CTF Crypto方向出题

原创

4XWi11 已于 2022-04-23 08:23:07 修改 346 收藏 2

分类专栏： 树哥让我天天写之Crypto 文章标签： vim python linux

于 2021-09-21 15:59:36 首次发布

本文链接：https://blog.csdn.net/m0_49109277/article/details/120402420

原文Fr.https://4xwi11.github.io/posts/921543e1/

CTF密码题是怎么炼成的

环境搭建

web和pwn的出题会用到docker的相关操作，出pwn题，有个大佬写了个脚本一键搭

而Crypto出题就不用那么麻烦（指环境搭建），可以只写下python脚本就好，主要还是对服务器进行配置吧，想起了某段时间，我一直在配置pwn环境的虚拟机，*先虚拟机，再vim，再换源，再zsh，on-my-zsh，p10k，再vim主题*，pwn用到的环境以及一些工具主要参考的是（Crypto交互会用到其中的pwntools

https://blog.csdn.net/Y_peak/article/details/112850307

这里到时候整一篇Crypto工具配置

https://4xwi11.github.io/posts/1cc01193/

接下来正式开始

主要参考

https://blog.soreatu.com/posts/how-to-setup-for-interactive-crypto-problems/

socat直接挂载

比较简便，适合校内比赛自己玩

主要是题目文件server.py，以及run.py

```
# admin @ iZbp1c5fxm09nfk44yamgiZ in ~/challenge/CSAWCTF-2021 [21:41:45]
$ tree
.
├── Bits
│   ├── run.py
│   └── server.rs
├── Forgery
│   ├── run.py
│   └── server.py
└── RSA_Pop_Quiz
    ├── run.py
    └── server.py

3 directories, 6 files
```

然后是启动程序，可以遵循以下模式，可以一直在这个端口运行（端口在运行的时候自己选择，阿里云的服务器是要在工作台开起来先）；还有开启防火墙端口的操作，可以直接看上面这位大师傅的，我一般复现只开一个端口

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import os
import sys

if len(sys.argv) != 2:
    print("Usage: %s [port]" % sys.argv[0])
    sys.exit(1)

port = sys.argv[1]
command = 'socat -d -d tcp-l:' + port + ',reuseaddr,fork EXEC:"python -u server.py" '
os.system(command)
```

将命令换成（前面加个nohup

```
$ nohup socat -d -d tcp-l:[port],reuseaddr,fork EXEC:"python -u server.py"
```

就可以在服务器上一直跑了（除非服务器关了

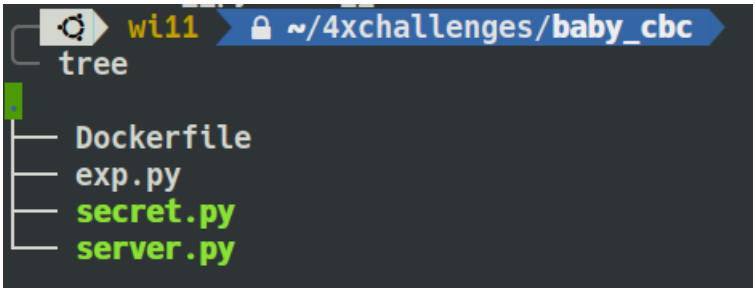
Docker打包的方式

之前的删了，太不实用，还是出不来题（校赛出题被赵总吐槽没有docker），因为照抄别人的没有灵魂，根本不会，所以这篇回炉重造

之前的一些问题

- 镜像搭建太慢（换阿里源之后还行，但更多的是安装方式（比如gmp2包是用wheel还是安装依赖
- 容器创建完程序就退出了
- 题目有问题……

这里以一道极其简单的题目为例（之前直接拿比赛的文件，结果各种环境和问题，其实不便我们学习



server.py

```python
from hashlib import sha256
import socketserver
from secret import flag
import signal
import string
import random
import os


class Task(socketserver.BaseRequestHandler):
    def _recvall(self):
        BUFF_SIZE = 2048
        data = b''
        while True:
            part = self.request.recv(BUFF_SIZE)
            data += part
            if len(part) < BUFF_SIZE:
                break
        return data.strip()

    def send(self, msg, newline=True):
        try:
            if newline:
                msg += b'\n'
            self.request.sendall(msg)
        except:
            pass

    def recv(self, prompt=b'[-] '):
        self.send(prompt, newline=False)
        return self._recvall()

    def proof_of_work(self):
        random.seed(os.urandom(8))
        proof = ''.join(
            [random.choice(string.ascii_letters+string.digits) for _ in range(20)])
        _hexdigest = sha256(proof.encode()).hexdigest()
        self.send(f"[+] sha256(XXXX+{proof[4:]}) == {_hexdigest}".encode())
        x = self.recv(prompt=b'[+] Plz tell me XXXX: ')
        if len(x) != 4 or sha256(x+proof[4:].encode()).hexdigest() != _hexdigest:
            return False
        return True

    def handle(self):
        signal.alarm(60)
        if not self.proof_of_work():
```

```
            self.send(b'[!] Wrong!')
            return

        self.send(b'here is your flag')
        self.send(flag)


class ThreadedServer(socketserver.ThreadingMixIn, socketserver.TCPServer):
    pass


class ForkedServer(socketserver.ForkingMixIn, socketserver.TCPServer):
    pass


if __name__ == "__main__":
    HOST, PORT = '0.0.0.0', 10001
    server = ForkedServer((HOST, PORT), Task)
    server.allow_reuse_address = True
    print(HOST, PORT)
    server.serve_forever()
```

DockerFile

```
FROM python:3.8
LABEL Description="baby_try" VERSION='1.0'

COPY server.py .
COPY secret.py .

RUN chmod +x server.py

EXPOSE 12345 # 仅仅是申明，没有实际用，容器都是随机映射的，这里方便编写者查看

CMD ["python", "server.py"]
```

exp

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import hashlib
from string import ascii_letters, digits
from pwn import *
from itertools import product

table = ascii_letters + digits


class Solve():
    def __init__(self):
        self.sh = remote('127.0.0.1', 12345)
        # self.sh = remote('121.36.197.254', 9999)

    def proof_of_work(self):
        # [+] sha256(XXXX+JaakUDSfxkW0xjzV) == 4dbfdc61cb88f5bd08d87493ac62e5ab174780f5f019051f91df8b3c36564ed0
        # [+] Plz tell me XXXX:
        proof = self.sh.recvuntil(b'[+] Plz tell me XXXX:')
        tail = proof[16:32].decode()
        _hash = proof[37:101].decode()
        for i in product(table, repeat=4):
            head = ''.join(i)
            t = hashlib.sha256((head + tail).encode()).hexdigest()
            if t == _hash:
                self.sh.sendline(head.encode())
                break

    def solve(self):
        self.proof_of_work()
        self.sh.recvline()
        flag = self.sh.recvline()[:-1].decode()
        print(flag)
        self.sh.close()


if __name__ == '__main__':
    solution = Solve()
    solution.solve()
```

step1 创建镜像

```
$ docker build . -t baby_try
```

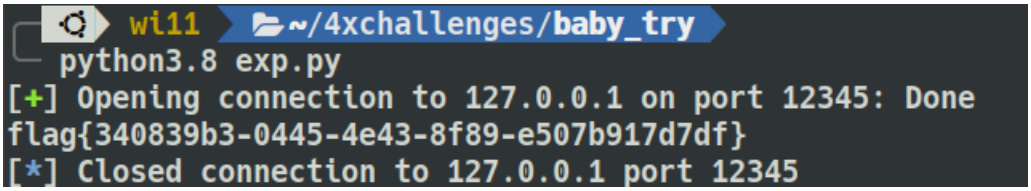
step2 启动容器

```
$ docker run --name trytry -d -idt -p 12345:10001 baby_try
```

将端口映射到本地12345端口

step3 尝试exp打本地

没什么问题



step4 挂载服务器

本地没问题后，就要部署到服务器上，操作和socat类似就不做了，以后碰到问题再说

Docker命令

创建镜像

```
$ docker build . -t [images_name]
```

查看创建的镜像

```
$ docker images
```

创建容器

```
$ docker run --name [container_name] -d [-idt] -p [host_port]:[container_port] [images_name]
```

- `-d`表示不进入容器内部
- `-p`表示将容器内部的port_inner端口映射到port_outer端口
- `-idt`表示创建守护进程（没有这个容器创建完程序就退出了，不太清楚

查看正在运行的容器

```
$ docker ps -a
```

停止容器

```
$ docker stop [container_id]
```

删除容器

```
$ docker rm [container_id]
```

删除镜像

```
$ docker rmi [image_id]
```

进入容器

```
$ docker attach [container_id]
$ docker exec -it [container_id] bash
```

常用Dockerfile语法

```
# 看菜鸟教程
```