HW防守 | 基于冰蝎的特征检测

SuPejkj 渗透Xiao白帽 今天

前言

临近HW，作为萌新的我在网上找了许多大佬们关于"冰蝎"流量特征的文章，以此作为分享，在我们正式HW时，可能会有所帮助，当监控设备发现这些流量，作为防守方也能做出准确的判断。避免丢分甚至服务器被端🐱

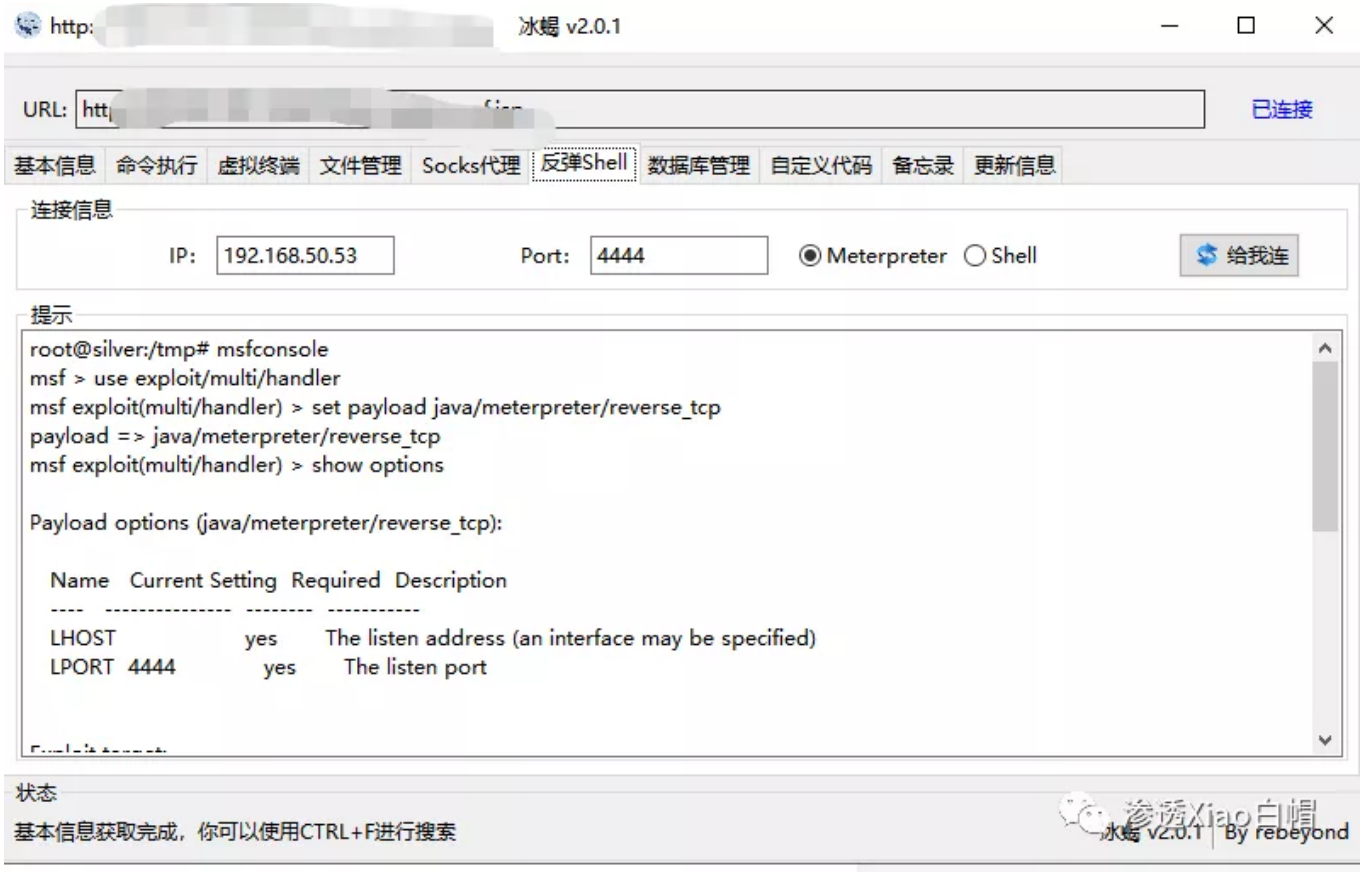
本文内容大多来自网络收集（出处会标在下文）如有问题还望各位大佬指正。

莫要喷俺

0x00 简介

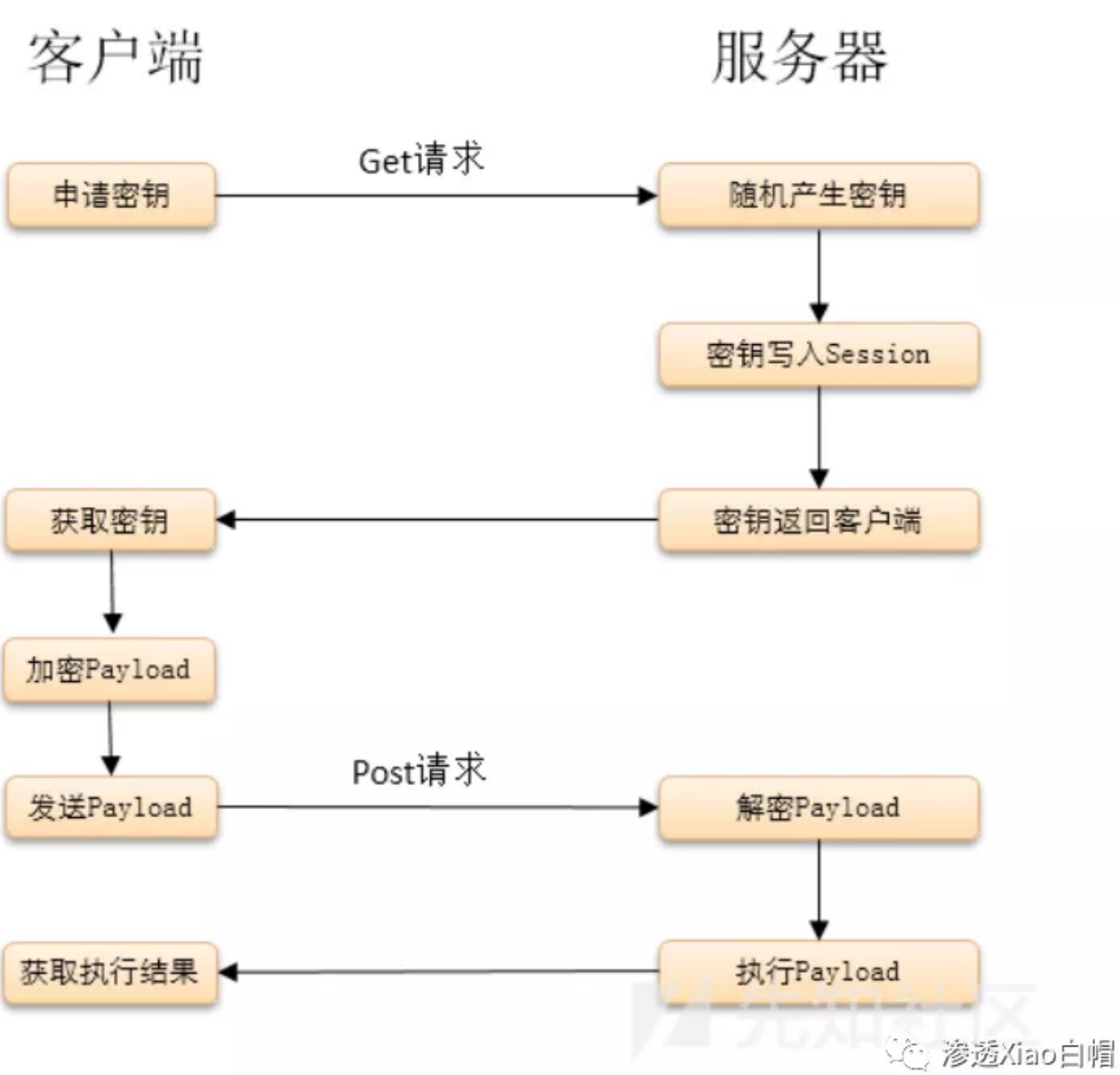
"冰蝎"是一款基于Java开发的动态加密通信流量的新型Webshell客户端。

老牌 Webshell 管理神器——***中国菜刀***的攻击流量特征明显，容易被各类安全设备检测，实际场景中越来越少使用，加密 Webshell 正变得日趋流行。由于通信流量被加密，传统的 WAF、IDS 设备难以检测，给威胁狩猎带来较大挑战。冰蝎其最大特点就是对交互流量进行对称加密，且加密秘钥是由随机数函数动态生成，因此该客户端的流量几乎无法检测。冰蝎目前最新版本为v2.0.1，兼容性已经日益完善，包括虚拟终端、Socks代理、文件管理、反弹shell、数据库管理等强大的功能，方便部署使用。



0x01 "冰蝎"获取秘钥的过程

图片来自红蓝对抗——加密Webshell"冰蝎"攻防



冰蝎在连接webshell的时,会对webshell进行两次请求访问

Request

Raw | Params | Headers | Hex

```
GET /wls-wsat/.conf.jsp?admin@123321=799 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727;
.NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; InfoPath.3)
Host: [redacted]
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: close
Content-Length: 267


HTTP/1.1 200 OK
Connection: close
Date: Fri, 05 Jun 2020 15:16:35 GMT
Content-Length: 16
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: JSESSIONID=ySWFDiFZdW0b2OMZz6qlD26D-jg7USx3J_odAxH-hfl0JFfQioRK!-1047449674; path=/; HttpOnly

bc4b7c3182d6377b
```

Request

Raw | Params | Headers | Hex

```
GET /wls-wsat/.conf.jsp?admin@123321=33 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727;
.NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; InfoPath.3)
Host: [redacted]
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: close
Content-Length: 267



HTTP/1.1 200 OK
Connection: close
Date: Fri, 05 Jun 2020 15:16:42 GMT
Content-Length: 16
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: JSESSIONID=CMGFDj1l4oj2Aksf5BVOS0bk2-Ly3OE1qLwAEtlUEMXiUoGGAyFN!-1047449674; path=/; HttpOnly

964229ad1a4e45cf
```


Request

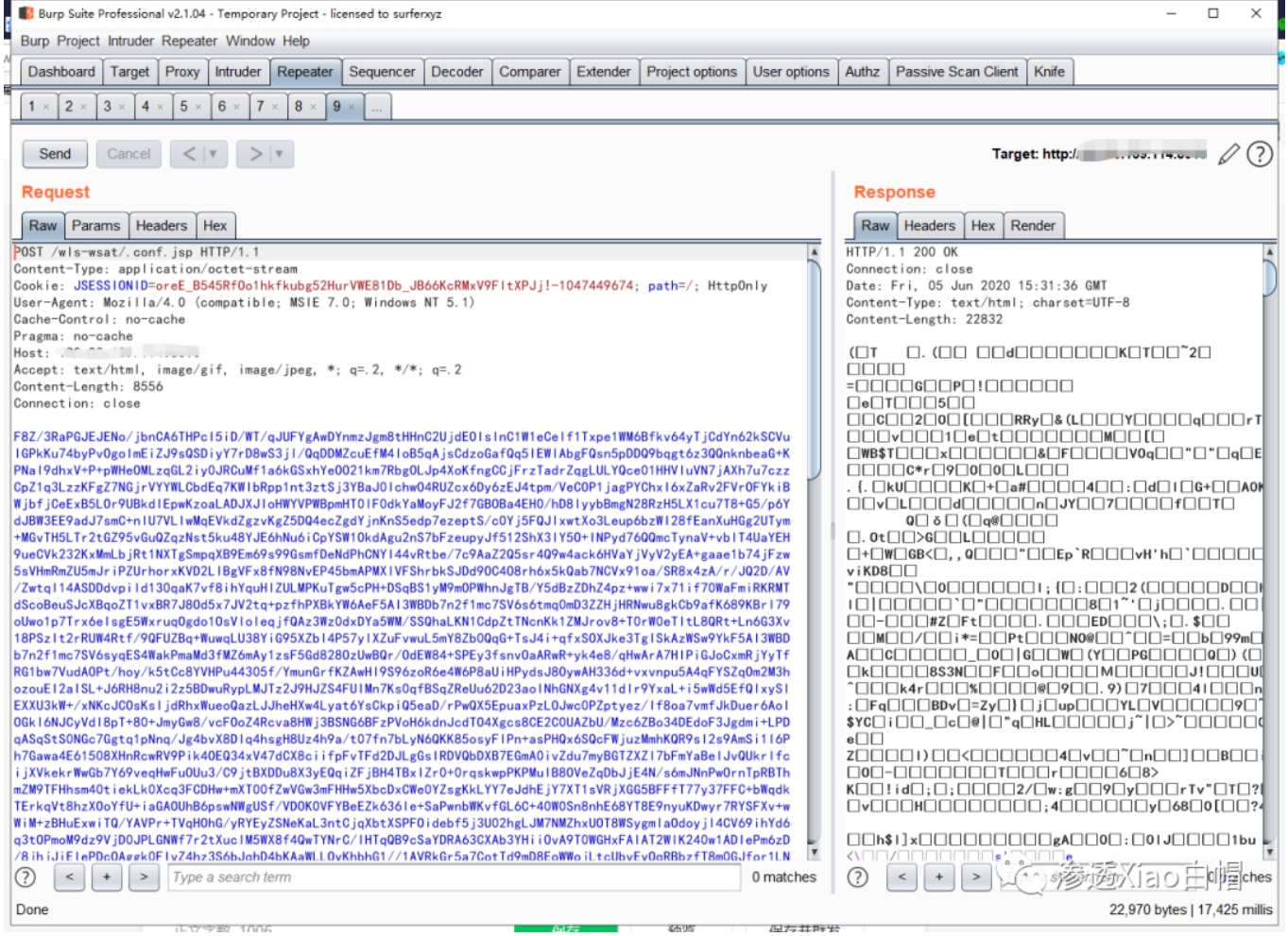
Raw | Params | Headers | Hex

```
POST /wls-wsat/.conf.jsp HTTP/1.1
Content-Type: application/octet-stream
Cookie: JSESSIONID=CMGFDj1l4oj2Aksf5BVOS0bk2-Ly3OE1qLwAEtlUEMXiUoGGAyFN!-1047449674; path=/; HttpOnly
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727;
.NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; InfoPath.3)
Cache-Control: no-cache
Pragma: no-cache
Host: [redacted]
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Content-Length: 8556
Connection: close

Hu23bXudxhGqx766N++ZXIwhPQ7E3Ofdlgh02P/WLWzR8HSkR0fXHjn7d5GJFnEgir5PdF8sBMq1NwqY19qmwqmyMWaTShHKIXbX8hWKT9o4EntN
D417XCWelg696w2uO5bCloIf+PBpWlJlJ8c/nNMqYxHxTi6ATyoxBx4kuoqomqg8xdVrP9gDWOe1GkQAExdXJchZNEs9okkJqGuJ37VZCjbVQos
CnzBH/tB6HYrRLnqRNQtlTQWBzWSQQEqjLVhNEFIhObqiaao7YOGjZeQ9SUBfxWP4OCxOMgu1dKicoUxSCEl6n7ENKgv4fN5BWHeu/kWIziXOEB
dPPYpVOiqLTdeZT6sjeFlYLpnsReH2tG5nj69n+65fbrLEVoE8K14aimrrknbzEsBdCb9x86TZJqVBEv9SlQ+skTxW2yEN4uog3D8Zzcad1zJtl
SMZ1XDVTDtT5rSpb81evpftWGFmkk/luHcatJzpH5nVHEgrpji6+HO1TY9cpdDPDD8a6Etf/dwfzME16SC6AonX5QOCXLCVhm+x8IrC687SOlxl
+OiRYyvFygYOUGy4deknRBmMtOyDPwsjT4pXlgFhkytYe8/1y7MCW8UoYlI4PBx9+cGKkQNWyXcN7RCteoyFeT+Z3LUT3sxZfpFmY52h1Ysri9F
n8YuUlpmZc0V14pOk/lWhSNGVzQ22FlLpZEPJyIHpxCxkvmdLscUt/s8/a0YYbO1ElAZhM8YmZEVeGzw5LPjmWjqsVSwOsxNiq4vs6wnbvl4Baf
v9FKHXLrmmpU15I8+ATKatTG7uz6ew1bV/drWEpOY4CLqHOpvUDfSUe+5waOYkuTPvu/Y5yqtsBlfZogn1chSxiGHacUpjrUUT9uH+Iq228An5w
wf3SyEgt2MaiEqFPOPHCW25I14kCdi3q1PrTr/FldkUzmHVnlkOJ6idj1zz/Wn9Uch1cEL2DayvezjjuWS3zBfTRnrpwFqrW5oCX7IG52iDjdfQ
IpyasU6068ixN8XKUC4IxKtv4AP3xLSGq6iR7ZVHshyWXJJDmhJRlez0L5hb3ZG7XDPWBUJRsmourlb673rmI1OOXNGWfr/eLeoVJIxybojEhwW
PvNrMjLdpJU3xQqaMGyXmchoGcpZkYPoc9e2VWDJHfq3ARGBkZXz1brz662xcUFT0ttIbjRMdOiqMiRJsYTfZyN88IfoQyVZAX4ziqifELdVQeB
iYJOb+5DCO+zBHX1cGVJXNBmnyflnVsbOsMPvGZd5Y2JKrUKI3zCpTqVXNOLL98LbXkYbfXu0e46HqFadssDJtMj0vbE5YfruVqXWdiWmbJyiNG
RRzCLkKwdXDXNGWfr/eLeoVJIxybojEh6T8uzniBKNZbTd8PnWOJloiozwLDJIMOoWjSUSIMyrXqk3VAgHVuBWzcZff8pvmIrrqC5eNTmeDCFEw
NNpGWEoOY816IZ+wLejNuE/jFjhd3vDpc348caSzA34yyZwna11vm4XPWJ1SuhQ9cUXw3Nm2XC4VwJpb6/qZcXQicRoSmWFQrojEOEqp5pcKw4i
H14fMJxgS3n2B+RUFKA3nBDJgOvb4lZjcOOUjGxTOtyrQjoFWx03Sje2Sc05h7YW1G17LivyxRAqMYUvMi7fHDUZGK6HqsxbaAUpAOAVIxyNe7Q
LOIsCfxZAWAByhwHjhyGW31UjSQYCjUhMzQtRKQRTiFYqq85ftFZswy7ObpKHShN/nSaFWe4GEosvTiYU/CyQ3W5PS+GQyLyjshXxEdGgvgchqF
Wc423ifSOi/4eC7ScZaVcxV2A8u6hctxS9oo7IQB4WIE2uzjscRhqVtw9kvJbghdiEm2KHSBeltm8IMzBoQr5eqJG3KxVzzUmntw3v5H9O/fpRH
VztiYfAmMGzJKk5HCkGvU2NTNJ6aHrzmzhzpdo5IZphJINcEAaGxxs+sAZqo5xUnM429kHeF+/JGlaxuhiqJQ4wtWTocUOgkpyVzcQey7U2pRs
12tBmzPpyAC4STAeKUh7zL2XjP3CtOgPcq+jCHiv59Sp50D4+6iJrEcL5GaQj+G+2WI8aA+V4mjVld/rGiIbhXf8M3AscyzFXfQ6gqenWs1G98y
fFvzFTnagLcvcToHVx5aVkh7CytxR1MicUs6oA8Guyp6+ExndIrNX/3zSLfIqNQJze9sNU5BpO9H58+Fmk2eM7wT4PwW2tI2Szz9xwJzcIR/lXq
Zng+x4NLhu5e0wPLhg6bLZirGGWRFbr66cnjIDSSz6uieEQLM3xHV8qoEA+eP1tDbnGbvjLASuTanTKzSwgPvOdLHbUGFAQ9cgESGx1TaWc9CKO
T8bM7vcXiUD+8icr/R/VthZHL8kK2clXHaCnvitH8AGOw6k/kt1PIOS/LBXpBXKPWeRx2yZTUb1zTOlJ3vO736lQqyjkfyI2VcWOtQW1ONtApJc
tNp8W5sQTmBpmTyPWYnGkQOV21av57S3+Xbxpx4DQ8KIPMaMq+Z7NwVQQQq2jYF4LsBlbKeiX2c5B2+ClTclr+9MorFB83/K46OmxjwIdQaiOqk
a3045jEinDVZUQFkDjmchdirAvm9j+GfpVUvLWEVfLPvtneXaL+T+L7tLd1tCi19tzbXBhFJc1OPOZfjTkYftIbk/30cZl3IjYkqtQqXfMKlOpV
```

Response

Raw | Headers | Hex | Render

```
HTTP/1.1 200 OK
Connection: close
Date: Fri, 05 Jun 2020 15:28:27 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 22832

|□□#□□□□□□-□□v□0□□□q□4z!b□□-□□
□□□%□□|V□□:□□[V□4U□□□=□□□□□□
□□□{□1□□T□N□□□{□[9□sr□□2□□□dp
□□Cm;7□6□\□□*□□□□~<h□;□>□□bGq□□
□□□A`□□□d□A□□□□□o;+□□8r□□*"□□
□;&□t,□l□□□;□XQ:□□□□□□~Y□$[□9□
□b□$□□9□□2□□i□:,9!b□□#□□□□□_□
□□□□□□□□□□□□□□□
h$lo□o>i□|fm3□□3□□□,□□□□□□□□□S□
□□□□□(WN□□Mj□h□□□□)□□Y□□f□a□c
uAl□□+V%□□□□□ail□□K□□□□`□□□□□
□l□}□□□□□Ou□□□□□□□/□□~z□□□
□V□`1□V□□□□□v□□□□□X□□□f□□z□□
M□□□□□□□
□□□□XnJt;]□4a□□□□c□□□□'G□□□□□
;g□JD□Q□□A□□□□□o□□F□□'L□□□TC□
□□□□r
□[V□X□□c□□□z□□□=w\W [

□□□0□□□'D□u□u□v\4□Dhs□□□Va□□□c
□□k□□i□□□□□□□□□□□□BX□□□E□□}$

□□R□□□□□M□V□□□E□□□□□□□□□□□□□
□□V}p□*□□j1□□(
n□□T□□7□□□.□□□□□m□□□w□N□,□□□
#w3.s□□□□m□□□□□□□□□□□□□□□□□□
□□□□□□□□□□8*m0□□□f□□□;□9□7i□
E□jLZ□□□n□B;□□.□]□mOV□□□□8□□[□
```



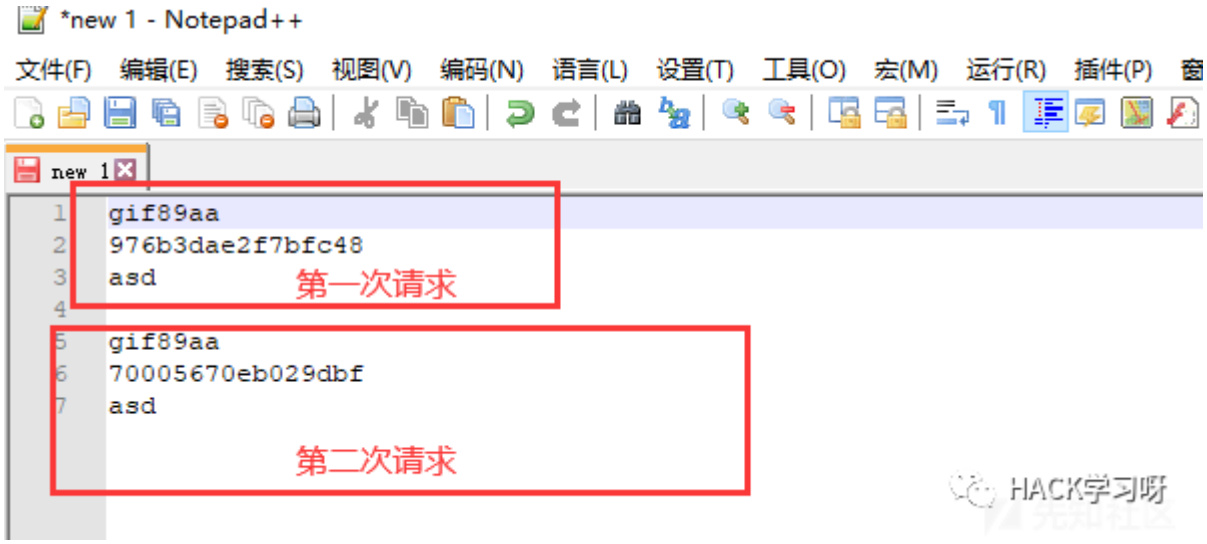
为什么进行两次访问?在别的文章没有看到关于这个问题的答案,于是去反编译冰蝎源码

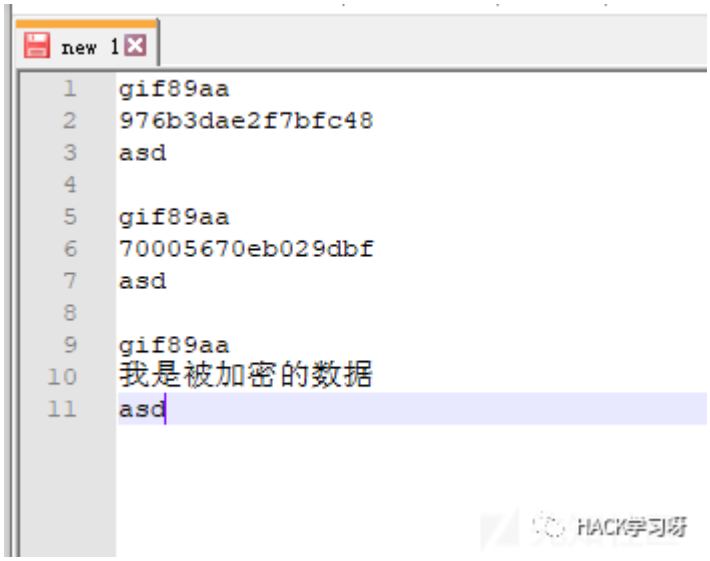
```
180     if (error) {
 98         throw new Exception(errorMsg);
        }
187     String rawKey_1 = sb.toString();
194     if (!Pattern.compile("[a-fA-F0-9]{16}").matcher(rawKey_1).find()) {
195         throw new Exception("页面存在，但是无法获取密钥!");
        }
198     int start = 0;
198     int end = 0;
199     int cycleCount = 0;
        while (true) {
201         Map<String, String> KeyAndCookie = getRawKey(getUrl, password, requestHeaders);
202         String rawKey_2 = KeyAndCookie.get("key");
203         byte[] temp = CipherUtils.bytesXor(rawKey_1.getBytes(), rawKey_2.getBytes());
204         int i = 0;
            while (true) {
204             if (i >= temp.length) {
                    break;
206             } else if (temp[i] <= 0) {
204                 i++;
207             } else if (start == 0 || i <= start) {
208                 start = i;
                }
            }
            int i2 = temp.length - 1;
            while (true) {
212             if (i2 < 0) {
                    break;
213             } else if (temp[i2] <= 0) {
                    i2--;
214             } else if (i2 >= end) {
215                 end = i2 + 1;
                }
            }
219         if (end - start == 16) {
220             hashMap.put("cookie", KeyAndCookie.get("cookie"));
221             hashMap.put("beginIndex", new StringBuilder(String.valueOf(start)).toString());
234             hashMap.put("endIndex", new StringBuilder(String.valueOf(temp.length - end)).toString());
238             hashMap.put("key", new String(Arrays.copyOfRange(rawKey_2.getBytes(), start, end)));
240             return hashMap;
227         } else if (cycleCount > 10) {
228             throw new Exception("Can't figure out the key!");
            } else {
231             cycleCount++;
```

```
}

public static Map<String, Object> requestAndParse(String urlPath, Map<String, String> header, byte[] data, int beginIndex, int endIndex) throws Exception {
    Map<String, Object> resultObj = sendPostRequestBinary(urlPath, header, data);
    byte[] resData = (byte[]) resultObj.get("data");
    if (!(beginIndex == 0 && endIndex == 0) && resData.length - endIndex >= beginIndex) {
        resData = Arrays.copyOfRange(resData, beginIndex, resData.length - endIndex);
    }
    resultObj.put("data", resData);
    return resultObj;
}
```

```
X-Powered-By: PHP/5.5.38
Set-Cookie: PHPSESSID=s3n3is01p1kb34sc5ffafcshb3; path=/
X-Powered-By: ASP.NET
Date: Tue, 14 Apr 2020 13:17:18 GMT
Content-Length: 25

gif89aa
976b3dae2f7bfc48
```


通过对代码阅读,可以发现冰蝎为了实现可以在webshell内添加任意内容 (比如gif89a子类的文件头或者其它标示字符) 冰蝎在初始化密钥时会对webshell进行两次访问,然后比较两次页面返回的差异,把两次请求都相同的字符记录一个位置,后续加密会用到这两个位置(beginIndex,endIndex)






```
if (!(beginIndex == 0 && endIndex == 0) && resData.length - endIndex >= beginIndex) {
    resData = Arrays.copyOfRange(resData, beginIndex, resData.length - endIndex);
}
resultObj.put("data", resData);
```

```
try {
  response.setContentType("text/html");
  pageContext = _jspxFactory.getPageContext(this, request, response,
            null, true, 8192, true);
  _jspx_page_context = pageContext;
  application = pageContext.getServletContext();
  config = pageContext.getServletConfig();
  session = pageContext.getSession();
  out = pageContext.getOut();
  _jspx_out = out;
```


)

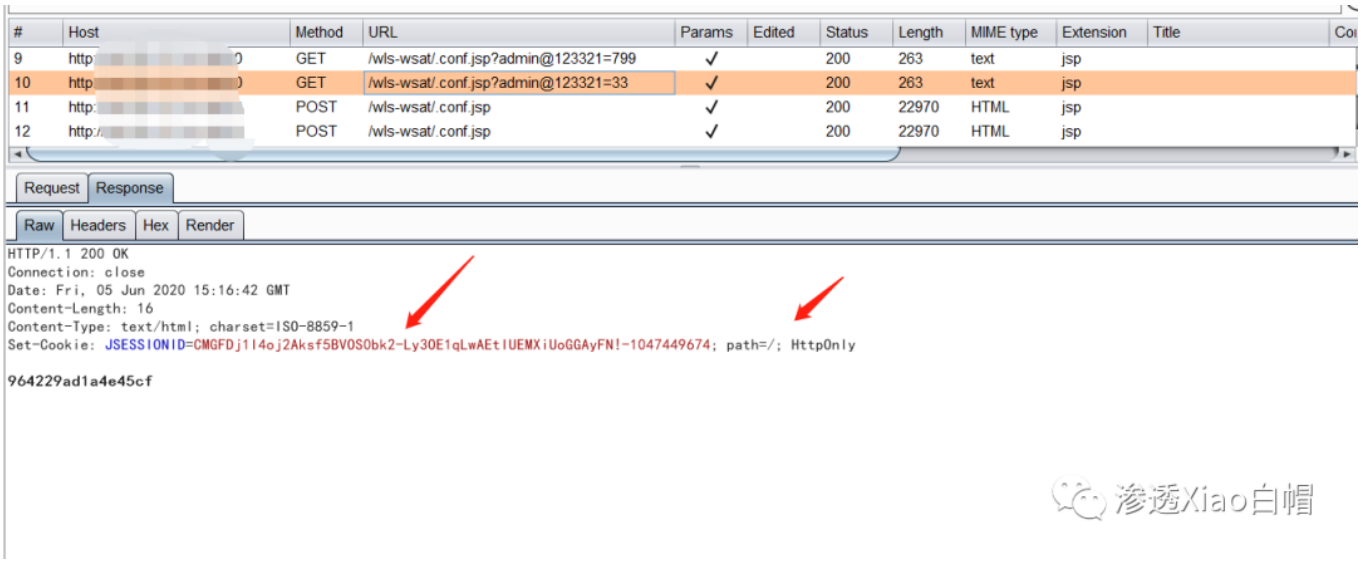
如图,根据数据包,beginIndex:8 endIndex:4 (含换行),冰蝎开始从数据流中截取被加密的数据从下标8开始到(数据包总长度-4)

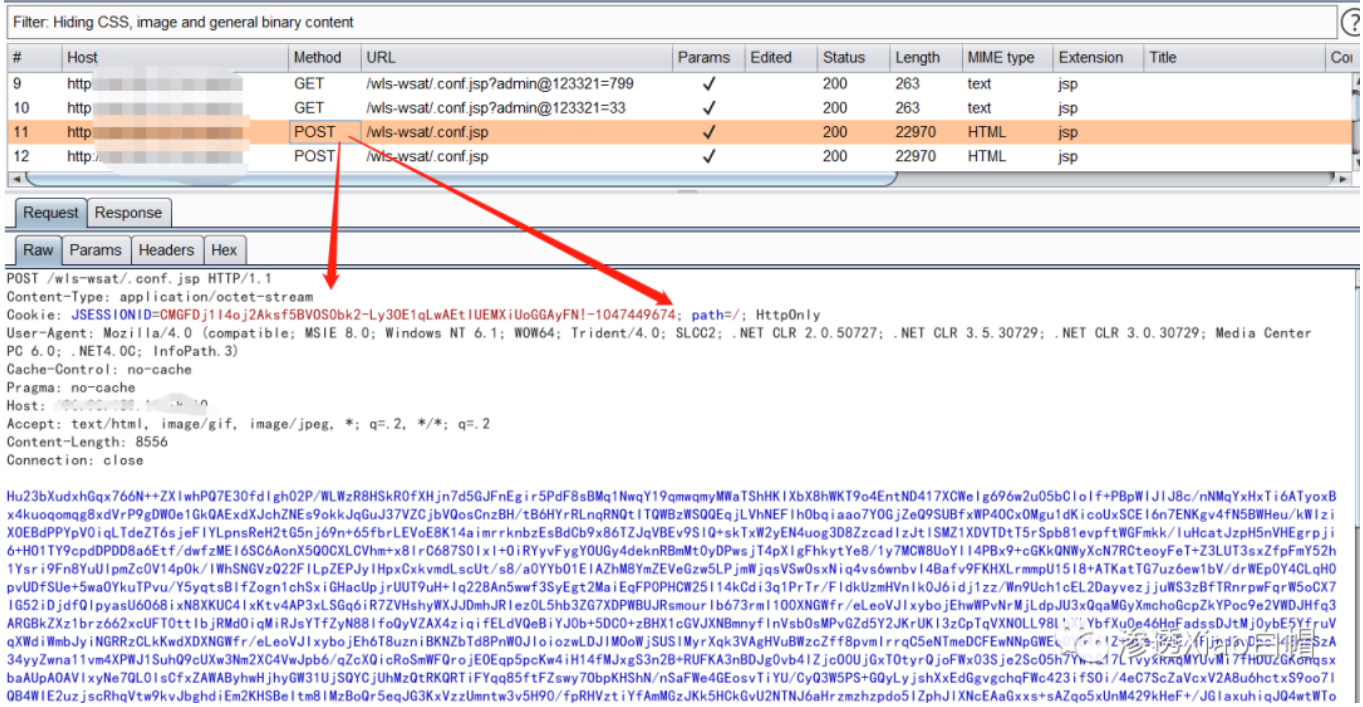
Waf可以针对于返回类型为 "text/html" 的数据包中加一些空格或者换行,来扰乱冰蝎的数据包,导致冰蝎无法运行。

(为什么要对返回类型为 "text/html" 的扰乱,别的格式不可以吗?

答案:jsp默认返回类型就是 "text/html" html添加一些空格或者换行,并不会影响网页的正常运行)

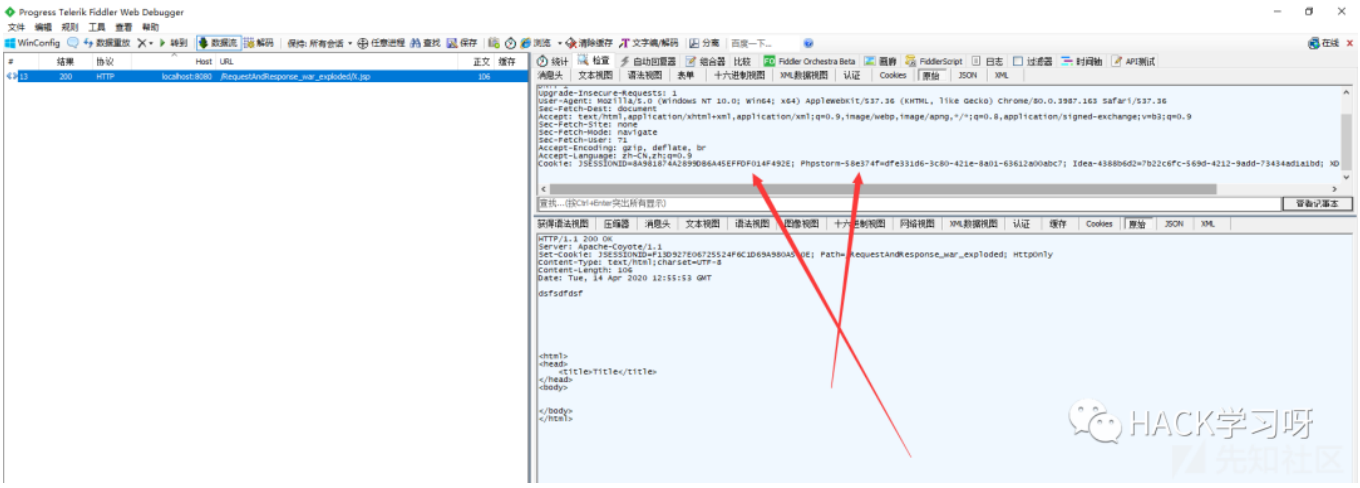
0x02 "冰蝎" 解析Cookie流程





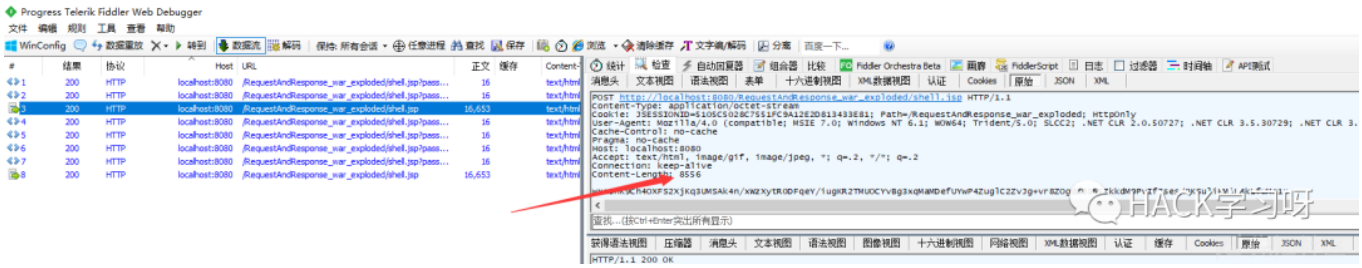

```
Iterator<String> it = headers.keySet().iterator();
while (true) {
    if (!it.hasNext()) {
        break;
    }
    String headerName2 = it.next();
    if (headerName2 != null && headerName2.equalsIgnoreCase("Set-Cookie")) {
        for (String cookieValue : headers.get(headerName2)) {
            cookieValues = String.valueOf(cookieValues) + ";" + cookieValue;
        }
        if (cookieValues.startsWith(";")) {
            cookieValues = cookieValues.replaceFirst(";", "");
        }
    }
}
```

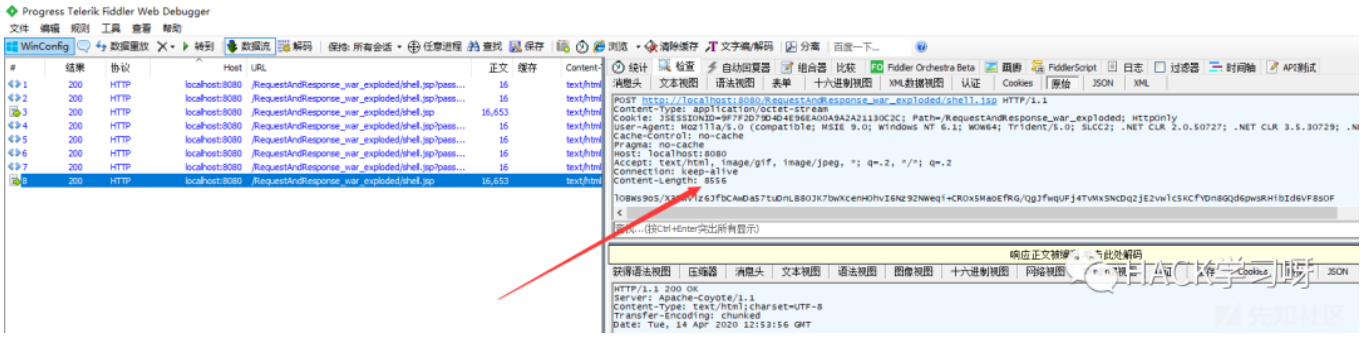
我们可以看到请求协议头中的Cookie字段,冰蝎在合并处理Cookie的时候没有考虑到,Cookie的一些属性 (比如 Path 或者 HttpOnly 之类或者其它的) 冰蝎直接把返回协议头中的Set-Cookie字段直接添加到下一个请求包的Cookie字段中



正常的请求是不会携带Cookie属性的,这可是识别冰蝎流量最直接的一种办法

0X03 "冰蝎" 动态加载





冰蝎动态加载的原理就是每次都发送一个class字节码(其它语言也一样) 冰蝎通过asm动态修改class字节码变量内容,实现携带参数动态执行,冰蝎在获取完密钥之后(2个请求),第三个请求就是获取BasicInfo(服务器的一些信息),冰蝎的BasicInfo功能并没有动态修改参数(一个获取服务器信息的能有啥参数),这会导致每次获取BasicInfo的数据包都是固定的大小。

0x04 UserAgent字段（可绕过）

冰蝎内置了十余种UserAgent，每次连接shell会随机选择一个进行使用。

以下UserAgent列表是从冰蝎的jar包中提取的，可见大多是比较早的浏览器，现在很少有人使用。而且有些国产浏览器甚至精确到了小版本，众所周知，很多国产浏览器是默认自动更新，正常用户很少用过早的版本，因此可以作为强特征使用。

如果发现历史流量中同一个源IP访问某个URL时，命中了以下列表中多个UserAgent，那基本确认就是冰蝎了。

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1(KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1

Mozilla/5.0 (Windows NT 6.1; WOW64; rv:6.0)Gecko/20100101 Firefox/6.0

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/534.50(KHTML, like Gecko) Version/5.1 Safari/534.50 " BOpera/9.80 (Windows NT6.1; U; zh-cn) Presto/2.9.168 Version/11.5

Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64;x64; Trident/5.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C; Tablet PC 2.0; .NET4.0E)

Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64;Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;Media Center PC 6.0; .NET4.0C; InfoPath.3)

Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1;Trident/4.0; GTB7.0)

Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1) , 7

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)

Mozilla/5.0 (Windows; U; Windows NT 6.1;)AppleWebKit/534.12 (KHTML, like Gecko) Maxthon/3.0 Safari/534.12

Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64;Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E)

Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64;Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E; SE 2.X MetaSr 1.0)

Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US)AppleWebKit/534.3 (KHTML, like

Gecko) Chrome/6.0.472.33 Safari/534.3 SE 2.XMetaSr
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64;Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E)
Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.1 (KHTML,like Gecko) Chrome/13.0.782.41 Safari/535.1 QQBrowser/6.9.11079.20
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64;Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729;Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E) QQBrowser/6.9.11079
Mozilla/5.0 (compatible; MSIE 9.0; WindowsNT 6.1; WOW64; Trident/5.0)

同样，UserAgent可由黑客自定义，因此该特征可能会被绕过。

0x05 总结

WAF可以对一个ip连续访问2次的数据包进行截取,比对相同字符,比对之后,截取两次不同的数据,如果剩下的是16位的key,就可以证明这两个数据包就是冰蝎发出的,第三个数据包通过***“冰蝎”解析cookie流程***和***“冰蝎”动态加载***中的一些bug,可以100%的匹配到冰蝎流量,不会误报。

0x06 参考链接

```
https://mp.weixin.qq.com/s/ZD34UQ0gP5cvDctVoIatIA // HACK学习呀 BeichenDream
https://www.freebuf.com/articles/web/216133.html
https://zhuanlan.zhihu.com/p/135227454
https://blog.csdn.net/dianzhongsou2379/article/details/100599116
https://www.cnblogs.com/guojia000/p/11641023.html
https://blog.csdn.net/qq_36334464/article/details/99978193
```

这里推荐一位大佬的文章聚合平台

所有资料应有尽有🤔

```
http://wechat.doonsec.com
```