代码安全审计报告

共发现 5 个安全问题

Webbuster.py:25

未设置请求超时 (MEDIUM)

```
"""循环扫描，直到web_path的路径被取完"""

    while not web_paths.empty():

        path = web_paths.get()  # 每次取一条路径

        url = f'{TARGET}{path}'

        time.sleep(2)  # 每个请求间隔2秒
```

requests.get()未设置timeout参数，可能导致线程长期阻塞

修复建议： 为requests.get()添加合理的超时时间，如：requests.get(url, timeout=10)

Webbuster.py:25

缺乏异常处理 (MEDIUM)

```
"""循环扫描，直到web_path的路径被取完"""

    while not web_paths.empty():

        path = web_paths.get()  # 每次取一条路径

        url = f'{TARGET}{path}'

        time.sleep(2)  # 每个请求间隔2秒
```

网络请求未添加try-except块，可能导致程序因网络异常崩溃

修复建议： 在test_remote()函数中添加异常处理机制，捕获requests异常

Webbuster.py:37

潜在资源耗尽 (LOW)

```
def run():

    mythreads = []  # 创建了一个空列表 mythreads 用于存储线程对象

    for i in range(THREADS):
```

未限制最大线程数，当THREADS设置过大时可能耗尽系统资源

修复建议： 添加线程数上限检查，建议不超过50个并发线程

Webbuster.py:45

敏感信息泄露 (HIGH)

```
t.start()



    for thread in mythreads:
```

```
        thread.join()
```

输出文件使用固定名称'myanswers.txt'，可能覆盖已有敏感文件

修复建议：使用唯一性文件名（如时间戳命名）或要求用户指定输出路径

Webbuster.py:7

硬编码凭证 (MEDIUM)

```
import time


TARGET = "http://localhost/WordPress-master"

THREADS = 10
```

目标地址硬编码在代码中，不利于灵活部署且可能泄露测试目标

修复建议：通过命令行参数或配置文件动态获取TARGET值