**NoSQL Injection**

**What is NoSQL Injection?**

**NoSQL Injection is a type of attack on NoSQL databases like MongoDB.**

**It happens when the application uses user input directly in database queries without checking it. This lets attackers change the query and see or change data they shouldn’t.**

**Why is NoSQL Injection Dangerous?**

1. **Attackers can see private or hidden data**
2. **They can log in without a password**
3. **They can change or delete data**
4. **It causes major problems for the entire system.**

**How to Protect Against NoSQL Injection?**

1. Always check and clean user input.
2. Don’t use user input directly in the database.
3. Give only the needed access to database users.
4. Show simple error messages only.

**Overview of Labs (4 labs):**

1. Detecting NoSQL Injection.
2. Bypassing Login with NoSQL Injection.
3. Extracting Data with NoSQL Injection.
4. Extracting Hidden Info with NoSQL Injection.

**LAB 1: Detecting NoSQL injection**

**Goal of the Lab:**

**Exploit a NoSQL injection vulnerability to make the website display unreleased products that are normally hidden.**

**Step-by-step Solution:**

1. **Open the lab and go to the products page.**
2. **Look at the URL when you select a category (?category=Gifts).**
3. **Change the URL to:**
4. **?category[$ne] =1**
5. **Press Enter.**

**new products (unreleased) appear — lab solved!**

A screenshot of a computer

AI-generated content may be incorrect.

A white background with black and white clouds

AI-generated content may be incorrect.

**LAB 2: Exploiting NoSQL Operator Injection to Bypass Authentication**

**Goal of the Lab:**

**We want to log in as admin.  
We don’t know the real password.**

**So, we will trick the website using a small code.**

**Step-by-step Solution:**

**Open the login page.  
Try normal login with:**

**Username: wiener**

**Password: peter**

**This is just to test the login.**

A screenshot of a computer

AI-generated content may be incorrect.

**Now we try the trick (the hack).**

**In the username box.**

**A screenshot of a computer

AI-generated content may be incorrect.In the password box.**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

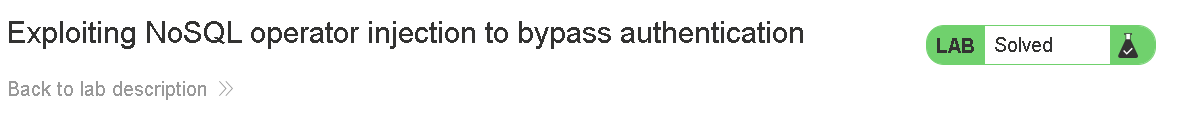
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.



**LAB 3: Exploiting NoSQL injection to extract data**

**Goal of the Lab:**

**The goal is to use a NoSQL Injection vulnerability to extract the admin's password from the database.  
Once we get the password, we will log in as admin and solve the lab.**

**We're not guessing the password randomly.  
Instead, we’re using a special trick called regex injection to ask the server:  
"Does the password start with this letter?"**

**This lets us find the password one letter at a time.**

**Step-by-step Solution:**

1. **Log in with username wiener and password peter.**
2. **Send the /user/lookup?user=wiener request to Repeater in Burp.**
3. **Test injection by submitting ' and simple payloads, URL-encoded.**
4. **Use boolean payloads like administrator' && this.password.length < N to find password length.**
5. **Use Intruder with payload administrator' && this.password[§0§]=='§a§' to brute-force the password.**
6. **Log in as administrator with the discovered password.**

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



**LAB 4: Exploiting NoSQL operator injection to extract unknown fields**

**Goal of the Lab:**

**The goal is to exploit a NoSQL Injection vulnerability using MongoDB operators to extract hidden fields (like the admin password) from the database.  
We will use NoSQL operator injection, such as $ne, $regex, and Boolean logic, to ask the server indirect questions like:**

**"Does this field exist?" or  
"Does it start with this value?"  
This helps us discover and extract unknown or hidden fields, such as Carlos's password, and then we use it to log in and solve the lab.**

**Step-by-step Solution:**

1. Log in with:

{"username": "Carlos", "password": {"$ne": ""}}

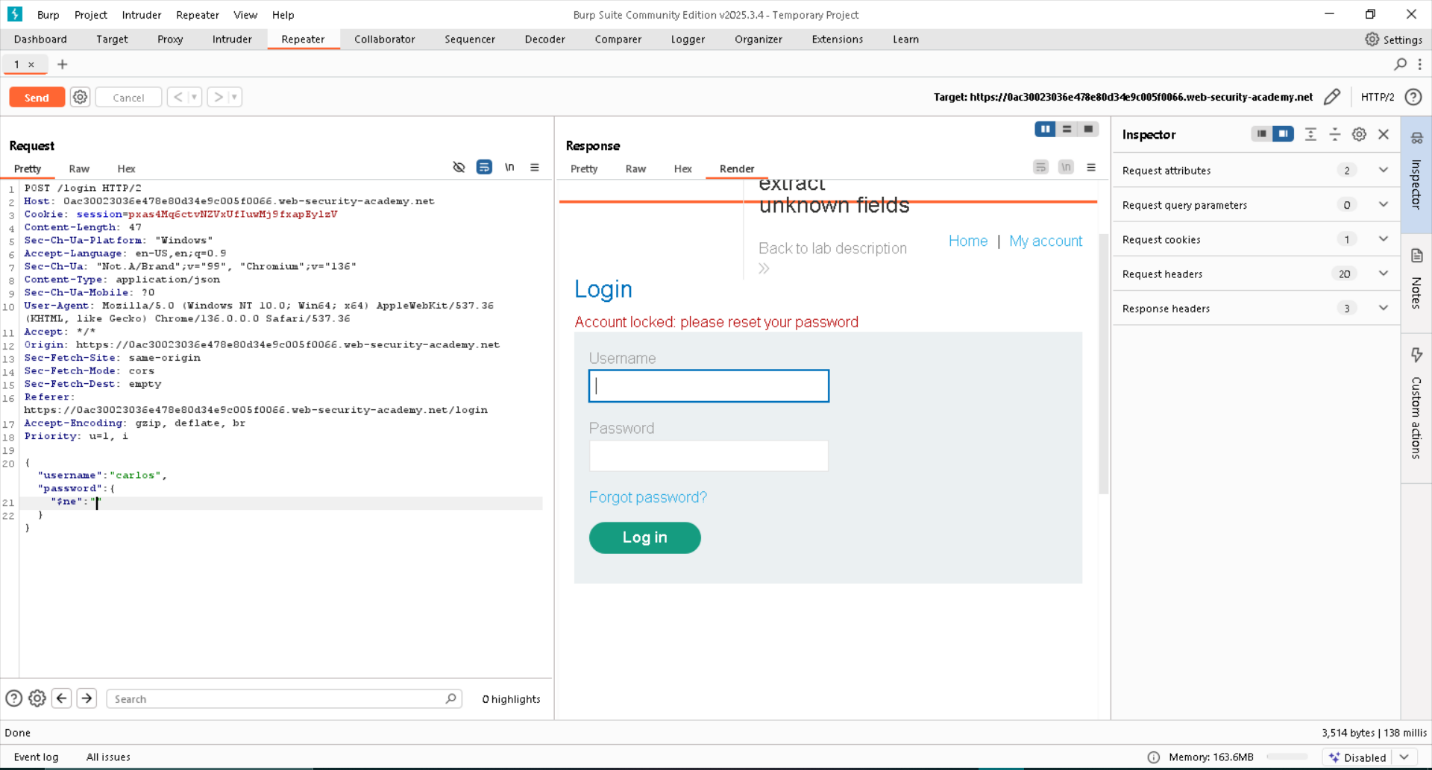
1. Use $where to check the first field:

"$where": "function () { if (Object. Keys(this)[0].match('\_id')) return 1; else return 0; }"

1. Change [0] to [4] or [5] to find hidden fields.
2. You find the hidden field name is email.
3. Check that email has length 5.
4. Use **Intruder** to brute-force the value of email, one character at a time.
5. Use the value to reset the password or log in as **Carlos**

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

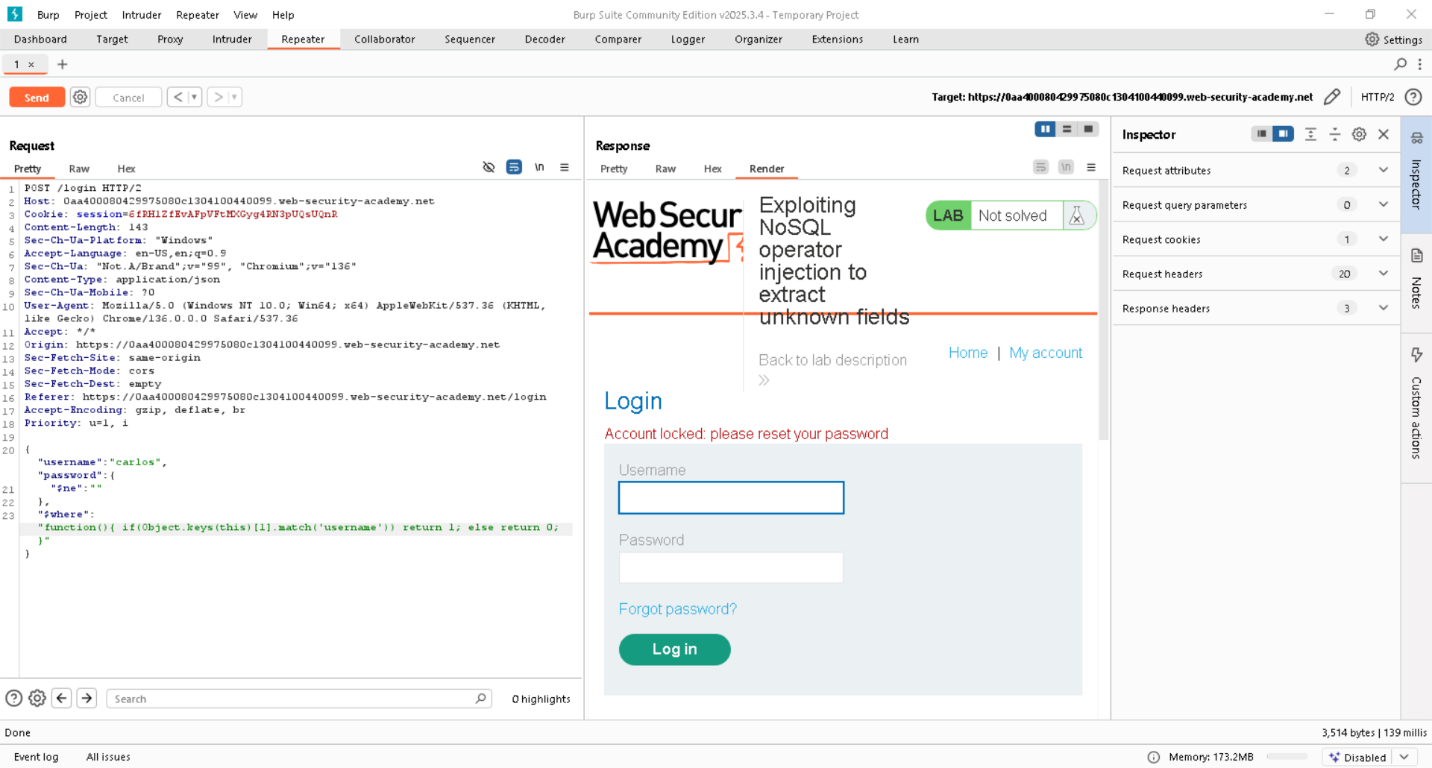
AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

