404页面识别

Bojack / 2019-03-18 09:20:00 / 浏览数 2631 安全技术 WEB安全 顶(1) 踩(0)

# 404页面识别

## 前言

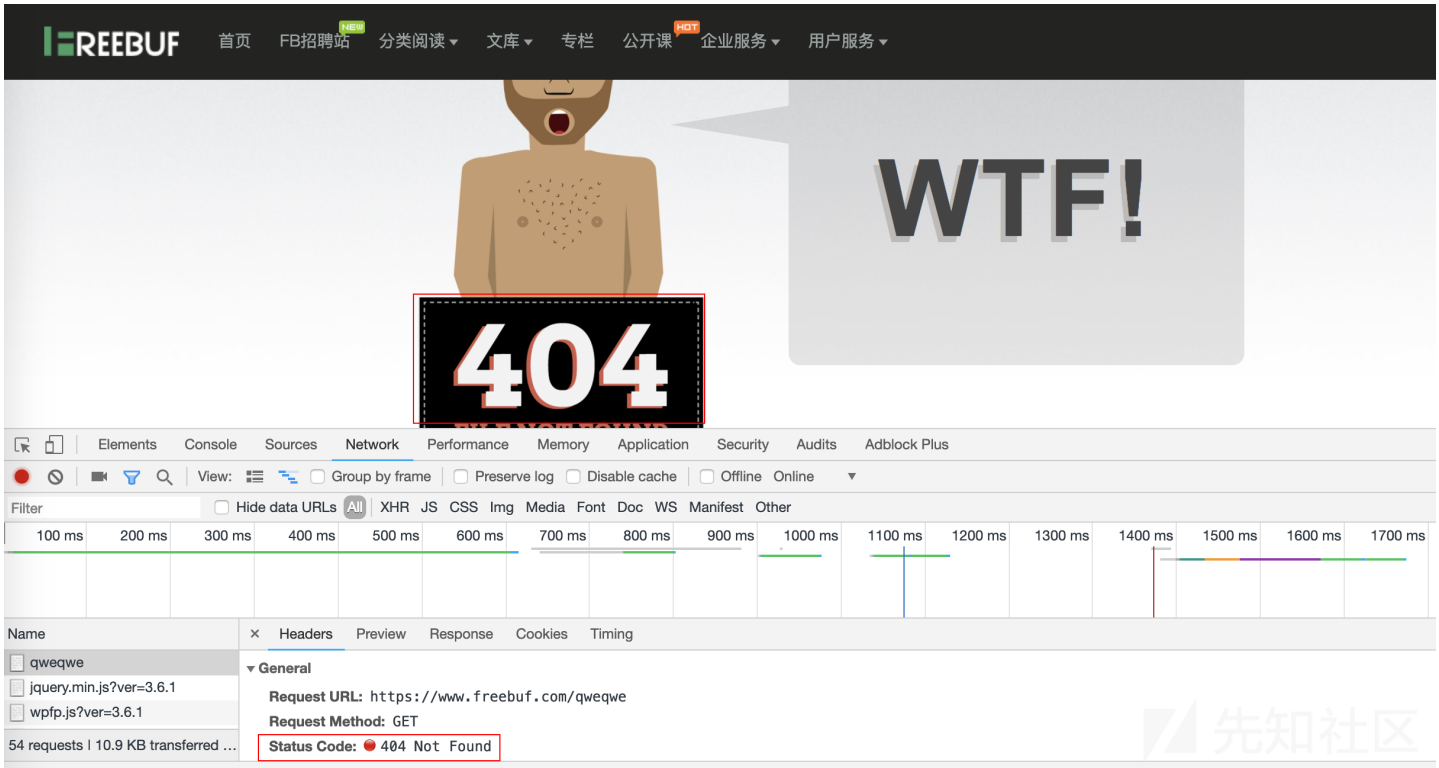
最近在挖洞前做资产收集的时跑了一波子域名，但是目前很多子域名挖掘机挖出来资产还是存在很多水分，准确率一般；于是乎写了个脚本用是否能请求成功作为筛选条件进

## 404页面识别思路

在编写之前先看一下目前网站404的呈现有哪些方式

1. web容器设置404错误页面，服务端返回404状态码

例如freebuf，当我们随便访问一个不存在的页面，返回的404页面，此时状态码返回了404。




这种情况下，可以根据返回response的状态码来直接判断是不是404页面。

1. 将404错误页面指向一个新的页面，页面上显示404信息，但是此时状态码并不为404，一般返回状态码有301、302，或者直接返回状态码为200的错误页面。

比如Baidu，当我们访问一个不存在的页面，会返回https://www.baidu.com/search/error。


根据以上这2种情况，大致得出404页面的识别思路。

- 从状态码是否为404判断
- 获取域名的404页面，然后判断请求的页面和404页面是否相似，相似则可以判断为404页面。

404页面过滤

页面相似度

根据上面提到的识别思路，首先要解决的第一个问题是■■■■■的判断。如何判断两个页面的相似度呢？

这里使用hashes.simhash，对两个页面的body计算hash值，再调用similarity获取两个页面的相似值，自定义一个阀值作为标准判断是否相似，radio可以根据具体情

```
from hashes.simhash import simhash

def is_similar_page(res1, res2, radio=0.85):
    if res1 is None or res2 is None:
        return False

    body1 = res1.body
    body2 = res2.body

    url1 = res1.get_url()
    url2 = res2.get_url()

    simhash1 = simhash(body1.decode('utf-8'))
    simhash2 = simhash(body2.decode('utf-8'))

    calc_radio = simhash1.similarity(simhash2)
    # print("[%s]■[%s]■■■■■■■■■■:%s" % (url1, url2, calc_radio))
    if calc_radio >= radio:
        return True
    else:
        return False
```

构造404页面

这里很简单，访问一个随机生成字符串为后缀的页面。

```
def generate_404_kb(self, url):
    # ■■URL■■■■
    domain = url.get_domain()               #www.freebuf.com
    domain_path = url.get_domain_path()     #https://www.freebuf.com
    rand_file = rand_letters(8) + '.html'
```

```
    url_404 = domain_path.urljoin(rand_file)
    resp_200 = requests.get(domain_path)
    resp_404 = requests.get(url_404)
    # ■■■■■■■■■■■■■■■■■■■404■■■■■■■■■■
    if is_similar_page(resp_200, resp_404):
        pass
    else:
        self._404_already_domain.append(domain)
        self._404_kb.append((domain, resp_404))
```

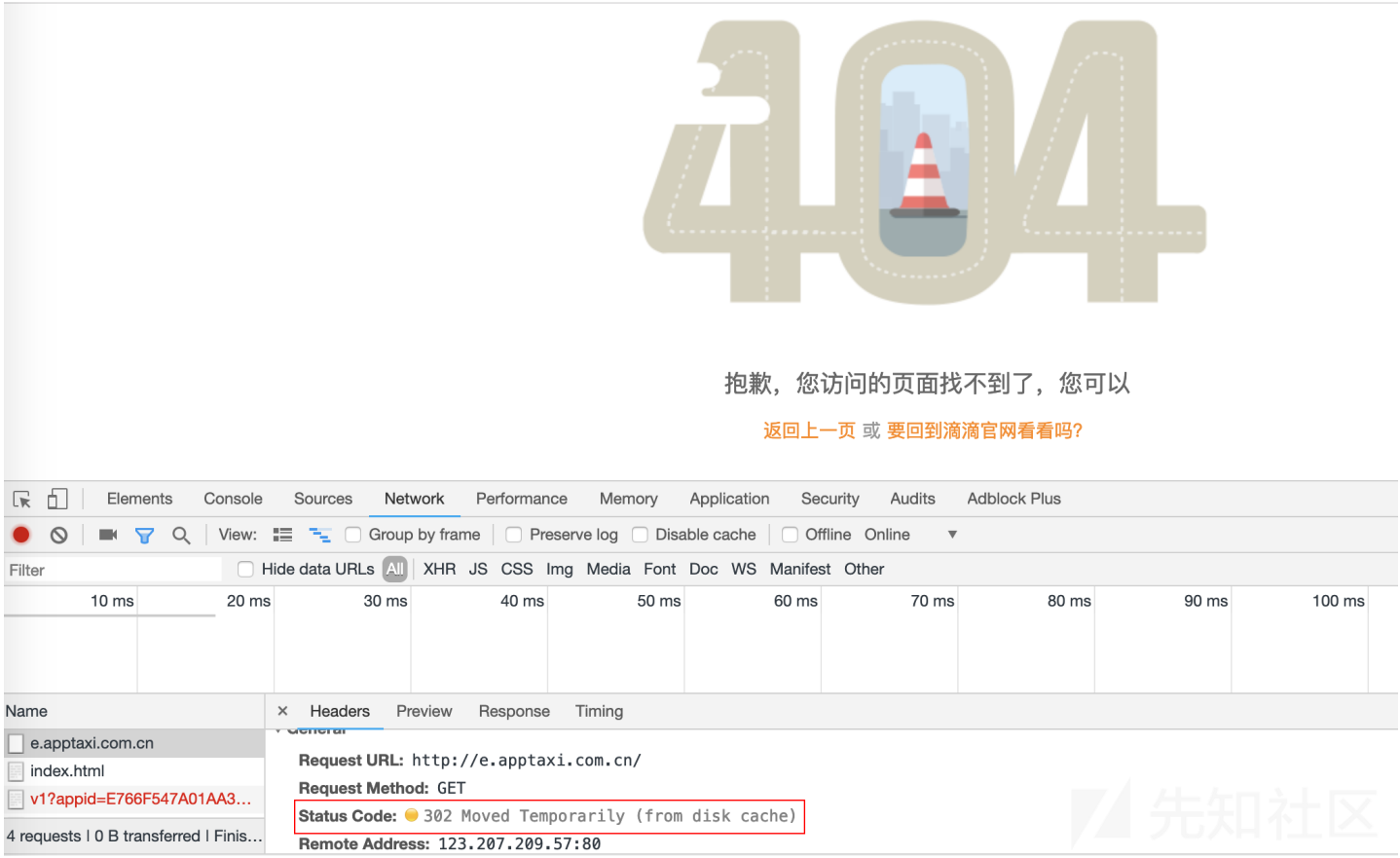
整体思路

```
class page_404:

    _instance = None

    def __init__(self):
        self._404_already_domain = []
        self._404_kb = []
        # ■■301■302■■■■■■■■200■404■■
        self._404_code_list = [200, 301, 302]

    def generate_404_kb(self, url):
        # ■■URL■■■■
        domain = url.get_domain()
        domain_path = url.get_domain_path()
        rand_file = rand_letters(8) + '.html'
        url_404 = domain_path.urljoin(rand_file)
        resp_200 = requests.get(domain_path)
        resp_404 = requests.get(url_404)
        # ■■■■■■■■■■■■■■■■■■■404■■■■■■■■■■
        if is_similar_page(resp_200, resp_404):
            pass
        else:
            self._404_already_domain.append(domain)
            self._404_kb.append((domain, resp_404))

    def set_check(self):
        self._404_kb = []
        self._404_checked = False

    def is_404(self, http_response):
        code = http_response.get_code()
        url = http_response.get_url()
        domain = url.get_domain()
        if domain not in self._404_already_domain:
            self.generate_404_kb(url)
        # ■■■■■■404■■■■
        if code == 404:
            return True
        if code in self._404_code_list:
            for domain_404, resp_404 in self._404_kb:
                # ■■■■■■■■
                if domain == domain_404:
                    if is_similar_page(http_response, resp_404):
                        return True
        return False


def is_404(http_response):
    if page_404._instance is None:
        page_404._instance = page_404()
    return page_404._instance.is_404(http_response)
```

荒废的域名

本来脚本过滤写到这里应该就差不多了，但是今天测试的时候还是发现了一个bug。当域名本身无效，即访问domain本身也会跳转到一个404页面，这时候情况就有点不一

比如直接访问http://e.apptaxi.com.cn，会发生302跳转到定义好的404页面https://img-ys011.didistatic.com/static/dfe_default_page/index.html，但是直接访问https




仔细看看上面■■404■■部分

```
resp_200 = requests.get(domain_path)
resp_404 = requests.get(url_404)
# ■■■■■■■■■■■■■■■■■■■■404■■■■■■■■■■
if is_similar_page(resp_200, resp_404):
   pass
```

我理所当然认为直接访问https://www.domain.com必定是一个200页面，即正常的页面。但是一般来说子域名挖掘机挖出来的不少子域名可能是无效的，或者是直接301、

这里我的思路是，假若resp_200本身就是一个404■■的话，那么它和随机生成的resp_404页面相似度radio可能大于0.95甚至相似度为1。所以这种情况分开做判断。

```
def generate_404_kb(self, url):
   # ■■URL■■■■
   domain = url.get_domain()               #www.freebuf.com
   domain_path = url.get_domain_path()     #https://www.freebuf.com
   rand_file = rand_letters(8) + '.html'
   url_404 = domain_path.urljoin(rand_file)
   resp_200 = requests.get(domain_path)
   resp_404 = requests.get(url_404)
   # ■■■■■■■■■■■■■■■■■■■■404■■■■■■■■■■
   if is_similar_page(resp_200, resp_404):
       # ■■■■■■■■1■■■■■■■■domain■■■■404■■
       if is_similar_page(resp_200, resp_404, 1):
           self._404_already_domain.append(domain)
           self._404_kb.append((domain, resp_200))
       pass
   else:
       self._404_already_domain.append(domain)
       self._404_kb.append((domain, resp_404))
```

后记

其实整个404页面识别过程中，我认为最难解决的是■■■■■判断这一个问题，这里简单的用response_body来计算hash并根据一个大概的radio来判断，然后根据具体网

另外文章中的代码是从自己的项目中截取出来，可能不能直接执行，但是大致已经提供了一个思路。师傅们有什么指点请问问指教。

参考《白帽子将web扫描》

上一篇：区块链安全——一则蜜罐DAPP欺骗手段分析 下一篇：JAVA代码审计-JEECG快速开...

1. 6 条回复


hpdoger 2019-03-18 16:06:13

这个hashes模块Pip下不能安装么？

0 回复Ta

---


Bojack 2019-03-18 16:10:59

@hpd**** 包名python-hashes，如果实在下载不了我也可以发你，你留个邮箱

0 回复Ta

---


Bojack 2019-03-18 16:30:21

页面相似度的计算还能参考 https://thief.one/2018/04/12/1/

0 回复Ta

---

zhangzhongnan 2019-03-19 15:07:10

非常巧合的是这篇帖子的ID是以404结尾的

2 回复Ta

---

张喵喵是小仙女 2019-04-11 15:35:26

小哥哥，源码可以发一下嘛～

0 回复Ta

---

Passer6y 2019-05-05 00:10:28

@hpdoger 踩了一遍坑，这个库在16年爆了个issue，到现在也没修... 然后有个人fork了他的仓库，帮他把洞修好了2333，安装:`pip install changanya`
导入：`from changanya.simhash import Simhash`

0 回复Ta

---

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板