



# **Web Application Penetration Testing Report**

Product Name: Security Shepherd

Product Version: v3.1

Test Completion: 22/04/2021

Lead Penetration Tester: Weronika Wolska

Prepared for: Mark Scanlon

# Consultant Information

Name: Weronika Wolska

Email: [Veronika.wolska@ucdconnect.ie](mailto:Veronika.wolska@ucdconnect.ie)

Location: University College Dublin, Belfield, Dublin 4, Ireland

Manager: Mark Scanlon

Manager email: [mark.scanlon@ucd.ie](mailto:mark.scanlon@ucd.ie)

## **NOTICE**

This document contains confidential and proprietary information that is provided for the sole purpose of permitting the recipient to evaluate the recommendations submitted. In consideration of receipt of this document, the recipient agrees to maintain the enclosed information in confidence and not reproduce or otherwise disclose the information to any person outside the group directly responsible for evaluation of its contents.

## **WARNING**

### **Sensitive Information**

This document contains confidential and sensitive information about the security posture of the OWASP Security Shepherd Application. This information should be classified. Only those individuals that have a valid need to know should be allowed access this document.

# Index

<b>Executive Summary</b>	5
<b>Scope</b>	6
<b>Test Cases</b>	7
<b>Findings</b>	8
Poor Session Management 1	8
Poor Session Management 2	11
Poor Session Management 8	13
Failure to Restrict URL Access 3	17
SQL Injection 1	19
SQL Injection 3	21
Insecure Direct Object Reference 2	25
Cross Site Scripting 3	28
Cross Site Scripting 5	31
Poor Data Validation 1	33
<b>Recommendations and Conclusions</b>	37

## Executive Summary

Lead Tester: Weronika Wolska

Number of days testing: 16

Test Start Date: 12/04/2021

Test End Date: 27/04/2021

### Project Information

Application Name: Security Shepherd

Application Version: v3.1

Release Date: 11/10/2018

Project Contact: Nikita Pavlenko

Findings:

OWASP Top 10:

Total Defects:

Severity	#Defects
Critical	4
High	3
Medium	3
Low	0

## Scope

### Challenges Attempted:

Poor Data Validation – Challenge 1  
Failure to Restrict URL Access – Challenge 1  
SQL Injection – Challenge 3  
SQL Injection – Challenge 1  
Insecure Direct Object Reference – Challenge 2  
XXS – Challenge 3  
XXS – Challenge 5  
Poor Session Management – Challenge 1  
Poor Session Management – Challenge 2  
Poor Session Management – Challenge 8

### Time Frame:

Testing start date: 12/04/2021  
Testing end date: 23/04/2021  
Report start date: 23/04/2021  
Report end date: 27/04/2021

### User roles

Username: admin  
Authorisation level: admin

### URLs:

<https://www.md5online.org/md5-decrypt.html>  
<https://www.md5online.org/md5-encrypt.html>  
<https://portswigger.net/burp/communitydownload>  
<https://www.base64decode.org/>  
<https://www.base64encode.org/>  
[http://qbarbe.free.fr/crypto/eng\\_atom128c.php](http://qbarbe.free.fr/crypto/eng_atom128c.php)  
<https://192.168.56.102/>

# Test Cases

## **Poor Data Validation – Challenge 1**

Business Logic Testing: Test Business Logic Data Validation (OTG-BUSLOGIC-001)

## **Failure to Restrict URL Access – Challenge 1**

Client Side Testing: Testing for Client Side Resource Manipulation (OTG-CLIENT-006)

## **SQL Injection – Challenge 3**

Input Validation: Testing for SQL Injection (OTG-INPVAL-005)

## **SQL Injection – Challenge 1**

Input Validation: Testing for SQL Injection (OTG-INPVAL-005)

## **Insecure Direct Object Reference – Challenge 2**

## **XXS – Challenge 3**

Client Side Testing: Testing for JavaScript Execution (OTG-CLIENT-002)

## **XXS – Challenge 5**

Client Side Testing: Testing for JavaScript Execution (OTG-CLIENT-002)

## **Poor Session Management – Challenge 1**

Authentication Testing: Testing for Credentials Transported over an Encrypted Channel (OTG-AUTHN-001)

## **Poor Session Management – Challenge 2**

Authentication Testing: Testing for weak password change or reset functionalities (OTG-AUTHN-009)

## **Poor Session Management – Challenge 8**

Authentication Testing: Testing for Credentials Transported over an Encrypted Channel (OTG-AUTHN-001)

## Findings

### Critical: Poor Session Management [CWE-930]

Broken Authentication and Session Management means that an application does not have secure procedures in place to authenticate its users or sufficient authorisation control. While authentication and session management can be broken through other attacks, such as Cross Site Scripting, it can also happen if an application has a flawed authentication and session management schema, just like below. Such flaws are commonly found in logout, password management, secret question and account update (Source: OWASP Security Shepherd). In this case, the web application does not have secure authorisation procedures in place and displays the email address of a user if you type in an incorrect password. This allows you to change the user's password and gain access to their account without their knowledge.

#### Steps to reproduce:

1. Download and run Burp Suite <https://portswigger.net/burp/communitydownload> (making sure you have Oracle Java Installed).
2. Utilising Firefox set the system proxy to route traffic through Burp - "Open Menu" button in the right hand corner -> Advanced -> Network (tab) -> Connection "Settings Button" -> Manual proxy configuration. The default for Burp is 127.0.0.1 with a port of 8080.
3. Go to Security Shepherd (<https://192.168.56.102/>) in Firefox and log in.
4. Go to Challenges -> Session Management -> Session Management Challenge 1.

## Session Management Challenge One

Only administrators of the following sub-application can retrieve the result key.

Administrator Only Button

5. Make sure that intercept is on in BurpSuite and click "Administrator Only Button". Navigate to BurpSuite.

```
Raw Params Headers Hex
POST /challenges/dfd6bfbal033fa380e378299b6a998c759646bd8aea02511482b8ce5d707f93a HTTP/1.1
Host: 192.168.56.103
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Referer: https://192.168.56.103/challenges/dfd6bfbal033fa380e378299b6a998c759646bd8aea02511482b8ce5d707f93a.jsp
Content-Type: application/x-www-form-urlencoded
X-Requested-With: XMLHttpRequest
Content-Length: 65
Cookie: checksum=dXNlclJybGU9dXNlbg==; JSESSIONID=BCEE3E1DEF6C60DCC0FB783D9A580168; token=-25832737592042591599627528598948371045; JSESSIONID3=0XnuISFeol1WzhSu4uiqq==
Connection: close

adminDetected=false&returnPassword=false&upgradeUserToAdmin=false
```

6. First, change the values of adminDetected, returnPassword and upgradeUserToAdmin to "true". Then copy the checksum value.
7. Go to <https://www.base64decode.org/>. Paste the checksum value and click „Decode“.



### Decode from Base64 format

Simply enter your data then push the decode button.

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

**AUTO-DETECT** Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

☒ Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

**< DECODE >** Decodes your data into the area below.

- Change to Encoding mode on the top bar. Type "userRole=admin" in the top text box and press encode.

### BASE64

Decode
Encode

Do you have to deal with Base64 format? Then this site is perfect for you! Use our super handy online tool to **encode** or **decode** your data.

#### Encode to Base64 format

Simply enter your data then push the encode button.

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Destination character set.

LF (Unix) Destination newline separator.

☐ Encode each line separately (useful for when you have multiple entries).

☐ Split lines into 76 character wide chunks (useful for MIME).

☐ Perform URL-safe encoding (uses Base64URL format).

☒ Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

**> ENCODE <** Encodes your data into the area below.

- Copy the encoded string and paste it as the checksum value in BurpSuite. Press "Forward".

```
POST /challenges/dfd6bfb1033fa380e378299b6a998c759646bd8aea02511482b8ce5d707f93a HTTP/1.1
Host: 192.168.56.103
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Referer: https://192.168.56.103/challenges/dfd6bfb1033fa380e378299b6a998c759646bd8aea02511482b8ce5d707f93a.jsp
Content-Type: application/x-www-form-urlencoded
X-Requested-With: XMLHttpRequest
Content-Length: 65
Cookie: checksum=dXNlcJvbGU9YWRTaW4=; JSESSIONID=BCEE3E1DEF6C60DCC0FB783D9A5BD168; token=-25832737592042591599627528598948371045; JSESSIONID3=OXnuISF*o11WzhSu4u1qg==
Connection: close

adminDetected=true&returnPassword=true&upgradeUserToAdmin=true
```

10. Navigate to Security Shepherd.

## Session Management Challenge One

Only administrators of the following sub-application can retrieve the result key.

Administrator Only Button

## HACK DETECTED

A possible attack has been detected. Functionality Stopped before any damage was done

It appears “admin” is not a valid userRole.

11. Go back to <https://www.base64encode.org/> and this time encode “userRole=administrator”. Copy the encoded string.

**Encode to Base64 format**  
Simply enter your data then push the encode button.

userRole=administrator

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Destination character set.  
LF (Unix) Destination newline separator.

☐ Encode each line separately (useful for when you have multiple entries).  
☐ Split lines into 76 character wide chunks (useful for MIME).  
☐ Perform URL-safe encoding (uses Base64URL format).

☒ Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

**> ENCODE <** Encodes your data into the area below.

dXNlclJvbGU9YWRTaW5pc3RyYXRvcg==

12. Navigate back to Firefox. Repeat steps 5 and 6. This time replace the checksum with the new encoded string. Press “Forward”.
13. You will see that you gained admin privileges.

## Session Management Challenge One

Only administrators of the following sub-application can retrieve the result key.

Administrator Only Button

## Admin Only Club

Welcome administrator. Your result key is as follows

+/1qMeyRC+jZ4snSgmG8k1FznnKeoyDUWUAadYM4H+79NX0GUmbIXc  
BwSzdLvAImucgh+RNA66xDSISicOrMH3rdJwZekDaxcoF/pLyuE=

### CVSS Score 9.8

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

### Mitigation:

It is recommended to use a salted hash instead of an encryption algorithm for transferring user roles. This makes the user role values more difficult to guess. It is also recommended to not transmit adminDetected, returnPassword and upgradeUserToAdmin in plain text, as it makes the value for these variables easy to modify for an attacker.

### **Critical: Poor Session Management [CWE-930]**

Broken Authentication and Session Management means that an application does not have secure procedures in place to authenticate its users or sufficient authorisation control. While authentication and session management can be broken through other attacks, such as Cross Site Scripting, it can also happen if an application has a flawed authentication and session management schema, just like below. Such flaws are commonly found in logout, password management, secret question and account update (Source: OWASP Security Shepherd). In this case, the web application does not have secure authorisation procedures in place and displays the email address of a user if you type in an incorrect password. This allows you to change the user's password and gain access to their account without their knowledge.

### Steps to reproduce:

1. Download and run Burp Suite <https://portswigger.net/burp/communitydownload> (making sure you have Oracle Java Installed).
2. Utilising Firefox set the system proxy to route traffic through Burp -"Open Menu" button in the right hand corner -> Advanced -> Network (tab) -> Connection "Settings Button" -> Manual proxy configuration. The default for Burp is 127.0.0.1 with a port of 8080.
3. Go to Security Shepherd (<https://192.168.56.102/>) in Firefox and log in.
4. Go to Challenges -> Session Management -> Session Management Challenge 2.

## Session Management Challenge Two

Only an **admin** of the following sub-application can retrieve the result key to this challenge.

User Name:

Password:

[Have you forgotten your password?](#)

5. Try to login with username = "admin" and password = "password".

Incorrect password for **zoidberg22@shepherd.com**

Username:

Password:

[Have you forgotten your password?](#)

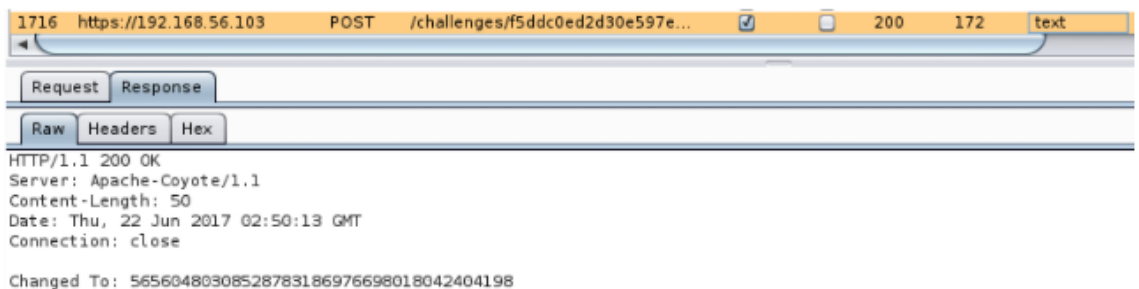
We now have an email address for admin.

6. Click on "Have you forgotten password?" and Type the above email into the text box. Make sure that intercept is on in BurpSuite and click "Reset Password".

## Reset Password

Please enter your **email address**. You will be sent an email with a new temporary password

7. Navigate to BurpSuite. Click Action -> Do intercept -> Response to this request and press "Forward". You will see the following response:



8. Copy the password from the "Changed To" field in the response request. Navigate to Security Shepherd and log in as admin with the new password. You will see that you have successfully logged in as admin.

### CVSS Score 9.8

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

### Mitigation:

It is recommended to show less detailed error messages within the web application. For example, use “username or password incorrect” as opposed to “incorrect password for ...”. This way, a potential attacker will not know if the entered username is valid since either field could be incorrect. It is also advised to never reference direct data within an error message to protect the confidentiality of users. As well as that, the changed password should not be sent back in plain text and ideally not sent in a HTTP packet entirely. The best protocol for password changing is to send an email to the user directly and send a confirmation of the email being sent in the HTTP response.

### **Critical: Poor Session Management [CWE-930]**

Broken Authentication and Session Management means that an application does not have secure procedures in place to authenticate its users or sufficient authorisation control. While authentication and session management can be broken through other attacks, such as Cross Site Scripting, it can also happen if an application has a flawed authentication and session management schema, just like below. Such flaws are commonly found in logout, password management, secret question and account update (Source: OWASP Security Shepherd). In this case, the web application does not have secure authorisation procedures in place and displays the email address of a user if you type in an incorrect password. This allows you to change the user’s password and gain access to their account without their knowledge.

### Steps to reproduce:

1. Download and run Burp Suite <https://portswigger.net/burp/communitydownload> (making sure you have Oracle Java Installed).
2. Utilising Firefox set the system proxy to route traffic through Burp -"Open Menu" button in the right hand corner -> Advanced -> Network (tab) -> Connection "Settings Button" -> Manual proxy configuration. The default for Burp is 127.0.0.1 with a port of 8080.
3. Go to Security Shepherd (<https://192.168.56.102/>) in Firefox and log in.
4. Go to Challenges -> Session Management -> Session Management Challenge Eight.

## Session Management Challenge Eight

Only highly privileged users of the following sub-application can retrieve the result key.

Privileged User Only Button

- Click "Privileged User Only Button".

You're not a privileged User!!!

Stay away from the privileged only section. The super aggressive dogs have been released.

- Make sure that intercept is on in BurpSuite and press the button again. Navigate to BurpSuite.

```
POST /challenges/714d8601c303bbef8b5cabab60b1060ac41f0d96f53b6ea54705bb1ea4316334 HTTP/1.1
Host: 192.168.56.101
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://192.168.56.101/challenges/714d8601c303bbef8b5cabab60b1060ac41f0d96f53b6ea54705bb1ea4316334.jsp
Content-Type: application/x-www-form-urlencoded
X-Requested-With: XMLHttpRequest
Content-Length: 61
Cookie: challengeRole=LmH6nmbC; JSESSIONID=6D27D5A83FBAC29AF449E2187BC78F1F; token=-135636708303766387706989682725427765061; JSESSIONID3="5gINJWrlrvk6fC7JTLsx/A=="
Connection: close

returnUserRole=false&returnPassword=false&adminDetected=false
```

- Change returnUserRole, returnPassword and adminDetected to "true".
- Open browser and go to [http://qbarbe.free.fr/crypto/eng\\_atom128c.php](http://qbarbe.free.fr/crypto/eng_atom128c.php). Paste the challengeRole value into the text box and click "decrypt".

CRYPTO.COM Free Online Encryption Services

ATOM-128 Encrypt or Decrypt message  
A simple way to encrypt / decrypt a text message is to use ATOM-128

Type or paste in the text you want to encrypt or decrypt

LmH6nmbC

encrypt decrypt clear

Type or paste in the text you want to encrypt or paste ATOM-128 encrypted message into the text field and click decrypt!

We get the following result:

CRYPTO.COM Free Online Encryption Services

ATOM-128 Encrypt or Decrypt message  
A simple way to encrypt / decrypt a text message is to use ATOM-128

Type or paste in the text you want to encrypt or decrypt

guest

encrypt decrypt clear

Type or paste in the text you want to encrypt or paste ATOM-128 encrypted message into the text field and click decrypt!

9. Type "admin" into the box and click "decrypt".

CRYPTO.COM Free Online Encryption Services

ATOM-128 Encrypt or Decrypt message  
A simple way to encrypt / decrypt a text message is to use ATOM-128

Type or paste in the text you want to encrypt or decrypt

admin|

encrypt decrypt clear

Type or paste in the text you want to encrypt or paste ATOM-128 encrypted message into the text field and click decrypt!

Result:

CRYPTO.COM Free Online Encryption Services

ATOM-128 Encrypt or Decrypt message  
A simple way to encrypt / decrypt a text message is to use ATOM-128

Type or paste in the text you want to encrypt or decrypt

KIA9MiNC

encrypt decrypt clear

Type or paste in the text you want to encrypt or paste ATOM-128 encrypted message into the text field and click decrypt!

10. Copy the value and paste it as the value for challengeRole in BurpSuite and press "Forward".
11. We get an invalid role detected error. After repeating steps 9 and 10 for the strings "administrator", "privilegedUser" and "superuser", we find that "superuser" is a valid role.
12. Click the "Privileged User Only Button" in Security Shepherd and navigate to BurpSuite.
13. Change returnUserRole, returnPassword and adminDetected to "true".
14. Replace the challengeRole value with "nmHqLjQknIHs" (ATOM-128 encoding of "superuser").
15. Press "Forward" and navigate to SecurityShepherd.

## Session Management Challenge Eight

---

Only highly privileged users of the following sub-application can retrieve the result key.

Privileged User Only Button

## Super User Only Club

---

Welcome super user! Your result key is as follows

2UMX99zPVN3soMOZEW8f3PNEzCGsryqk4C6TsQ+LnCDZtiq4M6mFDG  
YtLsazKcQShKbyAlz7f/taChC/ZucY25clYc0Dideh7/CEeLala7Dl5DuEerEK



You will see that we gained privileged user access.

### CVSS Score 9.8

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

### Mitigation:

It is recommended to use a salted hash instead of an encryption algorithm for transferring challenge roles. This makes the challenge role values more difficult to guess. It is also recommended to not transmit adminDetected, returnPassword and returnUserRole in plain text, as it makes the values of these variables easy to modify for an attacker.



## **Critical: Failure to Restrict URL Access [CWE-817]**

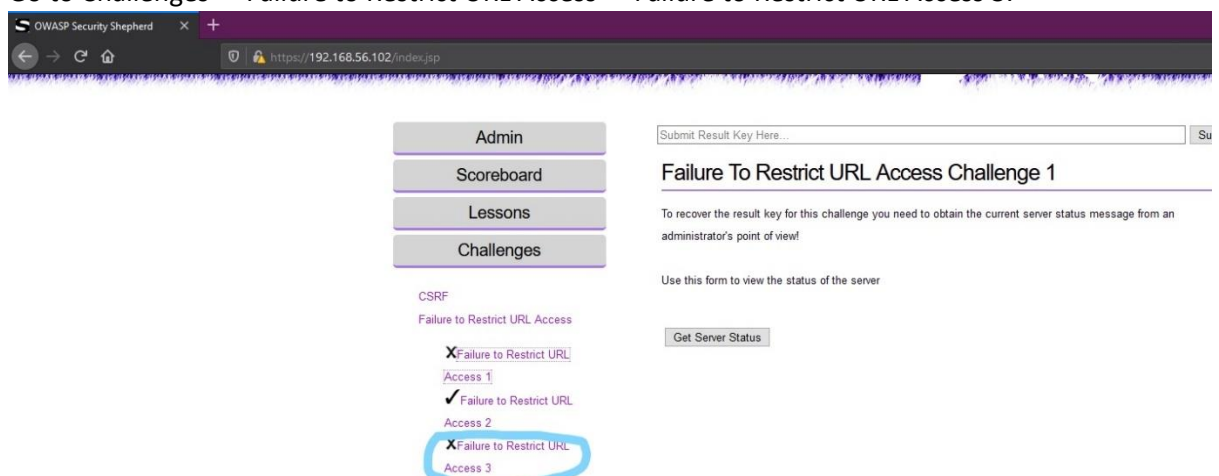
If your application fails to appropriately restrict URL access, security can be compromised through a technique called forced browsing. Forced browsing can be a very serious problem if an attacker tries to gather sensitive data through a web browser by requesting specific pages, or data files.

Using this technique, an attacker can bypass website security by accessing files directly instead of following links. This enables the attacker to access data source files directly instead of using the web application. The attacker can then guess the names of backup files that contain sensitive information, locate and read source code, or other information left on the server, and bypass the "order" of web pages.

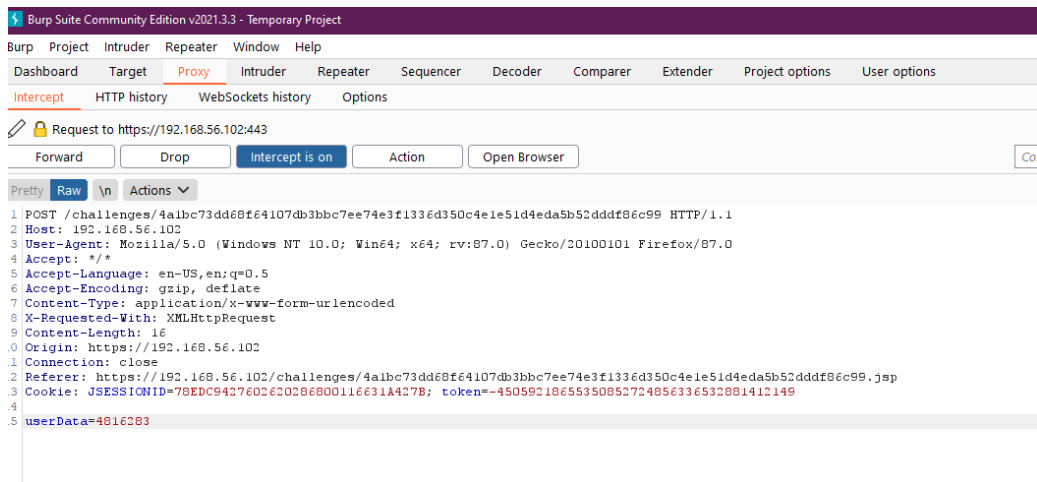
Simply put, Failure to Restrict URL Access occurs when an error in access-control settings results in users being able to access pages that are meant to be restricted or hidden. This presents a security concern as these pages frequently are less protected than pages that are meant for public access, and unauthorized users are able to reach the pages anonymously. In many cases, the only protection used for hidden or restricted pages is not linking to the pages or not publicly showing links to them. (source: <https://www.veracode.com/security/failure-restrict-url-access>)

### **Steps to reproduce:**

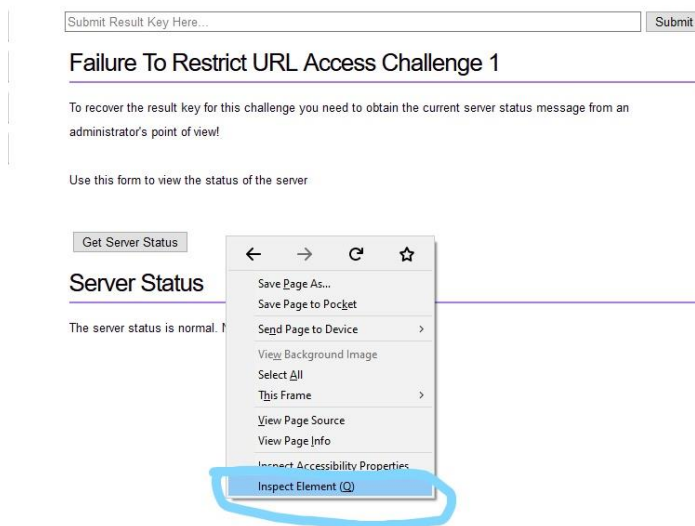
1. Go to Security Shepherd (<https://192.168.56.102/>) in Firefox and log in.
2. Go to Challenges -> Failure to Restrict URL Access -> Failure to Restrict URL Access 3.



3. Make sure that intercept is on in Burp Suite.
4. Click "Get Server Status" and navigate to Burp Suite.
5. You will see that there is a userData token assigned.



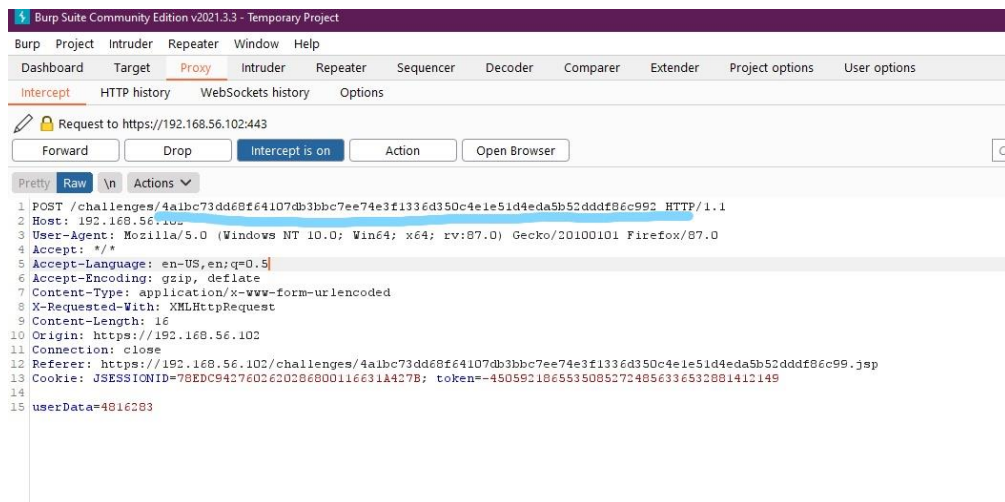
6. After several attempts, it turns out we cannot guess the userData value for administrators.
7. Left click on the page and select "Inspect Code".



8. After observing the source code for the page, we can see that there is a separate admin form:



9. The form has the same userData value but a different POST URL. Copy the URL for the admin form.
10. Click "Get Server Status" again and navigate to BurpSuite.
11. Copy the new URL value and press "Forward".



You will see that we got the server view from administrative perspective.

## CVSS Score 9.1

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	None

## Mitigation:

It is recommended that a permission or ACL is implemented to prevent unauthorised users from accessing pages they should not have access to. It is also recommended to use virtual directories for accessing web pages as opposed to the physical directories that contain sensitive data that should be secure.

## High: SQL Injection [CWE-727]:

SQL injection is the placement of malicious code in SQL statements, via web page input. SQL injection is a code injection technique that might destroy your database and is one of the most common web hacking techniques. It usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will unknowingly run on your database. (source: [https://www.w3schools.com/sql/sql\\_injection.asp](https://www.w3schools.com/sql/sql_injection.asp))

### Steps to reproduce:

1. Go to Security Shepherd (<https://192.168.56.102/>) in Firefox and log in.
2. Go to Challenges -> Injection->SQL Injection Challenge One.

## SQL Injection Challenge One

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

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

3. Type `"a' or 'a'='a"` into text box and click "Get user". We get a message saying "There were no results found in your search".
4. Let's try using the `"` character instead. Type `"a" or "a"="a"` into the text box and click "Get user". We get a table of all users.

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

### Search Results

Name	Address	Comment
John Fits	crazycat@example.com	null
Rubix Man	manycolours@cube.com	null
Rita Hanola	thenightbefore@example.com	null
Paul O'Brien	sixshooter@deaf.com	Well Done! The result Key is fd8e9a29dab791197115b58061b215594211e72c1680f1eac50b0394133a09f

### CVSS Score 7.5

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	None
Availability	None

## Mitigation:

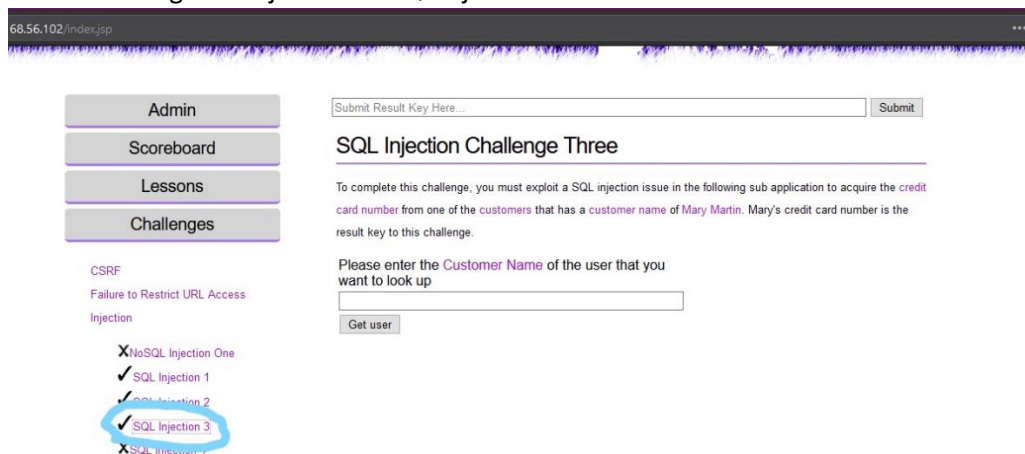
It is recommended to use prepared statements with Parameterized Queries when accessing the database based on user input. Parameterized queries force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied. Prepared statements ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker.

## High: SQL Injection [CWE-727]:

SQL injection is the placement of malicious code in SQL statements, via web page input. SQL injection is a code injection technique that might destroy your database and is one of the most common web hacking techniques. It usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will unknowingly run on your database. (source: [https://www.w3schools.com/sql/sql\\_injection.asp](https://www.w3schools.com/sql/sql_injection.asp))

## Steps to reproduce:

1. Go to Security Shepherd (<https://192.168.56.102/>) in Firefox and log in.
2. Go to Challenges -> Injection -> SQL Injection 3.



3. Type "Mary Martin" into text box and click "Get user".

Submit Result Key Here...

## SQL Injection Challenge Three

To complete this challenge, you must exploit a SQL injection issue in the following sub application to acquire the **credit card number** from one of the **customers** that has a **customer name** of **Mary Martin**. Mary's credit card number is the result key to this challenge.

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

Mary Martin

## Search Results

**Name**  
Mary Martin

We can see that the customer exists but only their name is displayed.

4. Type **"x' or 'x'='x"** into the text box and click **"Get user"**.

Submit Result Key Here...

## SQL Injection Challenge Three

To complete this challenge, you must exploit a SQL injection issue in the following sub application to acquire the **credit card number** from one of the **customers** that has a **customer name** of **Mary Martin**. Mary's credit card number is the result key to this challenge.

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

x' or 'x'='x

## Search Results

**Name**  
Mark Denihan  
Jason McCoy  
Mary Martin  
Joseph McDonnell  
John Doe

We get the list of names of all customers but still no credit card details.

5. Now let's try to figure out the name of the variable that stores the customer name. Type **"Mary Martin' group by customername having '1'='1"** into the text box and click **"Get user"**.

## SQL Injection Challenge Three

---

To complete this challenge, you must exploit a SQL injection issue in the following sub application to acquire the **credit card number** from one of the **customers** that has a **customer name** of **Mary Martin**. Mary's credit card number is the result key to this challenge.

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

## Search Results

---

**Name**

Mary Martin

We now know that "customername" is used to store the names of the customers.

- Now let's figure out the variable that stores the credit card number. Type "Mary Martin' group by creditcardnumber having '1'='1" into text box and press "Get user".

## SQL Injection Challenge Three

---

To complete this challenge, you must exploit a SQL injection issue in the following sub application to acquire the **credit card number** from one of the **customers** that has a **customer name** of **Mary Martin**. Mary's credit card number is the result key to this challenge.

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

## Search Results

---

**Name**

Mary Martin

No error has occurred so "creditcardnumber" is the variable corresponding to the credit card information.

- Now type "Mary Martin' UNION SELECT creditcardnumber FROM customers WHERE customername = 'Mary Martin'" and click "Get user".

Submit Result Key Here...

Submit

## SQL Injection Challenge Three

To complete this challenge, you must exploit a SQL injection issue in the following sub application to acquire the **credit card number** from one of the **customers** that has a **customer name** of **Mary Martin**. Mary's credit card number is the result key to this challenge.

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

ardnumber FROM customers WHERE customername = 'Mary Martin

Get user

## Search Results

### Name

Mary Martin

9815 1547 3214 7569

We now have access to the credit card numbers of the customer.

### CVSS Score 7.5

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	None
Availability	None

### Mitigation:

It is recommended to use prepared statements with Parameterized Queries when accessing the database based on user input. Parameterized queries force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied. Prepared statements ensure that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker.

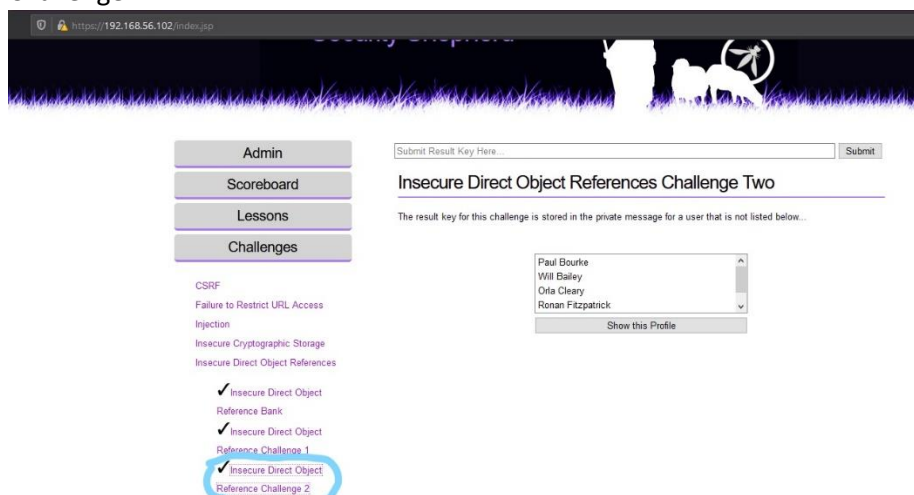


## **High: Insecure Direct Object Reference [CWE-715]:**

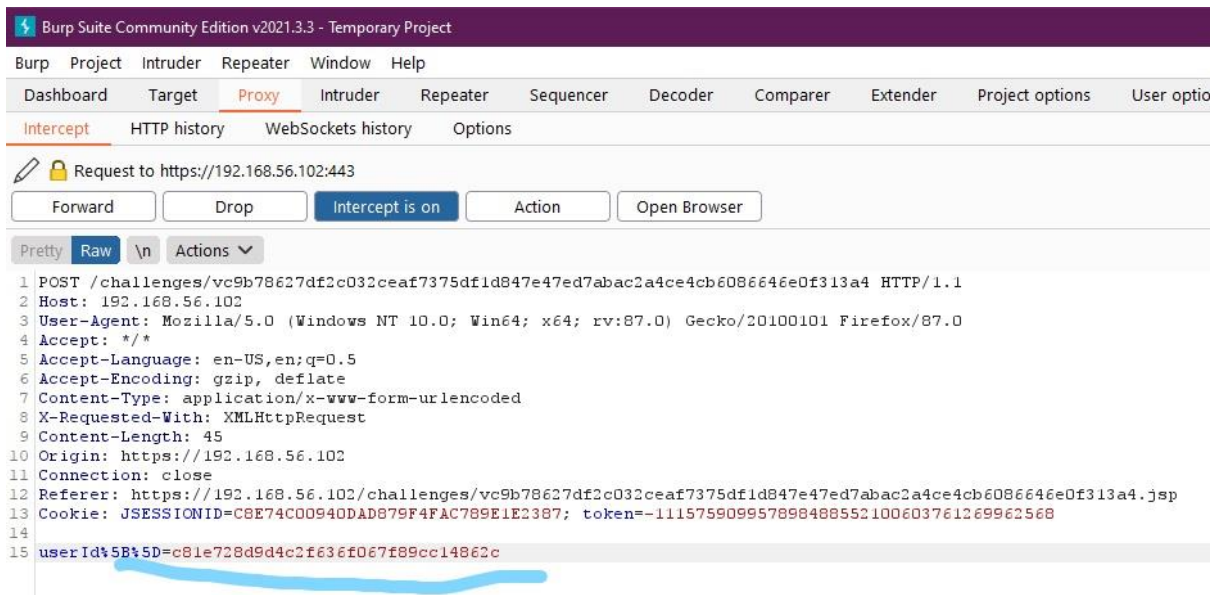
Insecure direct object references are a type of access control vulnerability that arises when an application uses user-supplied input to access objects directly. An attacker might be able to perform horizontal and vertical privilege escalation by altering the user to one with additional privileges while bypassing access controls. Other possibilities include exploiting password leakage or modifying parameters once the attacker has landed in the user's accounts page. (source: <https://portswigger.net/web-security/access-control/idor>)

### **Steps to reproduce:**

1. Download and run Burp Suite <https://portswigger.net/burp/communitydownload> (making sure you have Oracle Java Installed).
2. Utilising Firefox set the system proxy to route traffic through Burp - "Open Menu" button in the right hand corner -> Advanced -> Network (tab) -> Connection "Settings Button" -> Manual proxy configuration. The default for Burp is 127.0.0.1 with a port of 8080.
3. Go to Security Shepherd (<https://192.168.56.102/>) in Firefox and log in.
4. Go to Challenges -> Insecure Direct Object Reference-> Insecure Direct Object Reference Challenge 2.



5. Make sure that intercept is on in BurpSuite. Select "Paul Bourke" and click "Show this Profile".
6. Navigate to BurpSuite.

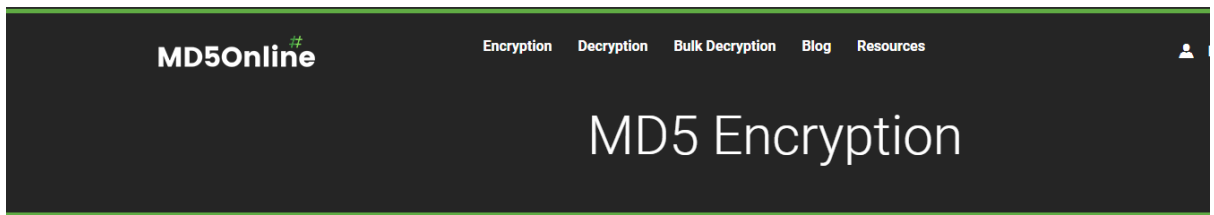


7. Go to <https://www.md5online.org/md5-decrypt.html> and copy the user ID into the text box. Press “Decrypt”.



You will see that the user ID decrypts to the number 2.

8. Repeat the process for the other users. You will find that the other IDs are: 3, 5, 7 and 11.
9. Let's try to see if a user with user ID = 13 exists. Go to <https://www.md5online.org/md5-decrypt.html>. Type „13” in the text box and click „Crypt”.



**MD5 Encryption**

Enter a word here to get its MD5 hash :

13

Crypt

No credit required in this tool

The MD5 hash for 13 is : **c51ce410c124a10e0db5e4b97fc2af39**

**Blog posts**

- How to Convert MD5 Passwords to SHA256?
- How to Open an MD5 File on Any System?
- Why MD5 Cannot Be Decrypted? (ar solutions)

10. Copy the hash and navigate back to Security Shepherd.
11. Select “Paul Bourke” and click “Show this Profile”.

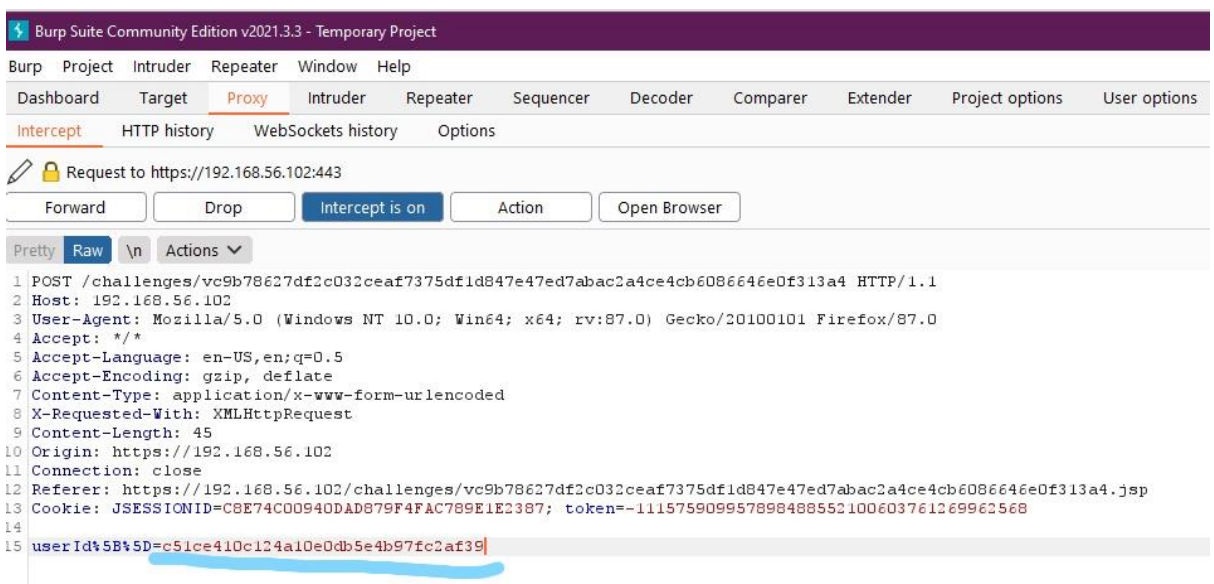
## Insecure Direct Object References Challenge Two

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

Paul Bourke  
 Will Bailey  
 Orla Cleary  
 Ronan Fitzpatrick

Show this Profile

12. Navigate to BurpSuite and replace the userID with the MD5 hash generated in step 9. Press “Forward”.



You will see that the application displays a private message from “Hidden User”.

### CVSS Score 7.5

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	None
Availability	None

### Mitigation:

It is recommended to use a salted hash instead of an encryption algorithm, to prevent an attacker from decrypting the user ID field and to prevent them from brute forcing other user ID's to gain access to other user data.

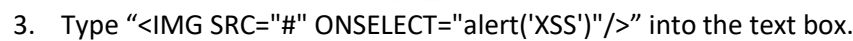
### Medium: Cross Site Scripting [CWE-725]:

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. (source: <https://owasp.org/www-community/attacks/xss/>)

### Steps to reproduce:

1. Go to Security Shepherd (<https://192.168.56.102/>) in Firefox and log in.
2. Go to Challenges -> XSS -> Cross Site Scripting 3.



You will notice that nothing happened. After doing some research, the common security measures for preventing XSS are removing the keyword “on”, e.g., onselect is filtered to select, removing “<script>” and replacing it with blank space, substituting “!” with “.”, and replacing “.” with “!”. Let us try finding out if the keyword “on” is being filtered.

- Nothing happens. It is possible that the system loops several times to remove the “on” keyword.

- 29

## Cross Site Scripting Three

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

`ONONONONONONONONONONONONONONSELECT="alert('XSS')"/>`

Get this user

Nothing happens. Perhaps the whole keyword “onselect” is being filtered.

6. Type "`<IMG SRC=# onseleconseleconseleconseleselecttttttt=alert('XSS')/>`" into the text box and click "Get this user".

Submit Result Key Here...

Submit

## Cross Site Scripting Three

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

Well Done

You successfully executed the JavaScript alert command!

The result key for this challenge is

D05F70A8C5DDFDAD4BB4A1205DFF7491963EA3A4C5C26A719D7C8486504  
324420552A70A7C205F774C2ED424C22408E6AC674B23CD040D06CE44C

## Search Results

Sorry but there were no results found that related to

The JavaScript code was executed.

## CVSS Score 6.5

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	Required
Scope	Unchanged
Confidentiality	None
Integrity	High
Availability	None

### **Mitigation:**

It is recommended to only store encrypted data within the HTML code if the data comes from an untrusted source, in this case the user. This includes HTML encoding, JavaScript encoding, JSON encoding, CSS encoding and URL encoding. It is also advised to sanitize HTML Markup by a designated library.

### **Medium: Cross Site Scripting [CWE-725]:**

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page. (source: <https://owasp.org/www-community/attacks/xss/>)

### **Steps to reproduce:**

1. Go to Security Shepherd (<https://192.168.56.102/>) in Firefox and log in.
2. Go to Challenges -> XSS->Cross Site Scripting 5.

#### **Cross Site Scripting Five**

---

Demonstrate a XSS vulnerability in the following form by executing a JavaScript alert command. The developers of this sub application wanted to demonstrate how HTTP links can be embedded in HTML. Have a look by putting in your own HTTP link. The Developers are white listing input so only HTTP URLs are allowed!

Please enter the **URL** that you wish to post to your public profile;

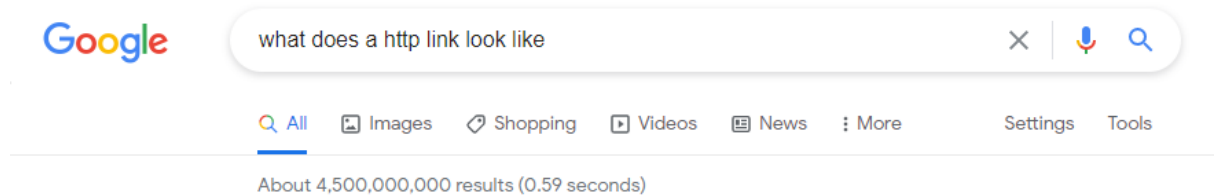
3. Type "onselect=alert('XSS')" into text box and click "Make Post". We get the following result:

# Your New Post!

You just posted the following link;

[Your HTTP Link!](#)

When we click on the link it brings us to a google search for “What does a HTTP link look like”.



The website replaced are URL which means that are input did not pass.

4. Type “http://a” and click “Make Post”. The link has passed, so “http://” is a valid URL format.
5. Type “http://a”.onselect=alert('XSS').b.c” and click “Make Post”. You will see that the script was successfully embedded.

Please enter the **URL** that you wish to post to your public profile;

http://a"" onselect=alert('XSS') .b.c

Make Post

## Well Done

You successfully executed the JavaScript alert command!

The result key for this challenge is

Lv49nhCJ8YvW/epyQUI1T/0z6vPZQQ1NqkDCjpPCp+a/MS6m7LSHR58xa5bfhe9aMS1hlnaQGmO7h3uAEV/ROnui00aSE7uNc+K7/LJH7Nc5VJSi1

### CVSS Score 6.5

Attack Vector	Network
Attack Complexity	Low
Privileges Required	None
User Interaction	Required
Scope	Unchanged
Confidentiality	None
Integrity	High



## Mitigation:

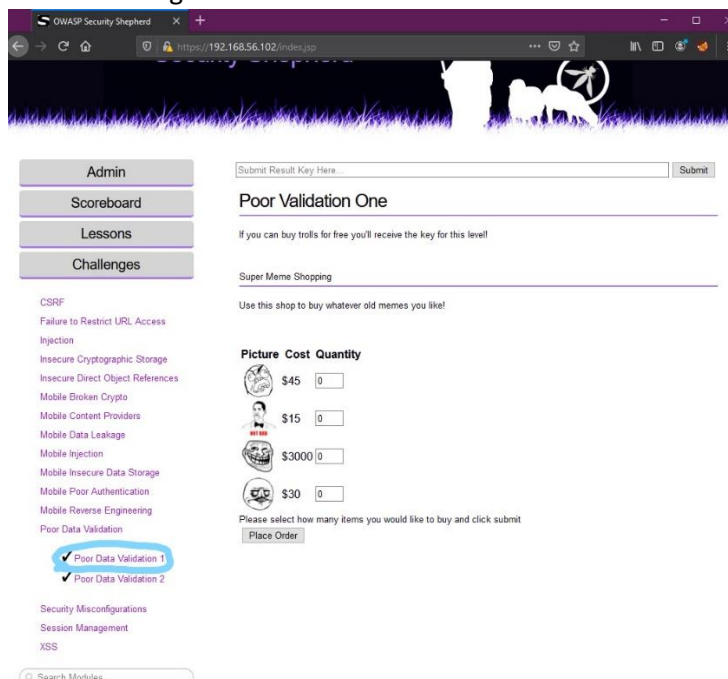
It is recommended to only store encrypted data within the HTML code if the data comes from an untrusted source, in this case the user. This includes HTML encoding, JavaScript encoding, JSON encoding, CSS encoding and URL encoding. It is also advised to sanitize HTML Markup by a designated library.

## Medium: Poor Data Validation [CWE-20]

Poor Data Validation occurs when an application does not validate submitted data correctly or sufficiently. Attackers can take advantage of poor data validation to perform business logic attacks or cause server errors. The data validation process should ideally be performed on the client side and again on the server side (Source: OWASP Security Shepherd). In this case, the web application does not check if the input is not a negative number.

### Steps to reproduce:

1. Go to Security Shepherd (<https://192.168.56.102/>) in Firefox and log in.
2. Go to Challenges -> Poor Data Validation 1.



3. First, put “-5” as the quantity for the trolls and click “Place Order” to see if the application checks for negative numbers:

### 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	<input type="text" value="0"/>
	\$15	<input type="text" value="0"/>
	\$3000	<input type="text" value="-5"/>
	\$30	<input type="text" value="0"/>

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

Place Order

Order Complete

Your order has been made and has been sent to our magic shipping department that knows where you want this to be delivered via brain wave sniffing techniques.

Your order comes to a total of \$0

You will see that the order total is \$-15,000.

### 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	<input type="text" value="0"/>
	\$15	<input type="text" value="0"/>
	\$3000	<input type="text" value="-5"/>
	\$30	<input type="text" value="0"/>

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

Place Order

Order Complete

Your order has been made and has been sent to our magic shipping department that knows where you want this to be delivered via brain wave sniffing techniques.

Your order comes to a total of \$-15000

- Now that we know that the application does not check for negative numbers, we can try to put in amounts that will equal to \$0. Put "2" as the quantity of trolls and "-20" as the quantity of me gusta:

### Super Meme Shopping


Use this shop to buy whatever old memes you like!

#### Picture Cost Quantity

 \$45

 \$15

 \$3000

 \$30

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


[Place Order](#)

5. Click "Place Order".
6. You will see that the order went through and that the total payment was \$0.

### Super Meme Shopping


Use this shop to buy whatever old memes you like!

#### Picture Cost Quantity

 \$45

 \$15

 \$3000

 \$30

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

[Place Order](#)

#### Order Complete

Your order has been made and has been sent to our magic shipping department that knows where you want this to be delivered via brain wave sniffing techniques.

Your order comes to a total of \$0

Trolls were free, Well Done -

### CVSS Score 5.9

Attack Vector	Physical
Attack Complexity	Low
Privileges Required	None
User Interaction	Required
Scope	Unchanged
Confidentiality	None
Integrity	High
Availability	High

**Mitigation:**

It is recommended to have the user data validated at both user and server side to ensure that the input amount is greater than or equal to 0. There should also be an upper limit enforced to prevent buffer overloading. The best solution to preventing poor data validation is to have the user select from a range of whitelisted values, e.g., have a “+” button beside the “quantity” box that increases the amount to order.

## Recommendations and Conclusions

It was found that the application is susceptible to attacks exploiting poor session management, failure to restrict URL access, SQL Injection, insecure direct object reference, cross site scripting and poor data validation.

The main recommendations are as follows:

- Use salted hashes to encrypt session ID's and authorisation checks.
- Do not transfer any data in plain text.
- Show less detailed error messages. Do not reference data directly within error messages.
- Implement a more secure password change protocol.
- Implement an ACL that prevents unauthorised users from accessing pages they should not have access to.
- Use virtual directories within HTML code rather than physical directories within the server.
- Use parameterized queries for functions that access the server's database.
- Always encrypt data from untrusted sources.
- Always validate data on both client and server side.
- Use whitelisted values where possible.