

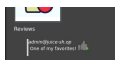
# OWASP Juice Shop

## Injection

Log in with the administrator's user account using SQL Injection

=====

found admin email



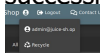
try to inject single quote (interesting)



try to perform SQL injection



successfully login as admin@juice-sh.op user



## Broken Authentication

reset Jim's password using the forgotten password mechanism - what was the answer to the secret question?

=====

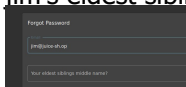
found jim email



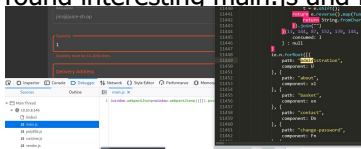
access forgot password link



jim's eldest siblings middle name?

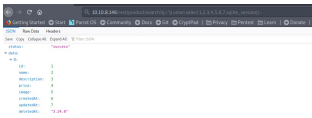


found interesting main.is and found /administration dir



able to access administration page

248



Getting information about the table and columns  
//the /login page (still rmb the object Object error?)



burpsuite result

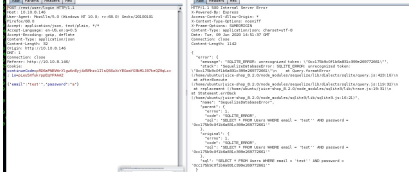
/\*

now we got the:-

tablename: Users

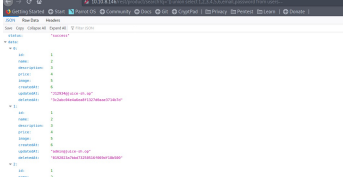
columns: email, password

\*/



Perform SQL Injection

//http://10.10.8.146/rest/product/search?q=%27))%20union%20select%201,2,3,4,5,6,email,password%20from%20users--



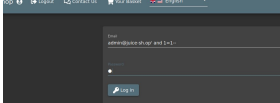
cracked admin hash credential



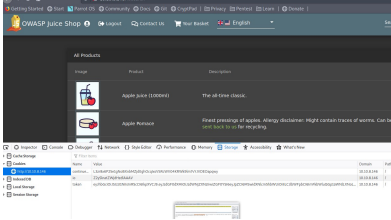
method 2: JSON Token in Cookies

=====

login as admin email with sql injection method

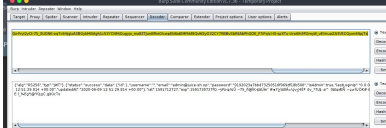


check the cookies -> token



decode it with base64

//found password in it!!



cracked admin hash credential

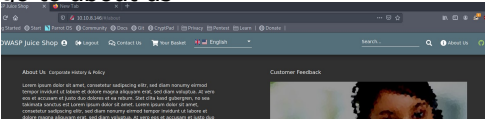


Sensitive Data Exposure

Access a confidential document and enter the name of the first file with the extension ".md"

=====

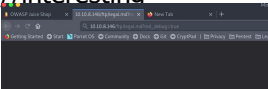
go to about us



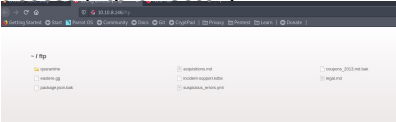
interesting green highlighted link



the link redirect to this path  
//legal.md ?  
// ftp directory?  
//interesting



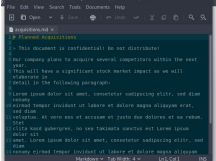
access /ftp directory



interesting file



confidential file voila!



Broken Access Control

Access the administration section of the store - What is the name of the page?

=====

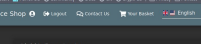
The screenshot shows a C# code editor with a 'Main' method. The code is as follows:

```






1  try
2  {
3      Person p = new Person();
4      p.SayHello();
5  }
6  }

```

A red box highlights the 'Person' class name in line 3, and a red arrow points to the 'SayHello()' method call in line 4. The 'Person' class is defined in a separate file, 'Person.cs', which is also visible in the background. The 'Person' class has a 'SayHello' method that prints 'Hello, my name is [Name]'.



Administrators

	Email	Role
	admin@juice.sh.jp	Admin
	jpr@juice.sh.jp	Admin
	lecher@juice.sh.jp	Admin
	team@juice.sh.jp	Admin
	cmi@juice.sh.jp	Admin

=====

The screenshot shows the 'Your Basket' page on the Amazon website. The page has a dark header with navigation links: Home, Community, Amazon, and a search bar. Below the header, there's a sub-header with 'Your Basket' and a language selector set to 'English'. The main content area is titled 'Your Basket (empty of 0)' and contains a table with 4 items. The table has columns for Product, Price, Quantity, and Total Price. The items are: Apple juice (1.00), Orange juice (2.00), Fajita juice (0.00), and Eggplant juice (0.00). At the bottom of the table, there's a 'Total' row showing a total price of 3.00. Below the table, there's a 'Proceed to checkout' button and a 'View cart' button.

Product	Price	Quantity	Total Price
Apple juice (1.00)	1.00	1	1.00
Orange juice (2.00)	2.00	1	2.00
Fajita juice (0.00)	0.00	1	0.00
Eggplant juice (0.00)	0.00	1	0.00
<b>Total</b>			<b>3.00</b>

[illegible]

The screenshot shows the OGAAP Juice Shop website. The navigation bar includes links for Home, Shop, Cart, Community, Data, and Profile. The main header features the OGAAP logo and the text 'OGAAP Juice Shop'. Below the header, there's a section titled 'Your basket' with a table showing one item: 'Fruit' with a price of 4.99. The total price is 4.99. There are buttons for 'Checkout' and 'View Cart'. The website has a dark theme and a navigation bar with links like Home, Shop, and Cart.

=====

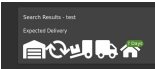
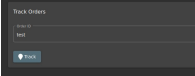
[illegible][illegible]

548

# Carry out reflected XSS using Tracking Orders

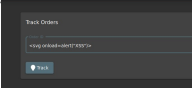
=====

test on entering text what would happen

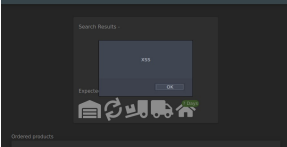


test XSS Injection

//<svg onload=alert("XSS")>



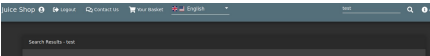
successfully perform XSS Injection



# Carry out XSS using the Search field?

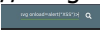
=====

test entering any text in search field



test XSS Injection

//<svg onload=alert("XSS")>



successfully perform XSS Injection

