API接口深度发现的动态爬虫实现(3. Web框架识别和移除鉴权)

原创 扫到漏洞的 李姐姐的扫描器 2025年05月06日 18:55 北京

现代Web应用中，SPA（Single Page Application）占比非常高。它接口丰富，容易发现漏洞，应该被优先关注和重点测试。

笔者认为漏扫动态爬虫有必要精准识别站点采用的Web框架，好处有几点：

1）直接提取得到完整的path

一种很常见的情况，安全测试人员打开站点，只能看到一个简单的登录页，其他页面都鉴权了（**请注意，这仅仅是前端拦截的**）。 通过识别框架，我们在简单登录页上，能轻松枚举出所有的path。

万一有功能页面鉴权属性配置错误，则可能发现漏洞。

2）直接修改auth相关属性的值，去掉前端拦截，触发接口请求

SPA的router组件通常配置特定的auth属性，来标记页面是否需要鉴权。

既然只是一个前端标记，直接查找到这个属性修改为不鉴权即可。

我们的扫描器可以修改auth属性，再导航到这些功能页面上，通过简单的交互（点击、下拉、输入、滑动等），有一定概率触发多个API接口请求。运气好时，甚至直接就捡到了未授权访问漏洞。

下次遇到面试官问你，看到一个没有帐号的登录页，你应该如何渗透测试时，记得介绍这样的测试技巧：

识别Web框架，枚举Path，前端一键移除鉴权，逐个页面遍历，交互触发API请求

识别Vue

第一步，检查全局变量

```
typeof window.__VUE__ !== 'undefined' ||
typeof window.Vue !== 'undefined'
```

如果第一步失败，继续检查DOM 元素上的 __vue__ 或 __vue_app__，

通常是一个div元素

```
const vueElements = document.querySelectorAll('*');
for (const el of vueElements) {
if (el.__vue__ || el.__vue_app__) {
  return 'vue';
}}
```

一旦识别目标网站采用了vue js，**接下来就需要拿到$router对象。**

根据vue版本的不同，最最常见的$router变量是

```
window.$router
```

```
divObject.__vue_app__.config.globalProperties.$router

divObject.__vue_app__.$router
```

其中divObject是页面中第一个div元素。

如果都没能找到，可以考虑递归查找window对象的属性直到发现**$router**，但这通常会导致非常严重的性能问题。

```
function recursiveFindRouterWithPush(obj, depth = 0, maxDepth = 5) {
if (depth > maxDepth) {
return null;
        }

for (const key in obj) {
try {
if (key === '$router' && obj[key] !== null && typeof obj[key] === 'object' &&
return obj[key];
                        }
                } catch (error) {

                }

try {
if (typeof obj[key] === 'object' && obj[key] !== null) {
const foundInSubObject = recursiveFindRouterWithPush(obj[key], depth + 1, max
if (foundInSubObject) {
return foundInSubObject;
                                }
                        }
                } catch (error) {

                }
        }

return null;
}
```

拿到router之后，存在2种语法拿routes

```
$router.getRoutes()
$router.options.routes
```

切记，尽可能通过getRoutes()方法来拿，拿到的path数量往往更多。

现在来修改所有的鉴权属性为false

```
foundRouter.getRoutes().forEach(route => {
    for (let key in route.meta) {
        if (route.meta.hasOwnProperty(key) && key.includes('auth') && route.me
            route.meta[key] = false;
        }
    }
})
```

此时，当你再访问原先需要鉴权的页面时，会发现页面已经不再跳转了。

一些API接口请求也开始出现，但大部分API接口，通常不会响应未登录用户的请求。你可以在web console导航到部分原先鉴权的页面，注意，不要在浏览器地址栏输入（会导致js context被销毁）

```
foundRouter.push('/some/secret/page/you/can/not/access')
```

识别Next.js React

优先识别Next.js， 采用如下方法

```
// Next.js, method 1: 检查 Next.js 的全局变量
```

```
if (typeof window.__NEXT_DATA__ !== 'undefined') {
return 'nextjs'; // Next.js 是基于 React 的，应该优先判断
}
// Next.js, method 2: 检查特定的 meta 和 script 标签
if (!!document.querySelector('meta[name="next-head"]')){
return 'nextjs';
};
const scripts = Array.from(document.querySelectorAll('script[src^="/_next/"]
if (scripts.length > 0) {
return 'nextjs';
}
```

若未识别Next.js，继续识别React。 大型应用使用React的比例极高，比如我们常用的chatgpt deepseek grock，都使用了React框架

```
// React, method 1: 检查 React 相关的全局变量
if (typeof window.__REACT__ !== 'undefined' || typeof window.React !== 'unde
return 'react';
}

// React, method 2: 检查 DOM 元素上的 React 属性
const rootElement = document.getElementById('root') || document.querySelecto
if (rootElement) {
for (const key in rootElement) {
if (key.startsWith('__react') || key.startsWith('__reactFiber$')) {
return 'react';
                }
        }
}
```

识别Angular

Google的应用使用Angular框架不少，简单识别方法如下

```
if (window.angular || window.ng) {
return 'angular';
}

if (document && document.querySelector('[ng-version]')) {
return 'angular';
}
```

本篇简单介绍了Vue Next.js React Angular的框架识别，枚举Router中登记的所有path，以及简单的鉴权属性移除技巧，通过界面交互，有一定几率发现隐蔽的API接口。