Bleichenbacher Signature Exponent 3 Attack RUS

14 февраля 2020 г.

1 Атака Блейхенбахера на подписи RSA с открытой экспонентой 3

Открытая экспонента 3 часто используется из-за того, что вычисления с ней намного проще (меньше энергии и времени). Очевидно, что она становится проблемой, если сообщение, которое вы пытаетесь зашифровать не переполняет модуль при возведении в степень. Чтобы предотвратить эту проблему были созданы несколько схем паддинга. Самая занимательная с точки зрения атак - это PKCS#1 v1.5, о которой мы узнаем больше в следующих тасках. Она использует похожий формат для подписей и шифрования, но сегодня нам интересны только подписи. Правильная подпись по PKCS имеет следующий формат:

`00 | 01 | FF | .. | FF | 00 | HASH`

Если N имеет длину k в байтах (обычно 128,256 или 384), то количество байтов FF должно быть равно $(k-3-len(HASH))$.

К сожалению, некоторые решения проверяли лишь последовательность

`00 | 01 | FF | .. | FF | 00`

с любым количеством байтов FF и потом брали необходимое для хеша количество байтов после паддинга. Проблема заключается в том, что настоящий паддинг уникален для каждого хеша, а при изменяющемся количестве байтов FF, появляется много возможных комбинаций, так как после хеша могут идти любые байты. При $e = 3$ это становится реальной проблемой, так как мы можем подделать подпись. Можно аппроксимировать кубический корень из $M =$

`00 | 01 | FF | 00 | HASH | 00 | .. | 00`

$S' \approx M^{1/3}$ и найти подпись $S = S' + 1$. Поскольку это изменить лишь байты после хеша, подделка пройдет проверку.

Подделайте подпись сообщения b'flag' и отправьте на сервер. Вместо хеша от b'flag' сервер просто проверяет наличие байтов 'flag' сразу после паддинга. Удачи!

[1]:
```
import socket
import re
from Crypto.Util.number import bytes_to_long,long_to_bytes,inverse,GCD
class VulnServerClient:
    def __init__(self,show=True):
        """Инициализация, подключаемся к серверу"""
        self.s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        self.s.connect(('cryptotraining.zone',1341))
```

```
        if show:
            print (self.recv_until().decode())
    def recv_until(self,symb=b'\n>'):
        """Получаем сообщения с сервера, по дефолту до знака приглашения"""
        data=b''
        while True:

            data+=self.s.recv(1)
            if data[-len(symb):]==symb:
                break
        return data
    def get_public_key(self,show=True):
        """Получаем открытый ключ с сервера"""
        self.s.sendall('public\n'.encode())
        response=self.recv_until().decode()
        if show:
            print (response)
        e=int(re.search('(?<=e: )\d+',response).group(0))
        N=int(re.search('(?<=N: )\d+',response).group(0))
        self.num_len=len(long_to_bytes(N))
        return (e,N)

    def checkSignature(self,c,show=True):
        """Проверить, является ли это число правильной подписью"""
        try:
            num_len=self.num_len
        except KeyError:
            print ('You need to get the public key from the server first')
            return
        signature_bytes=long_to_bytes(c,num_len)
        self.checkSignatureBytes(signature_bytes,show)

    def checkSignatureBytes(self,c,show=True):
        """Проверить, является ли эта последовательность байтов правильной␣
↪подписью"""
        try:
            num_len=self.num_len
        except KeyError:
            print ('You need to get the public key from the server first')
            return
        if len(c)>num_len:
            print ("The message is too long")
            return

        hex_c=c.hex().encode()
        self.s.sendall(b'flag '+hex_c+b'\n',)
        response=self.recv_until(b'\n').decode()
```

```
        if show:
            print (response)

        if response.find('Wrong')!=-1:
            print('Wrong signature')
            x=self.recv_until()
            if show:
                print (x)
            return
        flag=re.search('CRYPTOTRAINING\{.*\}',response).group(0)
        print ('FLAG: ',flag)

    def __del__(self):
        self.s.close()
```

[2]:
```
vs=VulnServerClient()
(e,N)=vs.get_public_key()
```

```
Welcome to Bleichenbacher's signature exponent 3 attack task
Available commands:
help - print this help
public - show public key
flag <hex(signature(b'flag'))> - print flag
quit - quit
>
e: 3
N: 23451721638450735837192936512705285084148016986437726204139579305692323143550
81964345787672240794572490003697517555394747462447654353297912996510869465982681
36790820845336349648480663904722604204569306361101850465776320279299837628291806
99297454189397229979451189326493503464132097157965882767371766584791340102547650
80712744161589731316860006929661017332925231984135770311579228265777992033597729
80378140505908286573965733673226813533790453648080805499611914254159619378643824
95328668937920235775400914630813078535281943770942554463414529423069019686396596
20114137170620726892914598776222266175315196918951747100927 7
>
```

[ ]: