

浅析绕过js加密

原创 六号刃部 酒仙桥六号部队

2020-11-10原文

这是 酒仙桥六号部队 的第 **104** 篇文章。

全文共计**3191**个字，预计阅读时长**9**分钟。

前言：在渗透测试过程中，我们会经常会碰到登录处用 js 加密字段的情况。在大多数情况下，看到这种加密方式，我们都会放弃对该登录处进行暴力破解。本文主要讲解对 js 加密进行绕过，以达到爆破或绕反爬的目的！

案例一：对登录处使用sm2国密加密算法的某网站进行爆破

抓包分析

该网站图形验证码失效，只要能对密码字段进行相应的加密，就可以爆破！

访问网站，输入用户名：`admin`、密码：`123456`以及正确的图形验证码进行登录。



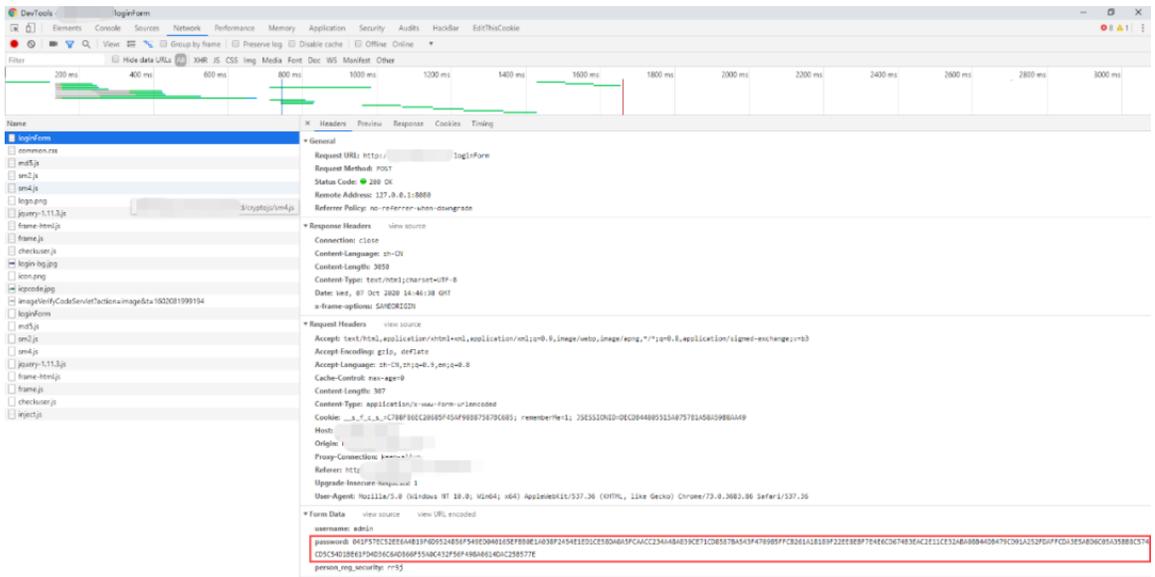
抓包，可以看到密码字段被加密为很长的一段字符。

```
username=admin&password=046985B0CB01EAB950ED081CF7E3A94E9F264EBA0BA0BD4E07DF026097EDC6EB0C4DCE7F48F10EB795FA9D32139D7B39E55566C220EF9DE44E036BD9A146BB60A4F343AD1521D74494A782975876DEF8A90B32F17AFB53300239CD0BF7454D4FEE1C592C5997989AF1C6A3FD3D151E812801933F807E25E3FF5DB27674241529&person_reg_security=71xz
```

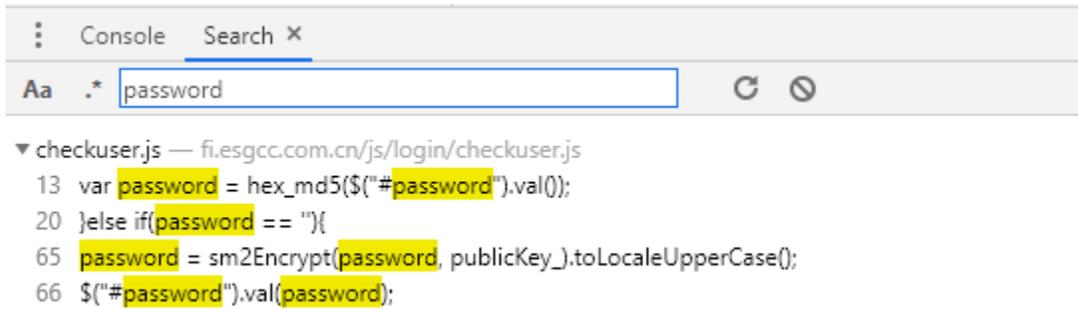
实战绕过

F12打开开发者调试模式，切换到Network选项卡。

重新登录一遍，可以看到password字段进行了加密。



切换到 Source 选项卡，`ctrl+shift+F` 调出全局搜索框，全局搜索 `password` 字段。



跳到 `checkuser.js` 文件，我们看看 `password` 字段经过哪些加密。

`password` 经过两次加密：

```
var password = hex_md5($("#password").val());
```

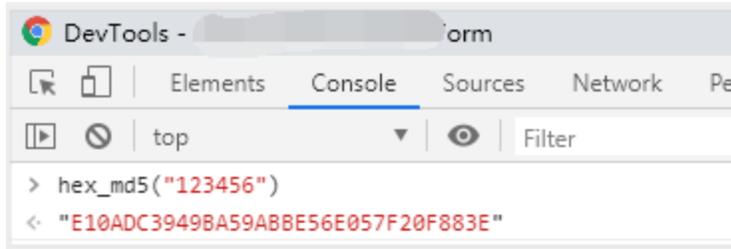
```
password = sm2Encrypt(password, publicKey_).toLocaleUpperCase();
```

```
10 /*
11 function checkUserNew(){
12     var username = $.trim($("#username").val());
13     var password = hex_md5($("#password").val());
14
15
16     if(username == ''){
17         $(".loginWrongTips").show();
18         $("#check_user_hide").show().html("请输入用户名! ");
19         return;
20     }else if(password == ''){
21         $(".loginWrongTips").show();
22         $("#check_user_hide").show().html("请输入密码! ");
23         return;
24     }
25     $(".loginWrongTips").hide();
26     var securityCode11=$("#person_reg_security").val();
27     if(securityCode11 == ''){//空值验证
28         $(".loginWrongTips").hide();
29         $("#check_user_hide").hide();
30         $(".loginWrongTips").show();
31         $("#check_user_hide").show().html("请输入验证码!");
32         return;
33     }
34     var url = "loginCode";
35     $.ajax({
36         type : "POST",
37         url : url,
38         dataType : 'json',
39         cache : false,
40         imageVerify: true,
41         data : {
42             username : username,
43             securityCode11 : securityCode11
44         },
45         async:false,
46         success : function(data) {
47             if( data && data.verify == false ) {
48                 $(".loginWrongTips").hide();
49                 $("#check_user_hide").hide();
50                 $(".loginWrongTips").show();
51                 $("#check_user_hide").show().html("验证码输入错误!");
52                 $("#person_reg_security").val("");
53                 changePersonSecurityCode();
54                 $("#authdiv").show();
55                 $(".sel .regcode_judge").show().find('span').removeClass('right').addClass('wrong');
56             }
57             return;
58         }
59         else if(data.success == "success"){
60
61             $(".loginWrongTips").hide();
62             $("#check_user_hide").hide();
63             $("#btn_loginbag").attr({style:"color:#666666"});//[0].style.color=" #666666";
64
65             password = sm2Encrypt(password, publicKey_).toLocaleUpperCase();
66             $("#password").val(password);
67             $("#form_submit").submit();
68         }
69     });
70 }
```

第一步的加密很简单，就是调用 `hex_md5` 加密函数对 `password` 进行加密。通过全局搜索 `hex_md5`，在 `md5.js` 文件中找到了该函数。如下：

```
18 /*
19  * These are the functions you'll usually want to call
20  * They take string arguments and return either hex or base-64 encoded strings
21  */
22 function hex_md5(s){ return binl2hex(core_md5(str2binl(s), s.length * chrsz));}
```

```
function hex_md5(s){ return binl2hex(core_md5(str2binl(s),
s.length * chrsz));}
```



然后我们看第二步加密，第二步加密调用了 `sm2Encrypt()` 函数对第一步加密后的字符串再进行加密。

我们在全局搜索 `sm2Encrypt`，最终在 `sm2.js` 文件中找到了该加密函数。通过百度搜索 `sm2` 加密算法，发现该算法是国密加密算法。

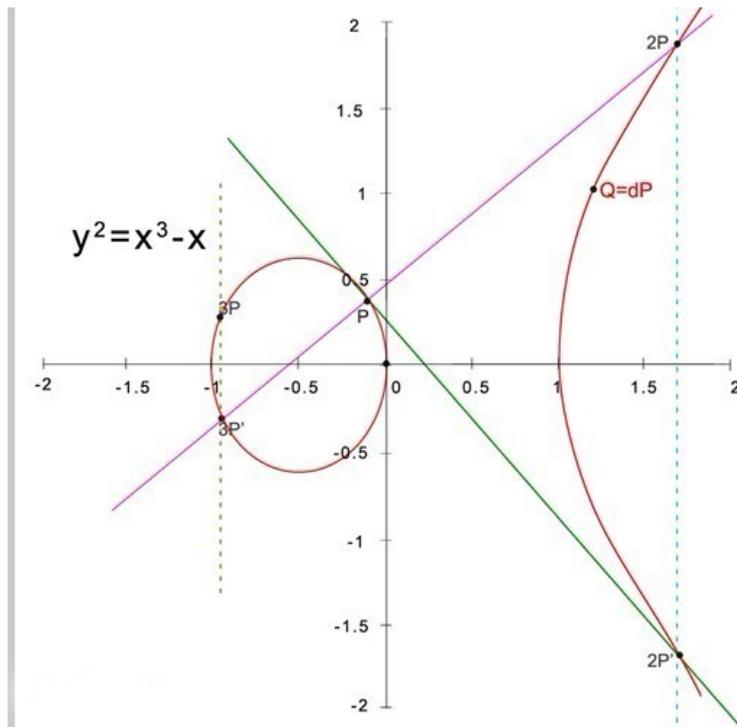
```
checkuser.js  sm2.js x
28840 /
28841 /
28842 /**
28843  * [SM2Encrypt description]
28844  * @param {[type]} data [待加密数据]
28845  * @param {[type]} publicKey [公钥 hex]
28846  * @param {[type]} cipherMode [加密模式 C1C3C2:1, C1C2C3:0]
28847  * @return {[type]} [返回加密后的数据 hex]
28848  */
28849 function sm2Encrypt(data, publicKey, cipherMode) {
28850     cipherMode = cipherMode == 0 ? cipherMode : 1;
28851     //msg = SM2.utf8to64(msg);
28852     var msgData = CryptoJS.enc.Utf8.parse(data);
28853
28854     var pubkeyHex = publicKey;
28855     if (pubkeyHex.length > 64 * 2) {
28856         pubkeyHex = pubkeyHex.substr(pubkeyHex.length - 64 * 2);
28857     }
28858
28859     var xHex = pubkeyHex.substr(0, 64);
28860     var yHex = pubkeyHex.substr(64);
28861
28862
28863     var cipher = new SM2Cipher(cipherMode);
28864     var userKey = cipher.CreatePoint(xHex, yHex);
28865
28866     msgData = cipher.GetWords(msgData.toString());
28867
28868     var encryptData = cipher.Encrypt(userKey, msgData);
28869     return '04' + encryptData;
28870 }
28871
```

SM2国密加密算法

SM2是国家密码管理局于2010年12月17日发布的椭圆曲线公钥密码算法。SM2算法和RSA算法都是公钥密码算法，SM2算法是一种更先进安全的算法，在我们国家商用密码体系中被用来替换RSA算法。

随着密码技术和计算机技术的发展，目前常用的1024位RSA算法面临严重的安全威胁，我们国家密码管理部门经过研究，决定采用SM2椭圆曲线算法替换RSA算法。

更多的关于椭圆曲线的加密方法就不细讲。



所以，现在我们需要对 `sm2Encrypt` 加密函数进行模拟。我使用 `nodejs` 来进行模拟。本地创建 `sm2.js` 文件，把网站上 `sm2.js` 文件中的 `sm2Encrypt()` 加密函数复制进来。末尾加一个 `console.log()` 打印，便于我们查看结果。再把网站的 `md5.js` 文件拷贝到 `sm2.js` 同目录下。而 `publickey` 则在 `sm2.js` 全局定义了。

```
1 /*
2 CryptoJS v3.1.2
3 code.google.com/p/crypto-js
4 (c) 2009-2013 by Jeff Matt. All rights reserved.
5 code.google.com/p/crypto-js/wiki/License
6 */
7 /**
8  * CryptoJS core components.
9  */
10 var publicKey = "0469623686396c766185cd705cbd517714b377ae80b4b919a9de2b688f1cf3ed060f67a13b6ecc8eef422577083d90844d635a675efef9cb6fa48386045a94518";
```

运行该 `js` 文件，提示 `CryptoJS is not defined`。于是在开头加入 `var CryptoJS = require("crypto-js");`

```

C:\Users\... \Desktop\sm2>node sm2.js
C:\Users\... \Desktop\sm2\sm2.js:13
  var msgData = CryptoJS.enc.Utf8.parse(data);
  ^

ReferenceError: CryptoJS is not defined
    at sm2Encrypt (C:\Users\... \Desktop\sm2\sm2.js:13:19)
    at Object.<anonymous> (C:\Users\... \Desktop\sm2\sm2.js:32:13)
    at Module._compile (internal/modules/cjs/loader.js:959:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:995:10)
    at Module.load (internal/modules/cjs/loader.js:815:32)
    at Function.Module._load (internal/modules/cjs/loader.js:727:14)
    at Function.Module.runMain (internal/modules/cjs/loader.js:1047:10)
    at internal/main/run_main_module.js:17:11

```

并且安装crypto-js模块：

`cnpm install crypto-js`

```

C:\Users\... \Desktop\sm2>cnpm install crypto-js
  Installed 1 packages
  Linked 0 latest versions
  Run 0 scripts
  All packages installed (1 packages installed from npm registry, used 178ms(network 177ms), speed 7.19kB/s, json 1(1.27kB), tarball 0B)

```

安装完之后，再次运行。提示 `SM2Cipher is not defined`。提示这个报错是因为该函数里面用到的一些其他函数我们没有复制出来。所以得一个个把相关的依赖函数复制出来。

```

C:\Users\... \Desktop\sm2>node sm2.js
C:\Users\... \Desktop\sm2\sm2.js:25
  var cipher = new SM2Cipher(cipherMode);
  ^

ReferenceError: SM2Cipher is not defined
    at sm2Encrypt (C:\Users\... \Desktop\sm2\sm2.js:25:18)
    at Object.<anonymous> (C:\Users\... \Desktop\sm2\sm2.js:33:13)
    at Module._compile (internal/modules/cjs/loader.js:959:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:995:10)
    at Module.load (internal/modules/cjs/loader.js:815:32)
    at Function.Module._load (internal/modules/cjs/loader.js:727:14)
    at Function.Module.runMain (internal/modules/cjs/loader.js:1047:10)
    at internal/main/run_main_module.js:17:11

```

在加密的地方打断点，F11进行跟进。

```

61 |         $("#loginWrongTips").hide();
62 |         $("#check_user_hide").hide();
63 |         $(".btn_loginbag").attr({style:"color:#666666"});//[0].style.color="#666666";
64 |
65 |         password = sm2Encrypt(password, publicKey).toLocaleUpperCase();
66 |         $("#password").val(password);
67 |         $("#form_submit").submit();
  ..

```

一步一步跳，找到了 `SM2Cipher` 函数，将其复制到我们的 `js` 文件中。

```
sm2.js x  checkuser.js
27870
27871
27872 function SM2Cipher(cipherMode) {
27873     this.ct = 1;
27874     this.p2 = null;
27875     this.sm3keybase = null;
27876     this.sm3c3 = null;
27877     this.key = new Array(32);
27878     this.keyOff = 0;
27879     if (typeof(cipherMode) != 'undefined') {
27880         this.cipherMode = cipherMode
27881     } else {
27882         this.cipherMode = SM2CipherMode.C1C3C2
27883     }
27884 }
```

再次运行，这次提示 `KJUR is not defined`

```
C:\Users\... \Desktop\sm2>node sm2.js
C:\Users\... \Desktop\sm2\sm2.js:82
    var ec = new KJUR.crypto.ECDSA({
                  ^
ReferenceError: KJUR is not defined
    at SM2Cipher.CreatePoint (C:\Users\... Desktop\sm2\sm2.js:82:18)
    at sm2Encrypt (C:\Users\... Desktop\sm2\sm2.js:28:26)
    at Object.<anonymous> (C:\Users\... Desktop\sm2\sm2.js:91:13)
    at Module._compile (internal/modules/cjs/loader.js:959:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:995:10)
    at Module.load (internal/modules/cjs/loader.js:815:32)
    at Function.Module._load (internal/modules/cjs/loader.js:727:14)
    at Function.Module.runMain (internal/modules/cjs/loader.js:1047:10)
    at internal/main/run_main_module.js:17:11
```

百度了下发现需要安装 `jsrsasign`

vue 使用jsrsasign加密，提示KJUR is not defined

vue.js

//安装

```
npm install jsrsasign --save
```

//main.js使用

```
import jsrsasign from "jsrsasign/lib/jsrsasign-all-min";
```

```
Vue.prototype.jsrsasign =jsrsasign;
```

于是安装该模块，并且在脚本的开头加入引入语句 `var`

```
KJUR=require("jsrsasign");
```



```

C:\Users\...\Desktop\sm2>node sm2.js
C:\Users\...\Desktop\sm2\sm2.js:88
    var point = ECPointFp.decodeFromHex(ec.ecparams['curve'], pubkeyHex);
    ^
TypeError: ECPointFp.decodeFromHex is not a function
    at SM2Cipher.CreatePoint (C:\Users\...\Desktop\sm2\sm2.js:88:31)
    at sm2Encrypt (C:\Users\...\Desktop\sm2\sm2.js:29:26)
    at Object.<anonymous> (C:\Users\...\Desktop\sm2\sm2.js:104:13)
    at Module._compile (internal/modules/cjs/loader.js:959:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:995:10)
    at Module.load (internal/modules/cjs/loader.js:815:32)
    at Function.Module._load (internal/modules/cjs/loader.js:727:14)
    at Function.Module.runMain (internal/modules/cjs/loader.js:1047:10)
    at internal/main/run_main_module.js:17:11

```

我们在导入的模块里面全局搜索该函数 `ECPointFp.decodeFromHex`，发现导入的模块中其实是有该函数的。

```

Searching 168 files for "ECPointFp.decodeFromHex"
C:\Users\...\Desktop\sm2\node_modules\jsrsasign@10.0.0\jsrsasign\lib\jsrsasign-all-min.js:
222 /*! (c) Stefan Thomas | https://github.com/bitcoinjs/bitcoinjs-lib
223 */
224: ECFieldElementFp.prototype.getBytesLength-function() {return Math.floor((this.toBigInteger().bitLength()+7)/8)}; ECPointFp.prototype.getEncoded-function(c) {var
d=function(h,f) {var g=h.toArrayUnsigned(); if(f < g.length) {g=g.slice(g.length-f)} else {while(f < g.length) {g.unshift(0)}} return g}; var
a=this.getX().toBigInteger(), e=this.getY().toBigInteger(), var b=d(a,32), if(c) {if(e.isEven()) {b.unshift(2)} else {b.unshift(3)}} else {b.unshift(4); b=b.concat(d(
e,32))} return b}; ECPointFp.decodeFrom-function(g,c) {var f=c[0]; var e=c.length-1; var d=c.slice(1,1+e/2); var b=c.slice(1+e/2,1+e); d.unshift(0); b.unshift(0); var a=new
BigInteger(d); var h=new BigInteger(b); return new ECPointFp(g,g.fromBigInteger(a),g.fromBigInteger(h));}; ECPointFp.decodeFromHex-function(g,c) {var f=c.substr(0,2); var
e=c.length-2; var d=c.substr(2,e/2); var b=c.substr(2+e/2,e/2); var a=new BigInteger(d,16); var h=new BigInteger(b,16); return new ECPointFp(g,g.fromBigInteger(
a),g.fromBigInteger(h));}; ECPointFp.prototype.add2D-function(c) {if(this.isInfinity()) {return c} if(c.isInfinity()) {return this} if(this.x.equals(c.x)) {if(this.y.equals(
c.y)) {return this} else {return this.curve.getInfinity()} } var g=c.x.subtract(this.x); var e=c.y.subtract(this.y); var a=e.divide(g); var d=a.square().subtract(
this.x).subtract(c.x); var f=a.multiply(this.x).subtract(d); return new ECPointFp(this.curve,d,f);}; ECPointFp.prototype.twice2D-function(c) {if(
this.isInfinity()) {return this} if(this.y.toBigInteger().signum()==0) {return this.curve.getInfinity()} var b=this.curve.fromBigInteger(BigInteger.valueOf(2)); var
e=this.curve.fromBigInteger(BigInteger.valueOf(3)); var a=this.x.square().multiply(e).add(this.curve.a).divide(this.y.multiply(b)); var c=a.square().subtract(
this.x.multiply(b)); var d=a.multiply(this.x).subtract(c); return new ECPointFp(this.curve,c,d);}; ECPointFp.prototype.multiply2D-function(b) {if(
this.isInfinity()) {return this} if(b.signum()==0) {return this.curve.getInfinity()} var g=b; var f=g.multiply(new BigInteger("3")); var l=this.negate(); var d=this; var
c; for(c=f.bitLength()-2; c>0; --c) {d=d.twice(); var a=f.testBit(c); var j=g.testBit(c); if(a!=j) {d=d.add2D(a?this:l)} return
d}; ECPointFp.prototype.isOnCurve-function() {var d=this.getX().toBigInteger(), i=this.getY().toBigInteger(), var f=this.curve.getA().toBigInteger(), var
c=this.curve.getB().toBigInteger(), var h=this.curve.getQ(); var e=i.multiply(i).mod(h); var g=d.multiply(d).multiply(d).add(f.multiply(d)).add(c).mod(h); return
e.equals(g)}; ECPointFp.prototype.toString-function() {return ("+"+this.getX().toBigInteger().toString()+"*"+this.getY().toBigInteger().toString()+"");}; ECPointFp.prototype
.validate-function() {var c=this.curve.getQ(); if(this.isInfinity()) {throw new Error("Point is at infinity.")} var a=this.getX().toBigInteger(), var
b=this.getY().toBigInteger(), if(a.compareTo(BigInteger.ONE)<0 || a.compareTo(c.subtract(BigInteger.ONE))>0) {throw new Error("x coordinate out of bounds")} if(
b.compareTo(BigInteger.ONE)<0 || b.compareTo(c.subtract(BigInteger.ONE))>0) {throw new Error("y coordinate out of bounds")} if(!this.isOnCurve()) {throw new Error("Point
is not on the curve.")} if(this.multiply(c).isInfinity()) {throw new Error("Point is not a scalar multiple of G.")} return true};
225 /*! Mike Samuel (c) 2009 | code.google.com/p/json-sans-eval
226 */
C:\Users\...\Desktop\sm2\node_modules\jsrsasign@10.0.0\jsrsasign\lib\jsrsasign.js:
227 /*! (c) Stefan Thomas | https://github.com/bitcoinjs/bitcoinjs-lib
228 */
229: ECFieldElementFp.prototype.getBytesLength-function() {return Math.floor((this.toBigInteger().bitLength()+7)/8)}; ECPointFp.prototype.getEncoded-function(c) {var
d=function(h,f) {var g=h.toArrayUnsigned(); if(f < g.length) {g=g.slice(g.length-f)} else {while(f < g.length) {g.unshift(0)}} return g}; var
a=this.getX().toBigInteger(), e=this.getY().toBigInteger(), var b=d(a,32), if(c) {if(e.isEven()) {b.unshift(2)} else {b.unshift(3)}} else {b.unshift(4); b=b.concat(d(
e,32))} return b}; ECPointFp.decodeFrom-function(g,c) {var f=c[0]; var e=c.length-1; var d=c.slice(1,1+e/2); var b=c.slice(1+e/2,1+e); d.unshift(0); b.unshift(0); var a=new
BigInteger(d); var h=new BigInteger(b); return new ECPointFp(g,g.fromBigInteger(a),g.fromBigInteger(h));}; ECPointFp.decodeFromHex-function(g,c) {var f=c.substr(0,2); var
e=c.length-2; var d=c.substr(2,e/2); var b=c.substr(2+e/2,e/2); var a=new BigInteger(d,16); var h=new BigInteger(b,16); return new ECPointFp(g,g.fromBigInteger(
a),g.fromBigInteger(h));}; ECPointFp.prototype.add2D-function(c) {if(this.isInfinity()) {return c} if(c.isInfinity()) {return this} if(this.x.equals(c.x)) {if(this.y.equals(
c.y)) {return this} else {return this.curve.getInfinity()} } var g=c.x.subtract(this.x); var e=c.y.subtract(this.y); var a=e.divide(g); var d=a.square().subtract(
this.x).subtract(c.x); var f=a.multiply(this.x).subtract(d); return new ECPointFp(this.curve,c,d);}; ECPointFp.prototype.multiply2D-function(b) {if(
this.isInfinity()) {return this} if(b.signum()==0) {return this.curve.getInfinity()} var b=this.curve.fromBigInteger(BigInteger.valueOf(2)); var
e=this.curve.fromBigInteger(BigInteger.valueOf(3)); var a=this.x.square().multiply(e).add(this.curve.a).divide(this.y.multiply(b)); var c=a.square().subtract(
this.x.multiply(b)); var d=a.multiply(this.x).subtract(c); return new ECPointFp(this.curve,c,d);}; ECPointFp.prototype.validate-function() {if(
this.isInfinity()) {return this} if(b.signum()==0) {return this.curve.getInfinity()} var g=b; var f=g.multiply(new BigInteger("3")); var l=this.negate(); var d=this; var
c; for(c=f.bitLength()-2; c>0; --c) {d=d.twice(); var a=f.testBit(c); var j=g.testBit(c); if(a!=j) {d=d.add2D(a?this:l)} return
d}; ECPointFp.prototype.isOnCurve-function() {var d=this.getX().toBigInteger(), i=this.getY().toBigInteger(), var f=this.curve.getA().toBigInteger(), var
c=this.curve.getB().toBigInteger(), var h=this.curve.getQ(); var e=i.multiply(i).mod(h); var g=d.multiply(d).multiply(d).add(f.multiply(d)).add(c).mod(h); return
e.equals(g)}; ECPointFp.prototype.toString-function() {return ("+"+this.getX().toBigInteger().toString()+"*"+this.getY().toBigInteger().toString()+"");}; ECPointFp.prototype
.validate-function() {var c=this.curve.getQ(); if(this.isInfinity()) {throw new Error("Point is at infinity.")} var a=this.getX().toBigInteger(), var
b=this.getY().toBigInteger(), if(a.compareTo(BigInteger.ONE)<0 || a.compareTo(c.subtract(BigInteger.ONE))>0) {throw new Error("x coordinate out of bounds")} if(
b.compareTo(BigInteger.ONE)<0 || b.compareTo(c.subtract(BigInteger.ONE))>0) {throw new Error("y coordinate out of bounds")} if(!this.isOnCurve()) {throw new Error("Point
is not on the curve.")} if(this.multiply(c).isInfinity()) {throw new Error("Point is not a scalar multiple of G.")} return true};

```

于是我们将之前的这条语句 `var KJUR=require("jsrsasign");` 改为 `var jsrsasign=require("jsrsasign");`

。然后再次运行，对运行报错的函数，全局搜索。如果在导入的模块中含有该函数，则在其前面加上 `jsrsasign`。


```
ReferenceError: SM2CipherMode is not defined
    at SM2Cipher.Encrypt (C:\Users\...\Desktop\sm2\sm2.js:58:18)
    at sm2Encrypt (C:\Users\...\Desktop\sm2\sm2.js:32:30)
    at Object.<anonymous> (C:\Users\...\Desktop\sm2\sm2.js:43:13)
    at Module.compile (internal/modules/cjs/loader.js:959:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:995:10)
    at Module.load (internal/modules/cjs/loader.js:815:32)
    at Function.Module._load (internal/modules/cjs/loader.js:727:14)
    at Function.Module.runMain (internal/modules/cjs/loader.js:1047:10)
    at internal/main/run_main_module.js:17:11

C:\Users\...\Desktop\sm2\node sm2.js
C:\Users\...\Desktop\sm2\sm2.js:59
  hexString = this.getHexString(c1.getText()).toBigInt(16) + this.getHexString(c1.getText().toBigInt(16))
  ^
TypeError: this.getHexString is not a function
    at SM2Cipher.Encrypt (C:\Users\...\Desktop\sm2\sm2.js:59:34)
    at sm2Encrypt (C:\Users\...\Desktop\sm2\sm2.js:32:30)
    at Object.<anonymous> (C:\Users\...\Desktop\sm2\sm2.js:43:13)
    at Module.compile (internal/modules/cjs/loader.js:959:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:995:10)
    at Module.load (internal/modules/cjs/loader.js:815:32)
    at Function.Module._load (internal/modules/cjs/loader.js:727:14)
    at Function.Module.runMain (internal/modules/cjs/loader.js:1047:10)
    at internal/main/run_main_module.js:17:11

C:\Users\...\Desktop\sm2\node sm2.js
C:\Users\...\Desktop\sm2\sm2.js:68
  + this.GetHex(c3).toString() + this.GetHex(data).toString();
  ^
TypeError: this.GetHex is not a function
    at SM2Cipher.Encrypt (C:\Users\...\Desktop\sm2\sm2.js:68:28)
    at sm2Encrypt (C:\Users\...\Desktop\sm2\sm2.js:32:30)
    at Object.<anonymous> (C:\Users\...\Desktop\sm2\sm2.js:44:13)
    at Module.compile (internal/modules/cjs/loader.js:959:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:995:10)
    at Module.load (internal/modules/cjs/loader.js:815:32)
    at Function.Module._load (internal/modules/cjs/loader.js:727:14)
    at Function.Module.runMain (internal/modules/cjs/loader.js:1047:10)
    at internal/main/run_main_module.js:17:11

C:\Users\...\Desktop\sm2\node sm2.js
04861d64315218180521d6c9db921bd7e2ecc9fd8db74940845cfe602efab722c9830d97237582ff4b4f3bdfcd352ea1f7d08520aa55465c1a15a0809b08d6145a2c9262d6d1ee1b29fe8cba97253f071e5f9cb4292b8643cb039e01879f42ca546f77634
17c27075c0b866d7d49ea57f804a0b742c1ecbe21108d7
```

sm2.js 代码如下：

```
sm2.js
1 var CryptoJS = require("crypto-js");
2 var jsrsasign=require("jsrsasign");
3 var md5=require("./md5.js")
4
5 var publicKey = "0469623686396c766185cd705cbd517714b377ae80b4b919a9de2b688f1cf3aed60f67a13b6cc0eeff422577083d90844d635a675efef9cb6fa48386045a94518";
6 var data = "123456"
7
8 /* SM2Encrypt description
9  * @param {[type]} data [待加密数据]
10  * @param {[type]} publicKey [公钥 hex]
11  * @param {[type]} cipherMode [加密模式, C1C2C1, C1C2C3C0]
12  * @return {[type]} [返回加密后的数据 hex]
13 */
14 function sm2Encrypt(data, publicKey, cipherMode) {
15   cipherMode = cipherMode == 0 ? cipherMode : 1;
16   //msg = SM2.utf8toB64(msg);
17   data = md5.hex_md5(data);
18   var msgData = CryptoJS.enc.Utf8.parse(data);
19
20   var pubkeyHex = publicKey;
21   if (pubkeyHex.length > 64 * 2) {
22     pubkeyHex = pubkeyHex.substr(pubkeyHex.length - 64 * 2);
23   }
24
25   var xHex = pubkeyHex.substr(0, 64);
26   var yHex = pubkeyHex.substr(64);
27
28
29   var cipher = new SM2Cipher(cipherMode);
30   var userKey = cipher.CreatePoint(xHex, yHex);
31
32   msgData = cipher.GetWords(msgData.toString());
33
34   var encryptData = cipher.Encrypt(userKey, msgData);
35   return '04' + encryptData;
36 }
37 function SM2Cipher(cipherMode) {
38   this.ct = 1;
39   this.p2 = null;
40   this.sm3keybase = null;
41   this.sm3c3 = null;
42   this.key = new Array(32);
43   this.keyOff = 0;
44   if (typeof(cipherMode) != 'undefined') {
45     this.cipherMode = cipherMode
46   } else {
47     this.cipherMode = SM2CipherMode.C1C3C2
48   }
49 }
50 SM2Cipher.prototype = {
51   getHexString: function(h) {
52     if((h.length & 1) == 0){
53
54       return h;
55     }else {
56       return "0" + h;
57     }
58   },
59   encrypt: function(pubKey, plaintext) {
60     var data = new Array(plaintext.length);
61     Array.Copy(plaintext, 0, data, 0, plaintext.length);
62     var c1 = this.InitCipher(pubkey);
63     this.EncryptSM2(data);
```

最终我们可以使用 burpsuite 的插件对这个 js 加密函数进行调用爆破，如下：

爬虫遭遇状态码521陷阱 破解js加密cookie

原创 咸糖 发布于2018-03-01 15:02:02 阅读数 8363 ☆ 收藏

展开

最近接了个小单，遇到一个很头疼的问题，返回的状态码无限521，在网上查阅了各种资料后，终于解决了问题返回200。

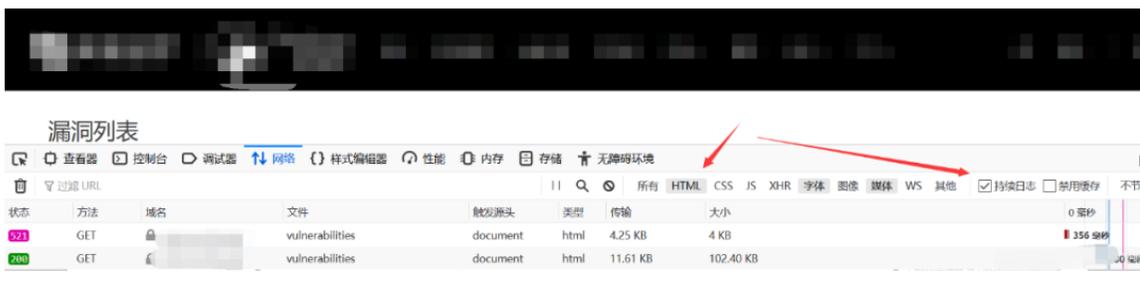
首先咱们先贴上网址：[点击打开链接](#)

抓包分析

接着，就需要开始绕过反爬了。

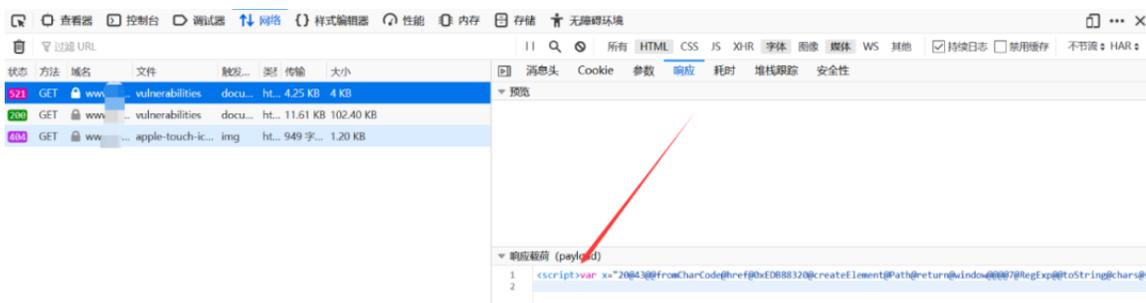
首先F12调试模式查看访问网站时的各种资源。我这里使用的是火狐浏览器。

查看网络一>html，访问网站分两步。

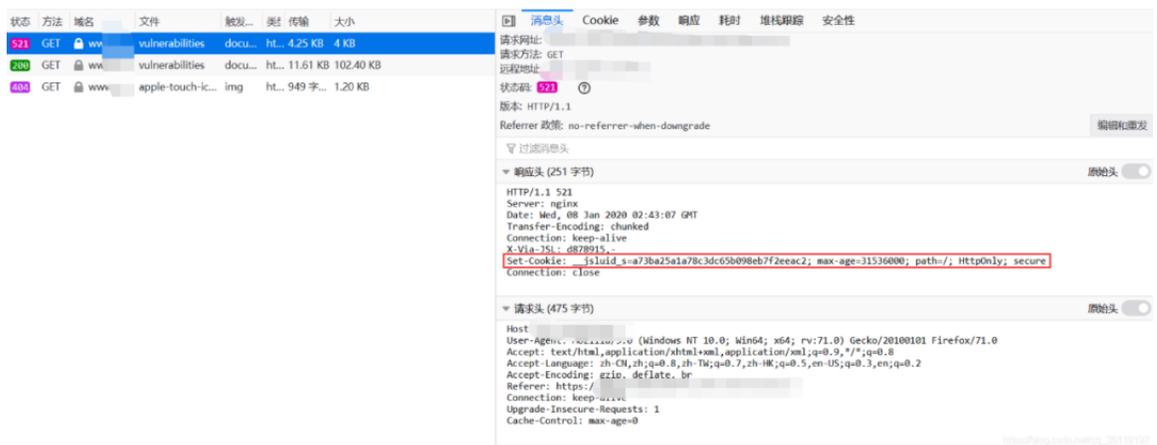


先看第一步521请求

返回的是 521 状态码，然后返回的数据是加密的js代码。

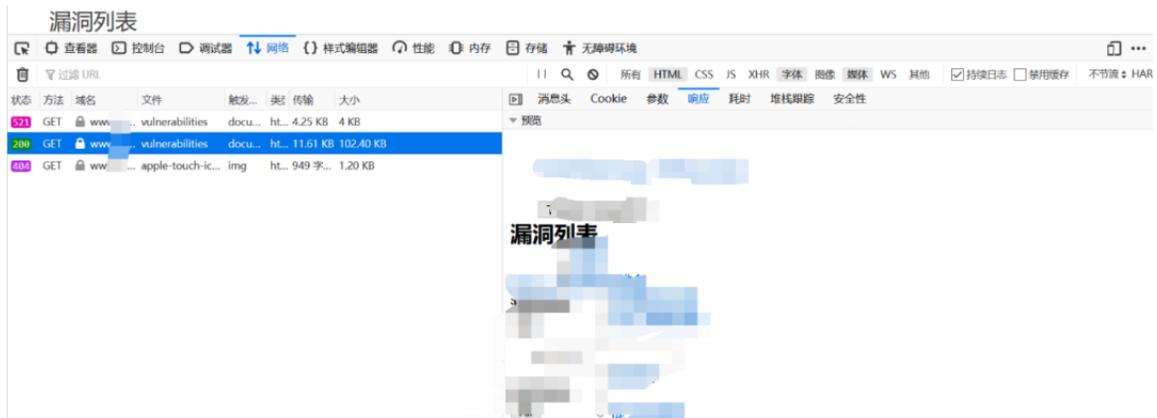


再看请求头和响应头，发现响应头有一个set-cookie参数值。

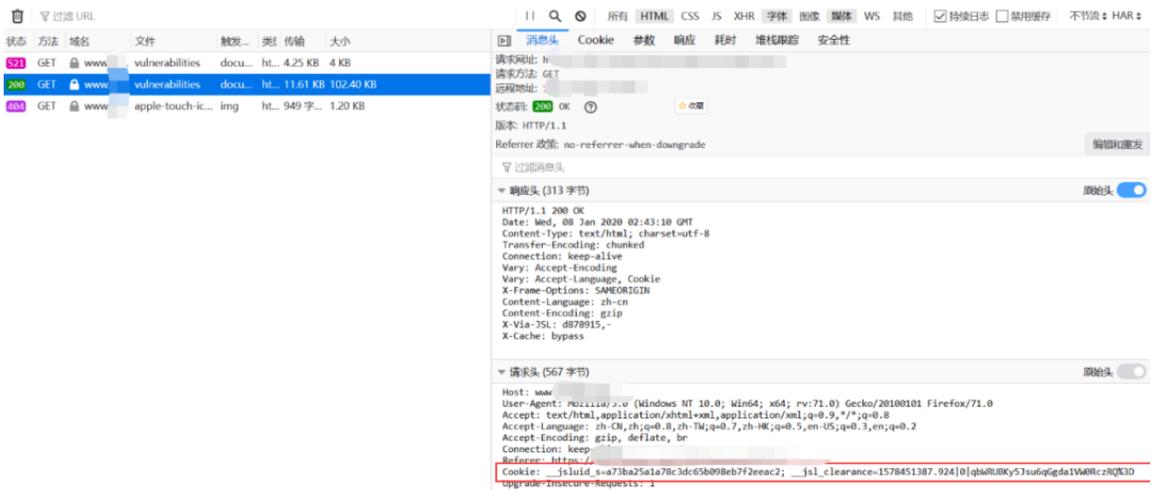


再看第二个200请求

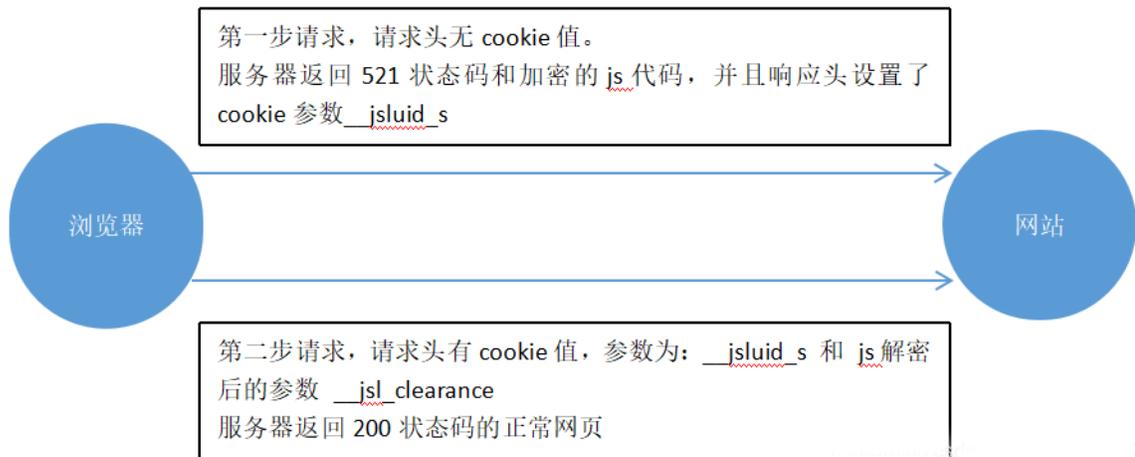
返回的是 200 状态码，然后返回的数据是网页的数据。



再看请求头和响应头，发现请求头的参数值有两个。其中一个参数 `__jsluid_s` 是第一步521请求响应包设置的，而另一个参数 `__jsl_clearance` 则是第一步 521 请求响应的 `js` 数据解密后的值。



我们来梳理一下流程：



所以，现在要想绕过反爬措施，最主要的是解密第一步 521 返回的 js 代码。以下是美化后返回的 js 代码。

实战绕过

我们来分析一下 js 代码。

首先设置了 x 变量和 y 变量。还设置了函数 f 和函数 z。我们现在姑且不看 x、y、f 和 z 的内容到底是什么。我们看最后的一个 while 循环，循环里面执行了 eval 函数。

```
<script>
var x = "20@43@fromCharCode@href@0xED888320@createElement@Path@return@window@@@07@RegExp@toString@chars@setTimeout@String@rOm9XFmtA3QKV/nYsPGI41iFyWwq5vcJh2Id>
y = "17 1h-18()j('1o.5-1o.u+1o.25.2c(/[\?]&]2h-21/,\\''',1k);1b.t="2D-1g.1z]2x]'+(18() (17 24-[18(1h)(9 2g('k.4('+1h+')'),18(1h)(2p(17 24-2x;24<1h.2t;24++) (1
f = function(x, y) {
  var a = 0,
      b = 0,
      c = 0;
  x = x.split("");
  y = y || 99;
  while ((a = x.shift()) && (b = a.charCodeAt(0) - 77.5)) c = (Math.abs(b) < 13 ? (b + 48.5) : parseInt(a, 36)) + y * c;
  return c
},
z = f(y.match(/\w/g).sort(function(x, y) {
  return f(x) - f(y)
}).pop());
while (z++) try {
  eval(y.replace(/b\w+\b/g,
  function(y) {
    return x[f(y, z) - 1] || ("_" + y)
  }));
  break;
} catch(_) {}
</script>
```

我们暂且不看eval函数里的内容是啥意思。我们将eval函数里的内容赋值给test，然后控制台输入这个内容最后调用eval函数执行这个test。这样，我们就可以在控制台看到最后执行的是啥东西了。

```
while (z++) try {
  var test=y.replace(/b\w+\b/g,
  function(y) {
    return x[f(y, z) - 1] || ("_" + y)
  });
  console.log(test);
  eval(test);
  break
} catch(_) {}
```

将以下js代码保存为后缀为 .html 的文件。

```

<script>
var x = "20@43@fromCharCode@href@0xE888320@createElement@Path@return@window@0@07@RegExp@toString@chars@setTimeout@String@0m9XFmtA3QKvNysPGT411fyWkq5vcjH21dx'
y = '17 1h-18([j]'1o.5-1o.1o.25.2c(/[\?&]2h-21h,\'\'\'\''),1k);1b.2=20-1g.1z[2k]'+(18([17 24-18(1h)9 2g('k.4('+1h+')'),18(1h)2p(17 24-2x;24c1h.2t;24+))1
f = function(x, y) {
  var a = 0,
  b = 0,
  c = 0;
  x = x.split("");
  y = y || 99;
  while ((a = x.shift()) && (b = a.charCodeAt(0) - 77.5)) c = (Math.abs(b) < 13 ? (b + 48.5) : parseInt(a, 36)) + y * c;
  return c;
};
z = f(y.match(/\w/g),sort(function(x, y) {
  return f(x) - f(y)
})).pop();
while (z++) try {
  var test=y.replace(/b\w\b/g,
  function(y) {
    return x[f(y, z) - 1] || ("_" + y)
  });
  console.log(test);
  eval(test);
  break;
} catch(e) {}
</script>

```

然后用浏览器打开该html文件，打开控制台输出以下js代码，并且一直在刷新。

将控制台输出的js代码美化，如下：

```

var _1h = function() {
  setTimeout('location.href=location.pathname+location.search.replace(/[\?&]captcha-challenge/,\'\'\'\'', 1500);document.cookie = '_jsl_clearance=1578451387.924|0|' + (function() {
    var _24 = [function(_1h) {
      return eval('String.fromCharCode(' + _1h + ')');
    }],
    function(_1h) {
      for (var _24 = 0; _24 < _1h.length; _24++) {
        _1h[_24] = parseInt(_1h[_24]).toString(36);
      }
      return _1h.join('');
    },
    _1h = [[[]] + (6 + [[]][0]), ( - {} + [] ) + ( - {} + [] )], [( - ~~~ [] + 7 + [] + [[]][0] ) + [ - ~ {} + ( ( - ~~~ [] << ~~~ [] ) ^ ~ {} ) + ( ( - ~~~
    for (var _2f = 0; _2f < _1h.length; _2f++) {
      _1h[_2f] = _24.reverse()[[ ( + !window['_p'] + 'hantom' ) ]][_1h[_2f]];
    }
    return _1h.join('');
  })() + ';Expires-Wed, 08-Jan-20 03:43:07 GMT;Path=/;';
  if ((function() {
    try {
      return ! window.addEventListener;
    } catch(e) {
      return false;
    }
  })()) {
    document.addEventListener("DOMContentLoaded", _1h, false)
  } else {
    document.attachEvent('onreadystatechange', _1h)
  }
}

```

我们发现cookie是通过如下函数生成的。

总结

无论是案例一还是案例二，都是网站为了加强安全性使用js加密做的防护。所以我们需要对网站的js代码进行深入分析，才能进行绕过。在工作中，碰到了js加密的网站不用慌，慢慢细心的分析，总会有意想不到的收获！



知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你



长按二维码关注 酒仙桥六号部队

精选留言

用户设置不下载评论