西湖论剑 网络安全技能大赛 HANGZHOU CHINA

雷神众测

攻防对抗 跨界交流

2020西湖论剑大赛品质论坛·雷神众测HACKINGDAY

主办单位 | 杭州市公安局 | 共青团杭州市委 | 杭州市学生联合会

承办单位 | 安恒信息 | 杭州市网络安全研究所 | 杭州市网络安全协会

协办单位 | 安恒信息海特实验室 | 安恒信息雷神众测 | 安恒信息AiLPHA大数据实验室

雷神众测

渗透Webpack等站点从此更加优雅

演讲人：Poc Sir

关于我 À propos de moi

Poc Sir

Cool-Guy@C-est.Cool

- 雷神众测等其他平台专业打酱油白帽子、内部已知贡献选手
- Hack Inn « www.hackinn.com » 安全议题分享平台运营者
- 安恒信息非著名 Ctrl C + V 工程师、不资深漏洞复现研究员
- 目前一直在法学习， 疫情不推荐大家来找我思考美食、人参
- 安全圈的一个小学生、新人， 活跃于微博喜欢聊聊两岸三地

目录 Table des matières

雷神众测

前端打包器 说 你 好

前端打包器：一个将网站所有的前端静态资源打包至一个或多个JS文件内的工具。以便于开发者快速开发、减少HTTP请求数量、提升用户的体验感等 —— 沃·兹基·宗杰德《找不到官方定义》

Grunt

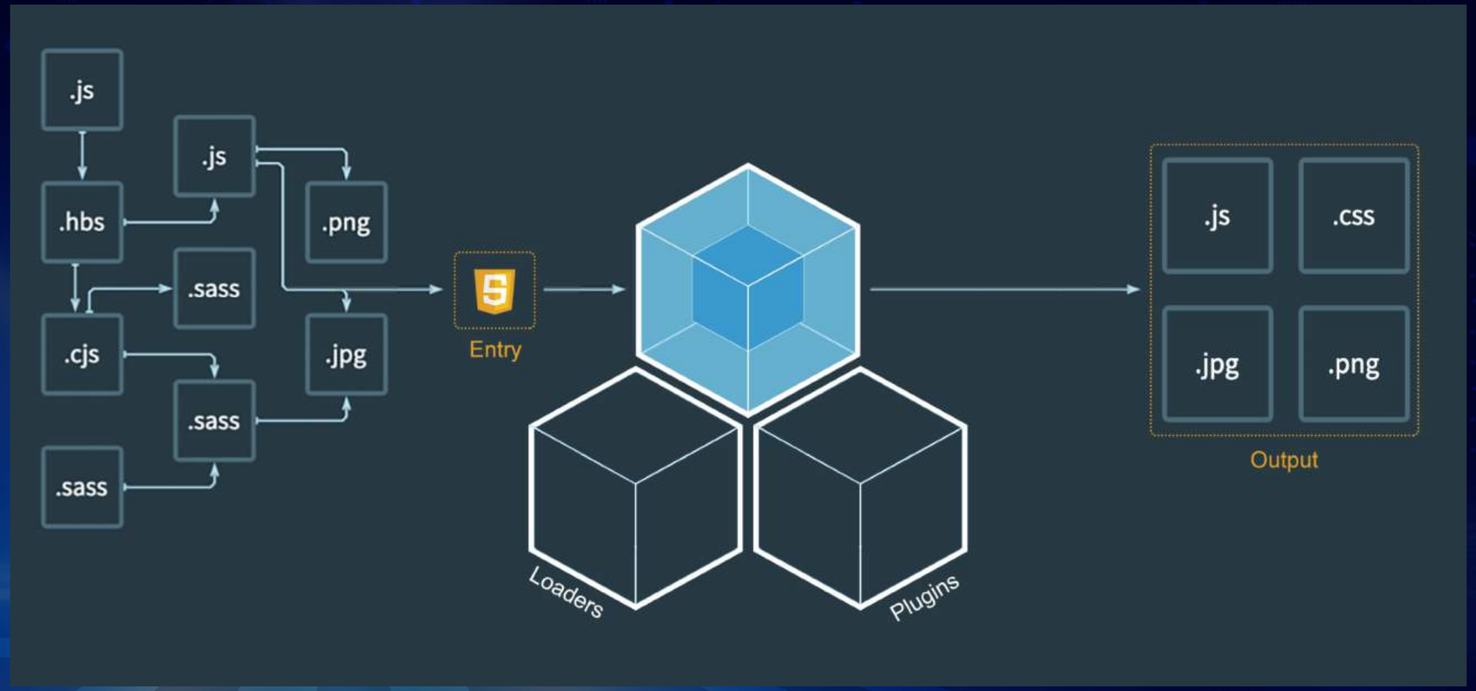
…

Rollup

…

Webpack

…

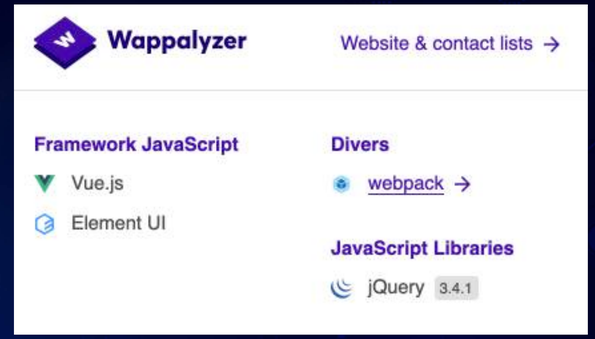


HTML也一并打包，浏览器端动态解析生成

如何快速识别


```
<!DOCTYPE html><html lang=en><head><meta charset=utf-8>
  <title>雷神众测</title></head>
  <body><noscript><strong>We're sorry but dasbounty
doesn't work properly without JavaScript enabled.
Please enable it to continue.</strong></noscript>
  <script src=/js/chunk-vendors.d1m05ef2.js></script>
  <script src=/js/app.d1m0h997.js></script></body>
</html>
```

webpackJsonp(XXXXXXX
window["**webpack**Jsonp"]



HTML源码速览

JS中找一找

借助浏览器插件

还有这种好事儿！

前端打包器：一个将所有网页元素、JS内容等打包在一起的工具。换句话说他把原来左一块右一块的**API以及其参数全部打包在了一起**~~以便于黑阔查看~~。

在？数万行的JS了解一下

开源公益项目

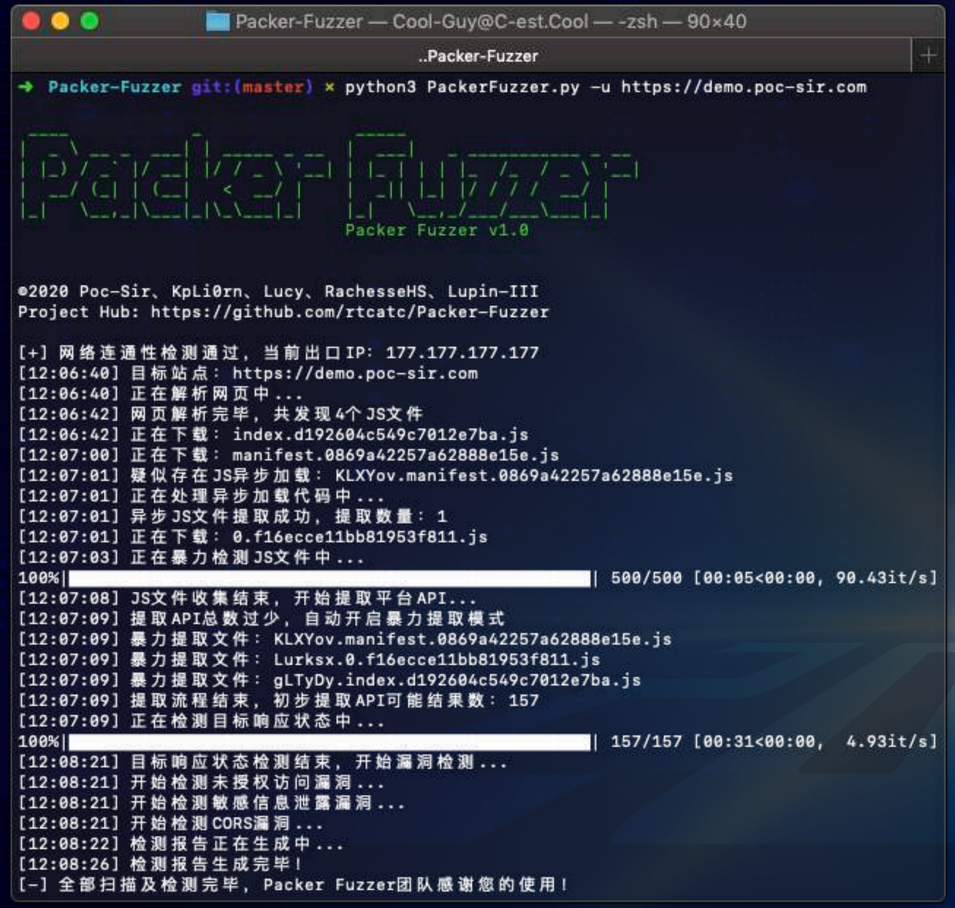
项目名称： **Packer Fuzzer**　版本号：v1.0

项目地址： https://github.com/rtcatc/Packer-Fuzzer

项目团队： Poc Sir、KpLi0rn、Lucy、RachesseHS、Lupin-III

Packer Fuzzer

雷神众测



项目介绍

一款针对Webpack等前端打包工具所构造的网站进行快速、高效安全检测的扫描工具。本工具支持自动提取对应目标站点API及其参数，并快速对其进行漏洞检测。自带Word及HTML扫描报告及5种主流语言翻译。

您的Star将是我们继续完善项目的巨大动力！

注：目前第一版只针对 Web Pack 规则做了优化

Word报告展示

雷神众测

编　号　PF-API-I77C2z

日　期　2020-10-01 12:17

demo.poc.sir.com.cn 平台 API 模糊检测报告

Version 1.0

2020 年 8 月

Poc-Sir、KpLi0rn、Lucy、RachesseHS、Lupin-III

■ 版权声明

■ 适用声明

本报告适用于一切使用本工具检测的目标系统，请妥善保管，未经目标系统所有者允许不得传出。

■ 免责声明

本报告为本工具根据使用者检测结果自动生成的报告，报告内容不代表本团队的立场及观点。由于传播、利用此工具提供的检测功能而造成的任何直接或者间接的后果及损失，均由使用者本人负责，本团队不为此承担任何责任。请在使用本工具时遵循使用者以及目标系统所在国当地的相关法律法规，一切未授权测试均是不被允许的。

■ 修订记录

版本号	修改日期	修改人	修改记录	批准人
1.0	2020/8/5	Lupin-III	无	Poc-Sir

一. 报告摘要

被扫描平台：demo.poc.sir.com.cn
输入参数值：https://demo.poc.sir.com.cn

本次扫描采用高级版扫描模式，使用 177.177.177.177 作为扫描 IP，共耗时 129 秒。

发起扫描时间：2020-10-01 12:15:40
扫描完成时间：2020-10-01 12:17:49

本次扫描共发现 3 个有效 API 接口。

共发现 4 个相关 JS 文件，分别是：
◆ https://demo.poc.sir.com.cn/static/js/index.d192604c549c7012e7ba.js
◆ https://demo.poc.sir.com.cn/static/js/manifest.0869a42257a62888e15e.js
◆ https://demo.poc.sir.com.cn/static/js/vendor.340bf150d19619148a1a.js
◆ https://demo.poc.sir.com.cn/static/js/0.f16ecce11bb81953f811.js

共发现 6 个安全漏洞，其中高危 0 个、中危 6 个、低危 0 个。分别是：
◆ 未授权访问漏洞：3 个
◆ 敏感信息泄露漏洞：3 个

附加 Cookies 信息为：未启用 Cookies 功能

附加传输头部信息为：未启用附加头部功能

经过本工具分析，目标平台的安全风险等级为：中风险

二. 漏洞详情

2.1 hello-demo 接口存在未授权访问漏洞（中危）

API 地址：https://demo.poc.sir.com.cn/thorsrc/hello-demo
关联 JS 地址：https://demo.poc.sir.com.cn/static/js/index.d192604c549c7012e7ba.js
响应内容：

```
{
    "code": "1",
    "msg": "hello-demo"
}
```

2.2 example.do 接口存在未授权访问漏洞（中危）

API 地址：https://demo.poc.sir.com.cn/thorsrc/example.do
关联 JS 地址：https://demo.poc.sir.com.cn/static/js/0.f16ecce11bb81953f811.js
响应内容：

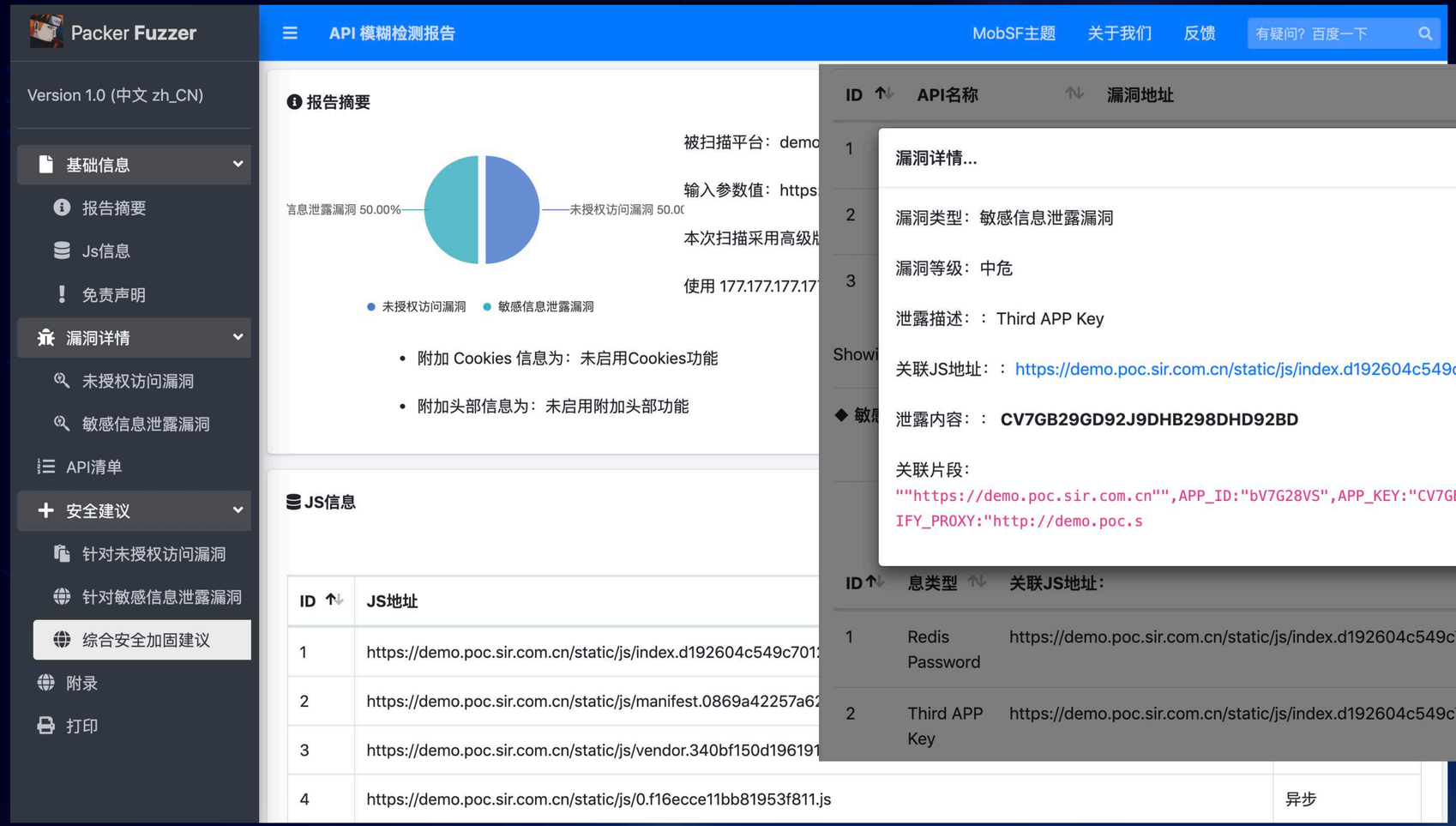
```
{
    "data": {
        "example": "G9F29F90UR02HGH029HN39",
        "flag": "1"
    },
    "message": "success",
    "status": "200"
}
```

2.3 shell 接口存在未授权访问漏洞（中危）

API 地址：https://demo.poc.sir.com.cn/thorsrc/shell
关联 JS 地址：https://demo.poc.sir.com.cn/static/js/0.f16ecce11bb81953f811.js
响应内容：

```
{
    "data": {
        "talk": "is cheap",
        "show": "me your shell",
        "success": "1"
    },
```

HTML报告展示



*我们使用了MobSF主题模板

一般JS文件提取


```
<!DOCTYPE html><html lang=en><head><meta charset=
  <title>雷神众测</title></head>
  <body><noscript><st
JavaScript enabled. P
  <script src=/js/chu
  <script src=/js/app
</html>
```

https://cdn.bootcdn.net/ajax/libs/jquery/3.5.1/jquery.js

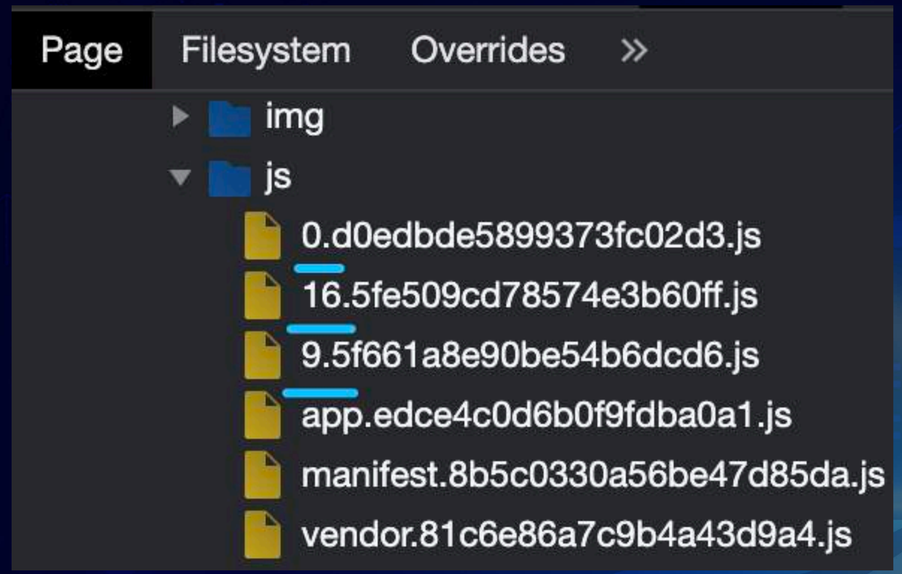
https://cdn.bootcdn.net/ajax/libs/jquery/3.5.1/jquery.min.js

https://cdn.bootcdn.net/ajax/libs/jquery/3.5.1/jquery.slim.js

https://cdn.bootcdn.net/ajax/libs/jquery/3.5.1/jquery.slim.min.js

```
<link rel=icon href=/favicon.ico><title>后台管理系统</title><link
css rel=stylesheet><link
=stylesheet></head><body><div id=app></div>
I.7dd3960f.js></script><script
></script><script>(function(c){function e(e)
,b=[];r<a.length;r++)d=a[r],f[d]&&b.push(f[d]
pe.hasOwnProperty.call(k,u)&&
shift()();return h.push.apply(h,t||
h.length;e++){for(var
=n[d];0!==f[a]&&(u=!1)}u&&(h.splice(e-
,d={runtime:0},f={runtime:0},h=[];function
-commons":"chunk-commons"}[c]||c)+"."+
f9b2158":"4c30b3b4","chunk-
:"f7ccb090","chunk-
:"4726d5db","chunk-
:"510ae4b6","chunk-
:"464cf4f3","chunk-
:"c36f7801","chunk-
:"caeedd10","chunk-
```

异步JS解析提取



只有 0、16、9
剩下的1、2、3、4、5... 哪里去了？？？？？？？

小小的眼睛，大大的疑惑.jpg

JS异步！JS按需加载！

雷神众测

异步JS解析提取

```
X.src = Z.p + "static/js/" + Y + "." + {
    0: "d1m0ecbdb81953f82a",
    1: "d1m0qe11bb81zz3f81"
}[Y] + ".js";
```

```
X.src = Z.p + "static/js/" + Y + "." + {
    "gre5g4e": "d1m0ecbdb81953f82a",
    "y5h1tr5": "d1m0qe11bb81zz3f81"
}[Y] + ".js";
```

```
> "static/js/" + 1 + "." + {
      0: "d1m0ecbdb81953f82a",
      1: "d1m0qe11bb81zz3f81"
  }[1] + ".js";
< "static/js/1.d1m0qe11bb81zz3f81.js"
> "static/js/" + "gre5g4e" + "." + {
      "gre5g4e": "d1m0ecbdb81953f82a",
      "y5h1tr5": "d1m0qe11bb81zz3f81"
  }["gre5g4e"] + ".js";
< "static/js/gre5g4e.d1m0ecbdb81953f82a.js"
>
```

异步JS解析提取


```
document.createElement(
"script");
```

通过判断JS文件中是否存在创建<script> dom 标签的代码，若无则必然是不存在JS异步加载的

```
re.compile(r"\w\.p\+(.*?)\.js",
re.DOTALL)
```

接着使用正则匹配出JS中的JS异步代码

异步JS解析提取


```
function js_compile(Y) {
  js_url= "XXXXXX" + Y + "." + {
      ......
  }[Y] + ".js";
return js_url}
```

在Python中调用Node.js运行拼接函数提取出异步JS的名称

```
if "exec" not in jsCode and "spawn" not in jsCode:
    jsCompileResult = execjs.compile(jsCodeFunc)
    for name in nameList:
        if "\"" in name:
            name = name.replace("\"", "")
```

别忘了初步过滤一些可被执行的函数名称（虽然应该遇不到）

异步JS爆破提取

1 . VZ777FZBIBF7777 . JS
2 . VZ777FZBIBF7777 . JS
3 . VZ777FZBIBF7777 . JS
...
N . VZ777FZBIBF7777 . JS

可猜测纯数字+固定内容+扩展名

平台API规则提取


```
var K = {
    postFile: F,
    postJSON: G,
    post: H,
    get: z,
    delete: q,
    patch: U
}
  , Z = function(e) {
    return K.get("/portal/emailcode", e)
}
  , Y = function(e) {
    return K.post("/portal/login", e)
}
  , Q = function(e) {
    return K.post("/portal/register", e)
}
```

```
self.regxs = [r'\w\.get\(\"(.*?)\"\,',
              r'\w\.post\(\"(.*?)\"\,',
              r'\w\.post\(\"(.*?)\"',
              r'\w\.get\(\"(.*?)\"',
              r'\w\+\"(.*?)\"\,']
```

```
[blacklist]
apiExts = *,+,=,{,},[,],(,),<,>,@,#,",',@,:,?,!,
,^,\,.docx,.xlsx,.jpeg,.jpg,.bmp,.png,.svg,.vue,
.js,.doc,.ppt,.pptx,.mp3,.png,.doc,.pptx,.xls,.m
p4
```

平台API暴力提取

示例	说明
/user/login	这是一个API
api/backdoor	这也是一个API
/	这个明显不是
a	这个明显也不是
/video/demo.mp4	后缀直接pass
/_&§:./+=	你是何方妖孽？
/bushishell.do	谁还不是个API了
/v1/test?dev=1	把"?"去了也是API
/安恒雷神众测	没人想不开用中文

通过右边表我们不难得出一个过滤非API的方式：首先API不能为空或者单纯“/”，其次API内不能存在有类似“*、§、&、=、+”等的怪异字符并且在正常情况下不会含有中文，另外他将不是以“mp4、mp3、ppt”等扩展名做结尾的，若提取出内容中有“?”则需把其后面内容一并去除，而且API长度不会过短（一个字符），排除了这些之后剩下的均是符合情理的。

我大不了用正则提取所有JS中的变量值

之后我按照规则过滤就是了

如何暴力提取👉？

API完整路径拼接

当然我们无需在此多花功夫，因为正常的网站一般只拥有一个BaseURL，并且只要打开目标站点在网站中触发任意一个API便可以快速的找出其对应的BaseURL。

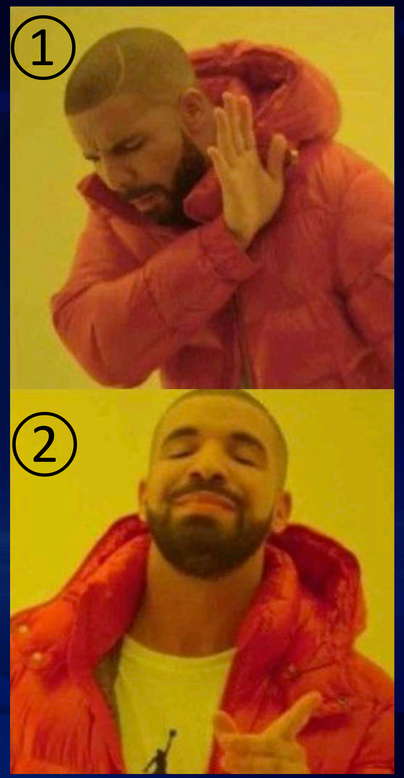
1. 域名 + API路径

https://woshipinyin.cn + /api/hello

2. 域名 + **BaseURL** + API路径

*https://woshipinyin.cn + **/demo** + /api/hello*

```
self.baseUrlRegxs = [r'url.?\s?\:\s?\"(.*?)\"',
                     r'url.?\s?\+\s?\"(.*?)\"',
                     r'url.?\s?\=\s?\"(.*?)\"',
                     r'host\s?\:\s?\"(.*?)\"', ]
```



Ta在、Ta不在

雷神众测

两个报错

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

[redacted] 16:50:23 CST 2020
There was an unexpected error (type=Method Not Allowed, status=405).
Request method 'GET' not supported

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Oct 1 16:51:55 CST 2020
There was an unexpected error (type=Unsupported Media Type, status=415).
Content type 'application/x-www-form-urlencoded;charset=UTF-8' not supported

两个解决方案

```
→  burp python3 魔镜啊魔镜，我该怎么办呢？.py
换POST请求！
再问蓝屏给你看！
→  burp
```

请求	Content-Type	说明
a=1	application/x-www-form-urlencoded	最常见的普通请求内容
{"a": 1}	application/json	JSON格式请求
<a>1	application/xml	XML格式请求

API参数提取之路

雷神众测

JS美化

打包器所生成的JS文件都是经过高度压缩的，所有的代码都缩在了一起。我们需要对其做一个简单的格式化，使这些脚本尽量的分离开来，以便后续分析操作。

找寻关联片段

在所有美化之后的JS之内搜索每个对应API的名称，并提取API名称出现处上下5行的关联内容（共11行），以便于后期分析使用。每个片段互相独立，储存于数组之中。

判断请求类型

之后我们需要判断每个片段是否存在“GET”、“POST”字符串，若存在则将会把当前片段归为对应的请求方式待提取文本，若这两个字符串不存在则默认归为“POST”请求。

正则提取参数

```
regxs_1 = r'method\:.*?
\,url\:.*?\,data\:
({.*?})'
regx_key = r'(.*?)\:.*?
\,|(.*?)\:.*?\}'
regx_value = r'\:
(.*?)\,|\:(.*?)\}'
```

随后使用已知正则规则在每个片段内提取对应的API参数内容并保存（图为规则一）

API参数提取之路

雷神众测

再次暴力提取

```
violent_regx = r'(?isu)"([^"]+)'
```

随后若对应片段内无法用已知规则匹配出参数内容，则使用暴力提取正则对参数进行提取操作。

判断参数类型

参数可以分为文本参数和字符参数，而我们可以使用规则库来进行简单的判定：

```
[FuzzerParam]
param = success,post,get
default = id,num,number,code,type
```

参数内容生成

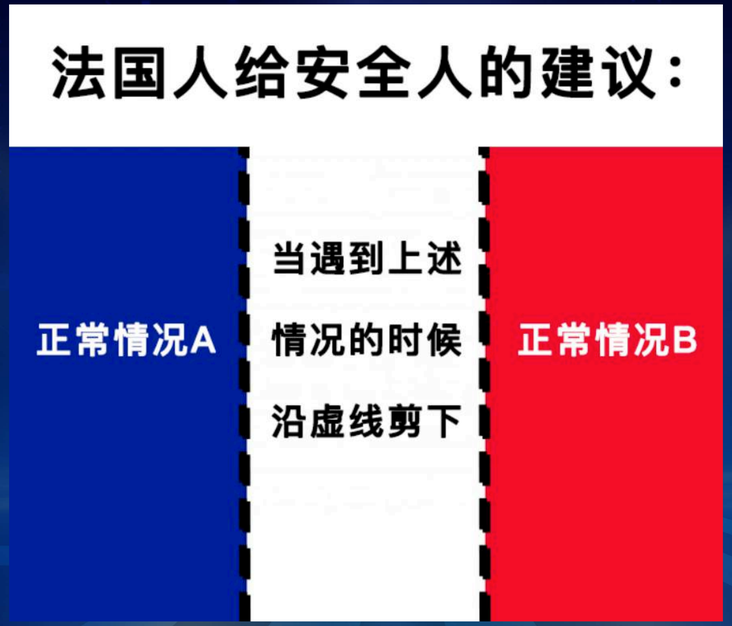
1. 对于文本型参数我们随机生成三位字符，2. 对于数字型参数我们随机生成三位数字，3. 对于已有的固定参数我们采取保留处理。

还有很多不足

至此现阶段的API及API参数提取思路全部结束。可以看到我们的思路目前还存在许多不足之处，若各位观众朋友们对此有高见及其他想法还愿提出！

对于一些情况

当脚本分的很开，混淆、加密很严重等等的情况则么办？？？



路漫漫 其修远 兮。。。

7大漏洞检测

敏感信息泄露漏洞

不测GIT，不测SVN，我们专注于JS内的泄露

```
rBKV: function(e, t, i) {
    "use strict";
    e.exports = {
        NODE_ENV: '"production"',
        PROXY: "http://XX.XX.XX.XX:80",
        无内鬼来个超管密码: "T1-10r__SRC",
        PORT: "5277",
        REDIS_HOST: "XX.XXX.XXX.XXX",
        REDIS_PORT: "9427",
```

祖传系统为何有着 Secret Key 泄露？是什么让开发小王留下了测试TOKEN？管理员密码究竟是JS自己留下的还是其他人留的？系统频频惨遭毒手是疏忽大意，还是服务器没拔掉网线？

让我们随着本页PPT走入敏感信息泄露漏洞 ...

思路：把一些特定的JS变量名对应的变量内容提取出来

比如： password、 PRIVATE-KEY、 APPSecret ...

未授权访问漏洞

API返回内容	说明
{"msg":"hello","code":"200"}	可未授权访问的API，只是不能进行敏感操作或无敏感内容
{"msg":"","errcode":"1","log":"您还没有登录！"}	不存在未授权访问漏洞
{"status":"200","message":"success","data":{"id":"1","user":"admin","pass":"1143720b05b5daf6f2bb83f9e4a9b5ba"}}	存在未授权访问漏洞，可以进行进一步测试

人工~~ZHI ZHANG~~语义（情感）分析（两类）：

未登录,请登录,权限鉴定失败,未授权,鉴权失败,unauth,状态失效,没有登录,会话超时,token???,login_failure

系统繁忙,系统错误,系统异常,服务器繁忙,参数错误,异常错误,服务端发生异常,服务端异常

CORS跨域漏洞

请求包：

Origin头内： https:// + 目标站网址 + .example.org + / +目标站网址

例如： Origin: https://lei-god666.com.example.org/lei-god666.com

返回包：

Access-Control-Allow-Credentials头内： 是否为true

Access-Control-Allow-Origin头内： 是否存在example.org字样

SQL注入漏洞

雷神众测

报错注入

震惊！一个单引号居然
引发了这样的惨案！！

You have an error in your SQL syntax
Oracle Text error
Microsoft SQL Server

布尔型盲注

and 1=1
and 1=2

返回包长度正常
返回包长度变小
（参考正常包）

基于时间延迟注入

~~NULL~~
and sleep(10)

正常包响应时间大于0秒，
但一般情况下小于 10 秒；
延迟包响应大于必定10秒。

水平越权漏洞

xx=1 xx=2 xx=3 xx=4 xx=5 五个包，发送五次请求，检测五次

```
get_repeat_nums = dict(Counter(get_all_list))
get_repeat_num = int("".join([str(key) for key, value in
get_repeat_nums.items() if value > 1]))  # 获取到我们重复的数据
    if len(get_select_list) >=3:
        ...
```

5个请求中存在若**3个及以上**的返回包内容长度不一致，则认定此API存在水平越权漏洞

虽然说可能会导致误报，但使用此方式**检测起来非常的快速并且基本不会错杀**，之后再对其进行进一步人工检测即可

弱口令漏洞

哪些API我们可以测弱口令：login.do, signin, login, user, admin ...

哪些是用户名参数：userCode, username, name, user, nickname ...

哪些是密码参数：userPass, password, pass, code, mima , token ...

好的 爆破就完事儿了

论如何从返回包中判断是否成功：登录成功, login success, 密码正确, 成功登录 ...

任意文件上传漏洞

1. 文件名使用“数字” + “.” + “特定扩展名”
2. 文件内容使用“PNG图片文件头” + 随机内容

```
files = {"file": (
                        "{}.{}".format(random.randint(1,100), ext),
(b"\x89\x50\x4E\x47\x0D\x0A\x1A\x0A\x00\x00\x00\x0D\x49\x48\xD7"+rands))}
```

关于扩展名，一堆现成名单：

```
ext_fuzz = tqdm(['asp;.jpg', 'asp.jpg', 'asp;jpg', 'asp/1.jpg', 'asp{}.jpg'.format(quote('%00')), 'asp .jpg',
            'asp_.jpg', 'asa', 'cer', 'cdx', 'ashx', 'asmx', 'xml', 'htr', 'asax', 'asaspp', 'asp;+2.jpg',
            'asp;.jpg', 'asp.jpg', 'asp;jpg', 'asp/1.jpg', 'asp{}.jpg'.format(quote('%00')), 'asp .jpg',
            'asp_.jpg', 'asa', 'cer', 'cdx', 'ashx', 'asmx', 'xml', 'htr', 'asax', 'asaspp', 'asp;+2.jpg',
            'asPx',
            'aspx .jpg', 'aspx_.jpg', 'aspx;+2.jpg', 'asaspxpx', 'php1', 'php2', 'php3', 'php4', 'php5', 'pHp',
            'php .jpg', 'php_.jpg', 'php.jpg', 'php.  .jpg',
            'jpg/.php', 'php.123', 'jpg/php', 'jpg/1.php', 'jpg{}.php'.format(quote('%00')),
            'php{}.jpg'.format(quote('%00')),
            'php:1.jpg', 'php::$DATA', 'php::$DATA.....', 'ph\np', '.jsp.jpg.jsp', 'jspa', 'jsps', 'jspx',
            'jspf',
            'jsp .jpg', 'jsp_.jpg'])
```

任意文件上传漏洞

哪些API我们可以测文件上传：upload, file, doc, pic, update ...

对于返回包是否上传成功我们要使用先黑名单过滤再白名单保留

白名单：上传成功, php, asp, html, jsp, success, 200, 成功, 已上传

黑名单：上传失败, 不允许, 不合法, 非法, 禁止, Fail, 失败, 错误

为什么要这样？

比如返回包为：{“code”：200，“msg”：“shell.php 文件不允许上传” }

雷神众测

谢谢 谢邀！

Merci à vous!