API接口深度发现的动态爬虫实现(4. 接口遗漏缺陷分析)

原创 扫到漏洞的 李姐姐的扫描器 2025年05月12日 16:48 北京

API接口对于现代Web漏扫至关重要，有时候，1-2个隐蔽接口，就能决定扫描任务的成败。笔者做了大量实验，想要定位清楚，是什么原因导致crawlergo radium项目产生API接口遗漏问题。

事件函数的触发执行

加载完成一个Web页面之后，程序需要自动化触发，执行该页面上绑定的所有事件函数。

CrawlerGo是通过override **Element.prototype.addEventListener** 方法，来收集全部的DOM事件。收集完成后，再统一地触发执行，相关代码如下

```
for (let node of nodes) {
    let loop = 0;
    let event_name_list = node.getAttribute("sec_auto_dom2_event_flag").split
    let event_name_set = new Set(event_name_list);
    event_name_list = [...event_name_set];
    for (let event_name of event_name_list) {
        let evt = document.createEvent('CustomEvent');
        evt.initCustomEvent(event_name, true, true, null);

        if (event_name == "click" || event_name == "focus" || event_name == '
            transmit_child(node, evt, loop);
        }
        if ((node.className && node.className.includes("close")) || (node.id
            continue;
        }

        try {
            node.dispatchEvent(evt);
        } catch (e) {}
    }
}
```

经笔者测试，**在一个for循环中不停node.dispatchEvent，会出现丢事件的现象。**举例说明，在笔者选定的测试站点中，遍历click菜单中的item，却出现只打开了最后一个。其他导航请求并没有正确被拦截到。

这自然是因为，某一类事件是依赖全局状态的，连续触发就等于触发最后1个。但笔者想要收集更全的事件。因此增加适当的延迟，如10ms

```
node.dispatchEvent(evt);
await new Promise(resolve => setTimeout(resolve, 10));
```

经测试，少量延迟可以解决这类问题，收集全我们需要的URL。

动态加载未触发的严重缺陷

上面的缺陷或许不算太严重，丢几个事件，运气好的话，其他逻辑还有机会补回来数据。

但下面我要介绍的这个缺陷，是最为致命的。**CrawlerGo在单个页面上只触发1次DOM事件。**看笔者在测试页打印的log

```
log: Found 60 nodes to dispatch events
log: After trigger all events, found 174 nodes to dispatch events
```

起初，程序收集到60个要触发事件的元素。

等程序把这60个元素上的dom事件都触发后，绑定事件的element已经增加到了174个。

出现这个现象的原因，是因为一些元素是动态添加的，甚至是动态加载的。

CrawlerGo没有收集到这些新增的绑定事件，没有将其触发。因此丢掉了相当比例的接口。

这些接口，一般是在Modal对话框上出现。

例如，笔者的测试页面中，就打开了多达4个Dialog。这些接口都被漏掉了。

解决方法也很简单，持续收集新增出现的绑定事件，一直触发，直到没有新的事件绑定出现再结束当前页面的处理。

小结

本篇总结了常见动态爬虫收集不全API接口的原因。

因现代web页面存在较多动态加载的元素、功能，因此，爬虫需要持续收集事件绑定，并且完整地触发所有事件。

尽可能多地收集完所有API接口