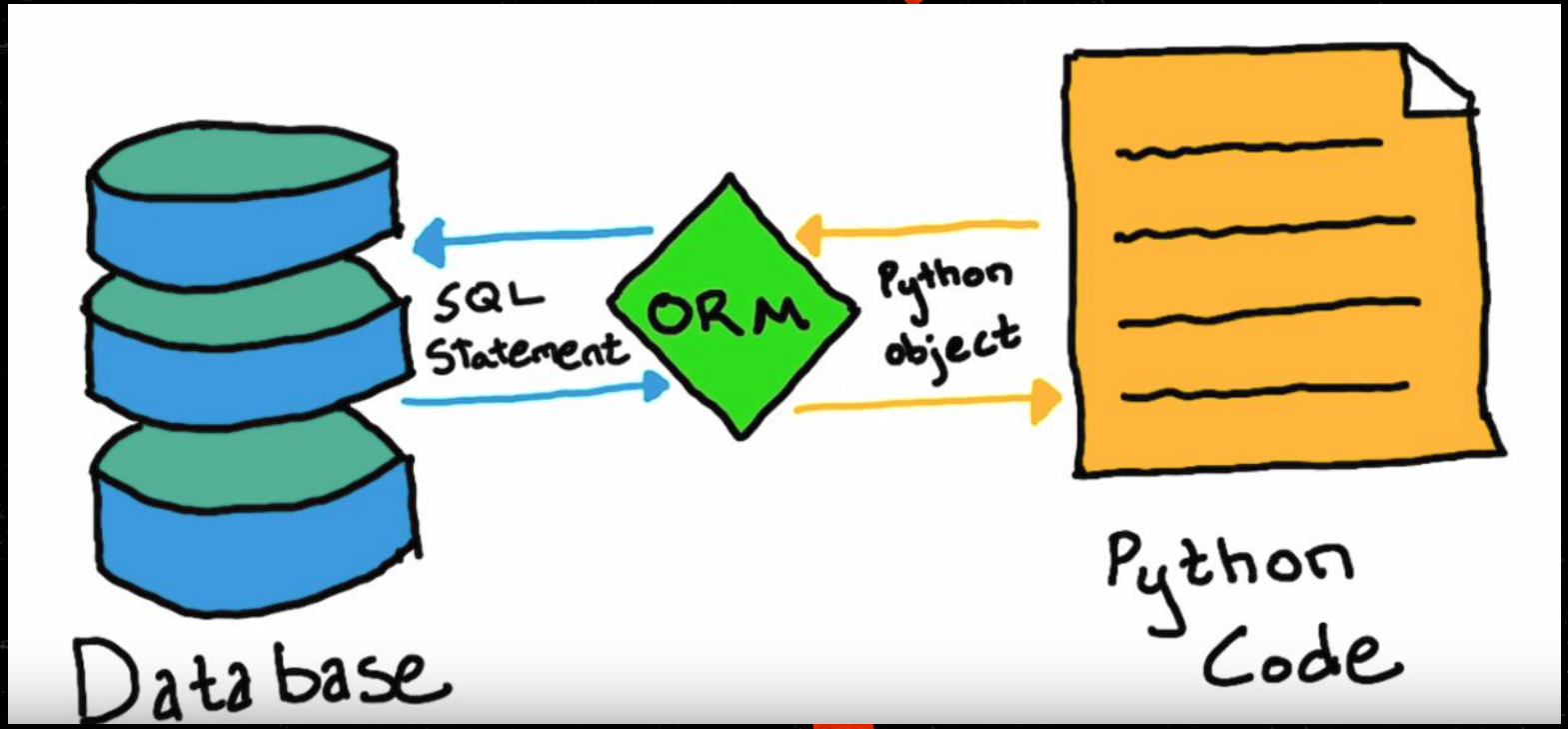
Document

Column store

Example of SQL

Example of NoSQL

edureka!


SQL
Statement
ORM
Python
object
Database
Python
Code

```
using System;
using System.Linq;
using System.Threading.Tasks;
using MarketPlace.DataLayer.Context;
using MarketPlace.DataLayer.Entities.Common;
using Microsoft.EntityFrameworkCore;

namespace MarketPlace.DataLayer.Repository
{
    public class GenericRepository<TEntity> : IGenericRepository<TEntity> where TEntity :
BaseEntity
        private readonly MarketPlaceDbContext _context;
        private readonly DbSet<TEntity> _dbSet;

        public GenericRepository(MarketPlaceDbContext context)
        {
            _context = context;
            this._dbSet = _context.Set<TEntity>();
        }

        public IQueryable<TEntity> GetQuery()
        {
            return _dbSet.AsQueryable();
        }

        public async Task AddEntity(TEntity entity)
        {
            entity.CreateDate = DateTime.Now;
            entity.LastUpdateDate = entity.CreateDate;
            await _dbSet.AddAsync(entity);
        }

        public async Task<TEntity> GetEntityById(long entityId)
        {
            return await _dbSet.SingleOrDefaultAsync(s => s.Id == entityId);
        }

        public void EditEntity(TEntity entity)
        {
            entity.LastUpdateDate = DateTime.Now;
            _dbSet.Update(entity);
        }

        public void DeleteEntity(TEntity entity)
        {
            entity.IsDelete = true;
            EditEntity(entity);
        }

        public async Task DeleteEntity(long entityId)
        {
            TEntity entity = await GetEntityById(entityId);
            if (entity != null) DeleteEntity(entity);
        }

        public void DeletePermanent(TEntity entity)
        {
            _dbSet.Remove(entity);
        }

        public async Task DeletePermanent(long entityId)
        {
            TEntity entity = await GetEntityById(entityId);
            if (entity != null) DeletePermanent(entity);
        }

        public async Task SaveChanges()
        {
            await _context.SaveChangesAsync();
        }

        public async ValueTask DisposeAsync()
        {
            if (_context != null)
            {
                await _context.DisposeAsync();
            }
        }
    }
}
```

MySQL Database

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language.

Installation Guide

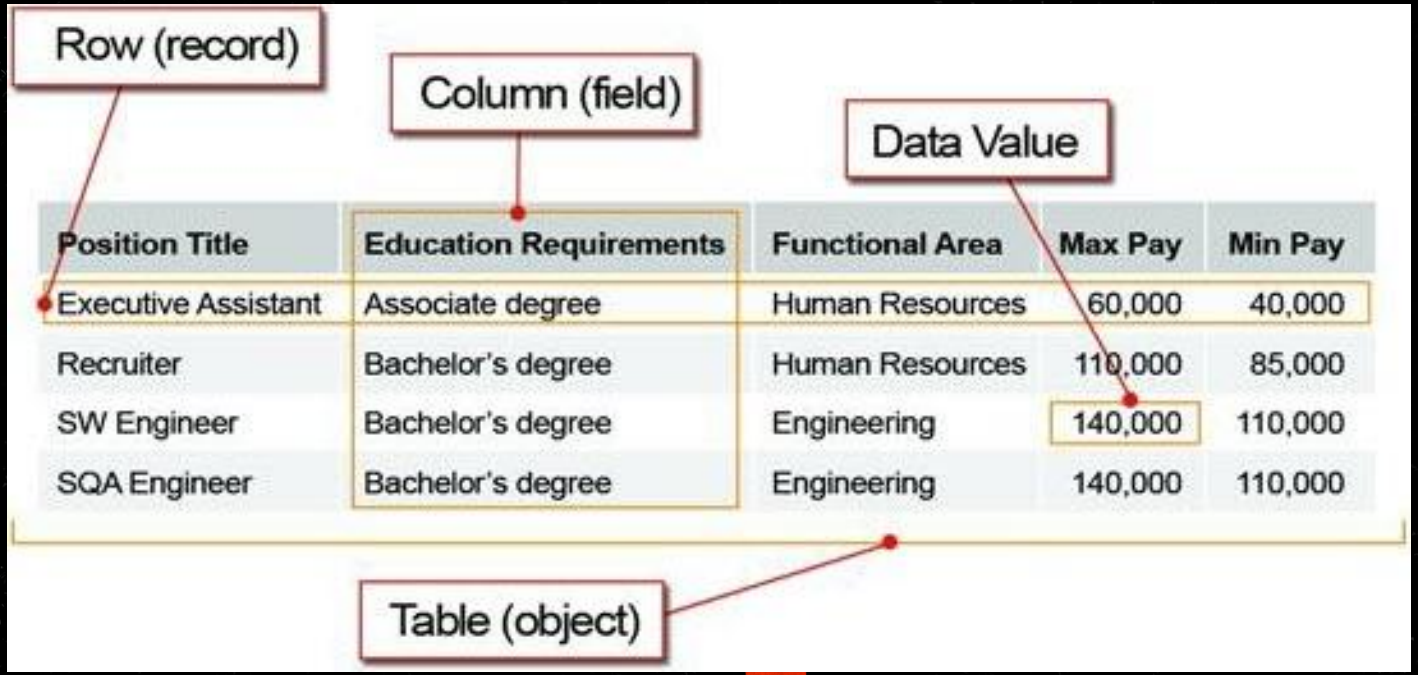
- r00t# apt update
- r00t# apt install mysql-server
- r00t# mysql_secure_installation
- r00t# mysql
- mysql> SELECT user,authentication_string,plugin,host FROM mysql.user;
- mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'P@ssw0rd';
- mysql> FLUSH PRIVILEGES;
- mysql> exit;

REF: https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-18-04

Installation Guide

- r00t# apt update
- r00t# apt install phpmyadmin php-mbstring php-zip php-gd php-json php-curl
- r00t# phpenmod mbstring
- r00t# systemctl restart apache2

REF: https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-phpmyadmin-on-ubuntu-20-04



Position Title	Education Requirements	Functional Area	Max Pay	Min Pay
Executive Assistant	Associate degree	Human Resources	60,000	40,000
Recruiter	Bachelor's degree	Human Resources	110,000	85,000
SW Engineer	Bachelor's degree	Engineering	140,000	110,000
SQA Engineer	Bachelor's degree	Engineering	140,000	110,000

CRUD Operation

- Create ::: INSERT
- Read ::: SELECT
- Update ::: UPDATE
- Delete ::: DELETE

Create Operation

```
INSERT INTO Customers (CustomerName, ContactName,
Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21',
'Stavanger', '4006', 'Norway');
```

Read Operation

```
SELECT * FROM customers;
```

Update Operation

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

Delete Operation

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

Union-Distinct

Table1	
column1	column2
a	b
a	c
a	d

U

Table 2	
column1	column2
b	c
a	d

=

Table1 Union Table2	
column1	column2
a	b
a	c
a	d
b	c

Duplicate row not repeated in results

Union Operation

```
SELECT City FROM Customers

UNION

SELECT City FROM Suppliers ORDER BY City;
```

Read/Write

```
SELECT LOAD_FILE("/etc/passwd");

SELECT "Payload" INTO OUTFILE

"/var/www/html/file.php";
```

REF::https://www.w3resource.com/mysql/string-functions/mysql-load_file-function.php

REF::https://dev.mysql.com/doc/refman/8.0/en/string-functions.html#function_load-file

REF::https://dev.mysql.com/doc/refman/8.0/en/select-into.html

REF::http://0haxor.blogspot.com/2012/08/into-outfile-uploading-your-shell-with.html

Hex Encode

SELECT HEX(“Value”)

Tip: Good for Bypass!

REF::https://www.w3resource.com/mysql/string-functions/mysql-hex-function.php

03. Understand PHP

Learn as much as need for a Hacker

PHP

PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

PHP Fundamentals

- Supports a full object-oriented programming interface
- It’s dynamically typed
- Does not provide any real memory safety features
- Does not provide any real support for concurrency
- Utilizes very little reflection
- Support custom serialization

PHP Frameworks & Template Engines

Frameworks

- Laravel
- CodeIgniter
- Zend
- Symfony
- Yii

Template Engines

- Smarty
- Twig
- Dwoo
- Blade
- Volt

PHP File and HTML


```
<!DOCTYPE html>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>
        <?php
            echo "Hi, I'm a PHP script!";
        ?>
    </body>
</html>
```

Or index.php file

Define Variables

```
$my_variable    = “string”;

$my_variable2 = 1337;
```

... (you already know that!)

Strings

- Single Quoted
- Double Quoted
- Heredoc Syntax
- Nowdoc Syntax

Double Quote

```
$p1var  = “Hello, th\151s is \u{0073}till \x61 simple string\n”;
```

The most important feature of double-quoted strings is the fact that variable names and function calls will be expanded.

Double Quote

```
$name = “Kousha”;

echo “Hello ${name}!”;

echo “Here is a phpversion function call ${phpversion()}”.PHP_EOL;
```

The most important feature of double-quoted strings is the fact that variable names and function calls will be expanded.

Double Expansion

```
$workplace = “company”;

$company = “liangroup”;

echo “Hello ${$workplace}!”;
```

The most important feature of double-quoted strings is the fact that variable names and function calls will be expanded.

Concatenation

```
$p1var  = “Hello”;

$p2var = “ name”;

$p3var = $p1var . $p2var;
```

Arrays

```
$cars = array("Volvo","BMW","Toyota");

var_dump($cars);
```

Conditions

```
if (condition) {
  code to be executed if this condition is true;
} elseif (condition) {
  code to be executed if first condition is false and
this condition is true;
} else {
  code to be executed if all conditions are false;
}
```

Loops

```
foreach ($array as $value) {
  code to be executed;
}
```

Break/Cont'

```php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
      break;
  }
  echo "The number is: $x <br>";
}
```

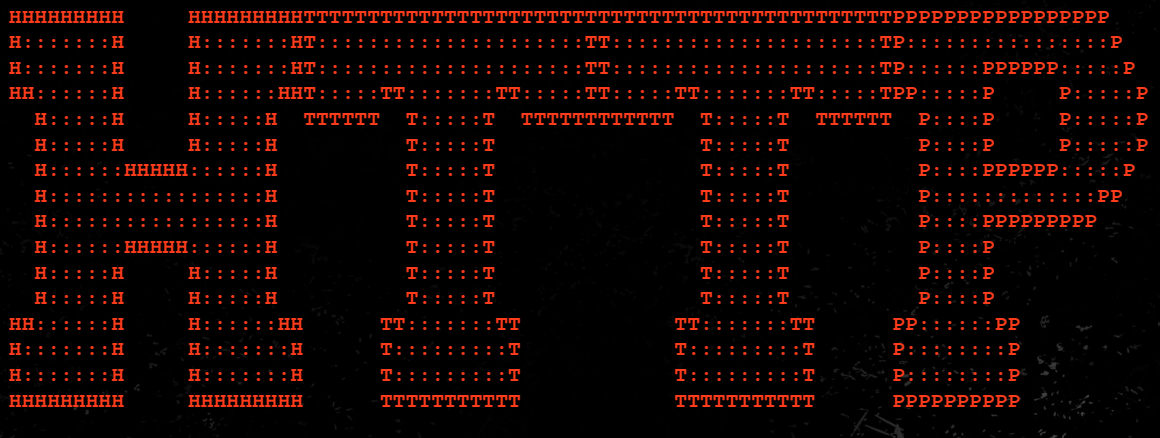
```php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
      continue;
  }
  echo "The number is: $x <br>";
}
```

Functions

```
function writeMsg() {
  echo "Hello world!";
}

writeMsg();
```

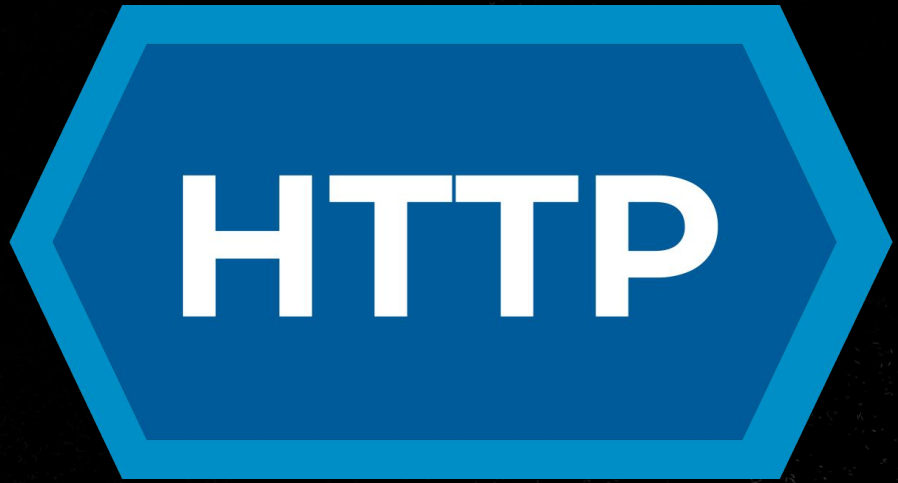
3-1. Understand HTTP

Let’s talk about HTTP Protocol



HTTP

In telecommunication, a communication protocol is a system of rules that allow two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. HTTP is one of them!



liangroup

HTTP Details

- HTTP is the abbreviation for HyperText Transfer Protocol and It just transfer HyperTexts.
- HTTP is an application-layer protocol that runs over TCP and It's Stateless!
- We have different versions of HTTP (e.g. 0.9, 1.0, 1.1, 2, 3)
- Methods (GET, POST, HEAD, OPTIONS, TRACE, etc.)
- You can make HTTP Packets everywhere! (e.g. Netcat, OpenSSL, Telnet, Burp Suite, etc.)
- HTTP is Simple! Complex tasks will be handled by TCP/IP.

REF: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

REF: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

URL Structure

- Scheme
- User [Optional]
- Password [Optional]
- Host (Hostname & TLD)
- Port
- Path
- Params (Query)
- Frag

URL

http://admin:password@site.com:8080/secret?id=1&s=true&q=fun/#section1

prot:// user : pass @ domain : port/ path ? params / frag

HTTP Packet

[Method] [DIR+PARAMS] [VER]

[HEADERS]

[CRLF][CRLF]

liangroup

HTTP Protocol Versions

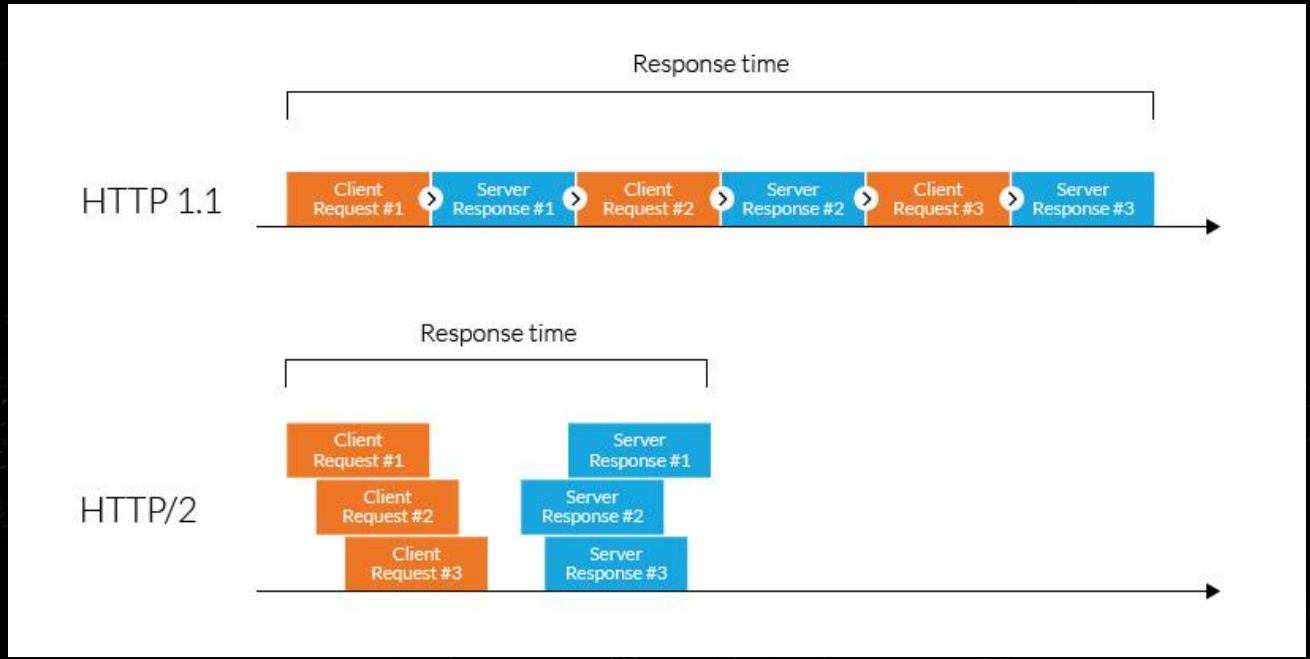
HTTP/1.0

- GET, HEAD, POST
- Connection terminate immediately after the response
- Some Header like Content-Type added
- Host Header (Optional)

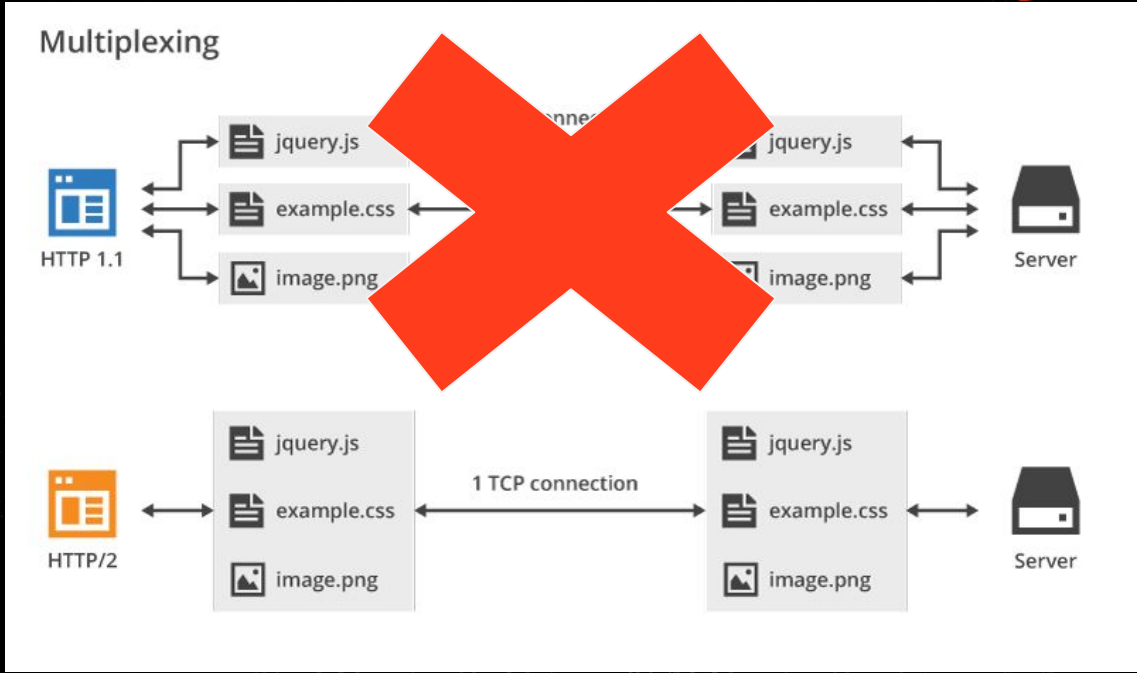
HTTP/1.1

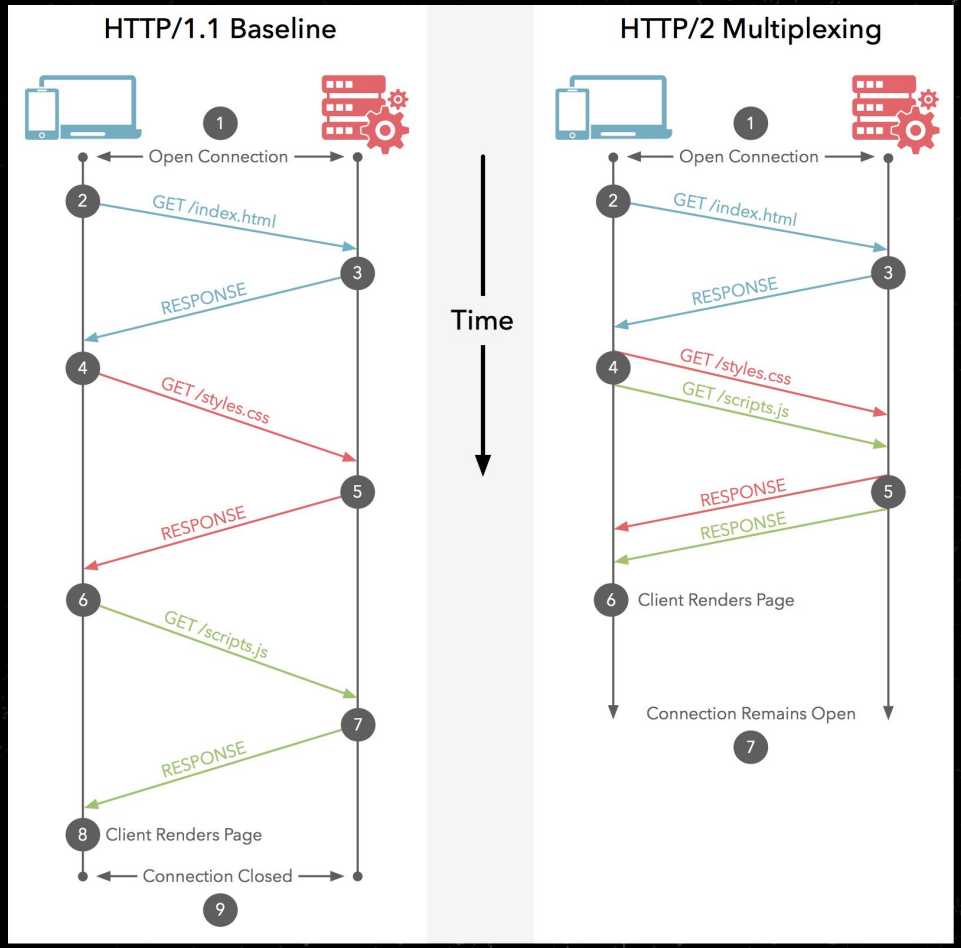
- GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, …
- Virtual-Host (Mandatory Host Header)
- Connection: keep-alive
- Upgrade: websocket

HTTP/2

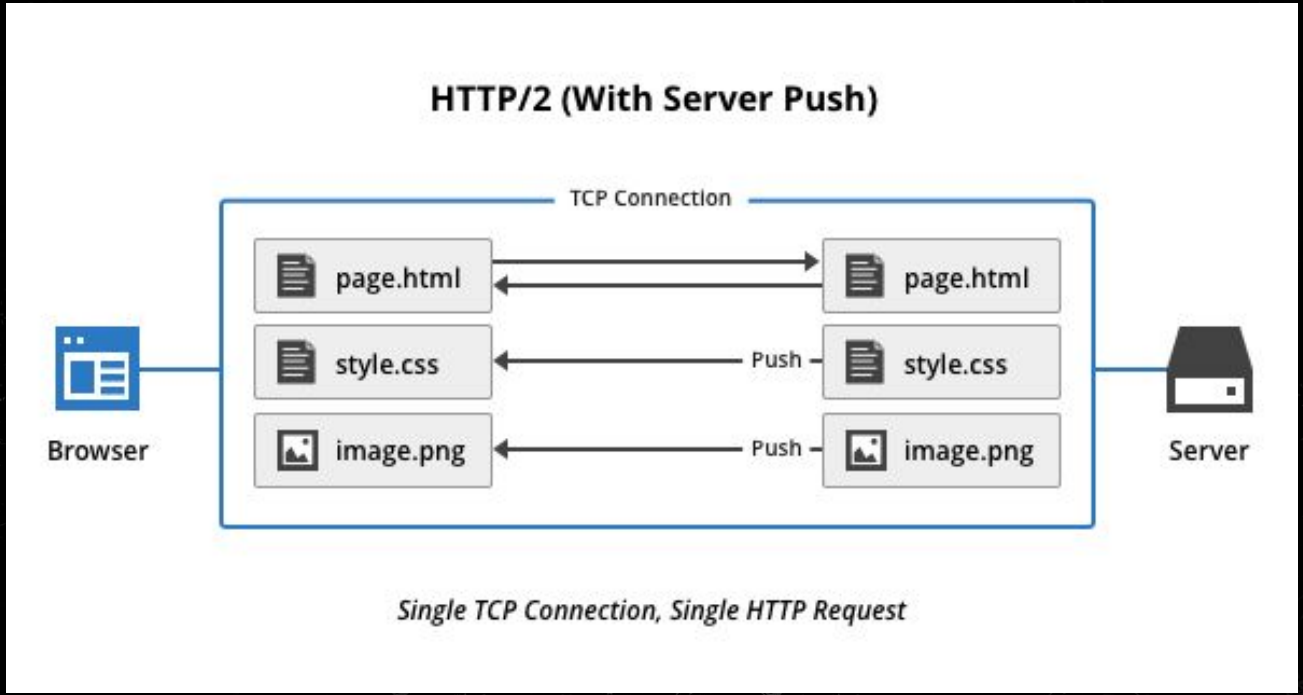

Response time
HTTP 1.1
Client Request #1
Server Response #1
Client Request #2
Server Response #2
Client Request #3
Server Response #3
Response time
HTTP/2
Client Request #1
Client Request #2
Client Request #3
Server Response #1
Server Response #2
Server Response #3

HTTP/2 Multiplexing



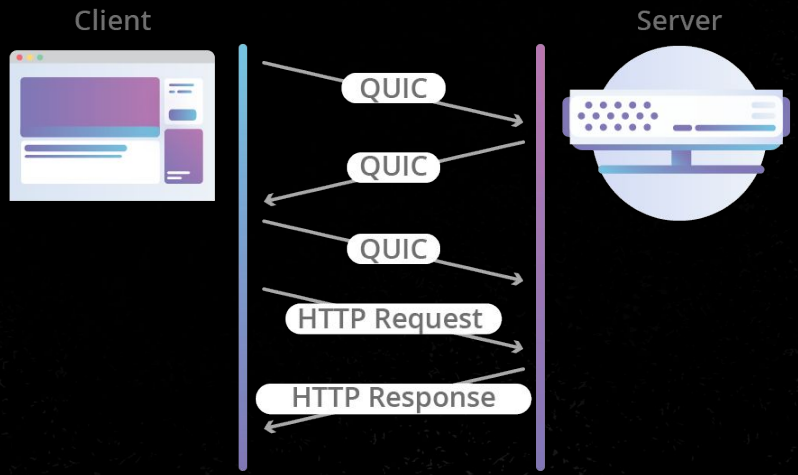

HTTP/1.1 Baseline
HTTP/2 Multiplexing
Time
1
Open Connection
2
GET /index.html
3
RESPONSE
4
GET /styles.css
5
RESPONSE
6
GET /scripts.js
7
RESPONSE
8
Client Renders Page
Connection Closed
9
1
Open Connection
2
GET /index.html
3
RESPONSE
4
GET /styles.css
GET /scripts.js
5
RESPONSE
RESPONSE
6
Client Renders Page
Connection Remains Open
7

HTTP/2 Server Push

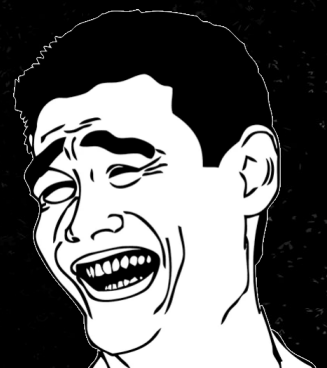


HTTP/3.0

HTTP Request Over QUIC



- A New Transport Protocol (QUIC)
- TLS/1.3 Handshake in QUIC Handshake



Create HTTP Packets Manually

```
nc www.web.site 80
  GET / HTTP/1.1
  Host: www.web.site
```

```
openssl s_client -connect web.site:443
  GET / HTTP/1.1
  Host: www.web.site
```

HTTP Methods

GET

- $URL/?p1=v1
- DO NOT send sensitive data
- 2KB Data limit (2048 char)

POST

- Bookmark?!
- Any type of data
- No length limit

HTTP Methods

HEAD

- Just give me Headers!
- Fast!
- There is nothing more actually

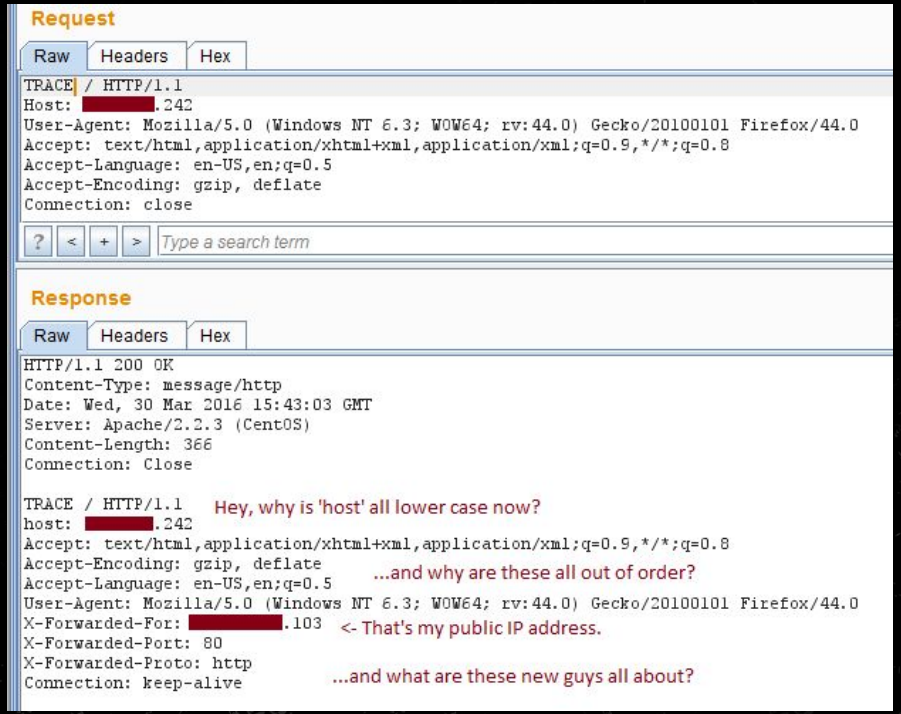
OPTIONS

- Available methods, That’s all

HTTP Methods

TRACE

- What server actually get?

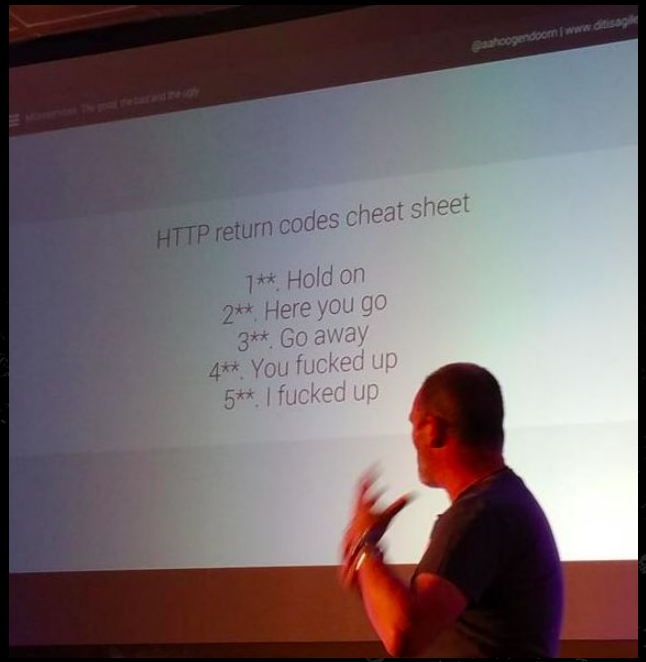


HTTP Methods

PUT/DELETE

- Upload a file to server
- Delete a file from server

HTTP Status Code



How Responses talk?!

https://http.cat

liangroup


I'M GETTING TOO
OLD FOR THIS SHIT
imgflip.com

PHP Application Input

- $GLOBALS - Global variables defined in the script(s)
- $_SERVER - Access client and response data
- $_REQUEST - Any GET/POST request parameters
- $_POST - Any POST request parameters
- $_GET - Any GET request parameters
- $_FILES - Any multi-part request parameters
- $_ENV - Any PHP environment variables
- $_COOKIE - Any cookies sent in any type of request
- $_SESSION - Session variables

HTTP Methods in PHP

- $_REQUEST['id']
- $_GET['id']
- $_POST['id']

$_SERVER[]

- PHP_SELF
- SERVER_NAME
- HTTP_HOST
- HTTP_REFERER
- HTTP_USER_AGENT

https://www.w3schools.com/php/php_superglobals_server.asp

Wrappers

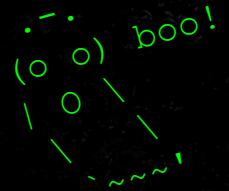
- ftp://
- http://
- file://
- data://
- etc.

https://www.php.net/manual/en/wrappers.php

Functions 0x01

- var_dump
- $_FILES
- mkdir()
- explode()
- count()
- uniqid()
- move_uploaded_file

Sample 0x01



Let’s check our first example

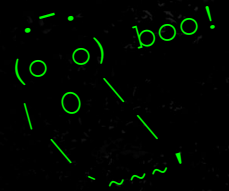
(uploader.php) code.

https://www.w3schools.com/php/php_file_upload.asp

Functions 0x02

- require(), include()
- str_replace()
- trim()
- isset()
- htmlspecialchars()
- getcwd()
- file_get_contents()
- ucword()

Sample 0x02



Let’s check our second example

(simple-php-website) code.

Functions 0x03

- file_put_contents()

Sample 0x03

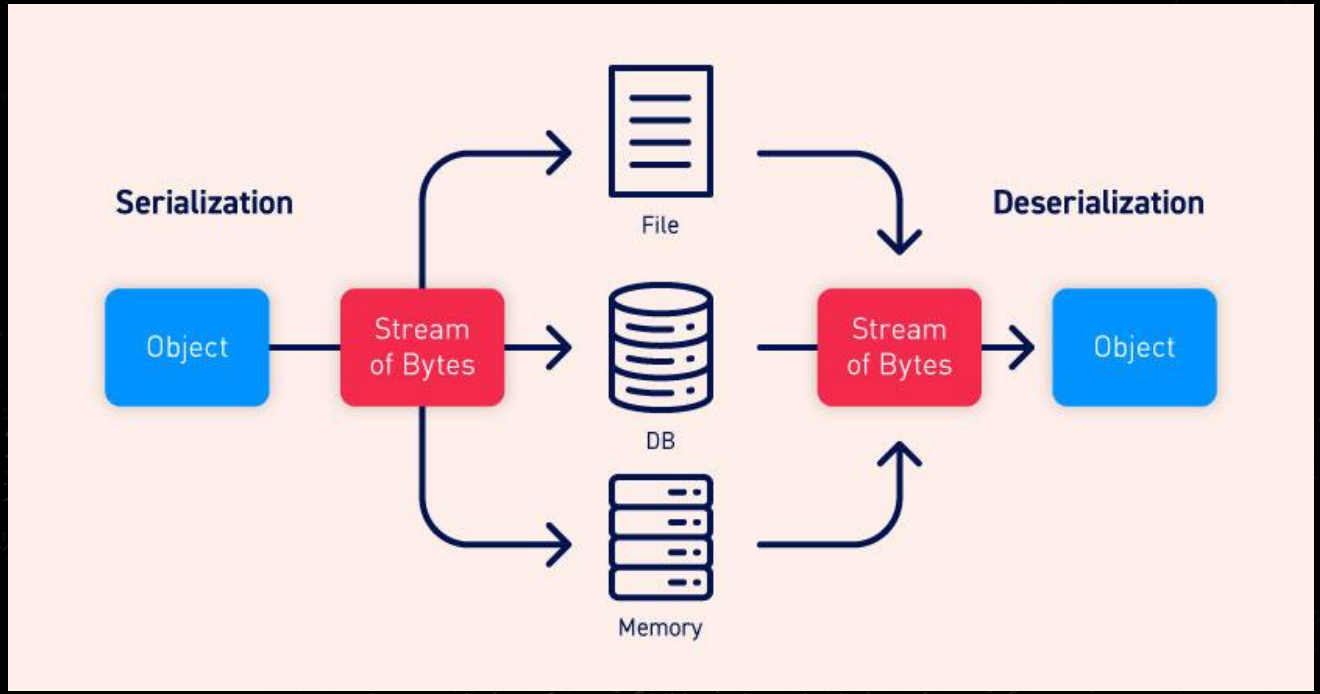
Let’s check our third example

(holy_bug.php) code.

Functions 0x04

- serialize()
- unserialize()

De/Serialization



Sample 0x04

Let’s check our fourth example (serialize) folder.

Vulnerable Functions

https://github.com/dustyfresh/PHP-vulnerability-audit-cheatsheet

Database Connection

```
$servername = "localhost";
$username = "username";
$password = "password";
$dbname    = “liangroup_db”;
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
```

liangroup

INSERT Operation

```php
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Lian', 'Group', 'info@liangroup.net')";
if ($conn->query($sql) === TRUE) {
  echo "New record created successfully!";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}
```

liangroup

Where Operation

```
$sql = "SELECT id, firstname, lastname FROM users_tbl WHERE lastname='Jafari'";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
  while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}
```

What is PHP Data Objects (PDO)?!

PDO refers to PHP Data Object, which is a PHP extension that defines a lightweight and consistent interface for accessing a database in PHP. It is a set of PHP extensions which provide a core PDO class and database-specific driver. Each database driver can expose database-specific features as a regular extension function that implements the PDO interface.

PDO Example

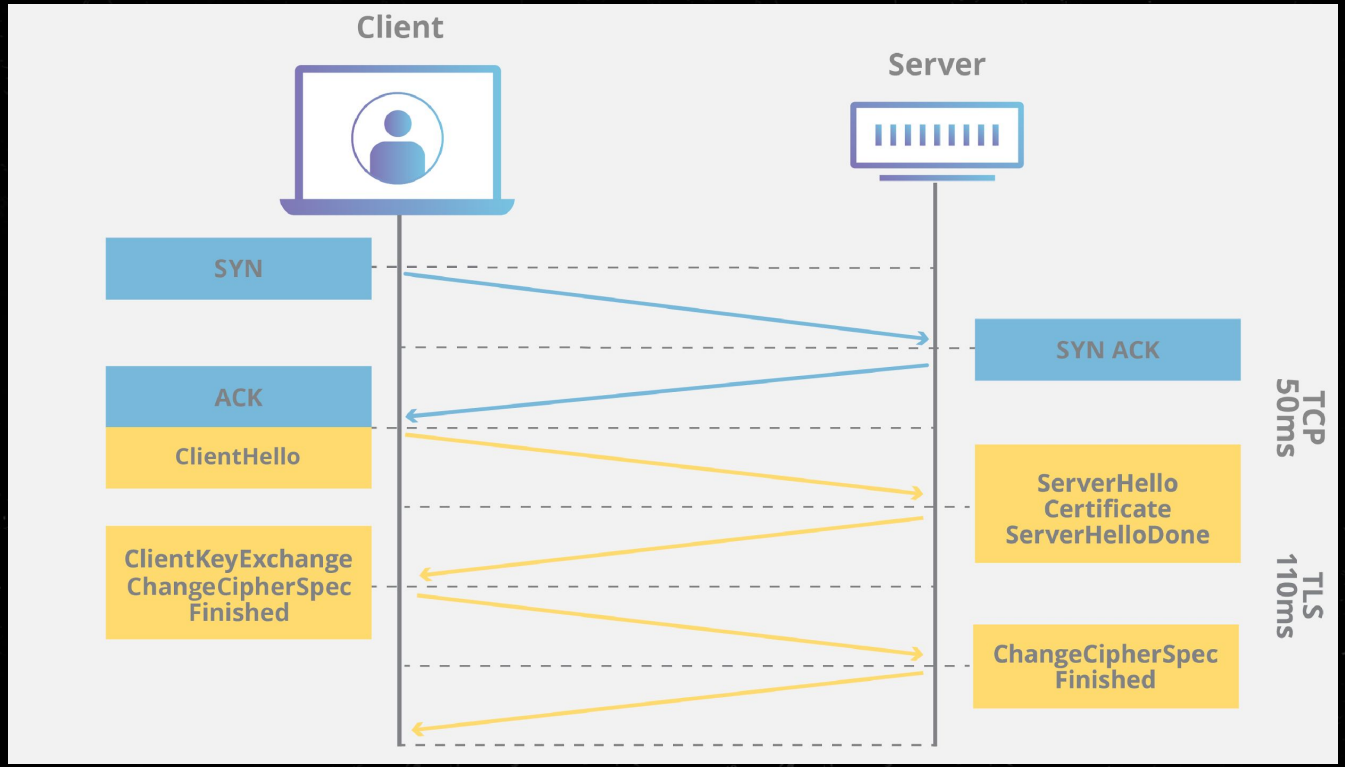
```
$pdo=new pdo("mysql:host=localhost;dbname=liangroup_db","root","123");
$sql=$pdo->prepare("select name from users where id=?");
$sql->bindValue(1,$_GET['id']);
$sql->execute();
if($sql->rowCount()>=1){
    echo "good";
} else{
    echo "bad";
}
```

liangroup

04. Let's Hack

Hack The Planet!

TLS Handshake



Step 0x01

The 'client hello' message: The client initiates the handshake by sending a "hello" message to the server. The message will include which TLS version the client supports, the cipher suites supported, and a string of random bytes known as the "client random".

Step 0x02

The 'server hello' message: In reply to the client hello message, the server sends a message containing the server's SSL certificate, the server's chosen cipher suite, and the "server random," another random string of bytes that's generated by the server.

Step 0x03

Authentication: The client verifies the server's SSL certificate with the certificate authority that issued it. This confirms that the server is who it says it is, and that the client is interacting with the actual owner of the domain.

Step 0x04

The premaster secret: The client sends one more random string of bytes, the "premaster secret." The premaster secret is encrypted with the public key and can only be decrypted with the private key by the server. (The client gets the public key from the server's SSL certificate.)

Step 0x04

Private key used: The server decrypts the premaster secret.

Step 0x05

Session keys created: Both client and server generate session keys from the client random, the server random, and the premaster secret. They should arrive at the same results.

Step 0x06

Client is ready: The client sends a "finished" message that is encrypted with a session key.

Step 0x07

Server is ready: The server sends a "finished" message encrypted with a session key.

Step 0x08

Secure symmetric encryption achieved: The handshake is completed, and communication continues using the session keys.

https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake

SSL/TLS Vulnerability

- POODLE (SSLv3)
- CRIME (TLS Compression) (CVE-2012-4929)
- BREACH (CVE-2013-3587)
- BEAST
- Sweet32
- Heartbleed (CVE-2014-0160)
- Raccoon

SSL/TLS Vulnerability

- SSLv2 and SSLv3 Connectivity
- CCS Injection
- TLS_FALLBACK_SCSV support
- DROWN
- FREAK
- LUCKY13
- Check for Weak Ciphers
- Logjam

SSL/TLS Vulnerability

- https://testssl.sh
- https://nabla-c0d3.github.io/sslyze/documentation
- https://github.com/rbsec/sslscan
- https://www.ssllabs.com/ssltest
- https://sslcheck.cert.ir/fa
- SSL Scanner (Burp Suite Extension)

How Browser Works?

- Add protocol (Recently HTTPS Become Default)
- Check /etc/hosts
- LIB_CALL getaddrinfo & gethostbyname (Obsolete)
- Local DNS cache
- External DNS Server