# 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<u>9分钟。</u>

前言: 在渗透测试过程中,我们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。

8	admin	
7cxz		XXX

登录

username=adminkgssaswrd= 046985800801EA89950ED00310F7E3A94E9F264EBACBACBACBACBACB04E007DF0206097ED06EB0C4AD0E7F4BF10EB795FA9D32139D7B39E555660220EF9DE44E036BD9A1468B60A4F343AD1521D74494A782975876DEF8A90B32F17AFB53300239C DDDF734540H=E10592C5997989AF10EA3FD3D151E812801933F807E25E3F50B276742415278person\_reg\_sourity=71xz

抓包,可以看到密码字段被加密为很长的一段字符。

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。

R (1)     Personant Consults Sources Nationaly Addition Sourcey A
●         ●         ▼         Q.         Vore: III: No. © Generally Vane:         □ Office: Office: Office: ●           Unr         ■         Hold and ULL (0) 308. R. CCS Ing. Media: For be: VS Monifed: Other         ■           20er         40° mi         60° mi         80° mi         130° mi         100° mi         180° mi         200° mi         240° mi         200° mi
Bit         Model Model XML (20)         XMR (20) SCS ing Models Forth Tac: VS Models Cheer           200 mi         400 mi         600 mi         1000 mi         1000 mi         1000 mi         1000 mi         3000 mi         2000
200 mi         400 mi         600 mi         1000 mi         1000 mi         1400 mi         1600 mi         2000 mi         2400 mi         2400 mi         2600 mi         2
lame × Headow Proview Response Cookies Timing krighterm + General
logisferm *General
commonizes Request URL: http://logieForm
ndSja Reguest Method: FOST
m2js Status Code: @ 280 OK
m2 js Remote Address: 127.0.0.1:8880
loga.ong Uryptos/um/s Referere Palicy no-roferrer-whon-downgrate
party 111 A p
i rane tangi ana kangana wangana wangana kangana kangana kangana kangana kangana kangana kangana kangana kangan
j rameja Conection: close
Joestmergen Joestm
Content Length 300
control in     control in     control in
manakimi Cadisevisti2ction_imageSta1502081998194 Date: Ned, 87 Oct 2020 14:146:38 GHT
konform x-frame-options: SA/SMIDN
netSia *Request Headers view source
m2js Accept text/html.spolication/shml+sml,spolication/shmlsm0,sper/seps_smg,*/*1g=0.8,spolication/signed-exchange:y=h3
umājs Acceptēncoding: gzip, dežiate
joury-1.11.3jb Accept Language: zh-C0, zh1q-0, 3, en1q-0, 8
frame-htmljs Cecher-Control: rax-age=0
frame/s Content Length: 307
checkuserjs Content Type: application/x-www-form-wilencoded
injectja Cookie: _s_f_c_s_r/2687865C20465745AF98875775C685; remember1e:1; 3555510x10=05C08448855553473728A58459880A49
Host
Origin: i
Proxy-Connection: here at the
Referen http
Upprade insecure weyse and 1
User-Agent Hozills/3.0 (Mindows HT 10.0; Min64; x64) Appleide052/337.36 (OOTHs, like Gecko) Chrome/73.0.3603.06 Safari/537.36
Form Data view source view URL encoded
username admin
BA15775C32E6A4B19F05F34B587548E39F059L638F7548E30463E58F888E1632E55DA6439F70A45A85RC71C0857BA4439F70585FC351A1338F22F888F7544EC0274859A42211(253A6489460A79C03L357Da9FC03L55FD47
C05CM0186617M0036660D86955Mc4327567486M6340AC288577E
person reg security re51

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

:	Console Search ×	
Aa	.* password	CO
▼ che 13 20 65 66	eckuser.js — fi.esgcc.com.cn/js/login/checkuser.js var <mark>password</mark> = hex_md5(\$("# <mark>password</mark> ").val()); }else if( <mark>password</mark> == "){ password = sm2Encrypt( <mark>password</mark> , publicKey_).toLocale \$("# <mark>password</mark> ").val( <mark>password</mark> );	UpperCase();
跳	到	checkuser.js
文作	+,我们看看password字段经过哪些力	日密。
pas	sword经过两次加密:	
var	<pre>password = hex_md5(\$("#password").va</pre>	1());

password = sm2Encrypt(password, publicKey\_).toLocaleUpperCase();

```
I checkuser.js ×
         */
  10
  11 function checkUserNew(){
           var username = $.trim($("#username").val());
var password = hex_md5($("#password").val());
 13
14
 15
16
              if(username == ''){
                    $(".longinWrongTips").show();
$("#check_user_hide").show().html("请输入用户名! ");
 17
18
  19
                     return;
 20
21
             }else if(password == ''){
    $(".longinWrongTips").show();
                    $("#check_user_hide").show().html("请输入密码! ");
 22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
                    return;
              $(".longinWrongTips").hide();
             var securityCode11=$("#person_reg_s
if(securityCode11 == ''){// 空值栽证
                                                               _reg_security").val();
                   $(".longinWrongTips").hide();
$("#check_user_hide").hide();
$(".longinWrongTips").show();
$("#check_user_hide").show().html("请输入验证码!");
                    return;
              3
               /
/ar url = "loginCode";
              $.ajax({
                   jax({
  type : "POST",
  url : url,
  dataType : 'json',
  cache : false,
  imageVerify: true,
 data : {
    username : username,
                           securityCode11 : securityCode11
                    },
                     async:false,
                    success : function(data)
                          cess : Tunclouted / 
if( data && data.verify == false ) {
    $(".longinWrongTips").hide();
    $("tcheck_user_hide").hide();
    $(".longinWrongTips").show().tml("脸证码输入描误!");
    $("#check_user_hide").show().tml("脸证码输入描误!");
    $("#check_user_hide").show().tml("脸证码输入描误!");
                                 $("#person_reg_security").val("");
                                 changePersonSecurityCode();
                                 $("#authdiv").show();
$('.sel .regcode_judge').show().find('span').removeClass('right')
                                                                                                                                             .addClass('wrong');
                                 return:
                          else if(data.success == "success"){
                                 $(".longinWrongTips").hide();
                                 $("#check_user_hide").hide();
$(".btn_loginbag").attr({style:"color:#6666666"});//[0].style.color=
                                                                                                                                                  #666666":
                                 password = sm2Encrypt(password, publicKey_).toLocaleUpperCase();
                                 $("#password").val(password
$("#form_submit").submit();
```

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:

```
def checkuser.js md5.js ×
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(s), s.lengt
```

```
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 ×
20040 3
28841
28842 /**
28843 * [SM2Encrypt description]
28844 * @param {[type]} data
                                   [符加密数据]
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) {
         cipherMode = cipherMode == 0 ? cipherMode : 1;
28850
28851
         //msg = SM2.utf8tob64(msg);
         var msgData = CryptoJS.enc.Utf8.parse(data);
28852
28853
28854
          var pubkevHex = publickev;
         if (pubkeyHex.length > 64 * 2) {
28855
28856
              pubkeyHex = pubkeyHex.substr(pubkeyHex.length - 64 * 2);
28857
          }
28858
28859
         var xHex = pubkeyHex.substr(0, 64);
         var yHex = pubkeyHex.substr(64);
28860
28861
28862
         var cipher = new SM2Cipher(cipherMode);
28863
28864
         var userKey = cipher.CreatePoint(xHex, yHex);
28865
28866
         msgData = cipher.GetWords(msgData.toString());
28867
28868
         var encryptData = cipher.Encrypt(userKey, msgData);
          return '04' + encryptData;
28869
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nod ejs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。

🖪 checkuser.js 🛛 sm2.js 🗡 CryptoJS v3.1.2 cryptods v5.1.2 code.google.com/p/crypto-js (c) 2009-2013 by Jeff Mott. All rights reserved. code.google.com/p/crypto-js/wiki/License \* CryptoJS core components. 10 var publicKey = "0469623686396c766185cd705cbd517714b377ae80b4b919a9de2b688f1cfa3edb60f67a13b6ecc8eef422577083d90844d635a675efef9cb6fa48386045a94518"; js 文件,提示 CryptoJS 行 该 云 i s not defined 。 于 是 在 开 头 加 入 var CryptoJS = require("crypto-js");

C:\Us	ers	s' \Desktop\sm2>node sm2.js
C:\Us	ers	s \Desktop\sm2\sm2.js:13
v	ar	msgData = CryptoJS.enc.Utf8.parse(data);
Refer	enc	ceError: CryptoJS is not defined
a	t٩	<pre>sm2Encrypt (C:\Users'Deckton\sm2\sm2.js:13:19)</pre>
a	t (	Object. <anonymous> (C:\User: Desktop\sm2\sm2.js:32:13)</anonymous>
		Modulecompile (internal/modules/cjs/loader.js:959:30)
		Module.load (internal/modules/cjs/loader.js:815:32)
a		Function.Moduleload (internal/modules/cjs/loader.js:727:14)
		Function.Module.runMain (internal/modules/cjs/loader.js:1047:10)
		internal/main/run_main_module.js:17:11

并且安装crypto-js模块:

#### cnpm install crypto-js



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。



在加密的地方打断点,F11进行跟进。



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

```
I sm2.js × checkuser.js
2/07
27871
27872 function SM2Cipher(cipherMode) {
          this.ct = 1;
this.p2 = null;
27873
27874
27875
           this.sm3keybase = null;
          this.sm3c3 = null;
this.key = new Array(32);
27876
27877
27878
           this.keyOff = 0;
          if (typeof(cipherMode) != 'undefined') {
27879
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



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");

\Users\\Desktop\sm2>cnpm install jsrsasign Installed 1 packages Linked 0 latest versions Run 0 scripts All packages installed (1 packages installed from npm registry, used 143ms(network 141ms), speed 34.82kB/s, json 1(4.91kB), tarball 0B)	
再次运行,提示 unregistered EC curve name: sm2,这是引入的 jsrsasign.js 文件报的异常。	
<pre>UBers'. \Desktop\sm2&gt;node sm2_js UBers'. \Desktop\sm2&gt;node find"  ISUB.crypto.fCMm.cr</pre>	) 1 F
Inregistered EC curve name: sm2	
基础释义	
unregistered EC curve name: sm2 搜索网络	
未注册EC曲线名称: SM2	
以上结果来自于百度翻译	

于是猜测需要去注册sm2曲线名称。我们继续翻阅网站的sm2.js文件,终于找到了注册该sm2曲线名称的代码。将其复制到我们的代码中。

27859	KJUR	.crypto	.ECPa	ramete	rDB.re	gist(									
27860		"sm2",	// nai	ne											
27861		256,													
27862		"FFFFFF	FEFFF	FFFFF	FFFFFF	FFFFFF	FFFFF	FFFFF	000000	ØFFFFF	FFFFF	FFFFF"	,//p		
27863		"FFFFFF	FEFFF	FFFFF	FFFFFF	FFFFF	FFFFF	FFFFF	000000	ØFFFFF	FFFFF	FFFFC"	, // a		
27864		"28E9FA	9E9D9I	5E344	D5A9E4	BCF650	9A7F39	9789F51	5AB8F9	2DDBCB	3D414D9	940E93"	,//b		
27865		"FFFFFF	FEFFF	FFFFF	FFFFF	FFFFF	FFF72	03DF6B2	1C6052	285388F	409390	054123"	, // n		
27866		"1", //	'h												
27867		"32C4AE	2C1F19	981195	F99044	66A390	.9948F8	E30BBFF	2660BE	1715A4	1589334	4C74C7"	, // gx		
27868		"BC3736	A2F4F	5779C5	9BDCEE	368692	2153DØ4	49877CC	62A474	4002DF3	32E5213	39F0A0"	, // gy		
27869		["sm2",	"SM2	']); /	/ alia	s									
27870															
再	1/17	굽	行		匝山	提	一一	切口	下	的	锆		Tvnel	Error	
1 1		$\sim$	14	,	71	JAC	11	AH	,	ЧЈ	۲Ц	•	TYPCI		•
			1	1	-					-					
ЕСР	ΟlΙ	ntFp	.de	cod	еFr	оmН	еx	ls i	not	a İ	unc	tio	[]		

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' sktop\sm2\sm2.js:29:26) at Object.<anonymous> (C:\Usens) Desktop\sm2\sm2.js:104:13) 我们在导入的模块 里面全局搜索 数 该 函

ECPointFp.decodeFromHex, 发现导入的模块中其实是有该函数的。

Searching	168 files for	r "ECPointFp	.decodeFrom	lex"									Ĩ
Searching C:\Users\ 222 /11 223 /2 224 /2 224 /2 224 /2 224 /2 224 /2 4 225 /1 226 /2 () 226 /2 () 226 /2 () 226 /2 () 226 /2 () 226 /2 () 226 /2 () () () () () () () () () ()	<pre>tWserst, Wserst, /pre>												
227 /*! 228 */ 229: ECF d=functid a=this.g e,32))}r BigInteg e=c.leng e=c.leng o,g.fro c.y)){re this.x).t this.isIL e=this.c this.y m	<pre>(c) Stefan 1 ieldElementF; on(h,f){var { etx().toBigIr eturn b};ECPC mBigInteger(h subtract(c.x) nfinity()){r urve.fromBigut</pre>	<pre>thomas   htt p.prototype. g=h.toByteAn hteger();van ew BigInteg substr(2,e/ h));ECPoint ice())return );van f=a.mu eturn this]i Integer(Bis] integer(Bis]i</pre>	<pre>ps://github. getByteLengt rayUnsigned( e=this.getY eFrom=functi er(b);returr 2);var b=c.s Fp.prototype this.curve. ltiply(this.y.toG nteger.value inly(this.y</pre>	<pre>com/bitcoin, h=function() );if(f<g.le; ().toBigInt. on(g,c){var new ECPoin' ubstr(2+e/2) .add2D=func' getInfinity x.subtract(igInteger() 0f(3));var cubtract(c)</g.le; </pre>	<pre>&gt;&gt;/stcoinj ){return Ma nggt){g=g.s gger();var f=c[0];var tFp(g,g.fro ,e/2);var a tion(c){if( ()}var g=c. d)).subtract .signum()== athis.x.sq ) cubtract()</pre>	s-lib th.floor((thi lice(g.length b=d(a,32);if( e=c.length-2 mBigInteger(a =new BigInteger(a this.isInfin) x.subtract(tt t(this.y);ret 0)(return th) eturne().multi this.y):returne()	is.toBigInt( a-f))else{wd (c)(if(e.isl l;var d=c.s) a),g.fromBig ger(d,16);va ity()){retur sis.x);var ( turn new ECF is.curve.gef ely(e).add( pn new ECP	sger().bitLer sile(f>g.leng ven()){b.uns lice(1,1+e/2) Integer(h)) in h=new Big] or c}if(c.is] ec.y.subtrac vointFp(this. linfinity()) whis curve.a}	ngth()+7)/8 th){g.unsh hift(2)}el ; <u>ECPointFp</u> Integer(b,1 infinity()) :t(this.y); :curve,d,f) var b=this .divide(th	));ECPointFp ift(0)})retu se{b.unshift lice(1+e/2,1 decodefromm 6);return nns (return this carve.fromBi };ECPointFp. curve.fromBi s.y.multip ECPointEn or	.prototype.getEncode rm gj;var (3))else(b.unshift(0)jb. wg-function(g,c)(var w ECPointFp(g,g,from jif(this.x.equals(c. de(g);var d-a.square prototype.tuice2Drft ginteger(BigInteger. y(b));var c-a.square otototome mitical.orbid	<pre>:d=function(c){vi (4);b=b.concat(d) mshift(0);var ar * f=c.substr(0,2) BigIntegen( x)){if(this.y.ec ().subtract( mction(){if( valueOf(2));var ().subtract( function(b))(if()</pre>	ar =new );var quals(
于	是	我	们	将	Ż	前	的	这	条	语	句	var	
КJU	R=re	quir	e ("	jsrs	asig	gn");		改		为		var	
jsr	sasi	g n = r	equi	lre (	"jsr	sasi	.gn"	);					
h			/		- 10 6	++ 44			trin de	1	田去日、	44 LH	

。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.



如果到导入的模块中不含有该函数,则说明该函数是该网站自己定义的,我们到网站的sm2.js中把该函数复制下来就行。比如function

SM3Digest函数,我们导入的模块中不含有该函数,但是在网站中 是定义了该函数的,我们将其复制下来到我们自己的代码中即可。



经过一个函数一个函数的跟踪其依赖,最终将其加密算法模拟了出来,运行截图如下:

ReferenceError: SHICipherMode is not defined	í.
at SM2Cipher.Encrypt (C:\Users\Desktop\sm2.js:58:18)	
at sm2Encrypt (C:\Users Desktop\sm2.js:32:30)	
at Object. <anonymous> (C:\Users\Desktop\sm2\sm2\s:431:13)</anonymous>	
at Modulecompile (internal/modules/cjs/loader.js:959:30)	
at Object.Moduleextensionsjs (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.rumMain (internal/modules/cjs/loader.js:1047:10)	
at internal/main/rum_main_module_js:17:11	
C:\Users\ \Desktop\sm2>node sm2.js	
C:\Users`\\Desktop\sn2\sn2.js:59	
hexString = this.getHexString(cl.getX().toBigInteger().toRadix(16)) + this.getHexString(cl.getY().toBigInteger().toBadix(16))	
Typerror: this gettexstring is not a function	
at solutions the second s	
at smither by (:: Users Users) (smither and smither and smi	
at Udject. (anonymous/ (c: (Users)udject) = (1) = (2)	
at Polaie_compile (internal/moulaes/c)s/1000er.js(35750)	
at Unject.Hooluleextensionsjs (internal/moolules/cjs/loaden-js/99/site) at Model land (internal/moolules/cjs/loaden-js/99/site)	
at Popule Load (internat/module)(c)s/Loader.)s/s15/52/	
at Function.Module_Load (Internal/Modules/c)s/iodader.js:////id/	
at Function. Module.rumMain (internat/Modules/c)s/100004/100	
at internal/main/pon_main_moule_j5:17:11	
C:\Users' \Desktop\sm2>nade sm2.js	
C:\Users\ \Desktop\sn2\sm2,js:68	
+ this.GetHex(c3).toString() + this.GetHex(data).toString();	
TypeError: this.GetHex is not a function	
at SM2Cipher.Encrypt (C:\Users`\Users\Use	
at sm2Encrypt (C:\Users`Desktop\sm2.js:32:30)	
at Object. <anonymous> (C:\Users\&gt; Desktop\sm2\sm2.js:443:jar/</anonymous>	
at Modulecompile (internal/modules/cjs/loader.js:959:30)	
at Object.Moduleextensionsjs (internal/modules/cjs/loadjs:995:10)	
at Module.load (internal/modules/cjs/loader.js:815:32)	
at Function.Moduleload (internal/modules/cjs/loado/js:727:14)	
at Function.Module.runMain (internal/modules/cjs/coader.js:1047:10)	
at internal/main/run_main_module.js:17:11	
040020003102510003242000952404C3009524004C09000074940043C0802081407225003077237027140413007C035520817/005502485548585C181C5800905900614582C9265089725376429208643C0039926187974223	46177634
1/2/0/3/200800/049833/1000000/42/36/06211000/	

sm2.js代码如下:



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:

sm2>node nodeis server.is										
04d4a1ec6b8c132bca7adcf40579e8f74fbf6f89ef7fa	a9feb848c846a9	d786bf8a7e6b2119	97e531a35f90a4d2	2391105b1bec2f6	a9c1ecbc623e7468188f7	ff7ccc2108d0f7e	:892a66d9841d698dcb	4158db1368ecaef6365	4bb3fbf4d76dbb997f88	ea8dceb12084d
77ac3bbf737028bf2e7c567c666b414227b1b6f38d0ada	¥6									
[1] ^ ^	Ruro Suite Profe	ussional v2020.2 - Tempo	wary Project - liceosed	to surferouz						- 0
[*] nodejs server start!				o tancinga						
[+] address: http://127.0.0.1:1664	Burp Project	Intruder Repeater	Window Help							
[+] 123456:04d757d192821ed96e2e72e240dbfc43f5		Deebboard	Ĭ	Tarnat	Provv	Intrude	r í	Penester	Sec	ancar
f49777		Daariooara		Talger	FIOXy	intrade	"	Repeater	0040	
[+] a123456:046fdb94a7226ec9410ed4c25e5efe584	Decoder	Comparer	Extender	Project option	user options	Sqimap	Bypass WAF	reCAPTCHA	captcha-killer	jsEncrypter
d62e652fa				/				_		
[+] 123456a:041d7fb550a2e8f670ca0731f19c98fe4 to two as a second seco	Host: 127.0.0	.1	Port:	1664	Timeout: 5000	Conne	ect IsConnect: True	Test		
09082382C f.1 E301214.044E-7037EL2.K-1L14-21-703.K-9040		ir					_			
[4] 3201314.04438702730300C101483187030030040	123456	04d757d192821	ed96e2e72e2400	ibfc43f5642463d	d8a652f21394f2c0fd748	a8c3a9cfd999ae	089856082ae63b87	e910c45e13a47a7a07	cf8b34106f235a6557e	827d8efdf4b9dct
[+] 111111:04ec89358d0d624b4fb904afcb20dc6f4c	a123456	046fdb94a7226	ec9410ed4c25e5	efe584d127c89c4	c43f2219cef1f3be29440	031116f83d8b74	dd3d1ac4edb625acc	33476e626a76fb08c6	ee246ecebb919690dd	d0f68abb4dfe35c
66093b	123456a	041d7fb550a2e8	3f670ca0731f19c	98fe4e8ca56b247	fec964e3deb028ddadba	a5e920b69b386	873fc4267526d0733	58a101af4a1537e402	ce6a3fb0a5fa9250ce4	ee27e1e674744b
[+] woaini1314:04fbef0af60e1aa859fa849e480f25	5201214	0445a70275b3d	bc1b14a31a789/	ba904060ada3a	20465346526720492495	5#25fb10b80ee6	c00#570b002cd172c	cf3ee7c06c83c45d45	00/00506-55066571	Gadb5/32d200f9
88292d958abbe35bb8	0201014	0445870273050	24646004-6602	deCide Ofer a Bb Ad	2000000000000700002400	01201010000ea0	-2-22d-2661dh-f=0	6776600=0E4b4fe4806	==200433de======	
[+] qq123456:041fd0ff71c61c8fa526b0f52d1b2c1c		0400893580006	240410904810020	000140218000040	13090040803836602541	030024898170	028330836610001890	00/00UUeU04D0IeI0U0	e529013c1a5653a527	aocoacooarzuzue
635744072261	woaini1314	04fbef0af60e1aa	a859fa849e480f2	54308de1caa173	7c232d15717785c2ad7	275219d47d826	ecf5544ebfeb57d023	2ce8a89d03afd5ee51	f0517b8859bd54632e	2bd396fbc17fb2c
[+] 123123:04c2edc5fee84c6e895a1742e065fd8755	qq123456	041fd0ff71c61c8	fa526b0f52d1b2	c1cda331a6f610a	42acedf409fbf8d69d9fb	e5faa201495273	3d6c88e8d943e525d	d4f9057bc596d05c73	1f82937139602758a4	3b45ec22a0cb2ct
fød1f2	123123	04c2edc5fee84c	6e895a1742e06	5fd8755fa6392fe5	53de29ccd3be665485d2	2435b50de5a6d	dc7788164121139ec	d6ecc16ed04ce53121	a3218041c87443274c	7942cebaffee097
[+] 000000:049855b45e877ab19d2f02dd7f432f72b4	000000	049855b45e877	ab19d2f02dd7f4	2f72b41bf4c9ed	d37aedf835c872e56ec4	d5cf3898d241ffb	03d70b55d5ca0b88	0346a5fdd2359472b7	8b96a7a544808f2a994	2f679f7bcf735b8
6c82b9	1gaz2wsx	044bd3a5d0d29	2c70a05197ab04	9d1e4c1719280	3c73442942ce07ad1c46	8446b6b881b3d	bc6c7c6d4b88bbf91	2487bfec61c5069b0cc	be16f5b37076738b47	34704fc5920e63
[+] 1qazzwsx:044005a5000292C70a05197a00490104 74238+5641db	102w3e4r	0489ab5999403	89a10058528aa	7acb04e78e756	834e4094c260add9ee80	136f435ae6a70d	5haab96f9d12506d2	78d8517df6d76e4180	46h55313f2601e69h2	d83de910ce4088
[+] 102w3e4c:8489ab599948389a18858528aad7acb8	muo122	04056500000400	Ochb907305091	06c0559b0o02o	7o2d22407Eca0d66b82	aa4acd03ac824	7c1ba0648daa0d0fcf	WeEGeeba1bd0daa14	04b307c0f1c0769f96	24266-920640641
4353616548b2	dwe 120	0426512070234	900091395061	0100300096926	7830234073085060063	ad4ecu03eco24	/cibaso4odeaodsici	neobeeba ibuguaa 140	5040297001100700100	3430080303000040
[+] gwe123:042651207d2349cbb89739508140f6c955	//58521	044/9566896310	21430001178187	20040398080000	9acc8aa0b9dc66239e02	201/5050894108	10150516/1449109/	a8380c49c692a/1eb	c/c523646951508201	c42c23ea289beb
7be456	123qwe	044456a9a41ea	d31c50eaa287a2	912c9d8a560620	be4f0ab69da925ffc00b8	89171602907d64	1e3c4d81603da8202	98935f104ab9bc5879	d2d2c9e9d8bb18d83d	15062cf12e694d6
[+] 7758521:0447956ce963fc2f43b0df1781872e646	a123123	042f0db4dd3556	5d6110d38c1c42	adefc5ca11d33a4	2a75d488a1a32ca8294	lbe9e96370bdd3	c253df8a302b3ad28	c200353a836e2cceac	df3383790dc3251c80	c21e1e7472892a
aa2e0eccd	123456aa	04fc76959174b7	cfb395ff23b110f	20d8866bb3a515	d9d181f99f055c9f0cf2dd	d8f99eed18db55	515d98ba531104129	733a286a3838bc443e	50977347d00570b69e	132cea6a2870et
[+] 123qwe:044456a9a41ead31c50eaa287a2912c9d8	woaini520	044cf662b3627b	of7a22ac2129302	1f668ff16f8eb73e	a7163957cd3318bd11b2	218329d4f1b6e7	96478372342a996db	dbb9b7d838182e12c	c04b445b15f2d239937	1f805be76e32bb
bf8f21	woaini	04a63a3ea8dd4	5014d7b2ce0b7t	78ba4ac7af817e	b0693eb36ab2ea6d2f78	36b15ab51ec770	e24bb7a7929232ef0	646392f035033f6986	c4664f8cb2a3e7da295	b2bededc18291
[+] a123123:04210db4dd3556d6110d38c1c42adetc5	100200	043689cfde1693	4a2e8f1ebb237	3d50059517d8e	a59f0bba83af1b5e0b2cf	246cecc58c70a	19b3ca041f9d3206d6	1171abf91ac11b8297	27dc8924553597c12	c56bdcfb5369bc
888143230 [+] 133456	1214520	04500501081050	Face200-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0	2272bd=014e=20	db076e7e5e0e97e98225	24000000000000000	4505ca041150320000	hEaseb9dee66de66ed	210003240000070120	47802642847644
(*) 12345088.04107053517407010355112301101200 c4ddbe88195e	1314520	040/4620009100	010003600300341	22/20009140830	009/00/e58200/e61335	b3b2b76a70elei	15acea310263666619	boccabecaelococroee	ee19340001315005911	4/60360360/1811
[+] woaini520:044cf662b3627bf7a22ac21293021f6										
79d85c9d75a497c										
[+] woaini:04a63a3ea8dd45014d7b2ce0b7b78ba4ac										
b8b115										
[+] 100200:043689cfde16934a2e8f1ebb237d3d5005										
e6b83c										
L+J 1314520:04b74e28689f06fc653e0c9d834f2272b										
201103120										

至此, js解密完成, 我们可以对该登录接口进行爆破了!

# 案例二:对某漏洞平台反爬进行绕过

以下是针对某漏洞平台反爬进行绕过,最终可以通过脚本爬取该漏 洞平台的漏洞列表。

直接使用爬虫脚本爬去漏洞,返回的是一段加密后的js代码,且返回状态码为521。

响应状态码: 521 响应数据: <script>var

 $x = "onready statchange@toLowerCase@020@__jsl_clearance@Wed@d@replace@_p@div@charCodeAt@dl@split@while@Array@000@00@document@dreverse@07d15784509830ttps@f@pathname@captha@r0m3KPtA30KV7N'sPGT41ifyWkdycjH2IdvUoChERLazBDNB6@cookie@return@parseInt@var@0xFF@dJgSe0upZ@decapthamtom@dlocathon@window@attachevert@e0@00ContentLoadedLese030fromCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCharCodePath@defunction@String@cHarsentomCh$ 

于是百度了下响应状态码521,果然,是反爬措施。

# 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->html,访问网站分两步。

														1					
	漏洞列	刘表										/							
R	○ 查看器	▶ 控制台	□ 调试器	↑↓ 网络	{}样式编辑器	Ω 性能	① 内存	∃ 存储	Ť	无障碍环境					_	_			Ó.
Û	冒过滤 URL							11	Q	◎ 所有	HTML CS	S JS	XHR 🦻	体图像	媒体	WS	其他	☑ 持续日志 □ 禁用缓存	不节流
状态	方法	域名		文作	+		触发源头	.	类型	传输	大小							0 臺秒	
521	GET			vul	nerabilities		docume	nt l	html	4.25 KB	4 KE							356 - 386	
200	GET	6		vuli	nerabilities		docume	nt l	html	11.61 KB	102.	40 KB							-30 REM

### 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。

R	¢ ₫	看器	▶ 控制台	口道	渴 ↑↓	网络	【} 样式	に編組器	Ω 性能	① 内存	- 7	字储 1	┢ 无障	碍环境										0 … ×
Û	冒过	g URL										11 C	0	所有	HTML	CSS	JS XI	IR 字体	图像	媒体	WS	其他	☑ 持续日志 □ 禁用緩存	不节流 ♥ HAR ♥
状态	方法	域名	文件		触发	毿 樹	n :	大小			Þ	消息头	Co	ookie	参数	响应	耗时	堆栈跟	踪	安全性				
521	GET	A www	. vuln	erabilities	docu	ht 4.2	5 KB 💚	4 KB			〒 预	揽												
200	GET	<b>≜</b> ww	vulne	erabilities	docu	ht 11.	61 KB 1	102.40 KB																
404	GET	🗎 ww	apple	e-touch-ic	img	ht 949	9字	1.20 KB								/								
											~ 8	麻藏荷	(pavlo	0	/									
											1		dint has		84289£c	on Charles	ada@bne	600~EDB9	932084	coatoE1	omonte	DathBee	tunolisi ndas 0000700 nativo0	BtoStningBchans@s
											2	csci	прома	IF X= 20	et seletti	oncharc	onsfaurs	reexeres	e 25 collect	reateti	tementig	eachere	couliewingewingerste	icosci. rugijenar sjis

再看请求头和响应头,发现响应头有一个set-cookie参数值。

状态	方法	域名	文件	触发	类 传输	大小	▶ 消息头 Cookie 参数 响应 耗时 堆栈跟踪 安全性	
521	GET	A ww	vulnerabilities	docu	ht 4.25 Ki	3 4 KB	请求网址	
200	GET	⊜ ww	vulnerabilities	docu	ht 11.61 H	(B 102.40 KB	远程地址	
404	GET	€ www	apple-touch-ic	img	ht 949 字.	1.20 KB	状态码: 523 ⑦	
							版本: HTTP/1.1	
							Referrer 政策: no-referrer-when-downgrade	编辑和重发
							▼ 过滤消息头	
							▼ 响应头 (251 字节)	原始头
							HI1P711-521 HI1P711-521 Date: Med. (00 Jan 2020 02:43:07 GHT Transfer-Encoding: chunked Connection: keep-alive X_Via_JSi: d872015 Set-Cookie: _sluid_s-m373ba25m1a78c3dc65b098eb7f2eeac2; max-age=31536000; path=/; HttpOnly; secure Connection: Close	
							* 请求是(475字号) Host User-Agenc.noclass.j.v (Windows NT 10.0; Win64; x64; rv:71.0) Gecko/20100101 Firefox/71.0 Accept: text/html,application/xhtml;aplication/xml;q=0.3,*/*;q=0.8 Accept:Incoding: grip.def.atc. br Commention: http://windows.atc.atc.atc.atc.atc.atc.atc.atc.atc.atc	殿設

## 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。

	漏洞列表																						
R		控制台	〇 调试器	1 👬 网络	绪 {}样	¥式编辑器	Ω 性能	① 内存	•	存储	<b>无障</b>	尋环境											<u></u>
Û	▼ 过滤 UR									11 C	0	所有	HTML	CSS	JS	(HR 🎐	体图	像 媒体	k WS	其他	✓ 持续日	志 🗌 禁用援存	不节流 🛊 HAR
状态	方法 域名	文件	1	触发 类	毬 传输	大小			Þ	消息外	c Co	okie	参数	响应	耗时	堆栈	眼踪	安全性					
521	GET 🔒 w	w vulne	rabilities	docu ht	t 4.25 KB	4 KB			₹ Ť	页范													
200	GET  🙆 w	w vulne	rabilities	docu ht	t 11.61 KE	в 102.40 КВ																	
404	GET 🔒 w	w apple	-touch-ic	img ht	t 949 字	1.20 KB																	
									<b>漏</b>	清河歹													

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@0xEDB88320@createElement@Path@return@window@@@@7@RegExp@@toString@chars@setTimeout@String@r0m9XFMtA3QKV7nYsPGT41ifyWwkq5vcjH2Idx
$\mathbf{y} = \frac{17}{1h-18()\{j(10.5=10.u+10.25.2c(/[))(10.5=10.u+10.25.2c(/[))(10.5=10.u+10.25.2c(/[](10.5=10.u+10.25.2c(/[](10.5=10.25.2c(/[](10.5=10.u+10.25.2c(/[](10.5=10.25.2c($
$f = function(x, y)$ {
x = x.split("");
<pre>while ((a = x.shift()) &amp;&amp; (b = a.charCodeAt(0) - 77.5)) c = (Math.abs(b) &lt; 13 ? (b + 48.5) : parseInt(a, 36)) + y * c;</pre>
<pre>z = f(y.match(/\w/g).sort(function(x, y) {</pre>
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)
}));
} catch(_) {}

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。



### 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。

R	① 查看器 ② 控制台 □ 调试器 1↓ 网络 {} 样式编辑器 ① 性能 ③ 内存 日 存储 1★ 无障碍环境								٥·
Û	<b>点</b> 印始時期用	错误	警告	日志(	18	调试	CSS	XHR	请求
	$ \begin{array}{llllllllllllllllllllllllllllllllllll$	8451387 g(_1E)} ^~~{})] ()+(-~[ ]<<-~~~ [])+-~ []<<-~~ [])+-~ []>^-~ []-~[]]+ 7845138 ){return	){retur ;retur ;( ]-~[]) []))+[ [])+ {})+(( ~[]))+ ([])]],[ ([] 7[matcl n 8;}}	nn []+7+[]+ "[-==[]=== ]),[===[]< []](07) ]<<====[] [](07) ]<<====[] [](07) ]<<====[] [](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<===[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<==[](07) ]<=	([]][( []]]],  +((-~  +[-~[] ))+[] /+[]+  []-~[] #m	97])+ .[((-~!] -~~[]<< []/^-~{} [-~(+1' ]][07] []][07 []]+(-~	)]+(-~ ))]+ )+ ])+ []+[])	1.h	tml:21:1
	$ar_{1} = \frac{1}{2} + \frac{1}{$	tarance [_24]).: []][0]). ([]- [])^([]- !{]+[( []])+[(6 []]+[]] []<< _2f]=_2	=15784 toStri +[2],( << })+(( ~[])*[ []< +[[]][( )+[2], ~-[]))+ 4.reve	51387.92 ng(36)}; []+7- [])]>>( [][]) ([]) ([]) a]), ( ((!{]+ [])+[!] rse()	1 0  '+ returr +[]+[] ]+ ]+( ]>>( -[]+7+ [( ])+((-	[]][0])+ [<-~~~ ~~~[})] ~~~[]<(-~~ []+[[]] []<<-~~ ~~~[]<(-~~~ ~~~~[]<(-~~~~ ~~~~~]<(-	+ [}]  ] 	1.h	tml:21:1

#### 将控制台输出的js代码美化,如下:



我们发现cookie是通过如下函数生成的。



我们直接在控制台执行该函数,如下,得到了生成的cookie。但是因为该cookie是有一定的时效性的。所以,我们得写一个脚本,快速的获取生成的cookie,然后访问网站。



最终的反爬脚本运行截图如下 :

🖬 cmd × + 🕚	v	乏 英 🦻 🔮 📟 🐁 👕 🔛
(([]<<[]) []<<[]) []][0]),[1{}+(( ~-()),[(1{}+(( ~())),[1{}+([]- ~())],[1{}+([]- ~()+(([]- []+7+[]+[])[ []<<[]+7+[]+[])[ []<<[])+(1),[6+[[])+(]) []]]+[]+[-1]+(( r(var 53-00_530_530_530_530_530_530_530_530_530_5	)^-())],[[2]+[1{}+(([]<<[])^{})+(([]<<[))^(})],[(6+[[]][0])+[1{}+([][])*[ ())+(([]<<[])^())+(2],(([]<<[]))+([]][0]),[[2]+[1]+([][]])],[[1](+([])]),[[1]]),[[1]]),[[1]]),[[1]]),[[1]]),[[1]]),[[1]]),[[1]]),[[1]]),[(1]]),[(1]]),[(1]]),[(1]]),[(1]]),[[1]]),[(1]]),[[1]]),[(1]]),[[1]]),[(1]]),[[1]]]),[[1]])),[[1]]),[[1]]),[[1]])),[[1]])),[[1]])),[[1]])),[[1]])),[[1]]	[-~[]-~[]], [-~!{}+(( [-~[]-~[]], [-~!{}+(( )+((-~~[]<<-~~[]))+[ )+((-~~[])^- (<-~~[])^- +(-~!{}+[-~(+!'')])/[ (-~~[]<<-~~[])^- +((-~~[])<-~~[])^- ~~[]>+((-~~[]))>(-~~ +([]][]))>(-~~ +(]+[]+[[]][0])+( )]>(-~~[]<<-~~[]) ]+(-~!{}+[]+[[]][0])];fo ()
解密cookie成切*** isl cleanance=1	-1578468775 868/0///CrAHiTinTC21g%28grV/Ynv0008%30	
	***************************************	*
('jsl_clearance 时间: 2020 时间: 2020 时间: 2020 时间: 2019 时间: 2019	e': '1578468775.868 0 kICrAHiTLpIC21g%28grYcXpxQ0A8%3D'} 标题: WebLogic 未授权远程代码执行漏洞(CVE-2019-2725) 链接: 标题: mongo-express远程代码执行漏洞(CVE-2019-10758) 链接: / 标题: Apereo CAS 4.X反序列化漏洞应急 链接: /vuldb/ssvid-98115 标题: D-Link DIR-859 -Unauthenticated RCE (CVE-2019-17621) 链接: 标题: Apache ActiveMQ Log4j 1.2.x远程代码执行漏洞 链接: /	
时间: 2019 时间: 2019 时间: 2019 时间: 2019	标题: Apache Log4j 1.2.x远程代码执行漏洞(CVE-2019-17571) 答~2 标题: TP-Link Archer路由器漏洞(CVE-2017-7405) 链接:	
时间: 2019 时间: 2019 时间: 2019 时间: 2019	标题: Confluence Pre-Auth SSRF(CVE-2019-3395) 链接: 标题: Apache Flink 1.9.1 通过RESTful API达成远程代码表,, 标题: Mindows证书对话框权限提升漏洞(CVE_2019-138P) <sup>#147</sup> 标题: Apache Shiro 721 Padding Oracle漏洞 链接:	
ny (c) · 2019	你题• Apache Solna 医能自MX KMI守我処柱代给执行漏洞(cvc zvz/ zzwz/) 如返• ,	

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。所以需要我们对网站的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!



精选留言

用户设置不下载评论