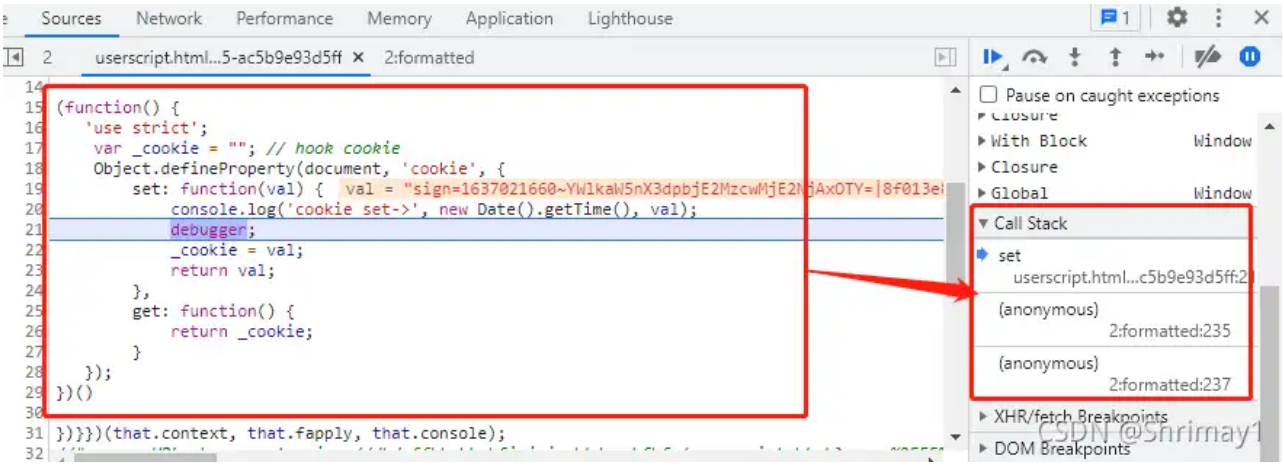
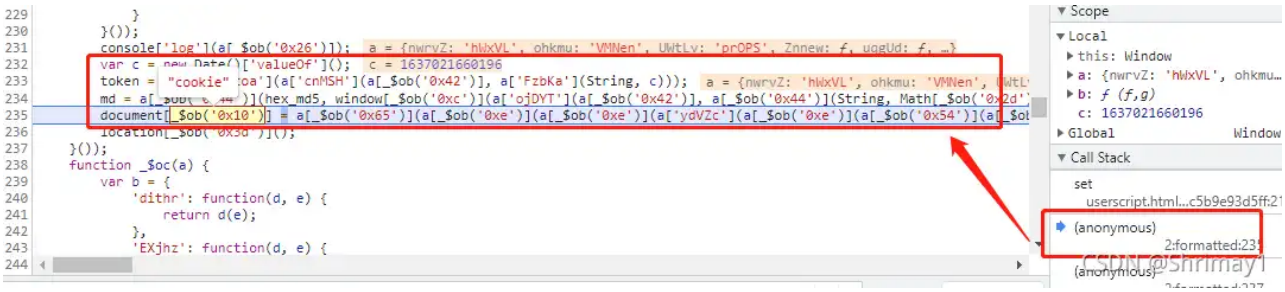
一、定位加密：

油猴添加hook_cookie的脚本，记得@match改为对应的网址，清掉网页cookie然后刷新页面，即可定位到cookie生成位置，接着看执行调用栈扣js



hook到cookie之后向上找堆栈；



二、解决方案如下：

将原有的ob混淆代码解混淆一下，翻译成可读的js代码，看到是一个MD5加密，通过对比是一个标准的MD5加密，可以直接使用crypto-js模块，也可以直接硬

CryptoJS.MD5加密函数；

```
// (function() {return !![];}['constructor']('debugger'))['call']('action')

// 过 debugger, hook的方式
// 学过原型链的应该清楚
//      已知:
//      1. Function.constructor = Function
//      2. 所有的函数定义，实际上都是 new Function
//      3. 也就是说，函数实际上是 Function 的实例化对象
//      4. 那么函数的constructor，实际上就是 Function.prototype.constructor
//      5. 所以 Function.prototype.constructor = Function
//      6. 所以只要修改 Function.prototype.constructor，就可以实现 hook 自定义函数的 constructor 目的
```

```
12
13  // 所以应该这样写:
14  /*
15      _Function = Function
16      Function.prototype. = function(){
17          if (arguments[0].indexOf('debugger') != -1){
18                  return _Function('')
19              }
20          return _Function(arguments[0])
21      }
22  */
23  // ====================================================业务代码
    ============================================================
24  var hexcase = 0;  /* hex output format. 0 - lowercase; 1 - uppercase        */
25  var b64pad  = ""; /* base-64 pad character. "=" for strict RFC compliance   */
26  var chrsz   = 8;  /* bits per input character. 8 - ASCII; 16 - Unicode      */
27
28  /*
29   * These are the functions you'll usually want to call
30   * They take string arguments and return either hex or base-64 encoded strings
31   */
32  function hex_md5(s){ return binl2hex(core_md5(str2binl(s), s.length * chrsz));}
33  function b64_md5(s){ return binl2b64(core_md5(str2binl(s), s.length * chrsz));}
34  function str_md5(s){ return binl2str(core_md5(str2binl(s), s.length * chrsz));}
35  function hex_hmac_md5(key, data) { return binl2hex(core_hmac_md5(key, data)); }
36  function b64_hmac_md5(key, data) { return binl2b64(core_hmac_md5(key, data)); }
37  function str_hmac_md5(key, data) { return binl2str(core_hmac_md5(key, data)); }
38
39  /*
40   * Perform a simple self-test to see if the VM is working
41   */
42  function md5_vm_test()
43  {
44    return hex_md5("abc") == "900150983cd24fb0d6963f7d28e17f72";
45  }
46
47  /*
48   * Calculate the MD5 of an array of little-endian words, and a bit length
49   */
50  function core_md5(x, len)
```

```
{
  /* append padding */
  x[len >> 5] |= 0x80 << ((len) % 32);
  x[(((len + 64) >>> 9) << 4) + 14] = len;

  var a =  1732584193;
  var b = -271733879;
  var c = -1732584194;
  var d =  271733878;

  for(var i = 0; i < x.length; i += 16)
  {
    var olda = a;
    var oldb = b;
    var oldc = c;
    var oldd = d;

    a = md5_ff(a, b, c, d, x[i+ 0], 7 , -680876936);
    d = md5_ff(d, a, b, c, x[i+ 1], 12, -389564586);
    c = md5_ff(c, d, a, b, x[i+ 2], 17,  606105819);
    b = md5_ff(b, c, d, a, x[i+ 3], 22, -1044525330);
    a = md5_ff(a, b, c, d, x[i+ 4], 7 , -176418897);
    d = md5_ff(d, a, b, c, x[i+ 5], 12,  1200080426);
    c = md5_ff(c, d, a, b, x[i+ 6], 17, -1473231341);
    b = md5_ff(b, c, d, a, x[i+ 7], 22, -45705983);
    a = md5_ff(a, b, c, d, x[i+ 8], 7 ,  1770035416);
    d = md5_ff(d, a, b, c, x[i+ 9], 12, -1958414417);
    c = md5_ff(c, d, a, b, x[i+10], 17, -42063);
    b = md5_ff(b, c, d, a, x[i+11], 22, -1990404162);
    a = md5_ff(a, b, c, d, x[i+12], 7 ,  1804603682);
    d = md5_ff(d, a, b, c, x[i+13], 12, -40341101);
    c = md5_ff(c, d, a, b, x[i+14], 17, -1502002290);
    b = md5_ff(b, c, d, a, x[i+15], 22,  1236535329);

    a = md5_gg(a, b, c, d, x[i+ 1], 5 , -165796510);
    d = md5_gg(d, a, b, c, x[i+ 6], 9 , -1069501632);
    c = md5_gg(c, d, a, b, x[i+11], 14,  643717713);
    b = md5_gg(b, c, d, a, x[i+ 0], 20, -373897302);
    a = md5_gg(a, b, c, d, x[i+ 5], 5 , -701558691);
    d = md5_gg(d, a, b, c, x[i+10], 9 ,  38016083);
```

```
    c = md5_gg(c, d, a, b, x[i+15], 14, -660478335);
    b = md5_gg(b, c, d, a, x[i+ 4], 20, -405537848);
    a = md5_gg(a, b, c, d, x[i+ 9], 5 ,  568446438);
    d = md5_gg(d, a, b, c, x[i+14], 9 , -1019803690);
    c = md5_gg(c, d, a, b, x[i+ 3], 14, -187363961);
    b = md5_gg(b, c, d, a, x[i+ 8], 20,  1163531501);
    a = md5_gg(a, b, c, d, x[i+13], 5 , -1444681467);
    d = md5_gg(d, a, b, c, x[i+ 2], 9 , -51403784);
    c = md5_gg(c, d, a, b, x[i+ 7], 14,  1735328473);
    b = md5_gg(b, c, d, a, x[i+12], 20, -1926607734);

    a = md5_hh(a, b, c, d, x[i+ 5], 4 , -378558);
    d = md5_hh(d, a, b, c, x[i+ 8], 11, -2022574463);
    c = md5_hh(c, d, a, b, x[i+11], 16,  1839030562);
    b = md5_hh(b, c, d, a, x[i+14], 23, -35309556);
    a = md5_hh(a, b, c, d, x[i+ 1], 4 , -1530992060);
    d = md5_hh(d, a, b, c, x[i+ 4], 11,  1272893353);
    c = md5_hh(c, d, a, b, x[i+ 7], 16, -155497632);
    b = md5_hh(b, c, d, a, x[i+10], 23, -1094730640);
    a = md5_hh(a, b, c, d, x[i+13], 4 ,  681279174);
    d = md5_hh(d, a, b, c, x[i+ 0], 11, -358537222);
    c = md5_hh(c, d, a, b, x[i+ 3], 16, -722521979);
    b = md5_hh(b, c, d, a, x[i+ 6], 23,  76029189);
    a = md5_hh(a, b, c, d, x[i+ 9], 4 , -640364487);
    d = md5_hh(d, a, b, c, x[i+12], 11, -421815835);
    c = md5_hh(c, d, a, b, x[i+15], 16,  530742520);
    b = md5_hh(b, c, d, a, x[i+ 2], 23, -995338651);

    a = md5_ii(a, b, c, d, x[i+ 0], 6 , -198630844);
    d = md5_ii(d, a, b, c, x[i+ 7], 10,  1126891415);
    c = md5_ii(c, d, a, b, x[i+14], 15, -1416354905);
    b = md5_ii(b, c, d, a, x[i+ 5], 21, -57434055);
    a = md5_ii(a, b, c, d, x[i+12], 6 ,  1700485571);
    d = md5_ii(d, a, b, c, x[i+ 3], 10, -1894986606);
    c = md5_ii(c, d, a, b, x[i+10], 15, -1051523);
    b = md5_ii(b, c, d, a, x[i+ 1], 21, -2054922799);
    a = md5_ii(a, b, c, d, x[i+ 8], 6 ,  1873313359);
    d = md5_ii(d, a, b, c, x[i+15], 10, -30611744);
    c = md5_ii(c, d, a, b, x[i+ 6], 15, -1560198380);
```

```
    b = md5_ii(b, c, d, a, x[i+13], 21,  1309151649);
    a = md5_ii(a, b, c, d, x[i+ 4], 6 , -145523070);
    d = md5_ii(d, a, b, c, x[i+11], 10, -1120210379);
    c = md5_ii(c, d, a, b, x[i+ 2], 15,  718787259);
    b = md5_ii(b, c, d, a, x[i+ 9], 21, -343485551);

    a = safe_add(a, olda);
    b = safe_add(b, oldb);
    c = safe_add(c, oldc);
    d = safe_add(d, oldd);
  }
  return Array(a, b, c, d);

}

/*
 * These functions implement the four basic operations the algorithm uses.
 */
function md5_cmn(q, a, b, x, s, t)
{
  return safe_add(bit_rol(safe_add(safe_add(a, q), safe_add(x, t)), s),b);
}
function md5_ff(a, b, c, d, x, s, t)
{
  return md5_cmn((b & c) | ((~b) & d), a, b, x, s, t);
}
function md5_gg(a, b, c, d, x, s, t)
{
  return md5_cmn((b & d) | (c & (~d)), a, b, x, s, t);
}
function md5_hh(a, b, c, d, x, s, t)
{
  return md5_cmn(b ^ c ^ d, a, b, x, s, t);
}
function md5_ii(a, b, c, d, x, s, t)
{
  return md5_cmn(c ^ (b | (~d)), a, b, x, s, t);
}

/*
```

```
 * Calculate the HMAC-MD5, of a key and some data
 */
function core_hmac_md5(key, data)
{
  var bkey = str2binl(key);
  if(bkey.length > 16) bkey = core_md5(bkey, key.length * chrsz);

  var ipad = Array(16), opad = Array(16);
  for(var i = 0; i < 16; i++)
  {
    ipad[i] = bkey[i] ^ 0x36363636;
    opad[i] = bkey[i] ^ 0x5C5C5C5C;
  }

  var hash = core_md5(ipad.concat(str2binl(data)), 512 + data.length * chrsz);
  return core_md5(opad.concat(hash), 512 + 128);
}

/*
 * Add integers, wrapping at 2^32. This uses 16-bit operations internally
 * to work around bugs in some JS interpreters.
 */
function safe_add(x, y)
{
  var lsw = (x & 0xFFFF) + (y & 0xFFFF);
  var msw = (x >> 16) + (y >> 16) + (lsw >> 16);
  return (msw << 16) | (lsw & 0xFFFF);
}

/*
 * Bitwise rotate a 32-bit number to the left.
 */
function bit_rol(num, cnt)
{
  return (num << cnt) | (num >>> (32 - cnt));
}

/*
 * Convert a string to an array of little-endian words
```

```
 * If chrsz is ASCII, characters >255 have their hi-byte silently ignored.
 */
function str2binl(str)
{
  var bin = Array();
  var mask = (1 << chrsz) - 1;
  for(var i = 0; i < str.length * chrsz; i += chrsz)
    bin[i>>5] |= (str.charCodeAt(i / chrsz) & mask) << (i%32);
  return bin;
}

/*
 * Convert an array of little-endian words to a string
 */
function binl2str(bin)
{
  var str = "";
  var mask = (1 << chrsz) - 1;
  for(var i = 0; i < bin.length * 32; i += chrsz)
    str += String.fromCharCode((bin[i>>5] >>> (i % 32)) & mask);
  return str;
}
/*
 * Convert an array of little-endian words to a hex string.
 */
function binl2hex(binarray)
{
  var hex_tab = hexcase ? "0123456789ABCDEF" : "0123456789abcdef";
  var str = "";
  for(var i = 0; i < binarray.length * 4; i++)
  {
    str += hex_tab.charAt((binarray[i>>2] >> ((i%4)*8+4)) & 0xF) +
           hex_tab.charAt((binarray[i>>2] >> ((i%4)*8  )) & 0xF);
  }
  return str;
}

/*
 * Convert an array of little-endian words to a base-64 string
 */
```

```
function binl2b64(binarray)
{
  var tab = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
  var str = "";
  for(var i = 0; i < binarray.length * 4; i += 3)
  {
    var triplet = (((binarray[i   >> 2] >> 8 * ( i   %4)) & 0xFF) << 16)
                | (((binarray[i+1 >> 2] >> 8 * ((i+1)%4)) & 0xFF) << 8 )
                |  ((binarray[i+2 >> 2] >> 8 * ((i+2)%4)) & 0xFF);
    for(var j = 0; j < 4; j++)
    {
      if(i * 8 + j * 6 > binarray.length * 32) str += b64pad;
      else str += tab.charAt((triplet >> 6*(3-j)) & 0x3F);
    }
  }
  return str;
}


// 思路:
// 因为是cookie加密的网站，所以直接HOOK cookie对应的sign值
// (function () {
//   var cookieVal = '';
//   Object.defineProperty(document, 'cookie', {
//     set: function (val) {
//       if (val.indexOf('sign') != -1) {
//         debugger;
//       }
//       console.log('Hook捕获到cookie设置->', val);
//       cookieVal = val;
//       return val;
//     }, get: function () {
//       return cookieVal;
//     },
//   });
// })();
// 在hook住设置cookie的地方找前一个堆栈，在堆栈之前下断点，尽量往前打断点或者当前函数的第一行，如果逻辑清晰的情况下可以准确下断
// 注意：hook debugger 同时 hook cookie，因为debugger过了之后整个js就执行完了，会重新加载，hook cookie就是为了断住js，方便找函数入口和调试
```

```
287
288 var c = new Date()['valueOf']();
289 token = global['btoa']('aiding_win' + String(c));
290 md = hex_md5(global['btoa']('aiding_win' + String(Math['round'](c / 0x3e8))));
291 // 'sign=1697801817~YWlkaW5nX3dpbjE2OTc4MDE4MTczMzQ=|5e1f336190a7c6d6fbb2c0aed6e2e41d;
    path=/'
292 sign =  String(Math['round'](c / 0x3e8))  + '~' + token + '|' + md
293 console.log(sign)
294 function _sign(time) {
295     token = global['btoa']('aiding_win' + String(time));
296     md = hex_md5(global['btoa']('aiding_win' + String(Math['round'](time / 0x3e8))));
297     return String(Math['round'](time / 0x3e8))  + '~' + token + '|' + md
298 }
299 console.log(_sign('1587102734000'))
```