链滴

CKFinder3 破解与防破解 (转)

作者：alanfans

原文链接：https://ld246.com/article/1510644721416

来源网站：链滴

CKFinder3破解与防破解

转自：http://www.petershi.net/ckfinder3-cracking-of

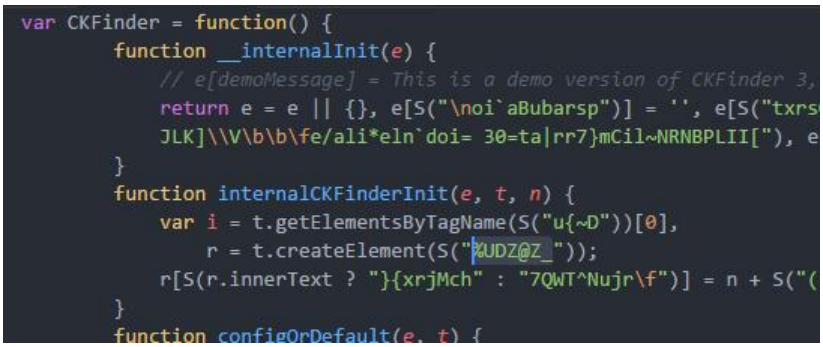
背景：后台开发时候频繁遇到图片选择组件，要求该组件可以选择服务器上已有的图片或者新上传一图片，选择之后还要编辑裁切图片。

试用了下CKFinder功能不错，就是多个站点得500刀确实贵了点，试试看能不能绕过授权试试看。

下载然后本地服务器配置好之后，深深地感受到了试用版真的是很Demo，网上流传有不少破解方法然而这些方法对于最新版并没有什么用，因为，最新版的js代码已经加密。里面的if判断基本全是加密的字符串。

本来事情就结束了，可是心中不服。

首先格式化js代码（工具），找到function S(e){ }; 加密后的内容都是在调用这个方法解密字符串。这子的话，貌似能试试看咯。即便是搞不定也能学个算法。

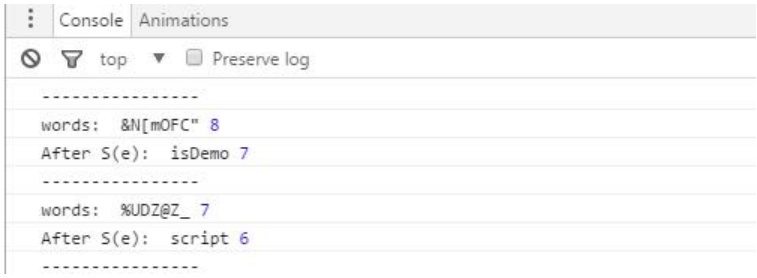


首先来个测试页面：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <script>
        /*
         * 原解密函数
         */
        function S(e) {
            for (var t = "", n = e.charCodeAt(0), i = 1; i < e.length; ++i) t += String.fromCharCode(
.charCodeAt(i) ^ i + n & 127);
            return t;
        }

        // 先试两个数据找下节奏
        log_info('&N[mOFC"');
        log_info(';tXRS/');


        /*记录测试数据*/
```

```
        function log_info(words) {
            console.log('----------------');
            var s = S(words);
            console.log("words: ", words, words.length);
            console.log("After S(e): ", s, s.length);
        }
    </script>
</body>

</html>
```

控制台输出：



words是解密后的字符串，After S(e)是用S方法解密后的数据

里面有一个细节是加密后的字符串比原字符串多了一位，尝试分别删除前后一个字符看看结果：

```
----------------
words:  &N[mOFC" 8
After S(e):  isDemo 7
----------------
words:  %UDZ@Z_ 7
After S(e):  script 6
----------------
words:  %UDZ@Z 6
After S(e):  scrip 5
----------------
words:  %UDZ 4
After S(e):  scr 3
----------------
words:  UDZ@Z_ 6
After S(e):  5
----------------
```

推测除了第一位数据后面的字母是一对一的关系，删除第一位之后后面数据全乱掉，说明，第一位应是“标志位”。

先明确下我们的目的：**找到程序中判断是否购买的的判断模块，或者是判断软件是适用版本的代码，示信息等。**

现在代码被加密了，Demo版本的提示信息显然是被加密了，要顺藤摸瓜，想想办法把提示信息找出。代码已经写死，加密后的提示信息 通过S函数可以直接换还原，那么我们有没有可能推出加密算法根据网页上显示的提示信息自己生成加密的提示信息，根据这个定位代码中哪块在判断授权。

开始正式尝试的分割线

解密算法有了，字符串结构了解了，那么接下来可以试着推一推“标志位”是怎么来的。

解密算法整理一下

```
function S(e) {
    for (var t = "", n = e.charCodeAt(0), i = 1; i < e.length; ++i) {
        t += String.fromCharCode(e.charCodeAt(i) ^ i + n & 127);
    }
    return t;
};
```

for循环里面分别对每一个字符串进行操作，确实要解密的字符串第一个字符并没有进行转换操作，这好解释了为什么前面发现长度差1位的原因

// 先拎出来第一个循环

```
// 涉及的变量
t=""
n=e[0]; // 标志位n : 字符串第一个字符的ASCII码对应值
i = 1;
```

算法公式（A）：

```
t = fromCharCode(e[i] ^ i + n & 127);
```

为了有效防止蒙圈，根据运算符优先级，用括号框起来便于理解

```
t = fromCharCode(e[i] ^ ( i + ( n & 127 ) ) );
```

拆解：
(n & 127) 目的是保证字符在ASCII之内，暂时忽略细节不考虑溢出，简化成 n;
e[i] 看作是密文
设 c = String.chaCodeAt(t);

那么c可看作原文得到：

```
c = e[i] ^ ( i + n );
```

用人话说就是

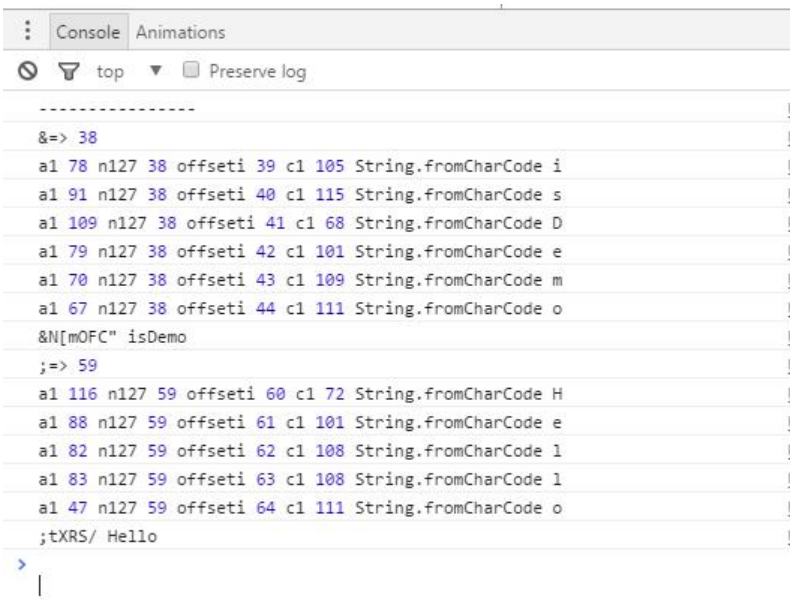
原文 = 密文 ^ (i + “标志位”);

根据异或的特性IF a ^ b = c THEN a ^ c = b

密文 = 原文 ^ (i + “标志位”);

进一步分析：i 在密文和原文之间，存在一对一的偏移关系，举例说:i=3, 密文[3] => 明文[3] ，因此且认为(i+ “标志位”) 是个定值。

从源代码中找了两段字符，测试了下输出结果，参照下图,注意看offset,和offset+i，



加密字符串为&N[mOFC" 开头字母 & 的ASCII值就是38，原文就是异或了这个offseti的值得出c1,原的ASCII值 比如N(78) 偏移 (39) 得到 i(105) 。

那么到此我们是有望推导出加密算法的。

显然第一个“标志位”是随机的，那么就意味着我们加密后的结果也是会非常多的。

当然好消息是要破解这个加密，找到我们想找的加密信息，不用算法也可以，因为我们已经确认了密的结构。

比如说：密文是：This(ASCII值：84,104,105,115)。甭管标志位怎么随机，把密文每个字符转成ASCI值[x1,x2,x3,x4,x5]。总有：x3-x2=104-84+1=21; x4-x3=105-104+1=2。

写个脚本穷举一下就出来了。

但是，但是，我想要加密算法。

那么就接着推吧

首先要来一个“标志位”。

ASCII中有不少控制字符是显示不出来的，还有双引号什么的，所以不要直接和ASCII码死磕,把他们先过去，挑一些正常的。

```
var charset = [
```

```
"!", "#", "$", "%", "&", "'", "(", ")", "*", "+", ",", "-",
".", "/", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
":", ";", "<", "=", ">", "?", "@", "A", "B", "C", "D", "E",
"F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q",
"R", "S", "T", "U", "V", "W", "X", "Y", "Z", "[", "\\", "]",
"^", "_", "`", "a", "b", "c", "d", "e", "f", "g", "h", "i",
"j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u",
"v", "w", "x", "y", "z", "{", "|", "}", "~",
];
// 第一步先随机一个字符串
var rand = Math.ceil(Math.random() * (0 - charset.length + 1) + charset.length - 1);
// 接下来逆推公式
var t = charset[rand];
var n = t.charCodeAt(0);
for (var i = 0; i < e.length; ++i) {
var a1 = e.charCodeAt(i); //把字母拿出来
var a2 = (i + 1) + n & 127; // 偏移量加 1
var b1 = a1 ^ a2; // a^b=c => a=b^c;
t += String.fromCharCode(b1);
}
return t;
```

找几个数据试试看：

```
log_info_all("Hello World!");
log_info_all("This is a demo");
```

结果：

```
----------------
words:  Hello World! 12
encrypt:  4}S[TV1SOR[a 13
decrypt:  Hello World! 12
true
----------------
words:  This is a demo 14
encrypt:  X22/}7,@B  15
decrypt:  This is a demo 14
true
```

然后就看到了这两句话：

```
----------------
words:  2g\\EQJZY[R/a4&66/(&i%-19?66&     35
After S(e):  This is a demo version of CKFinder 34
----------------
words:  ;tXRS/a$&())0h*8*/&+=qq6t4$2x+?:01'CHL
P
WZ nwq$dvwd`ijxdaa0<2dq5fbl9vthn>pFGEBJTS]EO_KNDTZ@ 122
After S(e):  Hello fellow cracker! We are really sad that you are trying to crack our
application - we put lots of effort to create it 121
----------------
words:  Ovowx=gpUNJO@SG MNXOVCWVv}qQW^^NRV#$,0!zf-,&k*?+*p%=s' 4:1-z"3(,
JLK]\Ve/ali*eln`doi= 30=ta|rr7}mCil~NRNBPLII[ 150
After S(e):  Would you like to get a free CKFinder license? Feel free to submit your
translation! http://docs.cksource.com/ckfinder3/#!/guide/dev_translations2<0: 149
----------------
```

好的吧。

那我就不搞了。