Vulnerability Assessment of Web Applications and Recommendations for Actions:

Penetration Testing Report

Introduction

Currently, there is no single system or application that is not vulnerable to attack. In the case of this study, the vulnerable web application is designed with a weak security policies that made it susceptible to most vulnerabilities for penetration testing. This report will cover vulnerable web application penetration test and report. The test report will include two vulnerabilities found, each with an explanation, impact score, the probability of exploitation, and security mechanism to protect the system against the attack. The investigation will cover the working mechanism of SQL injection and a novel way to protect against the attack.

Definitions of Impact and probability of exploitation

Impact of exploitation

|  |  |
| --- | --- |
| Consequences of exploitation | Explanation |
| Low | Even if the attack is successful, the attacker would not have control over what’s being exploited, modification of the contents is not also possible. Hence, the attack impact scope may be limited to only allowing an attacker to access a specific level of only viewing the system content. |
| Medium | The attack needs some pre-conditions for the attacker to exploit the whole system because the system is configured with some security measures. Therefore, a successful attack may allow little modifications on the specific system contents which may not affect the entire system |
| High | There is a total compromise of the whole system, and therefore, loss in system protections, which leads to full system files and configuration disclosure. Moreover, an attacker can modify or control the entire system |

The Probability of Exploitation

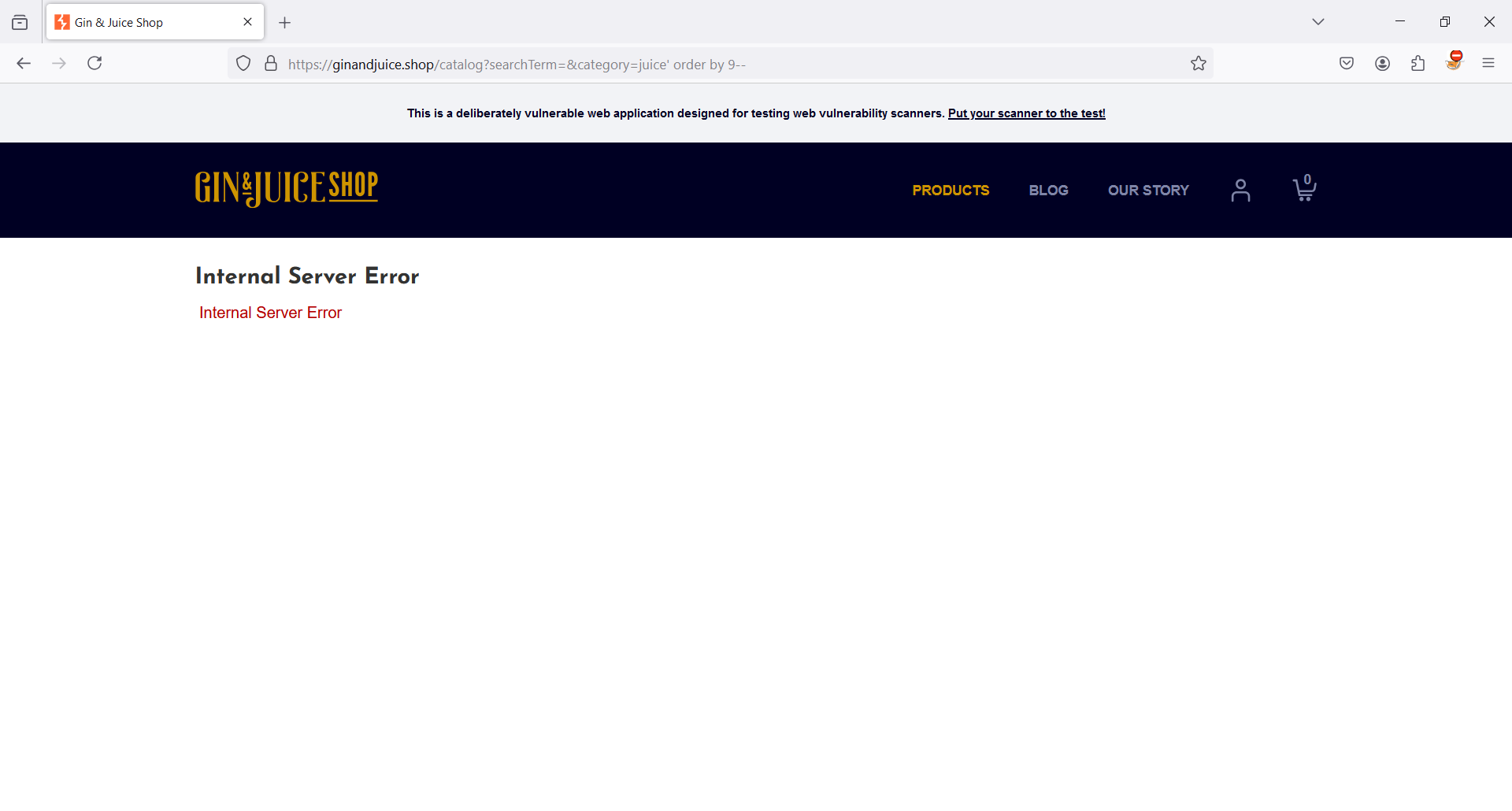
|  |  |
| --- | --- |
| Chances of Exploitation | Explanation |
| Low | There is a need for sophisticated tools to bypass systems restrictions to expose the vulnerability before starting to exploit the system. So, this vulnerability is very unlikely to occur |
| Medium | The chance for this kind of vulnerability to occur requires some preconditions. So, this vulnerability occurs moderately. |
| High | For this vulnerability to be exposed, require only some open available tools with some little knowledge. So, this weakness is very common to occur. |

**Vulnerability 1**

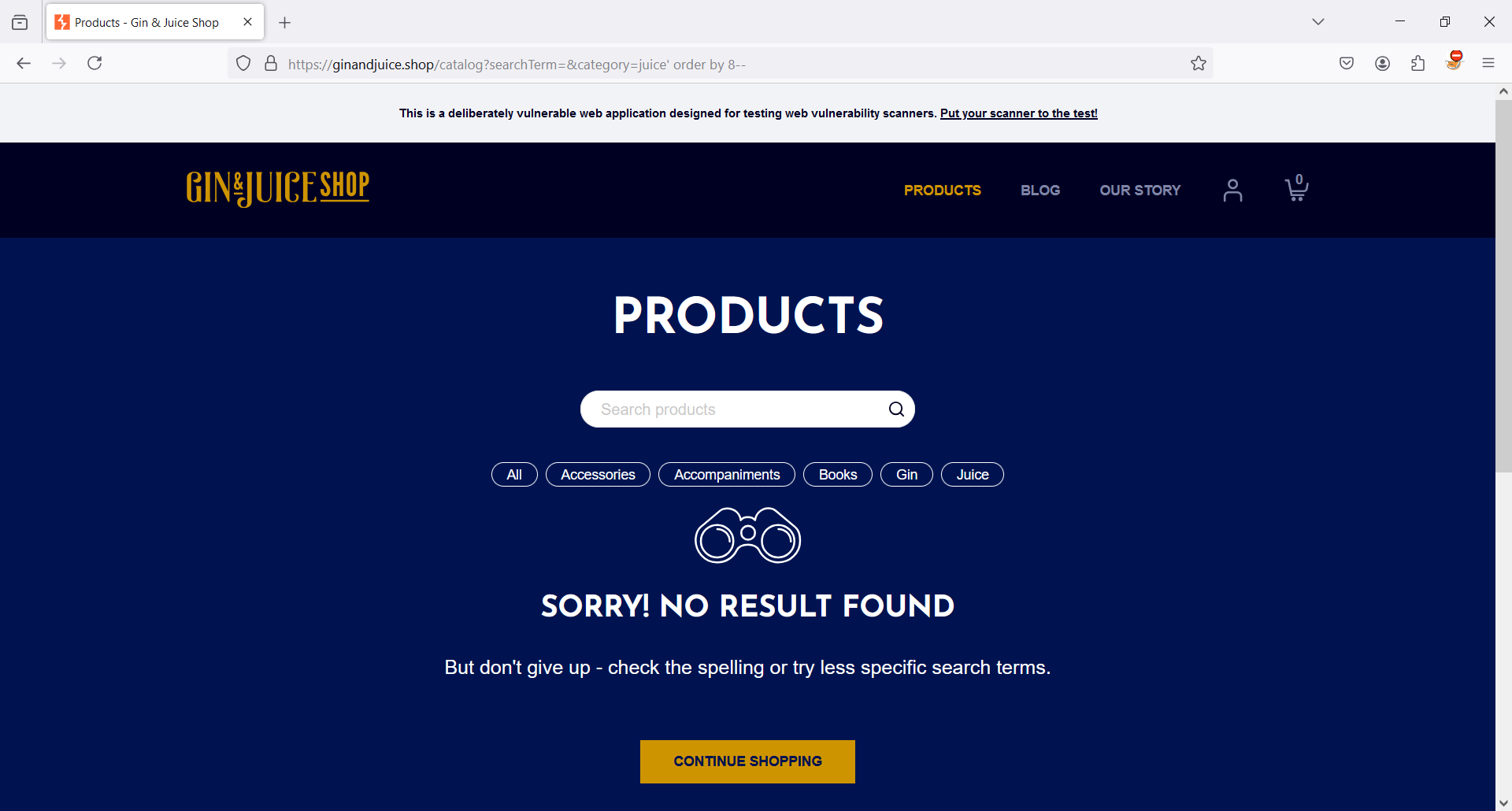
|  |  |
| --- | --- |
| SQL Injection | |
| Impact | Probability |
| Medium | Medium |

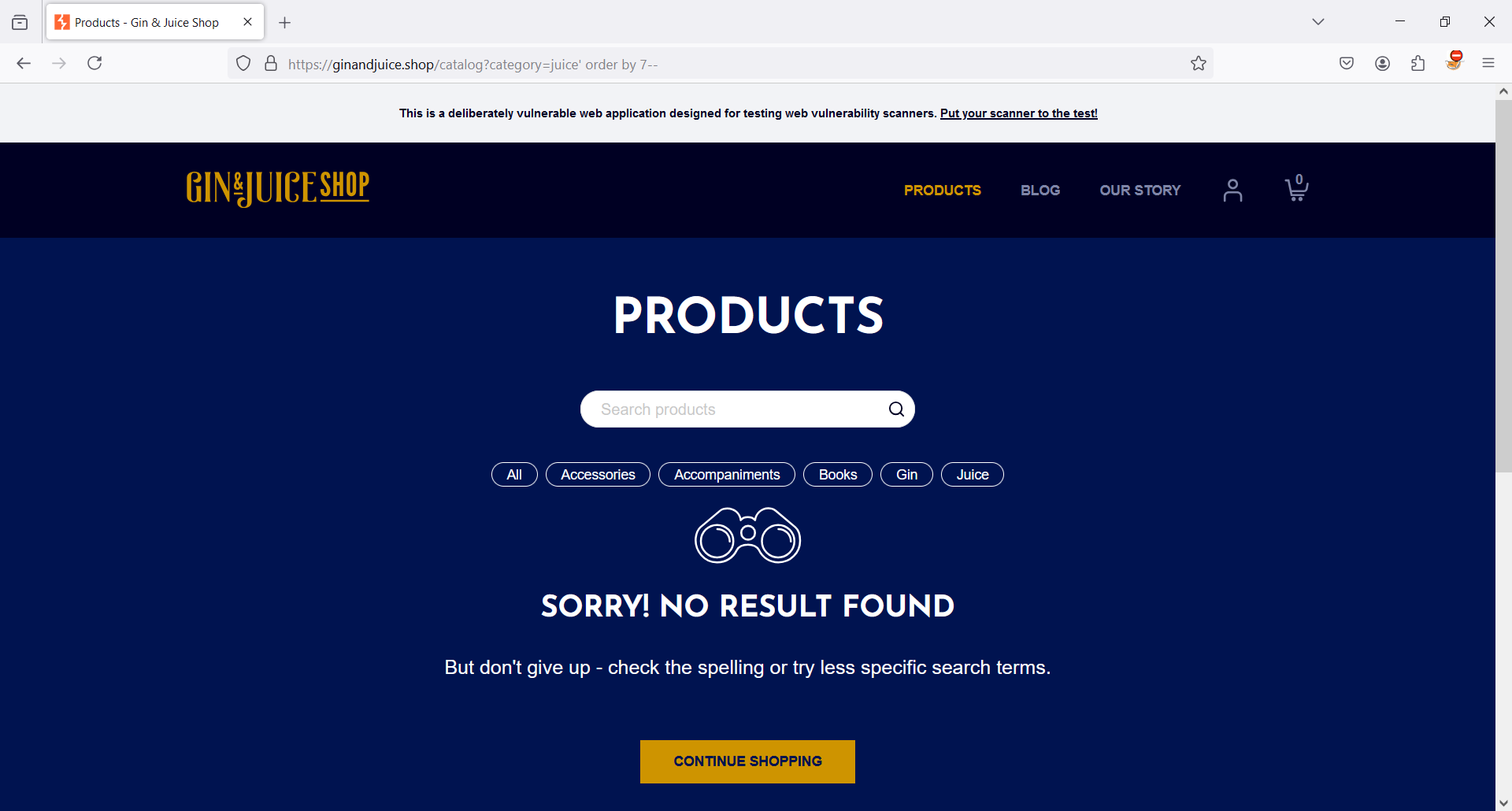
Finding

In the category page i found when i ordered more than 8 columns that give me Internal Server Error



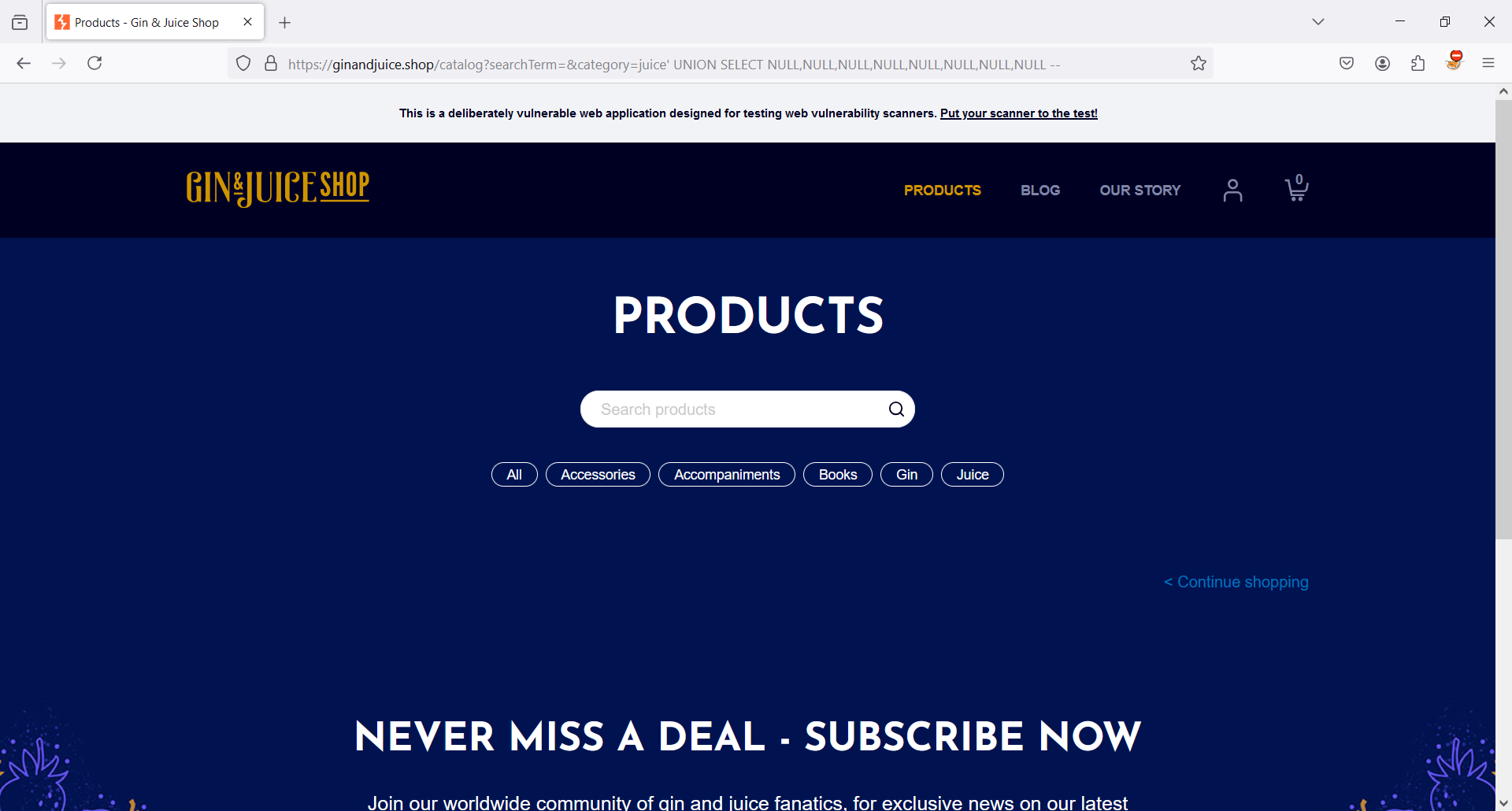
and when i ordered by 8 or less give me result



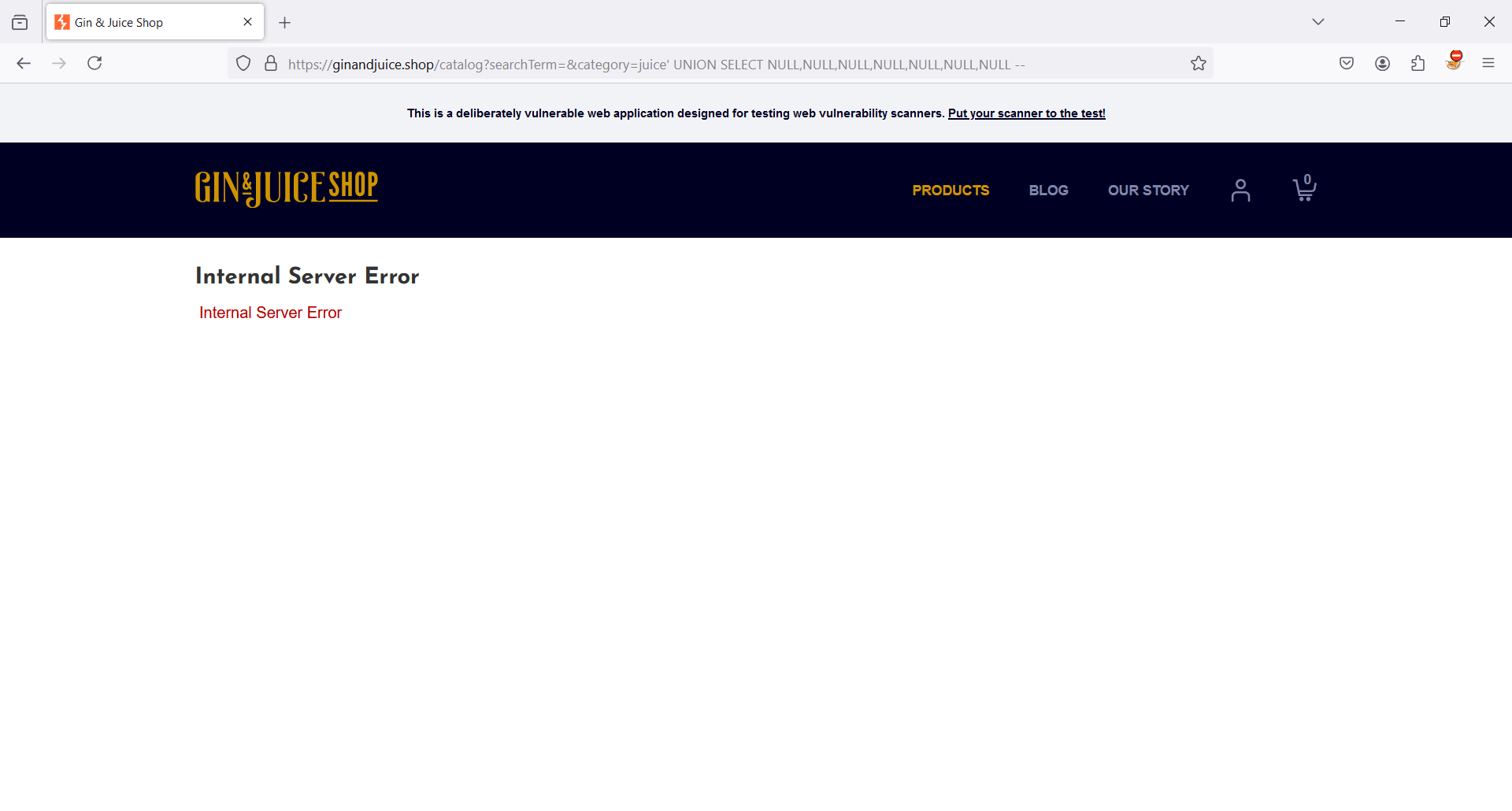


But we not sure about numbers of columns so I use UNION SELECT to prove that

When i use 8 columns this give me result

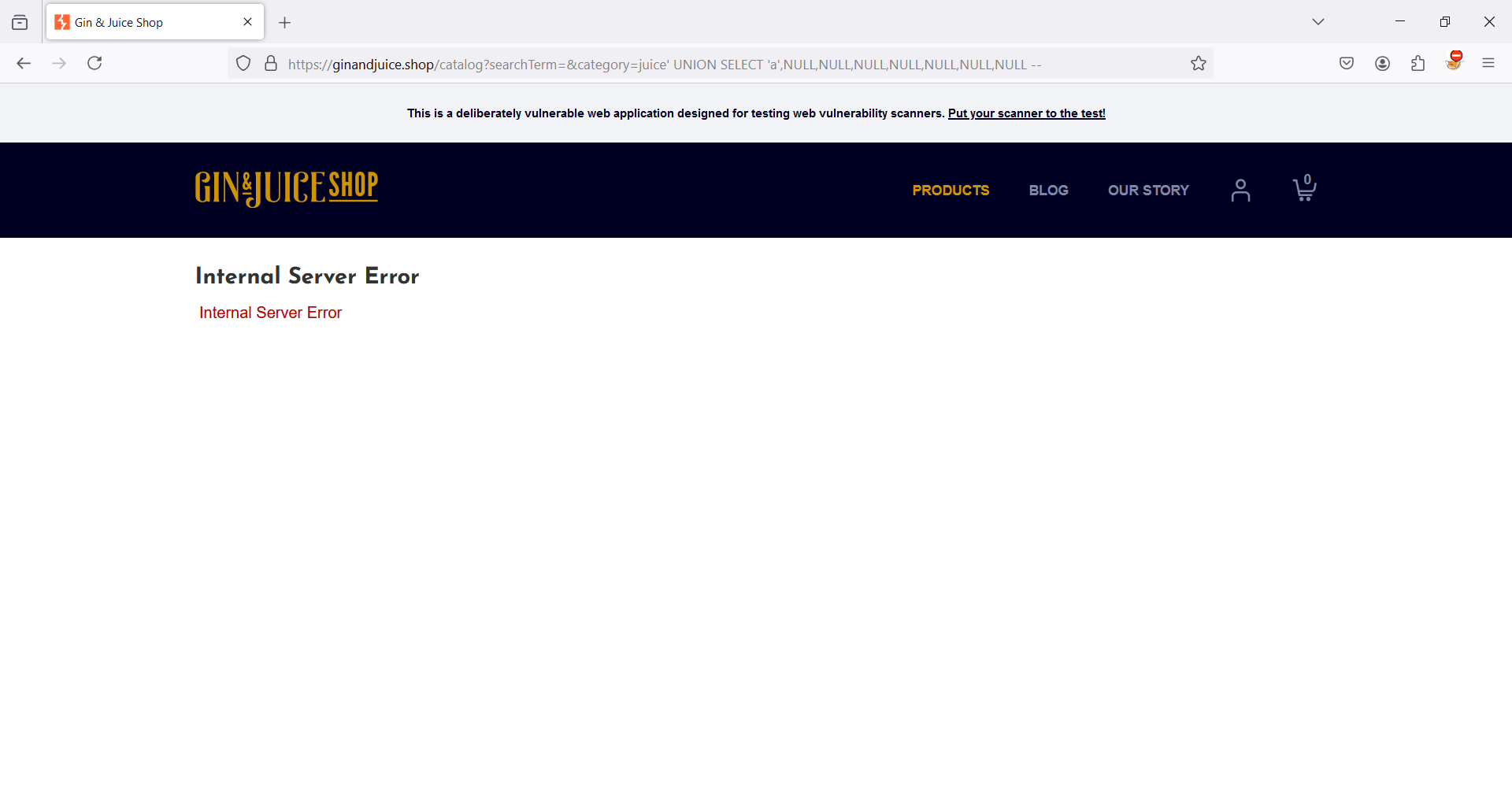


But when is use less than 8 columns that give me Internal Server Error so Number of Columns is 8

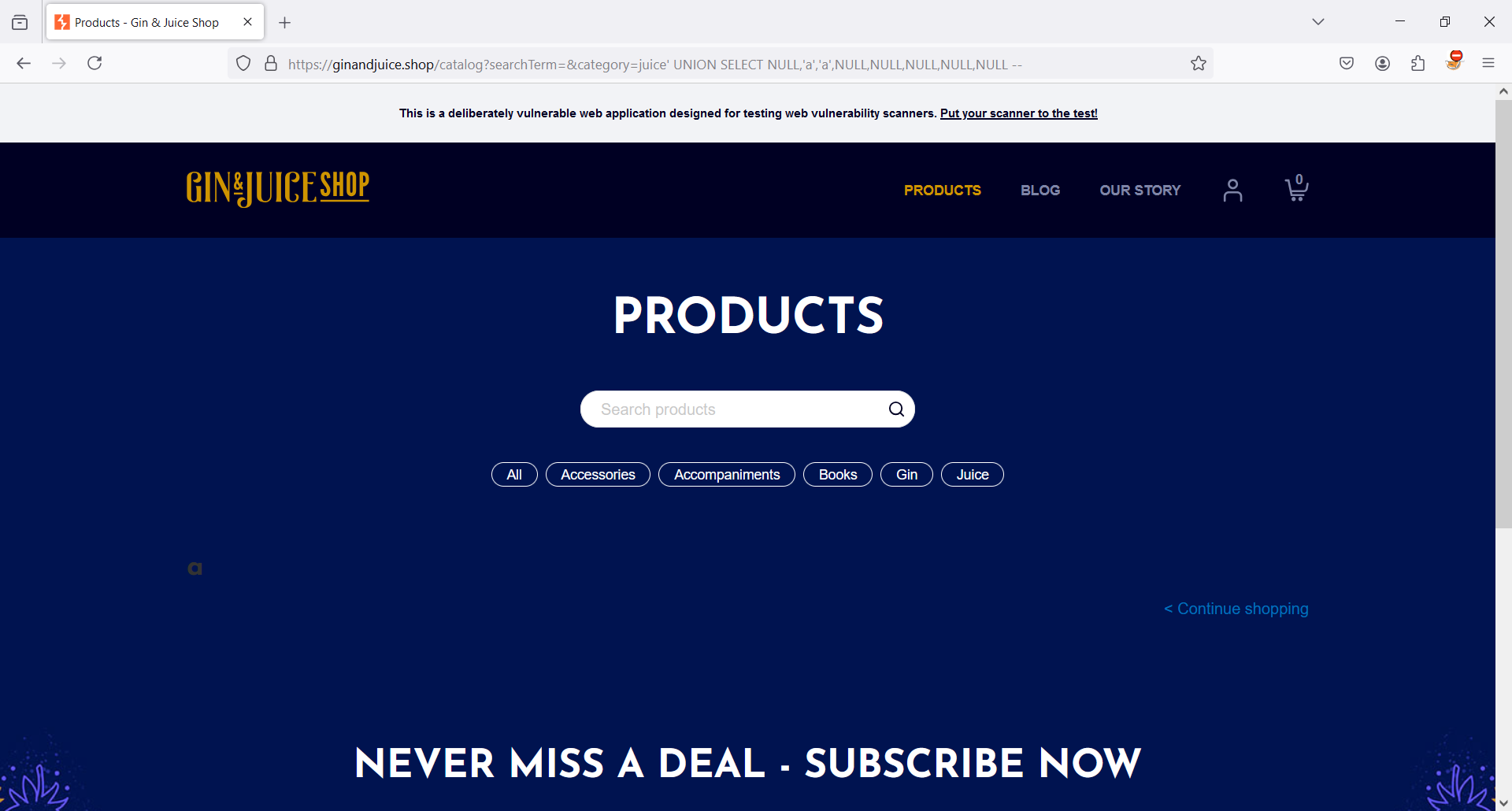


Next step i determine columns datatypes by put ‘a’ in each column and show result.

When i put ‘a’ in first column and 4,5,6,7,8 that give me Internal Server Error

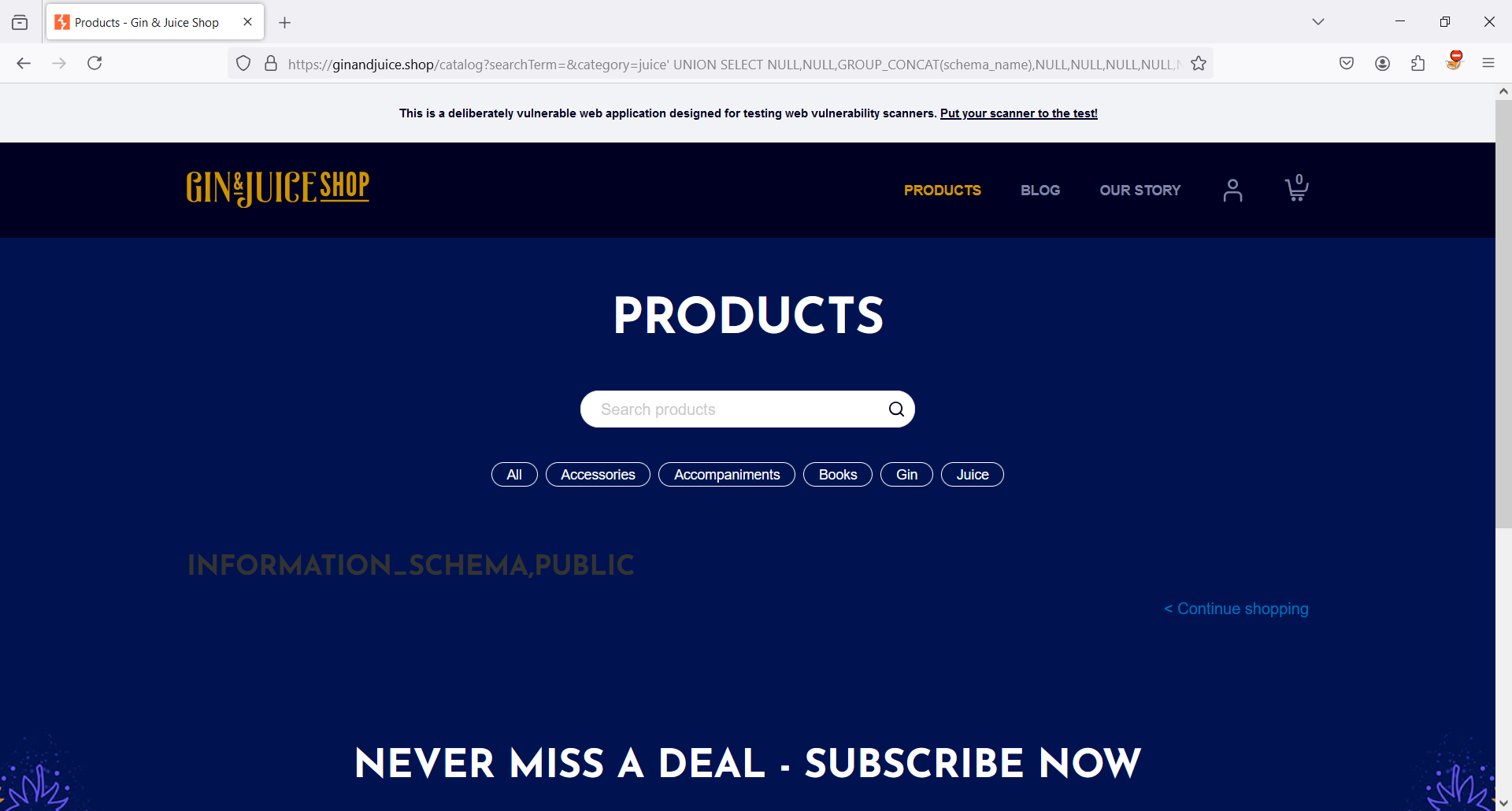


When is put ‘a’ in 2 and 3 column that give result



Then i will use column number 3 to Use UNION Atack Using :

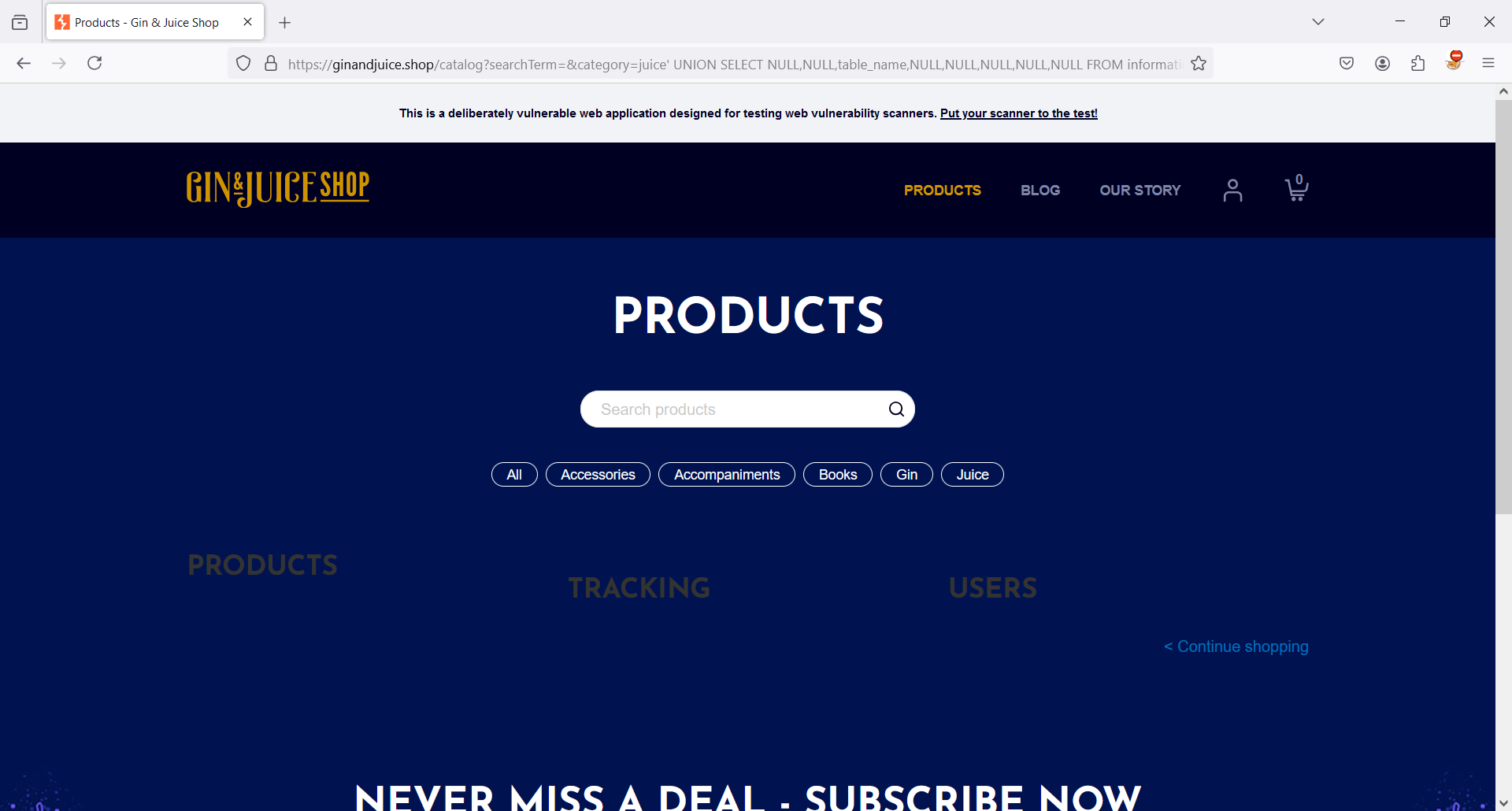
SELECT NULL,NULL,GROUP\_CONCAT(schema\_name),NULL,NULL,NULL,NULL,NULL FROM information\_schema.schemata --



As it show we reach to schema name is PUBLIC

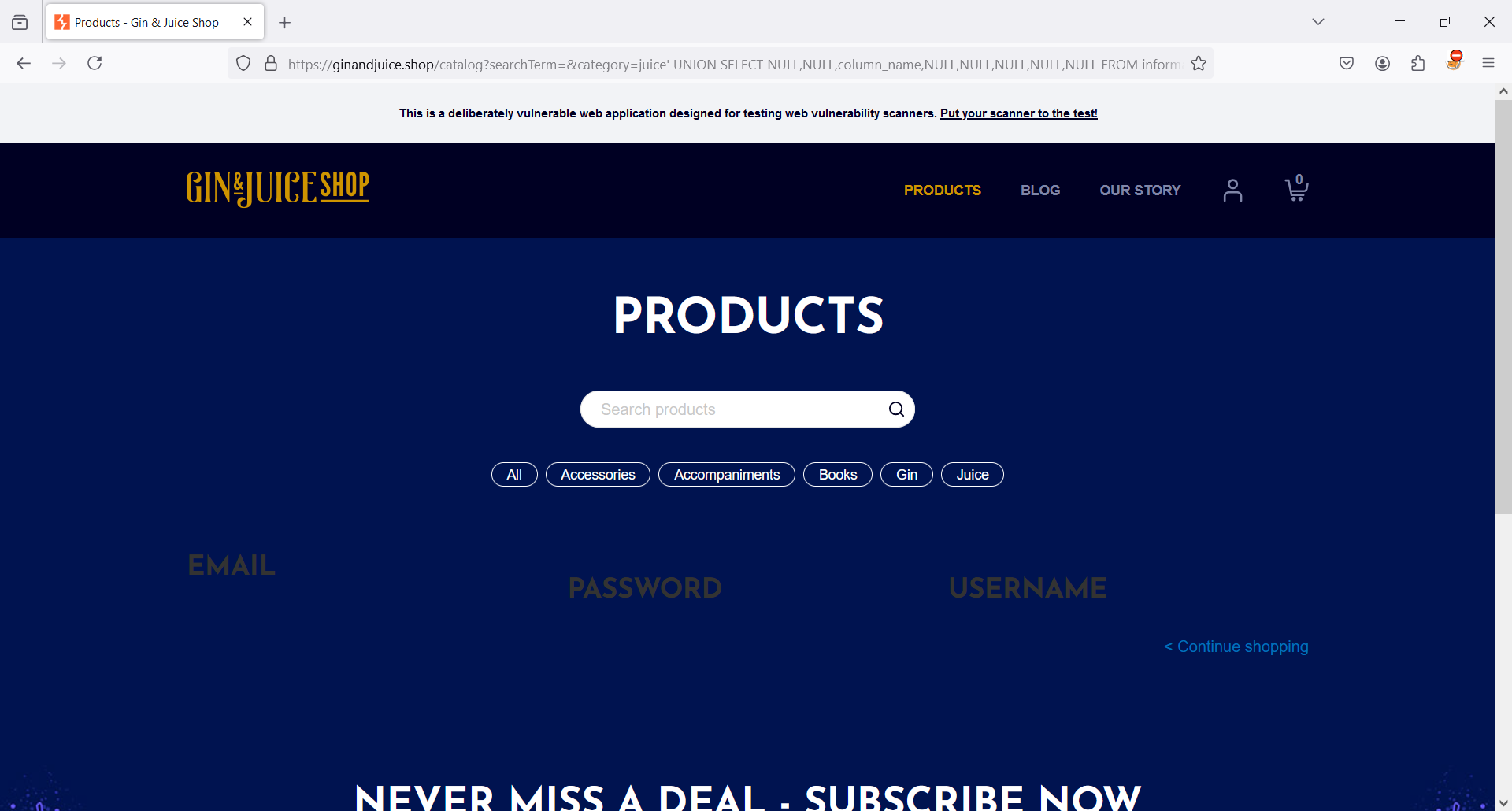
Then we catch three table (PRODUCTS – TRACKING - USERS) using :

UNION SELECT NULL,NULL,table\_name,NULL,NULL,NULL,NULL,NULL FROM information\_schema.tables WHERE table\_schema ='PUBLIC' –



Then we catch three columns (EMAIL – PASSWORD - USERNAME) using :

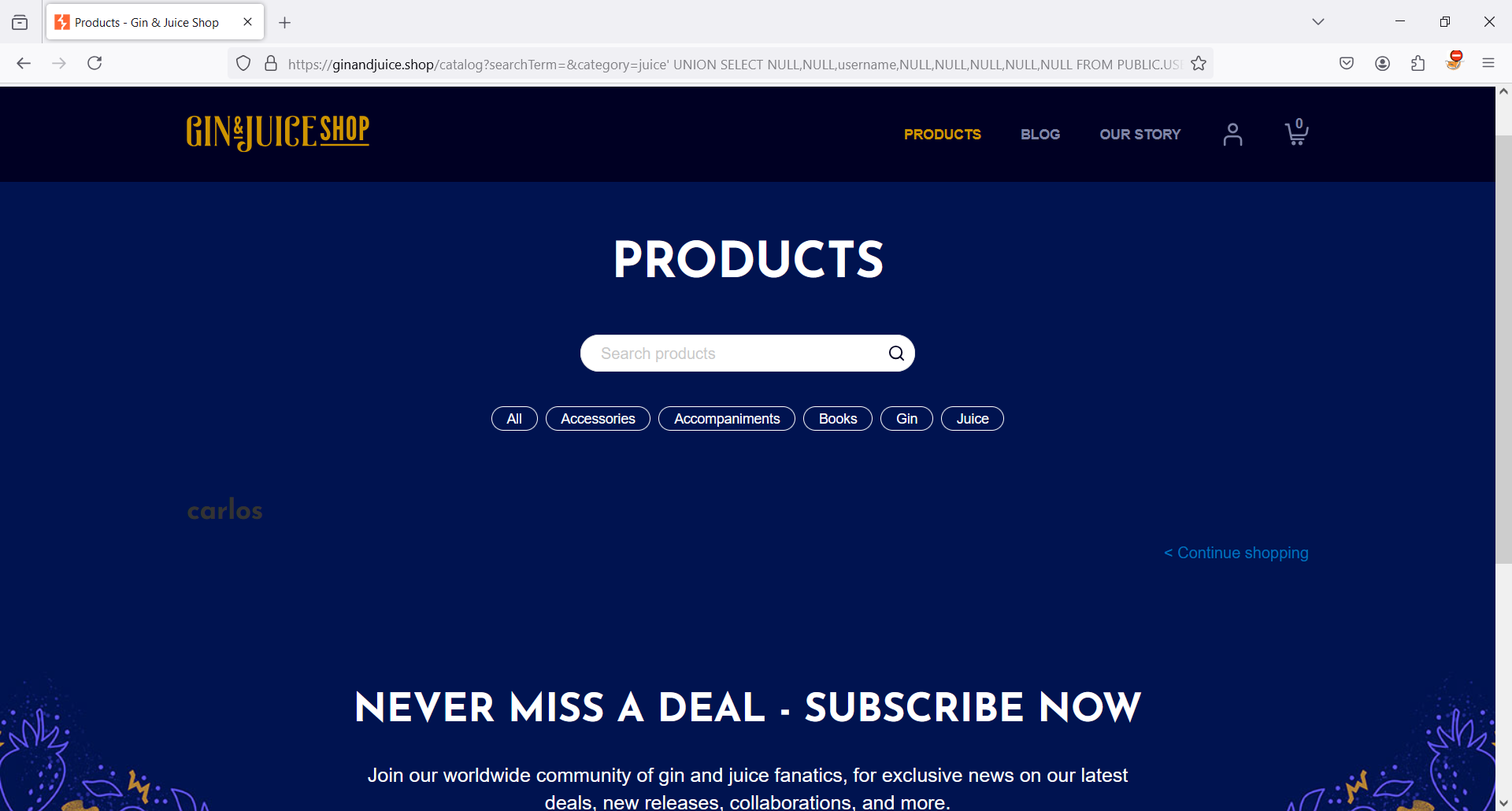
UNION SELECT NULL,NULL,column\_name,NULL,NULL,NULL,NULL,NULL FROM information\_schema.columns WHERE table\_schema ='PUBLIC' AND table\_name = 'USERS' --



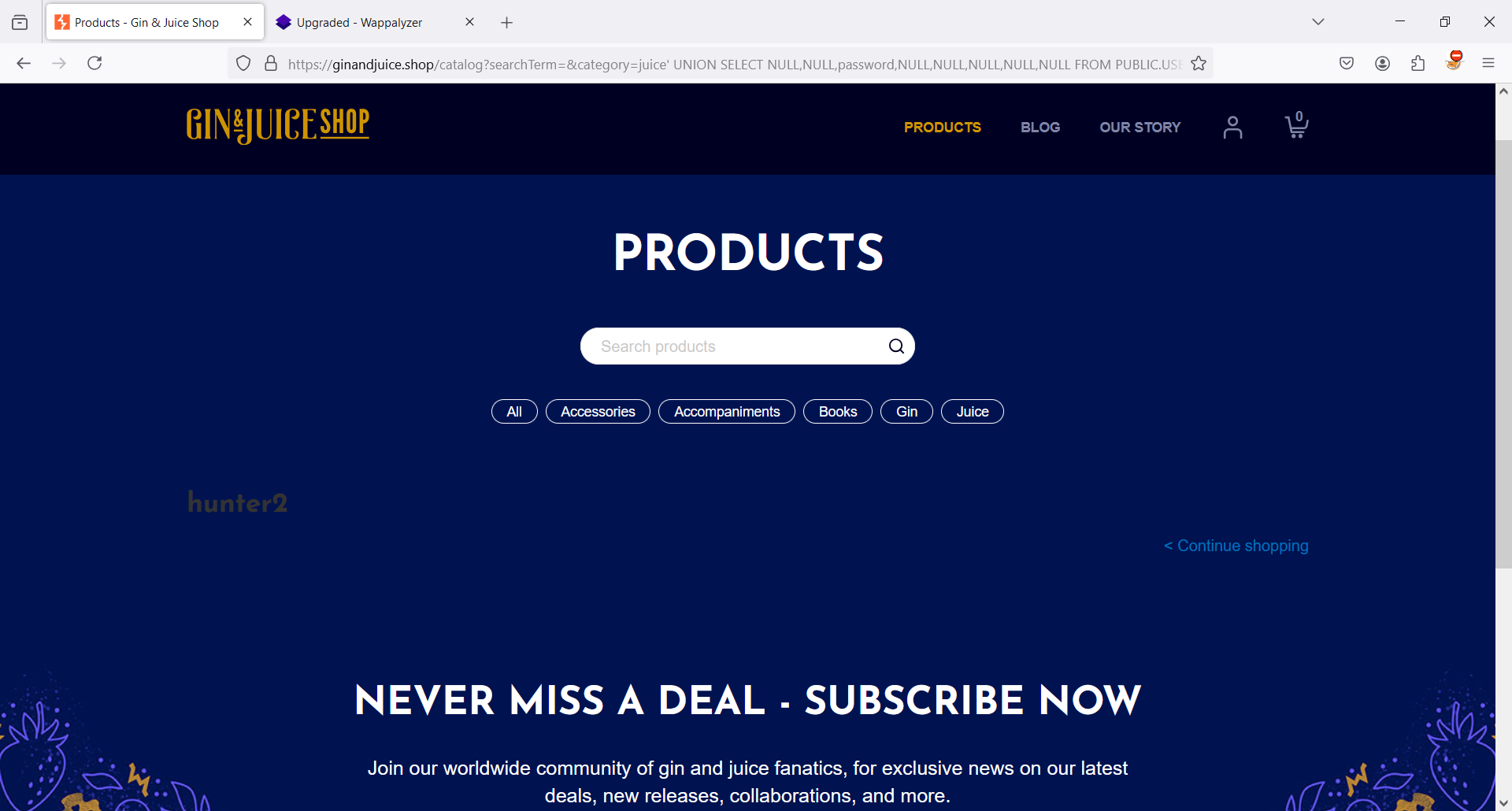
Then we exploit username column and catch it using:

UNION SELECT NULL,NULL,username,NULL,NULL,NULL,NULL,NULL FROM PUBLIC.USERS--

As it showen his name is Carlos



also we catch his password: hunter2

****

**Vulnerability 2**

|  |
| --- |
| Cross-Site Scripting: reflected |

|  |  |
| --- | --- |
| Impact | Probability |
| Medium | High |

***Finding***

In the path : ginandjuice.shop/catalog

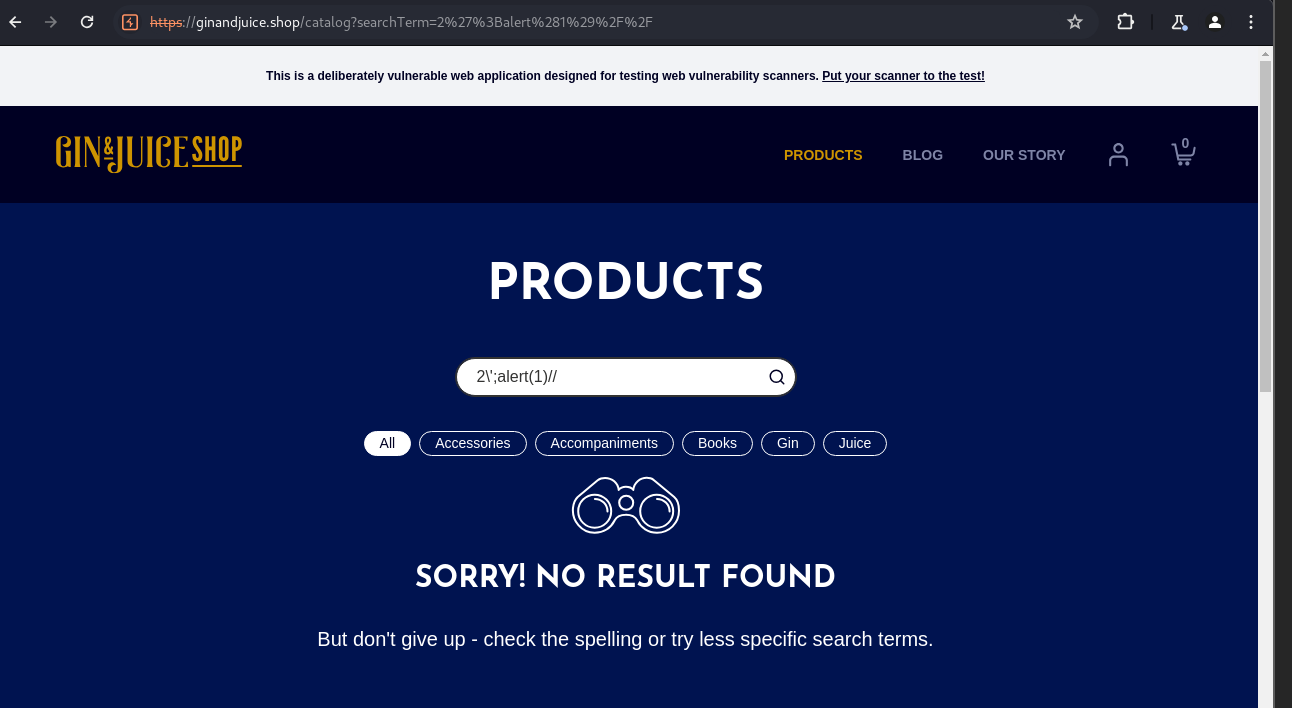
when submitting “ 2';alert(1)// “ In the search box, the application's response was returned as “ 2\';alert(1)//' ”

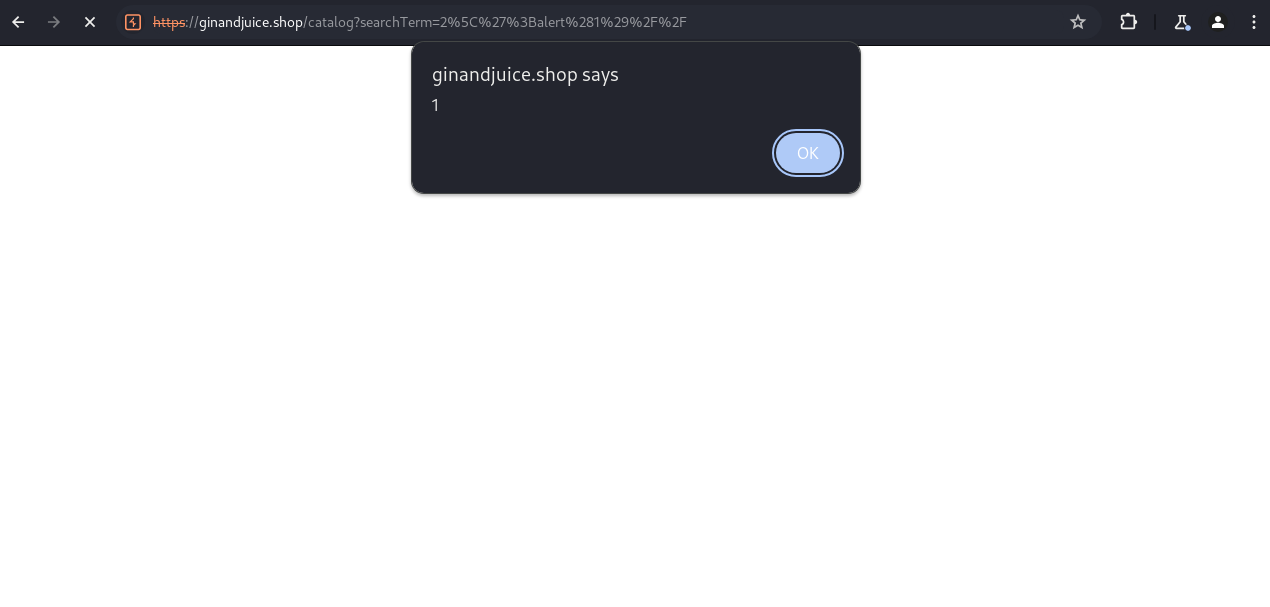
A screenshot of a computer

Description automatically generated

so the application add a backslash character before any quotation mark character in the input to prevent the effect of quotation mark .

but when submitting “ 2\';alert(1)// ” in the search box , the application fails to escape any backslash characters in the input , and the quotation mark remains unescaped and succeeds in terminating the string.





***Recommendation***

- Input should be properly validated. For example, restrict inputs to expected formats and lengths.

- Sanitize inputs by removing or encoding characters that could be harmful, such as quotes, backslashes, and angle brackets

jr penetration tester path

A certificate of completion

Description automatically generated