babygame

发表于 2021-01-04 分类于 Challenge， 2019， SCTF， Crypto
Challenge | 2019 | SCTF | Crypto | babygame

点击此处获得更好的阅读体验

作者：Delta、来源：安全客

解题思路

题目分析

题目首先需要proof_of_work，要求m和rsa加密m之后再解密的结果不相同，让m比n大即可绕过 进入系统之后有两个选项

*随机生成三组不同的a，b，n，使用相同的e=3，使得c=pow(a*m+b,e,n)，然后会给我们三组不同的a，b，n和c。最后再使用aes_ofb加密m，将结果也给我们。其中aes的iv和key都是随机生成的*

我们需要输入aes_ofb加密之后的m的结果，其中m需要将其中的afternoon替换为morning，如果构造的正确则返回flag 解题思路：

通过Broadcast Attack with Linear Padding解出m为I will send you the ticket tomorrow afternoon

将修改后的m，以及aes_ofb加密之后的m的结果进行异或，得到的最终结果就是修改后的m进行aes_ofb加密之后的结果。将此结果发送给服务器便得到flag

解题脚本

```
# hastads.sage
def hastads(cArray,nArray,e=3):
    """
    Performs Hastads attack on raw RSA with no padding.
    cArray = Ciphertext Array
    nArray = Modulus Array
    e = public exponent
    """
    if(len(cArray)==len(nArray)==e):
        for i in range(e):
            cArray[i] = Integer(cArray[i])
            nArray[i] = Integer(nArray[i])
        M = crt(cArray,nArray)
        return(Integer(M).nth_root(e,truncate_mode=1))
    else:
        print("CiphertextArray, ModulusArray, need to be of the same length, and the same size as the public exponent")
def linearPaddingHastads(cArray,nArray,aArray,bArray,e=3,eps=1/8):
    """
    Performs Hastads attack on raw RSA with no padding.
    This is for RSA encryptions of the form: cArray[i] = pow(aArray[i]*msg + bArray[i],e,nArray[i])
    Where they are all encryptions of the same message.
    cArray = Ciphertext Array
    nArray = Modulus Array
    aArray = Array of 'slopes' for the linear padding
    bArray = Array of 'y-intercepts' for the linear padding
    e = public exponent
    """
    if(len(cArray) == len(nArray) == len(aArray) == len(bArray) == e):
        for i in range(e):
            cArray[i] = Integer(cArray[i])
            nArray[i] = Integer(nArray[i])
            aArray[i] = Integer(aArray[i])
            bArray[i] = Integer(bArray[i])
        TArray = [-1]*e
        for i in range(e):
            arrayToCRT = [0]*e
            arrayToCRT[i] = 1
            TArray[i] = crt(arrayToCRT,nArray)
        P.<x> = PolynomialRing(Zmod(prod(nArray)))
        gArray = [-1]*e
        for i in range(e):
            gArray[i] = TArray[i]*(pow(aArray[i]*x + bArray[i],e) - cArray[i])
        g = sum(gArray)
        g = g.monic()
        # Use Sage's inbuilt coppersmith method
        roots = g.small_roots(epsilon=eps)
        if(len(roots)== 0):
            print("No Solutions found")
            return -1
        return roots[0]
    else:
        print("CiphertextArray, ModulusArray, and the linear padding arrays need to be of the same length," +
          "and the same size as the public exponent")
def LinearPadding():
    import random
    import binascii
    e = 3
    nArr = [
        0x81e620887a13849d094251e5db9b9160d299d2233244876344c0b454c99f7baf9322aa90b371f59a8ed673f666137df1f1e92d86e7b036479a2519827a81c7648543e16d4d0a334d0aa1124ad4c79429
        0x6c7c7935c58a586cf45e2e62ee51f6619ae2f6a7cef3865ed40a0d62ec31ba612e81045bcc6e50aa41d225b0f92b0d4a40051e2cf857ba61e91619e8fb3e2691d276c1abb5231c8012deb449e85752d2
        0x5e67f4953462f66d217e4bf80fd4f591cbe22a8a3eac42f681aea880f0f90e4a34aca250b01754dd49d3b7512011609f757cbaf8ae7c97d5894fb92fb36595aff4a1303d01e5c707284bbfdc20b8378e
    cArr = [
        0x3512b763bab0b45b2c6941cccd550c8b2628cea0f162dc3902951e48115d58d16ea25075da6331617e7a4ac6062190f8ce91c65c91cff57a845a21d2ebd792b46bdcb666bc4aeab2232f990956084003
        0x36bfe6fba6f34b93a0d2d44c890dfe44afc715a586bc1a44aa184571bb88a238187024b36b22a1f52a64f553fb52cf7ce193937e047307dd62e4c980601a3d20b1fbfe69888992726b11bf20330e48e4
        0x4961ba65469dfc17e663af04dfb8eeee16c61df4f85971495d0c7e706104060263896365179 1cfad28992312309c3179da27babf2a80fe41c062b21aa922da53bd793c614a0974ec5e5e18f9696df875
    aArr = [
        0xd0f458bc246d88f38e78076b36ad58981928594035b9e428401dc3ccf049a8012926dffb5be9fa225e8e128370581acc79ee24fa259d4ea895ce61d3d607ed2b,
        0xfbedf9c34170262e2ed0eee7512e935715400a8ce541285c98e5269d2cdf4dc1aa81e117bf5d62a3310064376e8c3d5d5c4fa67e5a434ad93e5875eaa7be9545,
        0xa2995200a4f252d7ba9959a3b7d51c4b138f3823869f71573f4ab61c581ce8879d40396a33ddc32a93fd100a1029dba53e41a0acbe9e023a0bf51c6e4ddc911d]
    bArr = [
        0xc2a6d47dc16824c86e92a9e88a931d215846052fe6787c11d0fcd9f4dde28f510707c33948290f69644a7fa64075d85e7761cfff3c627ee5156a03bd9f241c51,
        0xc2343fdbb6a351b387174db494e03d0879bea084e65b16f3f0ad106472bd3974813aec28a01fcceeae00db6d38b6c32bb6ce900dff236ae9c5814ad089591115,
        0xc4a2fb937c7441be58bfcb06208e0987423ab577041d0accf1f446545b9ebb7e4874fc56597ab1b842bb50e364a62f07a0afe7d6eff7a805361f8d3a12e79d65]
    randUpperBound = pow(2,500)
    msg = linearPaddingHastads(cArr,nArr,aArr,bArr,e=e,eps=1/8)
    msg = hex(int(msg))[2:]
    if(msg[-1]=='L'):
        msg = msg[:-1]
    if(len(msg)%2 == 1):
        msg = '0' + msg
    print(msg)
    print(binascii.unhexlify(msg))
if __name__ == '__main__':
    LinearPadding()
```

EXP

```
HOST = "47.240.41.112"
PORT = 54321
from Crypto.Util.strxor import strxor
from pwn import *
def pad(msg):
    pad_length = 16 - len(msg) % 16
    return msg + chr(pad_length) * pad_length
r = remote(HOST, PORT)
ru = lambda x : r.recvuntil(x)
rl = lambda  : r.recvline()
sl = lambda x : r.sendline(x)
# Give a large number bigger than n to break proof_of_work
ru('{65537, ')
n = ru('L}').strip('L}')
n = int(n[2:],16)
ru('Give me something you want to encrypt:')
sl(str(n**2))
# pad the message and target message we got in the first step
msg = pad("I will send you the ticket tomorrow afternoon")
target_msg = pad("I will send you the ticket tomorrow morning")
ru('message')
sl('1')
ru('this:')
message = ((ru('n').strip(' ')).strip('n')).decode('hex')
ru('message')
# message xor enc_message = middle_key_stream, middle_key_stream xor target_message = enc_target_message, so enc_target_message = xor(message,enc_message,target_message)
enc_target_message = strxor(strxor(target_msg,message),msg).encode('hex')
# choice 2 and send enc_target_message to get flag
sl('2')
ru('now:')
sl(enc_target_message)
flag = ru('}')
print "[+]FLAG IS: "+flag
r.close()
```

FLAG

```
sctf{7h15_ch4ll3n63_15_n07_h4rd_f0r_y0u_r16h7?}
```

- ***本文作者：*** *CTFHub*
- ***本文链接：*** *https://writeup.ctfhub.com/Challenge/2019/SCTF/Crypto/kXoshWQMJZcKURxr1VhQA1.html*
- ***版权声明：*** *本博客所有文章除特别声明外，均采用 BY-NC-SA 许可协议。转载请注明出处！*

 # Challenge # 2019 # SCTF # Crypto
强网先锋-打野
warmup