

弱密码检测工具：Webpass

场景说明

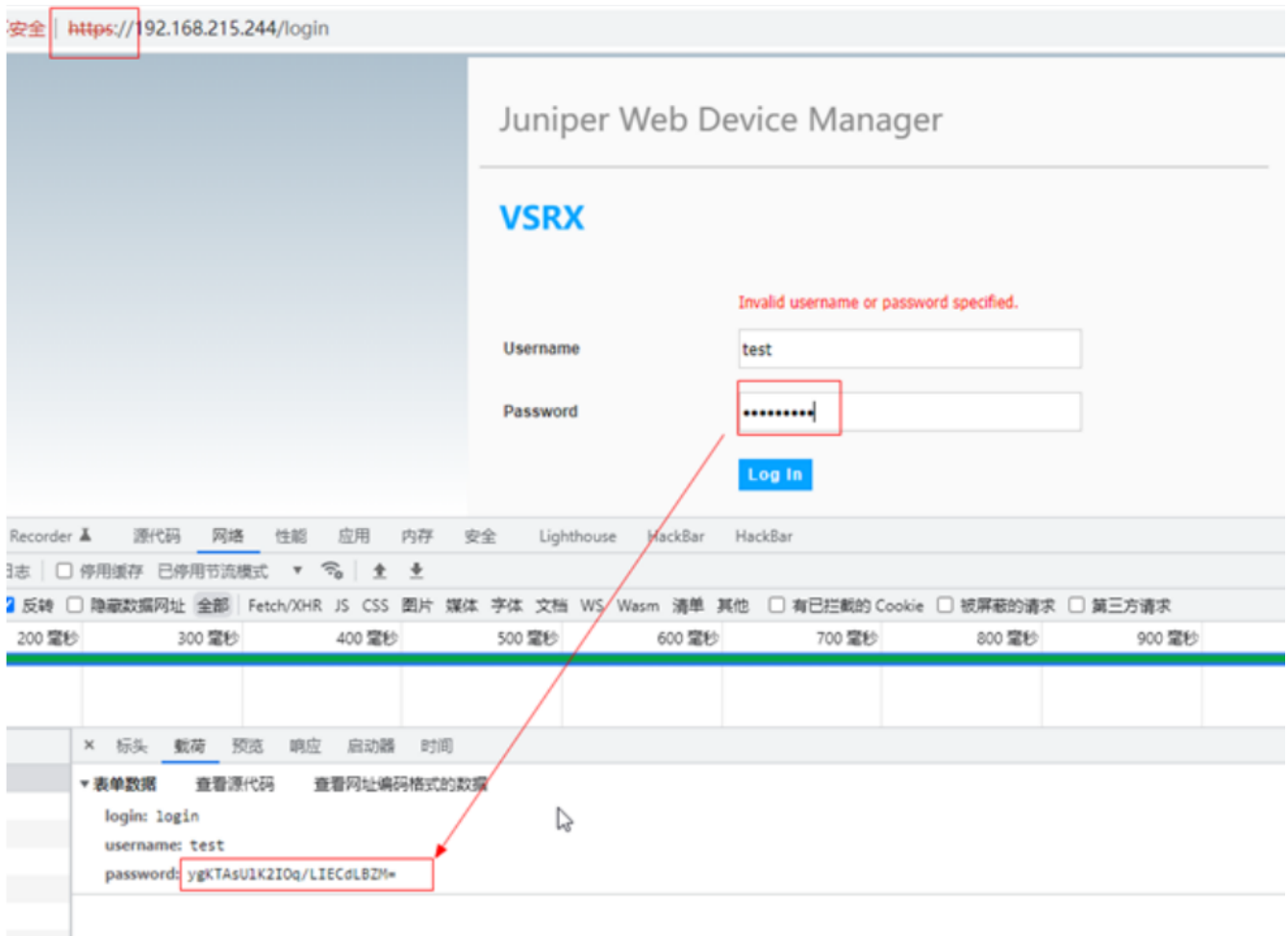
账号密码是信息安全的第一道屏障，在攻防中，弱密码拥有与高危漏洞同样的重视程度，除了传统协议(ftp, mysql, ssh 等) 的弱密码检测外，http(s) 层面的关键系统越来越多，例如 VPN 系统、SDP 系统等，如果能快速对 http(s) 业务系统的弱密码进行探测，相当于多了一条渗透路径。

但是 http(s) 层面有**验证码**、**密码加密**等灵活的防御机制，且经常配置**错误次数限制**，一个账号短时间错误次数太多，直接禁止登陆。这种情况下，万能的 burpsuite 也显得有些无力。

Webpass 工具通过驱动浏览器，模拟真实登录场景，形成通用的弱密码检测系统，可以支持 http(s) 协议的 web 登录弱密码检测，也支持针对不同的网络设备定制不同的字典，从而更容易检测到弱密码；

解决的问题

1. 密码加密传输



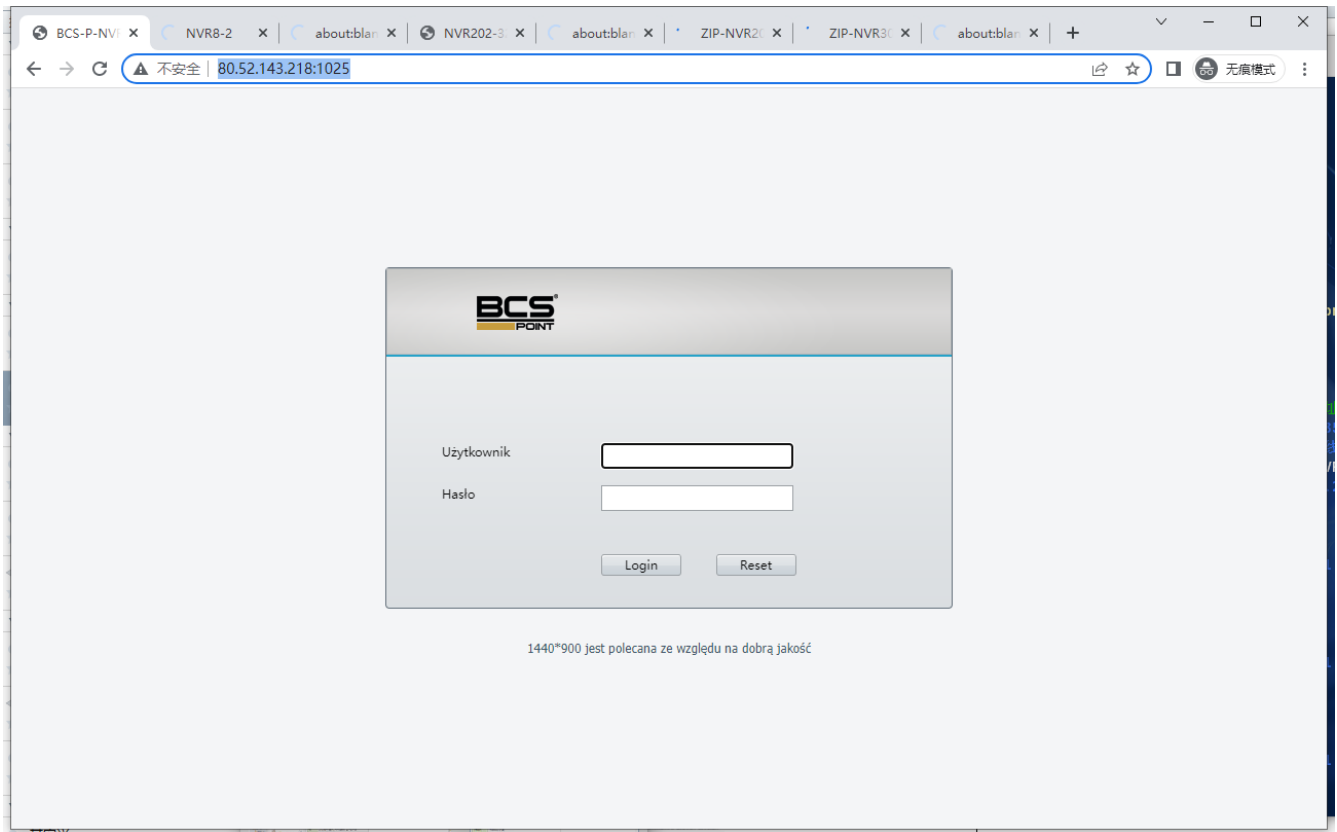
填写表单后，页面代码对填写内容进行加密，加密后发送到服务器后台，如图看起来像 base64 实际上不是，那么就因为不知道他怎么加密的，从而无法直接发包爆破。

这种情况只有驱动浏览器来进行爆破，让系统自己进行加密转化，最为简单直接

```
1 async def create_browser_loop(sema = None, ap = None):
2     try:
3         process = get_value('process', 8)
4         glob.ocr = dddocr.DdddOcr(old=True, show_ad=False)
5         # 创建指定数量的浏览器页面
6         browser = await ap.chromium.launch(headless=not get_value('debug', False), timeout=10 * 1000)
7
8         context = await browser.new_context(ignore_https_errors=True)
9         for _ in range(process):
10             page = await context.new_page()
11             await glob.pQu.put(page)
12             logger.debug(f"浏览器已创建页面总数: {glob.pQu.qsize()}")
13
```

```
14     except KeyboardInterrupt: kill_myself(1)
15     except Exception:
16         err_msg = get_err_msg()
17         logger.error(err_msg)
18         return err_msg
```

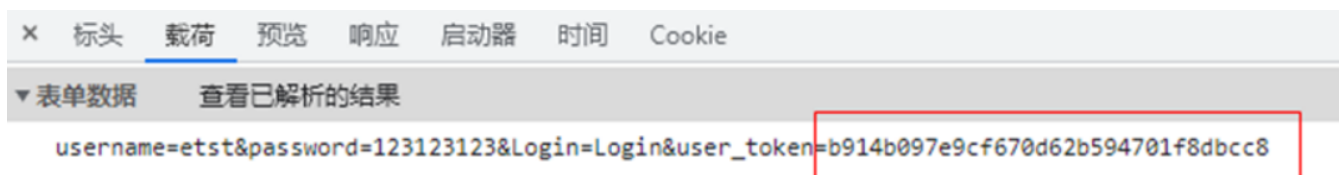
上述代码会打开指定数量的页面，自动填写配置好的用户名密码，效果图如下：



2. 验证码和 Token 验证

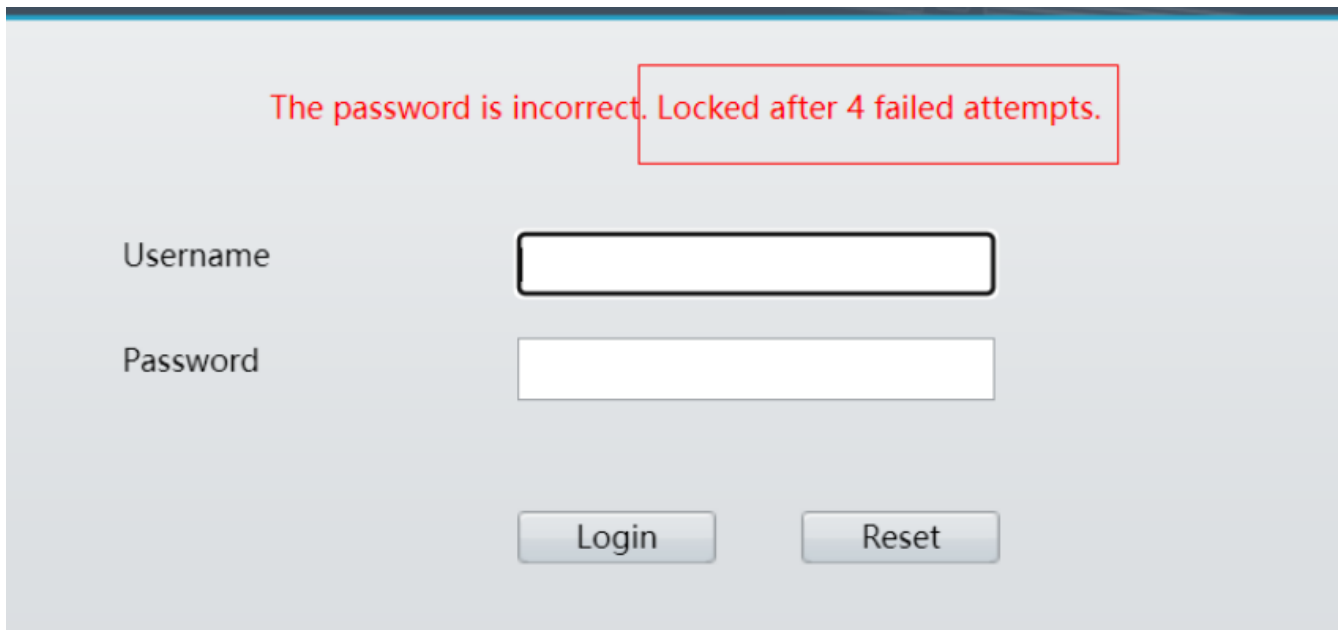


验证码是很常见的，当然单纯的验证码，发包模式也可以解决。但是有的时候，页面会带有隐藏表单，进行 token 提交，服务器会对 token 进行验证。



对于 token 验证不符合的请求，直接丢弃，这个类似于密码加密传输，不知道生成算法，直接发包也扑街。

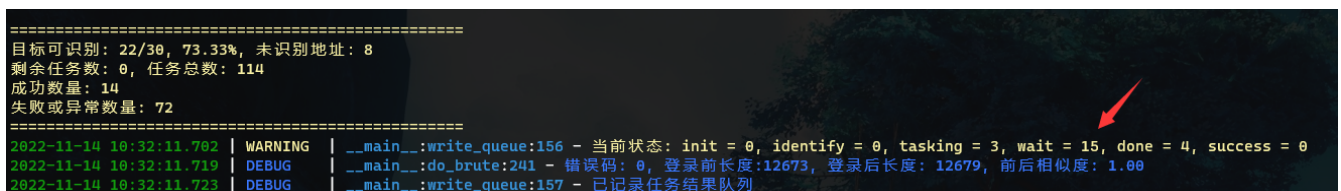
3. 错误次数限制



这种情况也很常见，当然发包的方式也可以解决这个问题，本程序主要是密码优先，枚举用户。

```
1 username_list = await aread_file(glob.cfg.pass_dict[key]['user_info'][1])
2 password_list = await aread_file(glob.cfg.pass_dict[key]['pass_info'][1])
3
4 user_pass = [[username_list[i], password_list[i]] for i in range(min(len(username_list), len(password_list)))] # 先写一对应的
5
6 for _pass in password_list: # 再添加剩余的，密码枚举用户
7     for _user in username_list:
8         if [_user, _pass] not in user_pass:
9             user_pass.append([_user, _pass])
```

单个目标枚举 3 次之后，测试下一个目标，这个目标切换为等待状态：


































另外，webpass 最初想法是爆破一些常见的网络设备，这样常见网络设备的默认密码自然是最高优先级，比通用弱密码更优先，上述代码也有体现。在配置中，可以配置每类产品独立的字典：

```

51 '奇安信网神全流量威胁发现系统':{
52     'keywords': ['奇安信网神', '全流量威胁发现系统'],
53     'user_info': ['[name="username"]', 'pass_dict/0-admin.txt'],
54     'pass_info': ['[name="password"]', 'pass_dict/1-qaxyty.txt'],
55     'button': ['[class="submit-btn"]'],
56     'CAPTCHA': ['[name="authcode"]', '[class="refresh-code"]'],
57     'suc': [],
58     'fail': ['username', 'password'],
59     'logic': ['or'],
60 },
61 '奇安信网神新一代安全感知系统':{
62     'keywords': ['奇安信网神新一代安全感知系统', '新一代安全感知系统'],
63     'user_info': ['//*[@id="app-main-component"]/div/div[4]/div[2]/div[1]/input', 'pass_dict/0-admin.txt'],
64     'pass_info': ['//*[@id="app-main-component"]/div/div[4]/div[2]/div[2]/input', 'pass_dict/1-qaxyty.txt'],
65     'button': ['//*[@id="app-main-component"]/div/div[4]/div[2]/button'],
66     'CAPTCHA': ['//*[@id="app-main-component"]/div/div[4]/div[2]/div[3]/input', '//*[@id="app-main-component"]'],
67     'suc': [],
68     'fail': [],
69     'logic': ['or'],
70 },
71 '奇安信网神流量传感器':{
72     'keywords': ['奇安信网神', '流量传感器'],
73     'user_info': ['[name="username"]', 'pass_dict/0-admin.txt'],
74     'pass_info': ['[name="password"]', 'pass_dict/1-qaxyty.txt'],
75     'button': ['[class="submit-btn"]'],
76     'CAPTCHA': ['[name="authcode"]', '[class="refresh-code"]'],
77     'suc': [],
78     'fail': ['username', 'password'],
79     'logic': ['or'],
80 },
81 '奇安信天眼分析平台':{
82     'keywords': ['奇安信天眼分析平台', 'update.skyeye.qianxin.com'],
83     'user_info': ['//*[@id="app-main-component"]/div/div[4]/div[2]/div[1]/input', 'pass_dict/0-admin.txt'],
84     'pass_info': ['//*[@id="app-main-component"]/div/div[4]/div[2]/div[2]/input', 'pass_dict/1-qaxyty.txt'],
85     'button': ['//*[@id="app-main-component"]/div/div[4]/div[2]/button']

```

代码配置如上图，字典文件如下图：

 0-0000.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-admin.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-admin1.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-administrator.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-ADSL.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-anonymous.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-broadmax.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-csadmin.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-empty.txt	2022/11/5 17:14	Notepad++ Doc...	0 KB
 0-epicrouter.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-eyouser.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-fiberhome.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-firewall.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-h3c.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-hdfw.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-hillstone.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-hwfw.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-IBMIMM.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-ishare.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-jboss.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-jonas.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-KQAdmin.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-leadsec.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-lenovo.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-manager.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-myywsj.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-netscreen.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-ns25000.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-qxql.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 0-root.txt	2022/11/5 17:14	Notepad++ Doc...	1 KB
 -	-----	-----	----

4. 用户名密码表单灵活多变

```

    </svg>
  </div>
  <input id="username" name="username" class="form-control" aria-descrip
    type="text" placeholder="请输入用户名" autocomplete="off" />
</div>
</div>
<div class="form-group">
  <div class="input-group">
    <div class="input-group-addon">
      <svg class="icon-font" width="18" height="18">
        <use xlink:href="#iconmima-blue"></use>
      </svg>
    </div>
    <input id="password" name="password" class="form-control" tabindex="2"
      placeholder="请输入密码" autocomplete="off" />
    </div>
  </div>

```

Web 页面的用户名密码表单，一般习惯是写 username, password 但是也有很多情况不是这两个名称，那么在程序填写用户名密码时，往往也需要配置或者智能识别，目前考虑的方案是：

1. 列举出所有的 input, 类型为 password 的肯定是 password
2. 计算表单 name 与 user 关键字的相似度，超过一定值考虑为 username 字段
3. 计算表单 name 与 pass 关键字的相似度，超过一定值考虑为 password 字段
4. 补充一些用户配置字典。

对于未识别的系统，可以使用通用字典来进行自动化爆破

5. 多目标批量

单个 web 目标的爆破，burpsuite 很灵活有效，但是多个目标，就比较麻烦了。新出的国产挑战者 Yakit 可以支持，但是在请求次数过多也会程序卡死

Request

[数据包扫描](#)[热加载](#)[URL](#)

```
1 GET /manager/html HTTP/1.1
2 Host: {{x(target)}}
3 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
  image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
4 Accept-Encoding: gzip, deflate
5 Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
6 Authorization: Basic {{base64({{x(user_top10)}}:{{x(pass_top25)}})}}
7 Cache-Control: max-age=0
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
  Gecko) Chrome/107.0.0.0 Safari/537.36
10
11
```

【优化】webfuzzer发包数量多导致前端页面卡死 #479

[Open](#)

Vernon818 opened this issue on Oct 10 · 1 comment



Vernon818 commented on Oct 10

Collaborator



目前数据存在前端，导致数据过多的时候会卡

强制停止

☒ 强制 HTTPS

高级配置: ☒

[分享 / 导入](#)

[历史](#)

[sending packets](#)

Intruder: [插入 yak.fuzz 语法](#)

渲染 fuzz: ☒

并发线程:

过滤器模式: [丢弃](#) [保留](#)

HTTPS: ☒

不修复长度: ☐

设置代理:

状态码: 关键:

请求 Host:

超时时间:

随机延迟: min: max:

正则:



WAY29 mentioned this issue 3 days ago

[需求]增加basic 认证 爆破 #512

[Closed](#)

WAY29 commented 3 days ago

Collaborator



yakit最新版尝试解决了这个问题，请升级到最新版yakit进行测试，如有问题再反馈

那么自己写程序，就能很好的解决这个问题：

```

09-21__Web弱密码扫描\_git__> python webpass_call.py -f env\t2.txt -d
INFO      | __main__:launch:50 - 目标总数 => 14
DEBUG     | __main__:launch:51 - 所有配置 =>

3.218:1025": {
nit",

DEBUG     | __main__:write_print_info_loop:120 - 已转换字典编码
DEBUG     | __main__:create_browser_loop:170 - 浏览器已创建页面总数:8
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
DEBUG     | __main__:main:103 - __step(232) >> main(103) >> 添加任务
WARNING   | __main__:back_run:68 - 当前状态: init = 14, identify = 0, tasking = 0, wait = 0, done = 0, success = 0
INFO      | __main__:manage_pages:201 - GET一个空闲页处理: [ init ] http://91.235.247.118:88
DEBUG     | __main__:identify_url:260 - manage_pages(204) >> identify_url(260) >> 识别

```

文件中一行一个目标，自动识别系统，调用指定的字典，未识别的使用通用字典

使用方法

```

1  INFO: Showing help with the command 'webpass_call.py -- --help'.
2
3  NAME
4      webpass_call.py -u: url -f: file -g: 配置模式 -w: 单目标等待时间，分钟 -j: 只探测识别到的目标
5
6  SYNOPSIS
7      webpass_call.py <flags>
8
9  DESCRIPTION
10     -u: url -f: file -g: 配置模式 -w: 单目标等待时间，分钟 -j: 只探测识别到的目标
11
12  FLAGS
13     --u=U
14         Default: ''
15     --f=F
16         Default: ''
17     --debug=DEBUG
18         Default: False
19     --gen=GEN
20         Default: False
21     --wait=WAIT

```

```
22         Default: 3
23     --process=PROCESS
24         Default: 8
25     --just_discern=JUST_DISCERN
26         Default: True
```

基本用法

-u <url> : 探测单个或者多个目标弱密码，含有 url 的字符串即可，会自动解析 url

-f <file> : 探测文件中所有的 url，带有 url 的文件即可，会自动解析 url

上述两个开关可以都配置，目标相加

```
1 python webpass_call.py -u http://1.2.3.4
2 python webpass_call.py -f /etc/target.txt
3 python webpass_call.py -u http://1.2.3.4 -f /etc/target.txt
```

Python

注意：url 需要带入协议类型如 http, https 等，如果不带，可能导致 url 无法识别。

配置模式

当然，在实际使用中，默认密码的情况比较多见，所以尽量还是经过配置后使用，而不是直接通用字典。

-g <url> : 配置模式，默认以 webkit 浏览器打开

```
1 python webpass_call.py -g http://baidu.com
```

Python



比较方便于生成配置。

其他开关

- w <mins> : 配置单目标等待时长，默认单目标尝试 3 次后等待 3 分钟
- p <process> : 浏览器最大页面数，即理解成多少个线程，实际上是协程的程序
- j <True>: 配置 True 则只爆破已识别的目标，不进行通用爆破，是 just_discern 的首字母
- d <True>: 调试开关，配置为 True, 多很多步骤打印，浏览器也会显示出来

配置文件说明

```
'奇安信网神全流量威胁发现系统':{
    'keywords': ['奇安信网神', '全流量威胁发现系统'],
    'user_info': ['[name=\"username\"]', 'pass_dict/0-admin.txt'],
    'pass_info': ['[name=\"password\"]', 'pass_dict/1-qaxty.txt'],
    'button': ['[class=\"submit-btn\"]'],
    'CAPTCHA': ['[name=\"authcode\"]', '[class=\"refresh-code\"]'],
    'suc': [],
    'fail': ['username', 'password'],
    'logic': ['or'],
},
```

配置文件基本结构如上图，是一个大字典，每类产品一个小字典，字典内部是 key:list 结构。

- Python
- 1 - keywords: 列表类型，识别产品的关键词，各个关键词之间与关系，即登录页html中某些关键词，支持base64编码；留空则直接用“key”作为关键词判断，如果没有找到则归为未识别
 - 2
 - 3 - user_info: 是包含username 字段名和字典目录的列表，pass_info类似
 - 4
 - 5 - button: 按钮名称，列表类型，一般用webpass_call.py -g url 获取按钮名称
 - 6
 - 7 - CAPTCHA: 长度一般为2，位置0位填充，位置1为截图，如果长度为0，则代表没有验证码
 - 8
 - 9 - suc: 登录成功后html中内容关键字，支持base64，根据logic计算整体真假
 - 10
 - 11 - fail: 登录失败后html中内容关键字，支持base64，根据logic计算整体真假
 - 12
 - 13 考虑到配置的便捷性，可以使用-g参数激活浏览器自动获取关键字段，便于填写，如果suc, fail都为空，则直接根据登陆前后页面相似度进行正确性判断

文件结构

- 0-errlog.txt # 报错信息

- 1-unknown_devices.txt # 未知设备
- 2-success.txt # 成功的目标
- 3-fail.txt # 失败的目标
- 4-check.txt # 需要检查的目标
- webpass_call.py # 主程序
- webpass_config.py # 主配置
- webpass_func.py # 功能库
- webpass_report.py # 上报 ES, 未完成

安装

STEP1. 官方网站下载 Python3.7+ 环境。

- python.org

STEP2. 安装调用库 PlayWright, Fire, loguru 等

- pip install playwright ddddocr fire loguru chardet aiofiles
- python -m playwright install

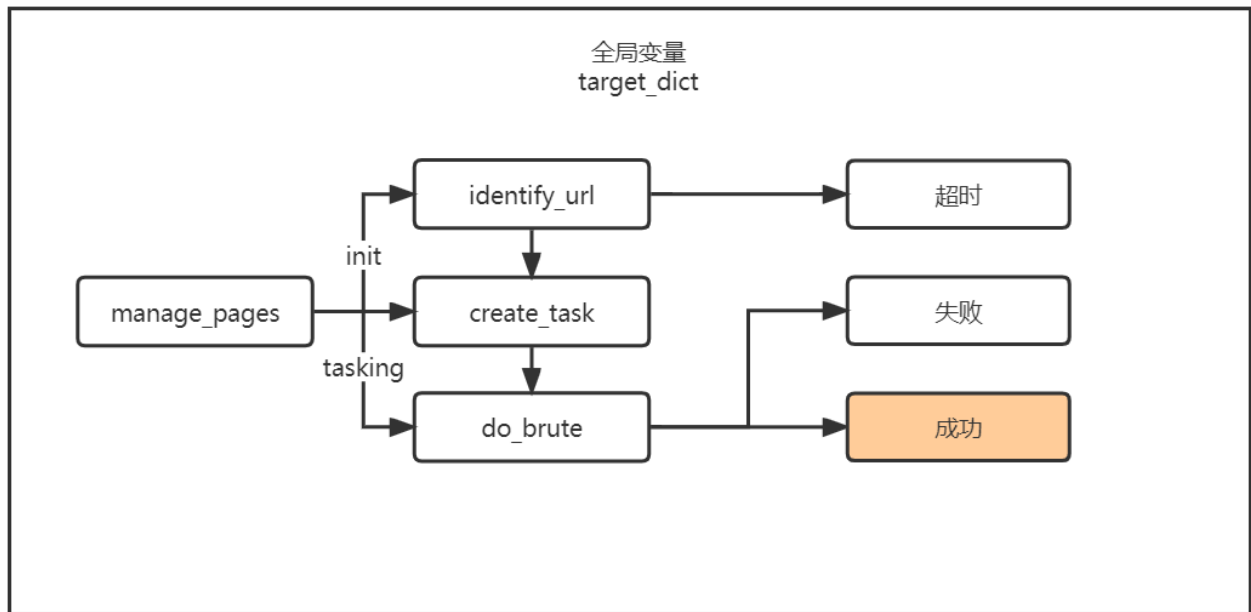
可能有一些未列举的, 后续提供 requirements.txt

装完库就可以运行。

代码结构

代码主要使用 Python3, 尽量使用 Python3.9 及以上的版本

整体结构



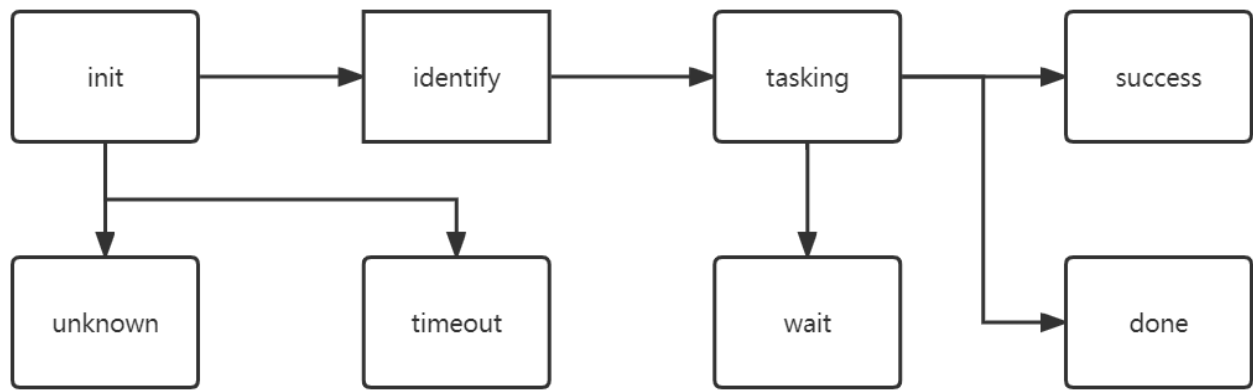
target_dict 数据结构

```
1 {
2   'target_dict':{
3     'http://1.2.2.2':{
4       'status':'init',
5       'title':'',
6       'content':'',
7       'key':'下一代防火墙',
8       'last_time':datetime 对象,
9       'user_pass':aQueue,
10    }
11  }
12 }
```

JSON

status 状态

- init
- identify, unknown, timeout
- tasking, wait
- done, success



后续计划

计划

- ☐ go 开发实现，便于编译成 exe 在目标内网执行
- ☐ 支持通用爆破

已实现

- ☒ 文档上述内容，除了计划之外，均已实现