Chrome调试技巧之js hook

1. js hook简介  
   js hook同frida hook使用思想基本一致，主要完成js代码调试中的加密位置定位、关键参数寻找等，可以用来验证一些猜想，并且大多时候，可以无需使用fiddler的autoresponser，直接使用js hook，在chrome中完成调试/或者对代码的修改。其主要使用**Object.defineProperty()** 方法，该方法会直接在一个对象上定义一个新属性，或者修改一个对象的现有属性，并返回此对象。  
     
   其中参数object为要被hook的对象实例，注：这里只能hook对象中的方法或者属性，如class，{}等，而不能hook一个函数---hook一个函数直接覆盖就可。  
   Prop为该函数中要被修改的属性名称，当然其既可以是一个属性，也可以是一个方法，或者也可以对该对象新增一个原来该对象不存在的属性或者方法，所以prop参数是一个很灵活的参数，尤其是：对于有的 值属性，可以在descriptor附加一个set、get方法，用来判断该属性值的传入内容，和debugger配合起来，灵活的下断，从而定位加密点。  
   descriptor为一个对象{}，其中最常用的参数  
   key为value---用来表示该属性要被重新赋予的值，可以是任何有效的 JavaScript 值（数值，对象，函数等）。如果没有被赋予则表示不改变原值；  
   set---当该属性被修改时，会调用这个函数，给其传入一个参数args，也就是该属性被赋予的新值，并且默认传入一个this对象，即该属性，set函数需要有返回值；  
   get---当该属性被访问时，会调用此函数，不传入任何参数，默认传入this对象，该函数的返回值被用作get到的属性值。可以用this.变量名引用当前属性值。  
   详细的案例介绍见：<https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty>
2. 简单的应用  
   

当前内存中有一个实例化的对象my\_hook，我们hook该对象下的test方法，通过value属性重新给test指定（覆盖）一个新的方法，然后后续在调用my\_hook的test方法时，调用的就是新覆盖的方法了。

1. 给hook对象新增属性/方法



当Object.defineProperty第二个参数为prop为以前对象没有的新方法时，就会给该对象新增方法/属性，当value的值为属性时，就新增属性，为方法时，就新增方法。

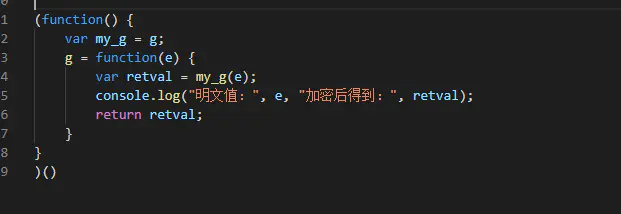
1. 带参数的hook



当hook覆盖某一函数时，新定义的函数中无需指定接受的形参，系统会默认将原函数的参数，以一个arguments数组的形式传入，我们只需要对arguments进行索引就可以取到传入的参数了，可以使用arguments.length获取该数组的长度，用于循环处理。

注：在js hook中应该是有和frida hook对应的this.xxx语法，用来引用hook之前的函数或者值。当然在本例中，他是先把hook前的函数，赋值给了另外一个函数，然后直接调用的这个新函数\_test。

1. Hook非对象属性的普通函数



对于非对象的属性我们就不用那么麻烦的依赖object.defineproperty去hook了，直接原地覆盖即可，只不过覆盖前需要做一些相应的处理。

首先把原函数赋值给另一个函数，以便二次调用计算原来的返回值，然后直接给 函数名赋值新的函数，---注意：此时需要填入相应的形参，而不是依赖于arguments进行了，最后需保证和原来一样有一个return即可。

1. Set方法的应用



当document中新增的cookie str中包含固定字符串时，就执行断点，然后断点执行完毕后，返回该返回的cookie值。



当给window中新增（修改）\_$ss属性时，就断下来，打印该属性，然后断点执行完毕后，返回该新设置的值，完成值的设置。

Cookie之所以要新增一个包含固定字符串的判断是因为，cookie经常被修改，这样会断很多次，而我们只想让他断在固定cookie块被设置的时候，因此这么写，而\_$ss属性只设置一次，我们就想断这一次，所以无需有额外的条件。

1. chrome hook的标准写法

我们一般使用一个立即执行的匿名函数完成hook替换，hook普通函数格式如下

(function() {

var new\_function\_name = ori\_function\_name;

ori\_function\_name = function(canshu,xx,xx){

var retval = ori\_function\_name(canshu,xx,xx);

console.log(retval);

return retval;

}

})()

Hook对象时采用如下格式

(function() {

'use strict';

Object\_name .new\_function\_name = Object\_name .ori\_function\_name;

Object.defineProperty(

Object\_name, object\_shuxing\_str, {

value/set: function(如果确定参数个数，也可以指定){

var retval = Object\_name .new\_function\_name(arguments[0]);

Debugger;

Return retval;

}

}

);

})()

注意：js hook一旦刷新网页就清除掉了，因此需要在要hook的代码调用前，网页内容加载后的时间点下一个断点，在断下来的时候在console中执行js hook，等hook好了，在过掉这个断点。

还有另一种方法，就是在网页回车键加载前打开chrome开发者工具，然后在sources中点击右上方的||按钮，让其变为debugger模式，此时就不会执行js，然后回车加载网页，也不会执行新的网页的js，然后赶紧console中执行hook代码，在点击原来位置，过掉这个断点即可。