湖湘杯 | Misc Wp

原创

Sn0w/ 于 2020-11-09 16:14:00 发布 1990 收藏 2

分类专栏： CTF_Writeup 文章标签： 安全

本文链接：https://blog.csdn.net/qq_43431158/article/details/109514249

版权

CTF_Writeup 专栏收录该内容

32 篇文章 4 订阅

订阅专栏

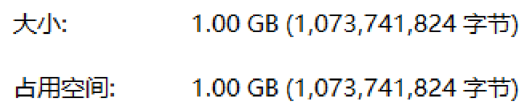
前言

全程卡死在验证码处，体验感爆棚，上午打铁三，下午吃完饭就进不去，就随便做做MISC算了。

MISC

0x00:passwd(50pt)

> 提示：we need sha1(password)!!!

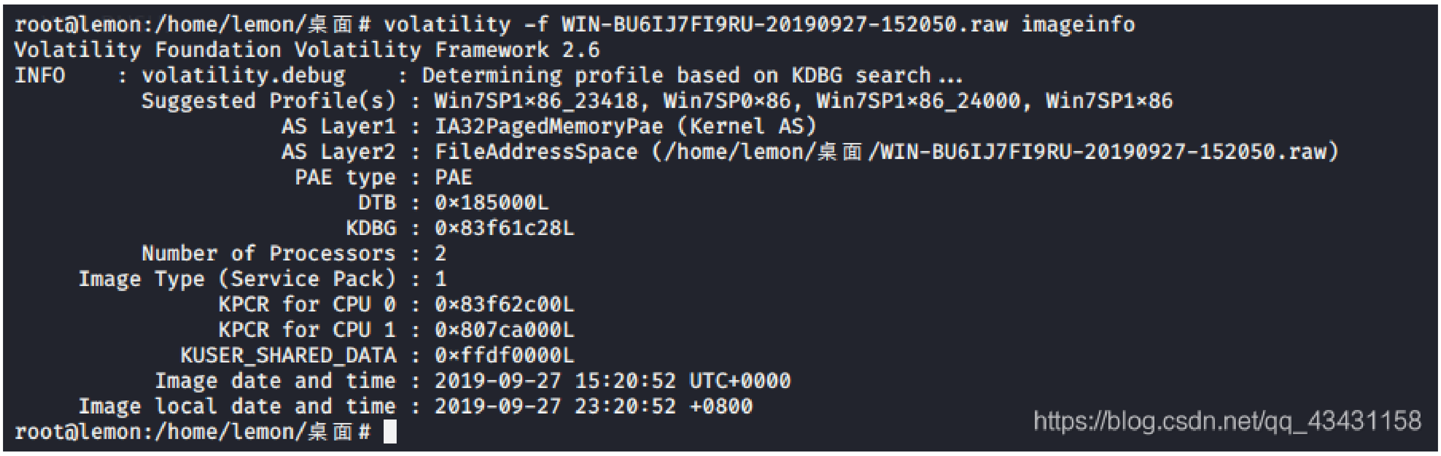

大小: 1.00 GB (1,073,741,824 字节)

占用空间: 1.00 GB (1,073,741,824 字节)

这么大，结合提示就是内存取证了

先看一下镜像信息

```
volatility -f WIN-BU6IJ7FI9RU-20190927-152050.raw imageinfo
```

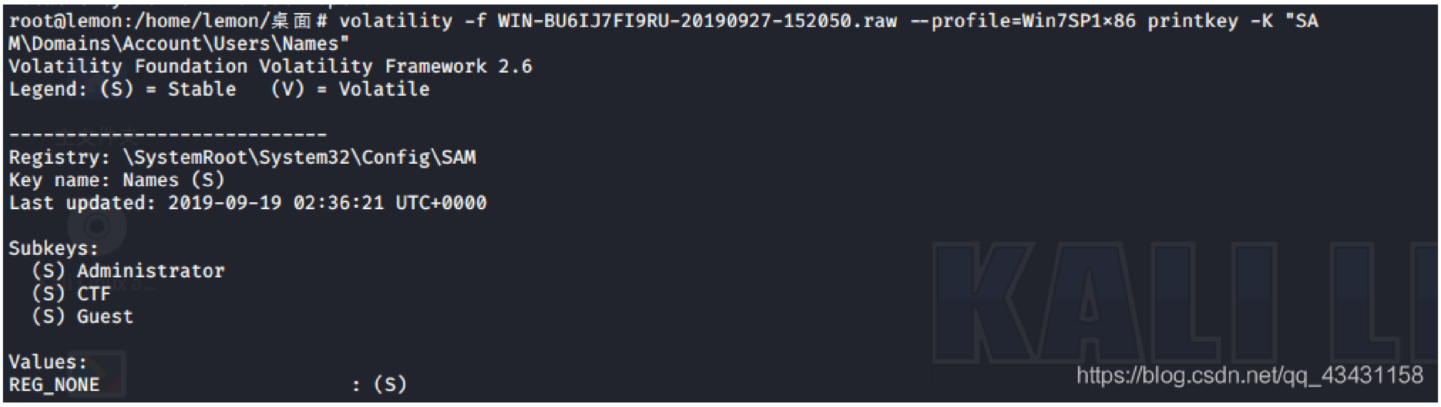


```
root@lemon:/home/lemon/桌面# volatility -f WIN-BU6IJ7FI9RU-20190927-152050.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/lemon/桌面/WIN-BU6IJ7FI9RU-20190927-152050.raw)
                      PAE type : PAE
                           DTB : 0x185000L
                          KDBG : 0x83f61c28L
          Number of Processors : 2
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0x83f62c00L
                KPCR for CPU 1 : 0x807ca000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2019-09-27 15:20:52 UTC+0000
     Image local date and time : 2019-09-27 23:20:52 +0800
root@lemon:/home/lemon/桌面# 
```

https://blog.csdn.net/qq_43431158

再列举一下用户及密码看看

```
volatility -f WIN-BU6IJ7FI9RU-20190927-152050.raw --profile=Win7SP1x86 printkey -K "SAM\Domains\Account\Users\Names"
```

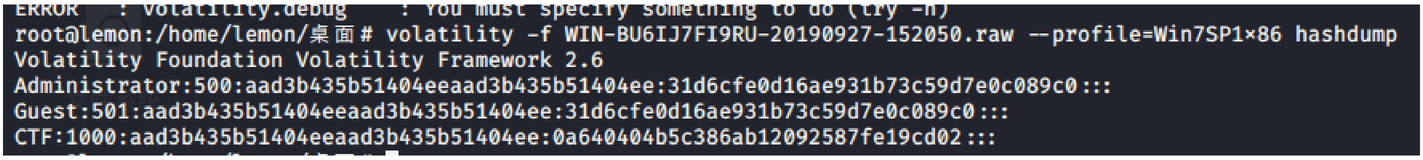


```
root@lemon:/home/lemon/桌面# volatility -f WIN-BU6IJ7FI9RU-20190927-152050.raw --profile=Win7SP1x86 printkey -K "SA
M\Domains\Account\Users\Names"
Volatility Foundation Volatility Framework 2.6
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \SystemRoot\System32\Config\SAM
Key name: Names (S)
Last updated: 2019-09-19 02:36:21 UTC+0000

Subkeys:
  (S) Administrator
  (S) CTF
  (S) Guest

Values:
REG_NONE                      : (S)
```

https://blog.csdn.net/qq_43431158

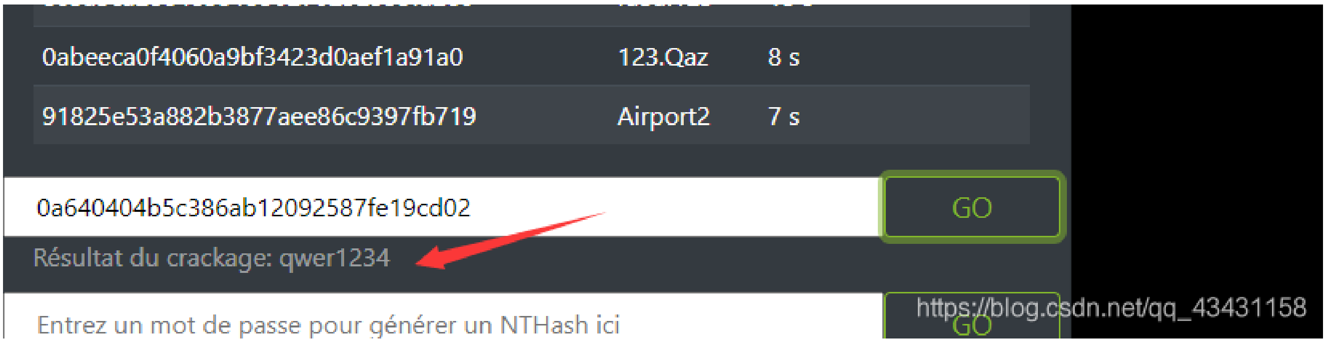
发现有一个CTF用户，那接下来获取一下CTF用户的哈希值

```
volatility -f WIN-BU6IJ7FI9RU-20190927-152050.raw  --profile=Win7SP1x86 hashdump
```



```
ERROR   : volatility.debug    : You must specify something to do (try -h)
root@lemon:/home/lemon/桌面# volatility -f WIN-BU6IJ7FI9RU-20190927-152050.raw --profile=Win7SP1x86 hashdump
Volatility Foundation Volatility Framework 2.6
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
CTF:1000:aad3b435b51404eeaad3b435b51404ee:0a640404b5c386ab12092587fe19cd02:::
```

在线破解一下即可



0abeeca0f4060a9bf3423d0aef1a91a0	123.Qaz	8 s
91825e53a882b3877aee86c9397fb719	Airport2	7 s

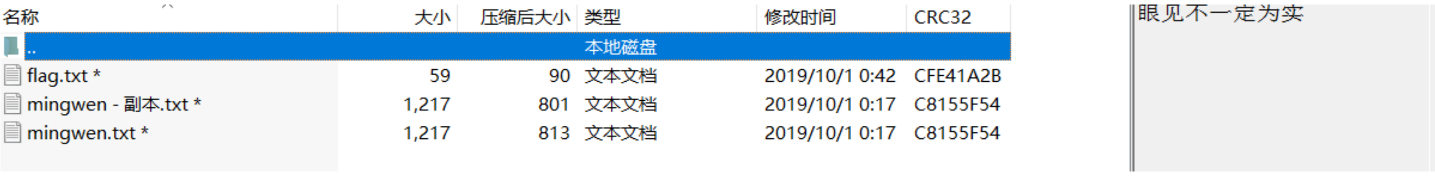
0a640404b5c386ab12092587fe19cd02

Résultat du crackage: qwer1234

Entrez un mot de passe pour générer un NTHash ici

https://blog.csdn.net/qq_43431158

0x01: 虚实之间



名称	大小	压缩后大小	类型	修改时间	CRC32
..			本地磁盘		
flag.txt *	59	90	文本文档	2019/10/1 0:42	CFE41A2B
mingwen - 副本.txt *	1,217	801	文本文档	2019/10/1 0:17	C8155F54
mingwen.txt *	1,217	813	文本文档	2019/10/1 0:17	C8155F54

眼见不一定为实

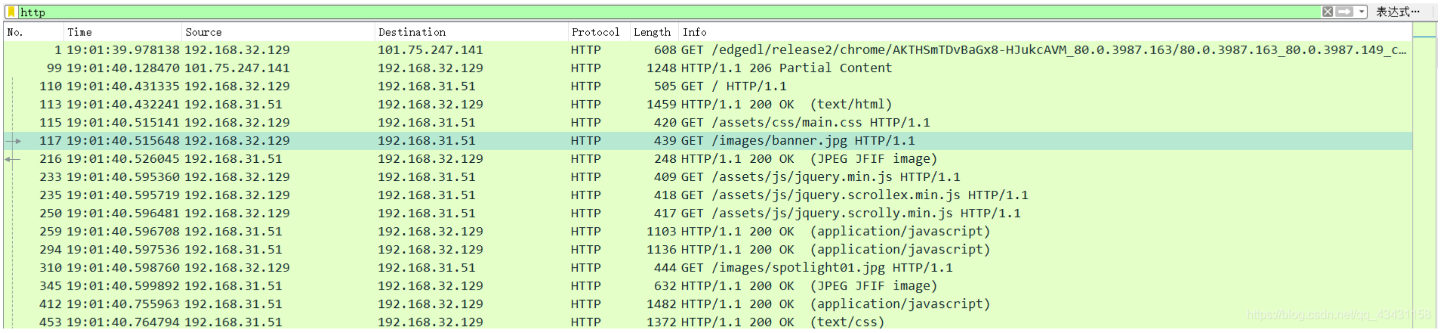
用7z打开可以直接提取出副本文件，是伪加密，看拼音和这种格式的应该就是明文攻击了，但是使用明文攻击时发生报错，两个加密文件的加密算法不一样，可能是这个原因无法进行明文攻击，删除flag.txt即可

爆破出秘钥为`123%asd!O`，之后打开flag.txt文件发现是栅栏加密，key值为5，偏移一下就可以了。

0x02: 颜文字之谜

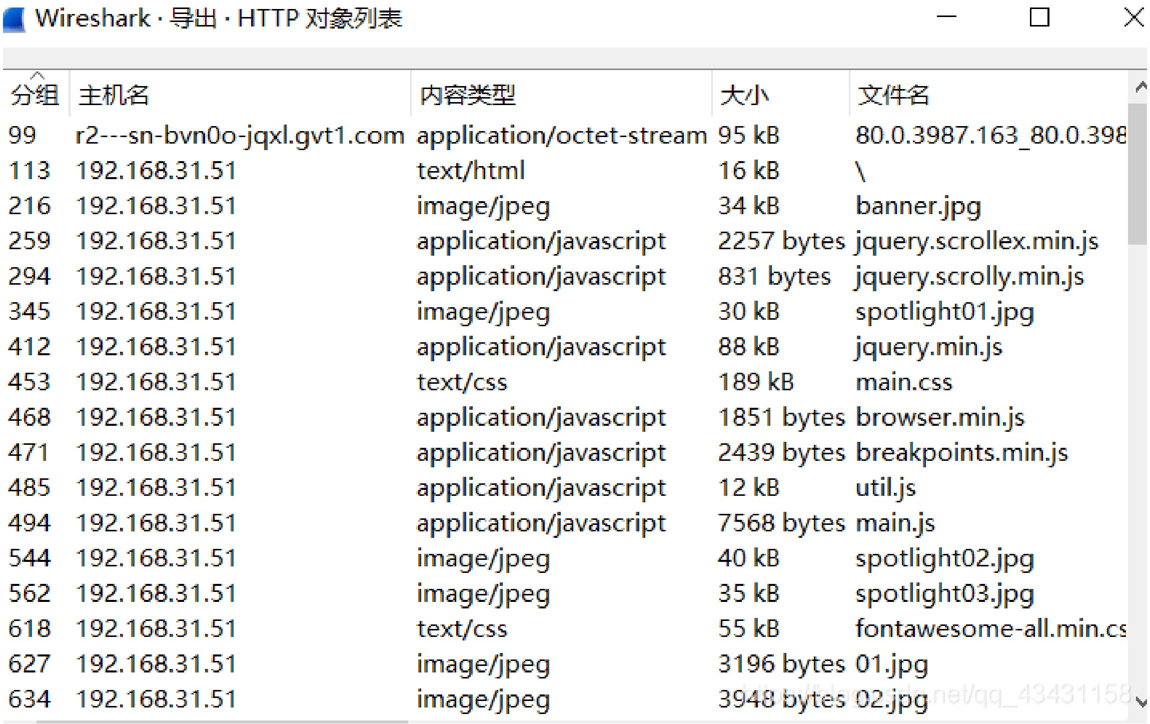
提示：颜文字把flag抢走了，你能拿到吗

给了一个流量包，过滤一下http观察一下



No.	Time	Source	Destination	Protocol	Length	Info
1	19:01:39.978138	192.168.32.129	101.75.247.141	HTTP	608	GET /edgedl/release2/chrome/AKTHSmTDvBaGx8-HJukcAVM_80.0.3987.163/80.0.3987.163_80.0.3987.149_c…
99	19:01:40.128470	101.75.247.141	192.168.32.129	HTTP	1248	HTTP/1.1 206 Partial Content
110	19:01:40.431335	192.168.32.129	192.168.31.51	HTTP	505	GET / HTTP/1.1
113	19:01:40.432241	192.168.31.51	192.168.32.129	HTTP	1459	HTTP/1.1 200 OK (text/html)
115	19:01:40.515141	192.168.32.129	192.168.31.51	HTTP	420	GET /assets/css/main.css HTTP/1.1
117	19:01:40.515648	192.168.32.129	192.168.31.51	HTTP	439	GET /images/banner.jpg HTTP/1.1
216	19:01:40.526045	192.168.31.51	192.168.32.129	HTTP	248	HTTP/1.1 200 OK (JPEG JFIF image)
233	19:01:40.595360	192.168.32.129	192.168.31.51	HTTP	409	GET /assets/js/jquery.min.js HTTP/1.1
235	19:01:40.595719	192.168.32.129	192.168.31.51	HTTP	418	GET /assets/js/jquery.scrollex.min.js HTTP/1.1
250	19:01:40.596481	192.168.32.129	192.168.31.51	HTTP	417	GET /assets/js/jquery.scrolly.min.js HTTP/1.1
259	19:01:40.596708	192.168.31.51	192.168.32.129	HTTP	1103	HTTP/1.1 200 OK (application/javascript)
294	19:01:40.597536	192.168.31.51	192.168.32.129	HTTP	1136	HTTP/1.1 200 OK (application/javascript)
310	19:01:40.598760	192.168.32.129	192.168.31.51	HTTP	444	GET /images/spotlight01.jpg HTTP/1.1
345	19:01:40.599892	192.168.31.51	192.168.32.129	HTTP	632	HTTP/1.1 200 OK (JPEG JFIF image)
412	19:01:40.755963	192.168.31.51	192.168.32.129	HTTP	1482	HTTP/1.1 200 OK (application/javascript)
453	19:01:40.764794	192.168.31.51	192.168.32.129	HTTP	1372	HTTP/1.1 200 OK (text/css)

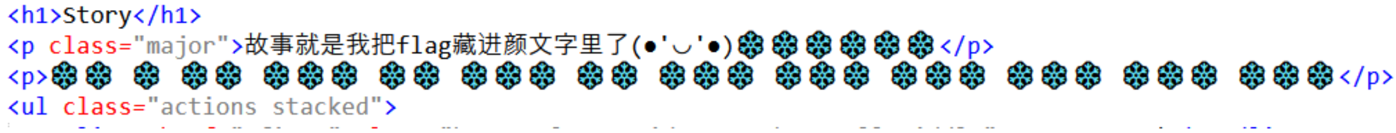
有图片还是杂项题，直接http导出看能得到什么



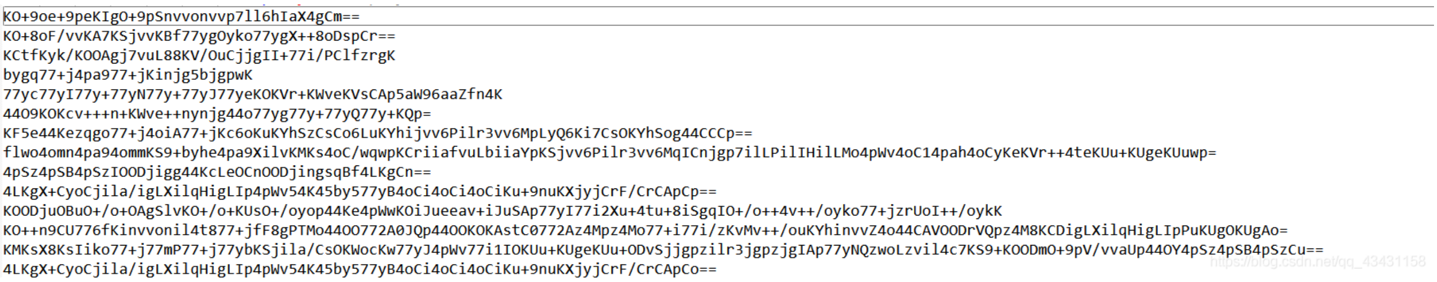
Wireshark · 导出 · HTTP 对象列表

分组	主机名	内容类型	大小	文件名
99	r2---sn-bvn0o-jqxl.gvt1.com	application/octet-stream	95 kB	80.0.3987.163_80.0.398
113	192.168.31.51	text/html	16 kB	\
216	192.168.31.51	image/jpeg	34 kB	banner.jpg
259	192.168.31.51	application/javascript	2257 bytes	jquery.scrollex.min.js
294	192.168.31.51	application/javascript	831 bytes	jquery.scrolly.min.js
345	192.168.31.51	image/jpeg	30 kB	spotlight01.jpg
412	192.168.31.51	application/javascript	88 kB	jquery.min.js
453	192.168.31.51	text/css	189 kB	main.css
468	192.168.31.51	application/javascript	1851 bytes	browser.min.js
471	192.168.31.51	application/javascript	2439 bytes	breakpoints.min.js
485	192.168.31.51	application/javascript	12 kB	util.js
494	192.168.31.51	application/javascript	7568 bytes	main.js
544	192.168.31.51	image/jpeg	40 kB	spotlight02.jpg
562	192.168.31.51	image/jpeg	35 kB	spotlight03.jpg
618	192.168.31.51	text/css	55 kB	fontawesome-all.min.cs
627	192.168.31.51	image/jpeg	3196 bytes	01.jpg
634	192.168.31.51	image/jpeg	3948 bytes	02.jpg

导出后，有一个index-demo.html文件，打开发现有段提示和一些base64编码

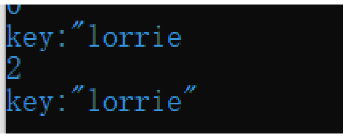


```
<h1>Story</h1>
<p class="major">故事就是我把flag藏进颜文字里了(•'◡'•)❄❄❄❄❄❄</p>
<p>❄❄ ❄ ❄❄ ❄❄❄ ❄❄ ❄❄❄ ❄❄ ❄❄❄ ❄❄❄ ❄❄❄ ❄❄❄ ❄❄❄</p>
<ul class="actions stacked">
```



```
KO+9oe+9peKIgO+9pSnvvonvvp7ll6hIaX4gCm==
KO+8oF/vvKA7KSjvvKBf77ygOyko77ygX++8oDspCr==
KCtfKyk/KOOAgj7vuL88KV/OuCjjgII+77i/PClfzrgK
bygq77+j4pa977+jKinjg5bjgpwK
77yc77yI77y+77yN77y+77yJ77yeKOKVr+KWveKVsCAp5aW96aaZfn4K
44O9KOKcv+++n+KWve++nynjg44o77yg77y+77yQ77y+KQp=
KF5e44Kezqgo77+j4oiA77+jKc6oKuKYhSzCsCo6LuKYhijvv6Pilr3vv6MpLyQ6Ki7CsOKYhSog44CCCp==
flwo4omn4pa94ommKS9+byhe4pa9XilvKMKs4oC/wqwpKCriiafvuLbiiaYpKSjvv6Pilr3vv6MqICnjgp7ilLPilIHilLMo4pWv4oC14pah4oCyKeKVr++4teKUu+KUgeKUuwp=
4pSz4pSB4pSzIOODjigg44KcLeOCnOODjingsqBf4LKgCn==
4LKgX+CyoCjila/igLXilqHigLIp4pWv54K45by577yB4oCi4oCi4oCiKu+9nuKXjyjCrF/CrCApCp==
KOODjuOBuO+/o+OAgSlvKO+/o+KUsO+/oyop44Ke4pWwKOiJueeav+iJuSAp77yI77i2Xu+4tu+8iSgqIO+/o++4v++/oyko77+jzrUoI++/oykK
KO++n9CU776fKinvvonil4t877+jfF8gPTMo44OO772A0JQp44OOKOKAstC0772Az4Mpz4Mo77+i77i/zKvMv++/ouKYhinvvZ4o44CAVOODrVQpz4M8KCDigLXilqHigLIpPuKUgOKUgAo=
KMKsX8KsIiko77+j77mP77+j77ybKSjila/CsOKWocKw77yJ4pWv77i1IOKUu+KUgeKUu+ODvSjjgpzilr3jgpzjgIAp77yNQzwoLzvil4c7KS9+KOODmO+9pV/vvaUp44OY4pSz4pSB4pSzCu==
4LKgX+CyoCjila/igLXilqHigLIp4pWv54K45by577yB4oCi4oCi4oCiKu+9nuKXjyjCrF/CrCApCo==
```

之前也没有接触过base64隐写（懒狗，还是做题太少了），网上有通用脚本，直接跑就可以了

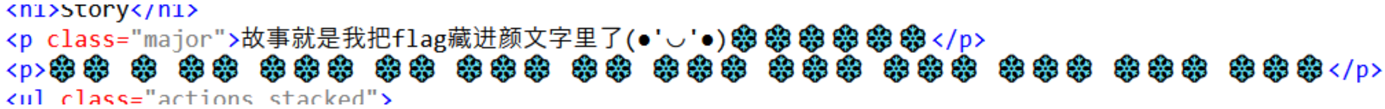


```
0
key:"lorrie
2
key:"lorrie"
```

这道题还涉及到了SNOW隐写，简单介绍一下SNOW隐写

snow 隐写是在html嵌入隐写信息，它的原理是通过在文本文件的末尾嵌入空格和制表位的方式嵌入隐藏信息，不同空格与制表位的组合代表不同的嵌入信息。

其实源码中已经给了提示



```
<h1>Story</h1>
<p class="major">故事就是我把flag藏进颜文字里了(•'◡'•)❄❄❄❄❄❄</p>
<p>❄❄ ❄ ❄❄ ❄❄❄ ❄❄ ❄❄❄ ❄❄ ❄❄❄ ❄❄❄ ❄❄❄ ❄❄❄ ❄❄❄</p>
<ul class="actions stacked">
```

Github找一下工具，得到的 `lorrie` 应该就是密钥

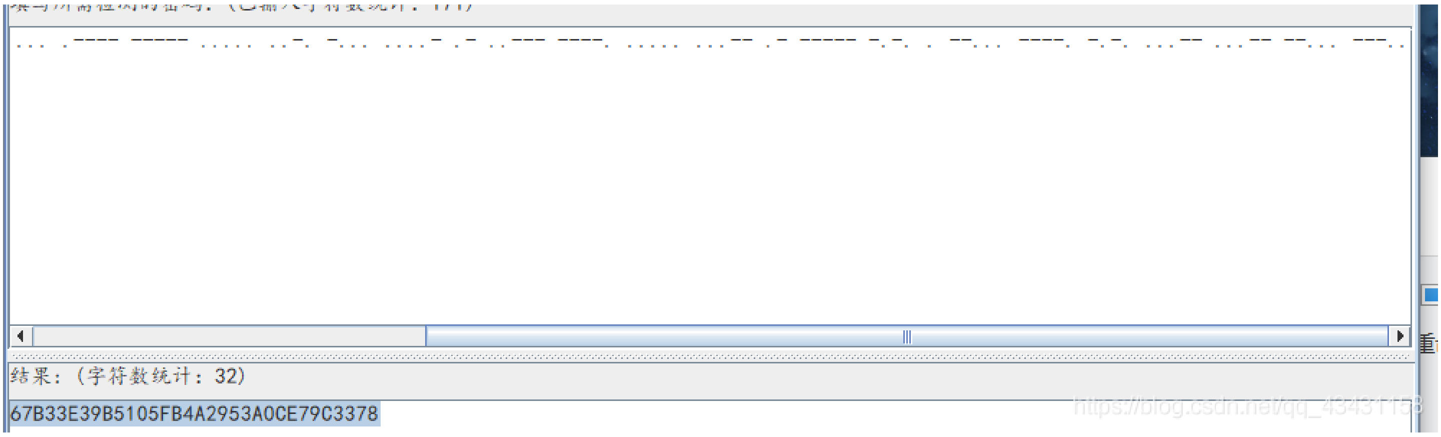
```
SNOW.EXE -p lorrie -f index-demo.html > shy.txt
```

得到flag

```
flag{→_→←_←←_←←_←←_← →_→→_→←_←←_←←_← →_→←_←←_←←_← ←_←←_←←_←→_→→_→ ←_←←_←←_←→_→→_→ ←_← ←_←←_←←_←→_→→_→ →_→→_→→_→→
_→←_← →_→←_←←_←←_← ←_←←_←←_←←_←←_← ←_←→_→→_→→_→→_→ →_→→_→→_→→_→→_→ ←_←←_←←_←←_←←_← ←_←←_←→_→←_← →_→←_←←_←←_← ←_←
←_←←_←←_←→_→ ←_←→_→ ←_←←_←→_→→_→→_→ →_→→_→→_→→_→←_← ←_←←_←←_←←_←←_← ←_←←_←←_←→_→→_→ ←_←→_→ →_→→_→→_→→_→→_→ →_→←_
←→_→←_← ←_← →_→→_→←_←←_←←_← →_→→_→→_→→_→←_← →_→←_←→_→←_← ←_←←_←←_←→_→→_→ ←_←←_←←_←→_→→_→ →_→→_→←_←←_←←_← →_→→_→→
_→←_←←_←}
```

但这样的格式肯定是不对的，所以还需要进一步解密，只有两种格式，就先替换成摩斯密码试试

```
→_→ =》 -
←_← =》 .
```



填写所需检测的密码：（已输入字符数统计：171）

```
... .---- ----- ..... ..-. -... ....- .- ..--- ----. ..... ...-- .- ----- -.-. . --... ----. -.-. ...-- ...-- --... ---..
```

结果：（字符数统计：32）

67B33E39B5105FB4A2953A0CE79C3378

最终得到flag

0x03: 隐藏的秘密

隐藏的秘密。附件当时没下载起来，等有附件了再来补充一下