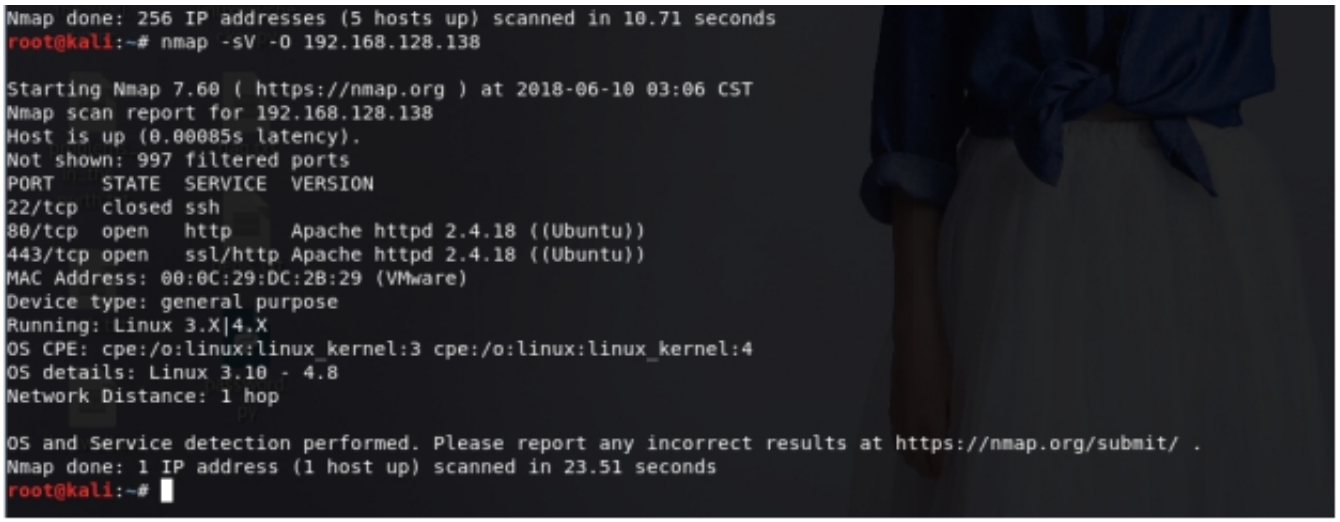
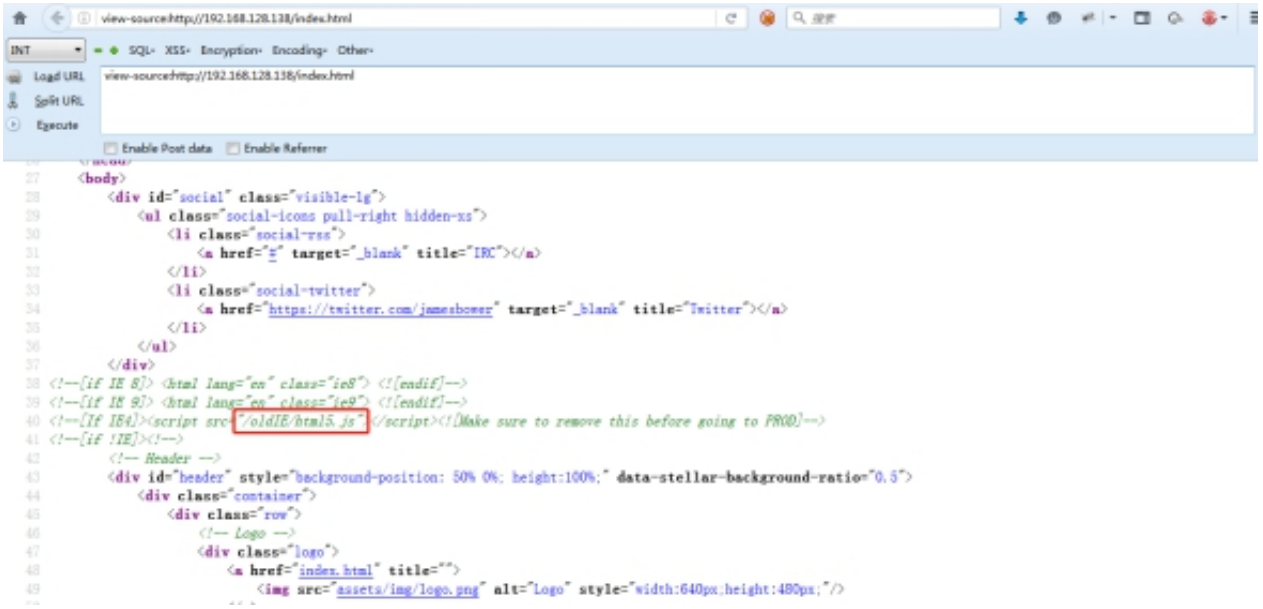
IP发现：

端口扫描：



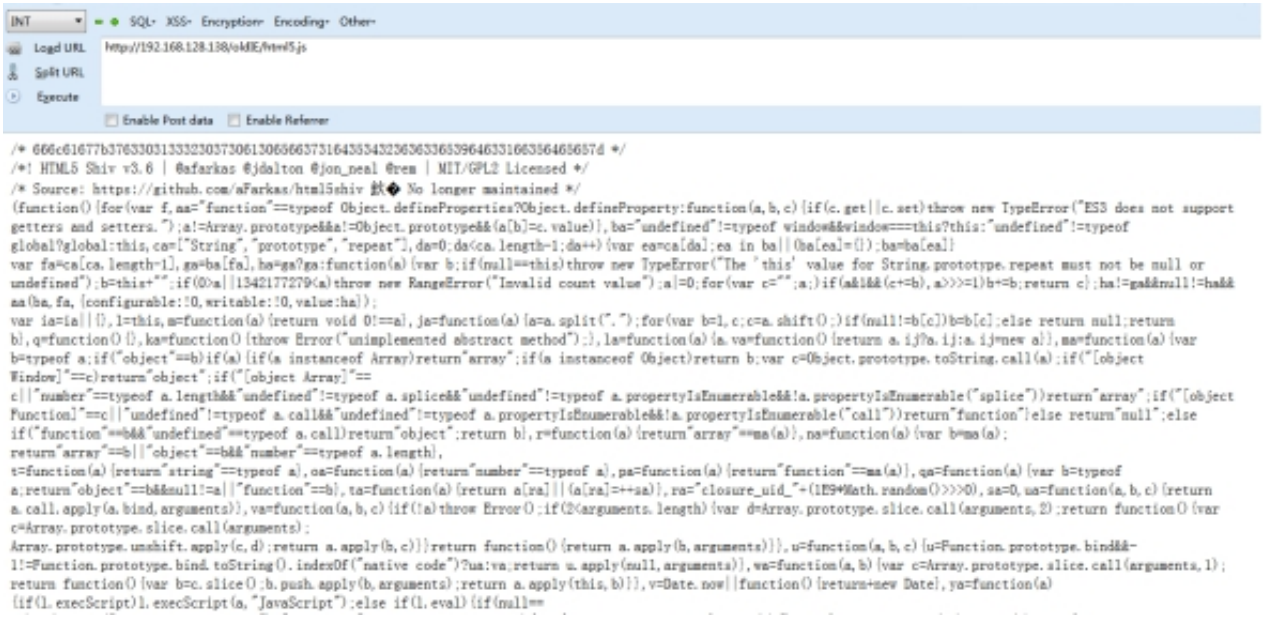
访问web页面：

查看网站源码，发现一个目录：

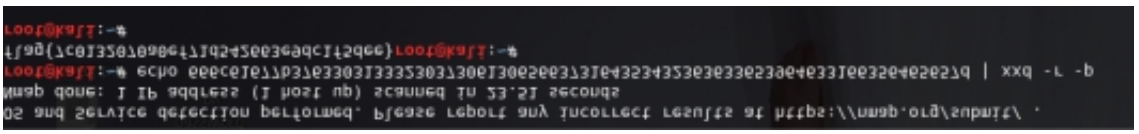


访问文件，发现第一个flag，是十六进制编码：

```
666c61677b376330313332303730613065663731643534323636336539646331663564
65657d
```




```
现在我使用以下xxd命令将十六进制转换为文本。
echo
666c61677b376330313332303730613065663731643534323636336539646331663564
65657d | xxd -r -p
```

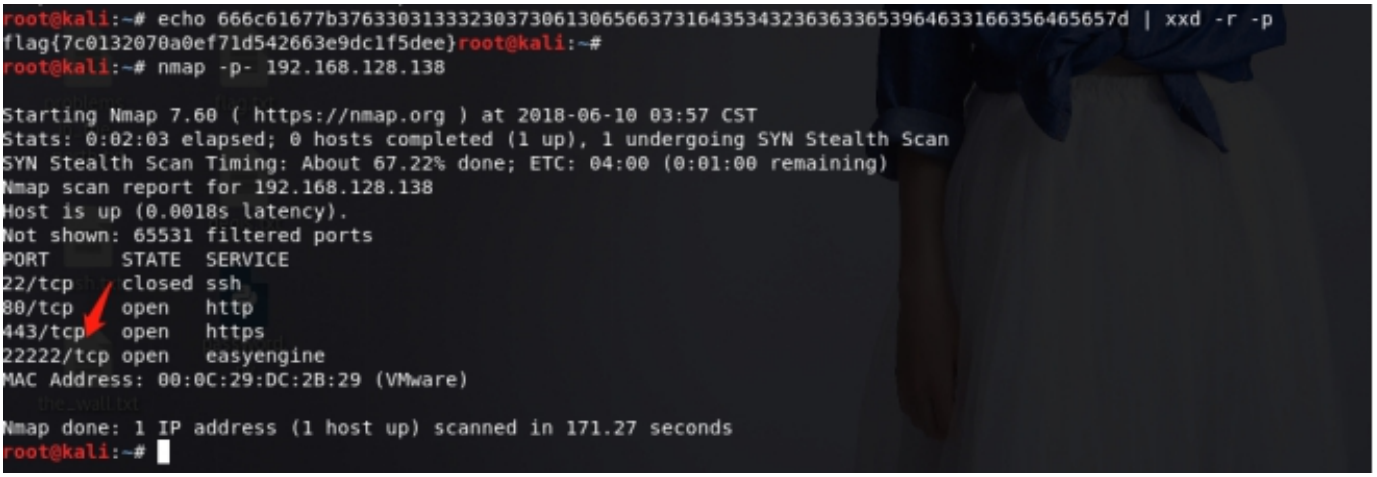


现在拿到了第一个flag:

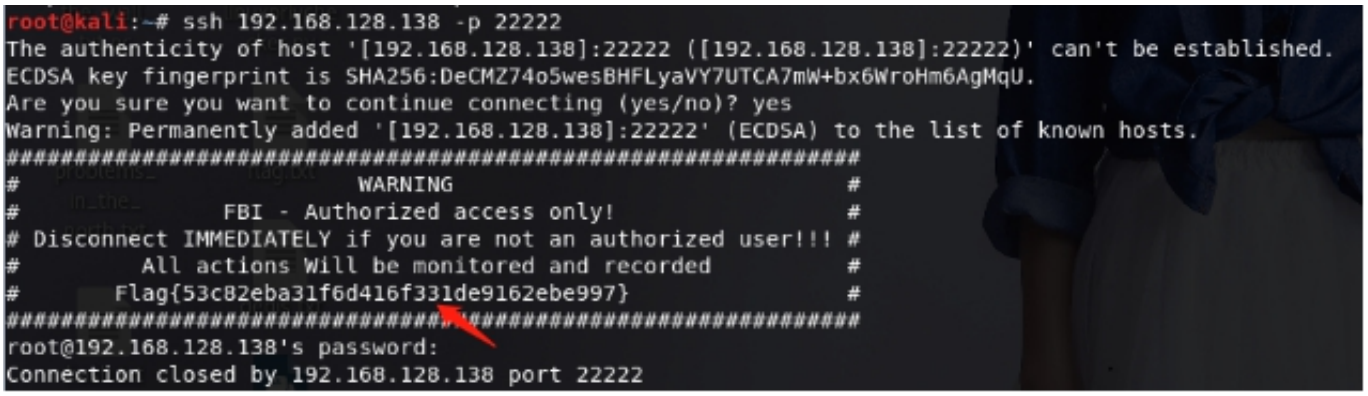
flag{7c0132070a0ef71d542663e9dc1f5dee}

flag用md5解密:

所给的线索是nmap，用nmap 对所有65535端口进行更完整的扫描，确定服务器正在端口22222上运行SSH服务器。这一定是我进入服务器的方式

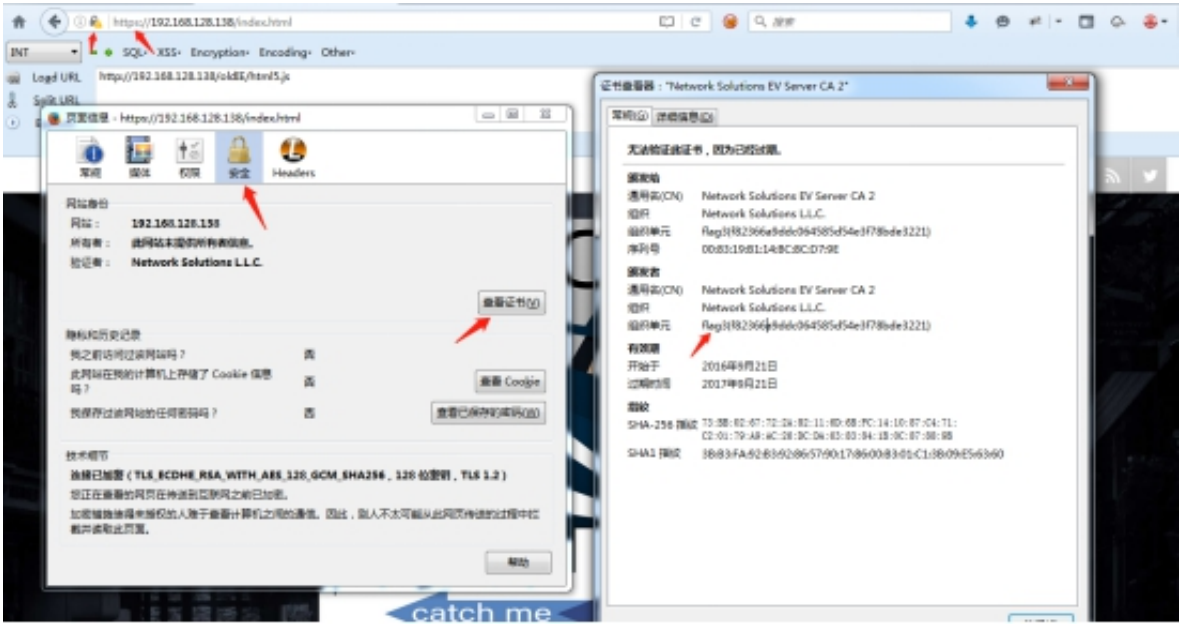


通过ssh连接，拿到了第二个flag：Flag{53c82eba31f6d416f331de9162ebe997}



md5解密flag，解密后的线索是：encrypt

1 线索是“加密”。到目前为止，截获流量和加密的唯一方法就是用于默认站点的SSL。所以我仔细看看SSL证书和BOOM。



拿到第三个flag：flag3{f82366a9ddc064585d54e3f78bde3221}

解密后的线索： personnel

发现personnel是一个目录，结合flag信息，又看了看之前找到的

http://192.168.128.138/oldIE/html5.js

发现如下信息

(a.compareDocumentPosition(b)&16);for(;b&&a!=b;)b=b.parentNode;return b==a},de=function(a){return 9==a.nodeType?a:a.ownerDocument||a.document},Ae=function(a)
{try{return a.contentWindow||(a.contentDocument?ae(a.contentDocument):null)}catch(b){}return null},Be=function(a,b){if("textContent"in a)a.textContent=b;else
if(3==a.nodeType)a.data=b;else if(a.firstChild&&
3==a.firstChild.nodeType){for(;a.lastChild!=a.firstChild;)a.removeChild(a.lastChild);a.firstChild.data=b}else
te(a),a.appendChild(de(a).createTextNode(String(b)))},De=function(a){var b=[];Ce(a,Hc,b,!1);return b},Ce=function(a,b,c,d){if(null!=a)for(a=a.firstChild;a;){if(b(a)&&
(c.push(a),d)||Ce(a,b,c,d))return!0;a=a.nextSibling}return!1},Ee={SCRIPT:1,STYLE:1,HEAD:1,IFRAME:1,OBJECT:1},Fe={IMG:" ",BR:"\n"},Ge=function(a,b){b?a.tabIndex=0:
(a.tabIndex=-1,a.removeAttribute("tabIndex"))},Je=function(a){var b;
/* maindev - 6/7/02 Adding temporary support for IE4 FBI Workstations */
/* newmaindev - 5/22/16 Last maindev was and idoit and IE4 is still Gold image - @Support doug.perterson@fbi.gov */
(b="A"==a.tagName||"INPUT"==a.tagName||"TEXTAREA"==a.tagName||"SELECT"==a.tagName||"BUTTON"==a.tagName?!a.disabled&&(!He(a)||Ie(a)):He(a)&&Ie(a))&&
B?(a=!pa(a.getBoundingClientRect)||B&&null==a.parentElement?{height:a.offsetHeight,width:a.offsetWidth}:a.getBoundingClientRect(),a=null!=a&&0<a.height&&
0<a.width):a=b;return a},He=function(a){a=a.getAttributeNode("tabindex");return null!=a&&a.specified},Ie=function(a){a=a.tabIndex;return oa(a)&&0<=a&&
32768>a},Le=function(a){if(ae&&null!==a&&"innerText"in
a)a=a.innerText.replace(/(\r\n|\r|\n)/g,"\n");else{var b=[];Ke(a,b,!0);a=b.join("")}a=a.replace(/ \xAD /g," ").replace(/\xAD/g,"");a=a.replace(/\u200B/g,"");
ae||(a=a.replace(/ +/g," "));" "!=a&&(a=a.replace(/^\s*/,""));return a},Me=function(a){var b=[];Ke(a,b,!1);return b.join("")},Ke=function(a,b,c){if(!(a.nodeName in
Ee))if(3==a.nodeType)c?b.push(String(a.nodeValue).replace(/(\r\n|\r|\n)/g,"")):b.push(a.nodeValue);else if(a.nodeName in Fe)b.push(Fe[a.nodeName]);else
for(a=a.firstChild;a;)Ke(a,b,

FLAGS

Flag#1 - "Don't go Home Frank! There's a Hex on Your House"

Flag#2 - "Obscurity or Security? That is the Question"

Flag#3 - "During his Travels Frank has Been Known to Intercept Traffic"

Flag#4 - "A Good Agent is Hard to Find"

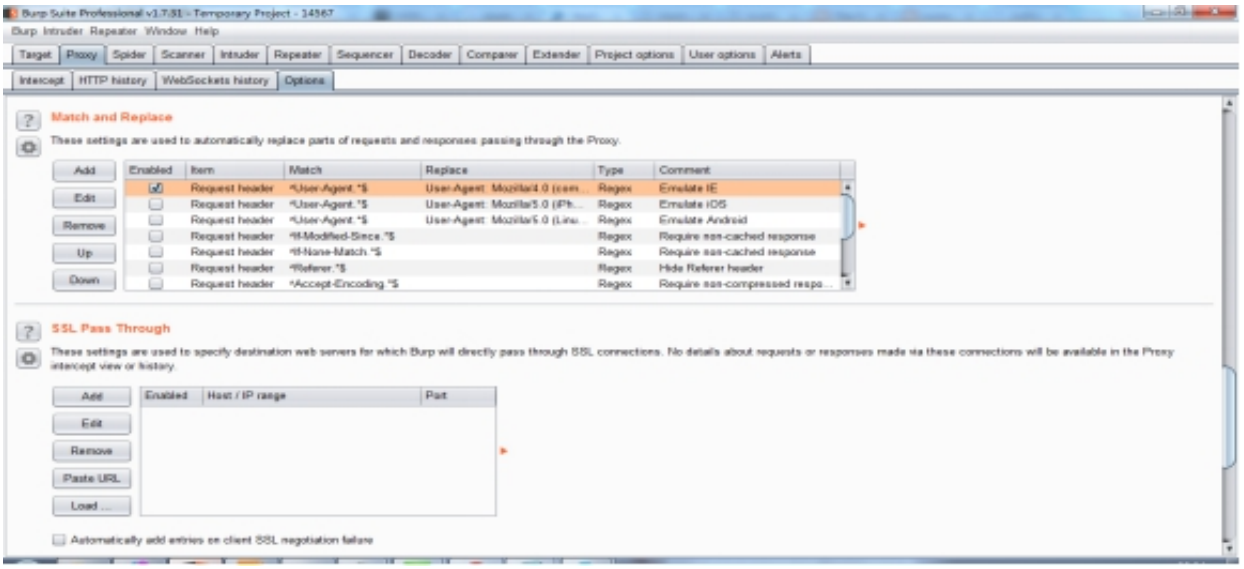
Flag#5 - "The Devil is in the Details - Or is it Dialogue? Either Way, if it's Simple, Guessable, or Personal it Goes Against Best Practices"

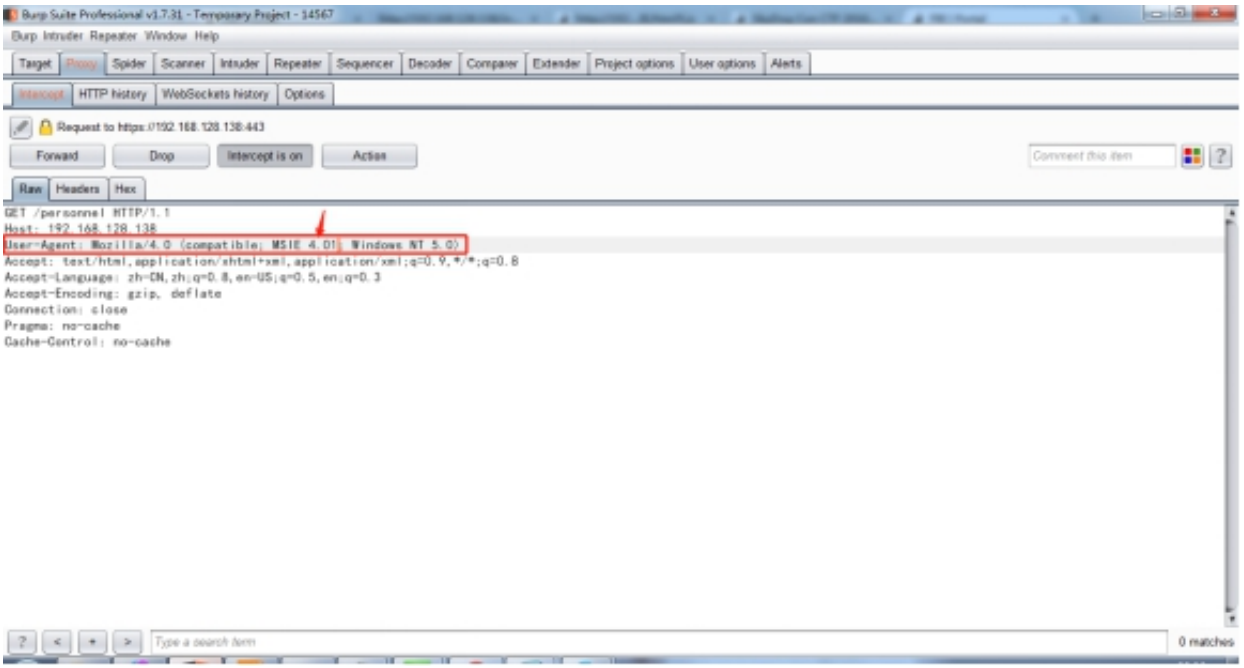
Flag#6 - "Where in the World is Frank?"

Flag#7 - "Frank Was Caught on Camera Cashing Checks and Yelling - I'm The Fastest Man Alive!"

Flag#8 - "Franks Lost His Mind or Maybe it's His Memory. He's Locked Himself Inside the Building. Find the Code to Unlock the Door Before He Gets Himself Killed!"

用burp挂上代理，修改如下地方， 修改User-Agent，使用IE4访问：

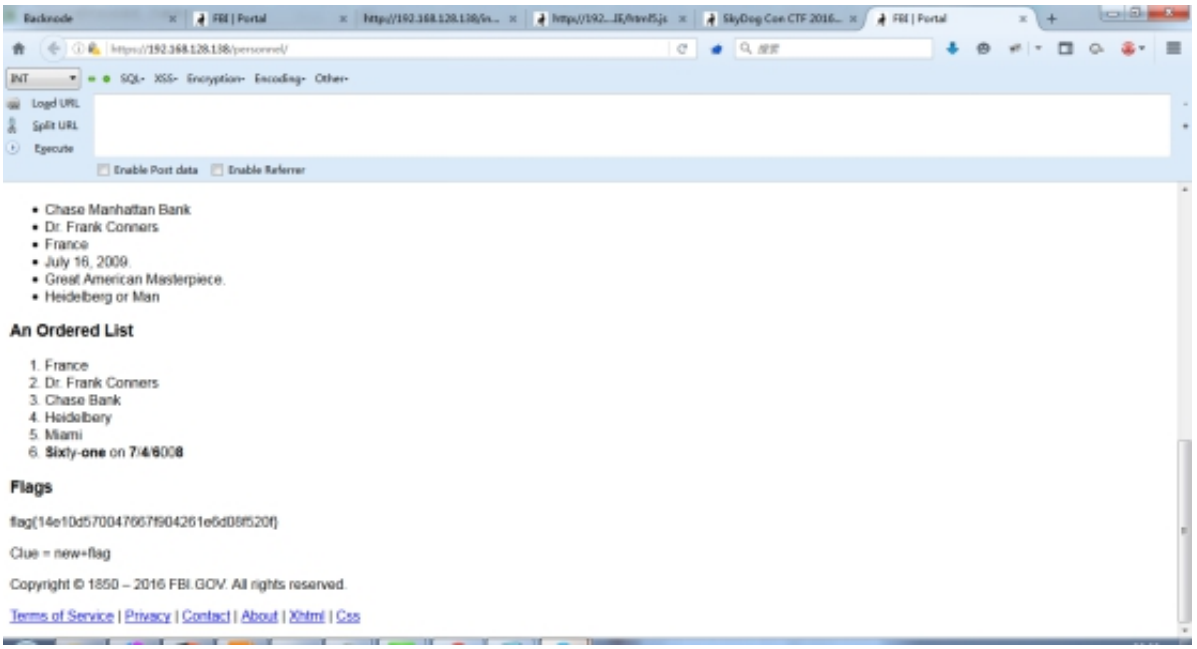




现在我用IE4用户代理刷新页面，我们迎接着欢迎Agent Hanratty的FBI门户网站

浏览器css做了过滤，但是不影响找到flag,第四个flag:

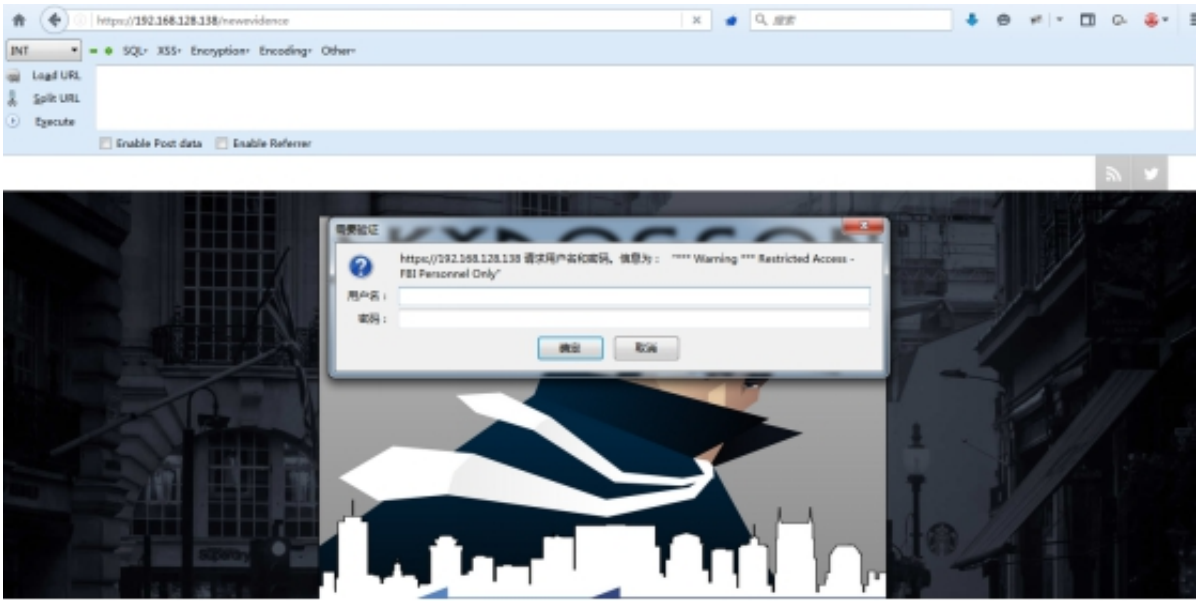
flag{14e10d570047667f904261e6d08f520f}



在门户网站的底部，我们发现我们的第四个标志{14e10d570047667f904261e6d08f520f}和一个新线索“Clue = new +flag”。

解密后线索是： evidence

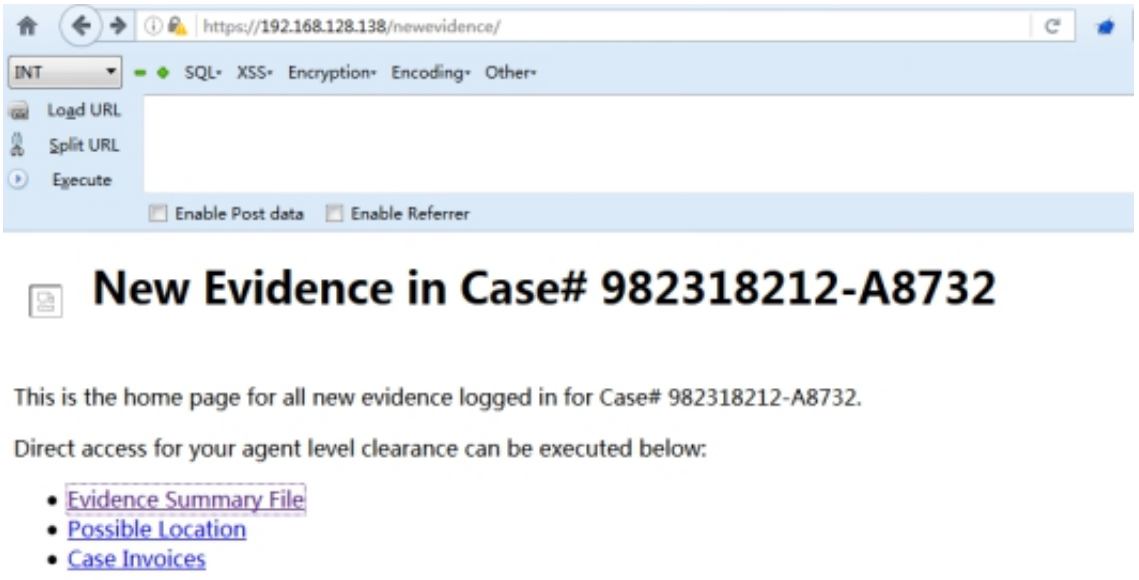
通过我们刚刚从Flag4得到的信息，猜测/ newevidence是否是一个目录。确实是，但需要用户名/密码，而且要使用IE4用户代理登录。

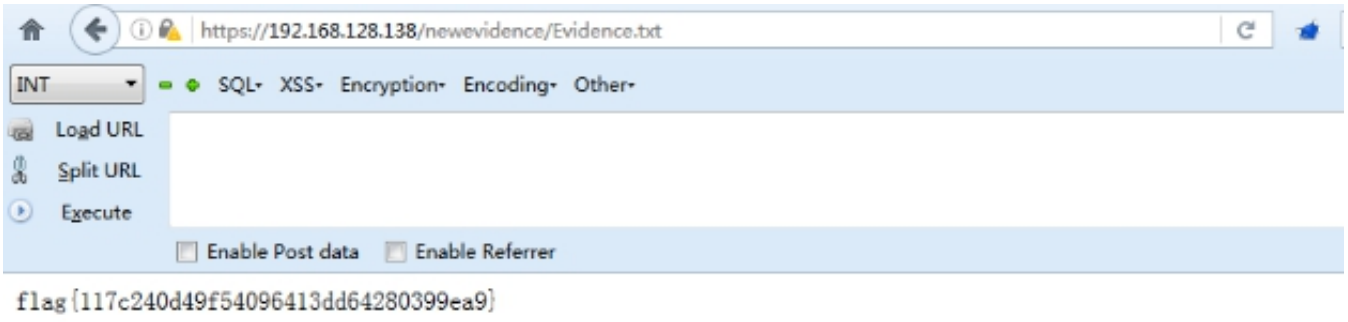


用户名，密码怎么办，我在上一个页面用cewl生成了一个字典，用hydra果然跑出了口令没跑出来

然后又去搜了这个电影相关的信息，反正是找了好多信息终于找到口令： carl.hanratty/ Grace

登录任然需要代理采用IE4浏览器



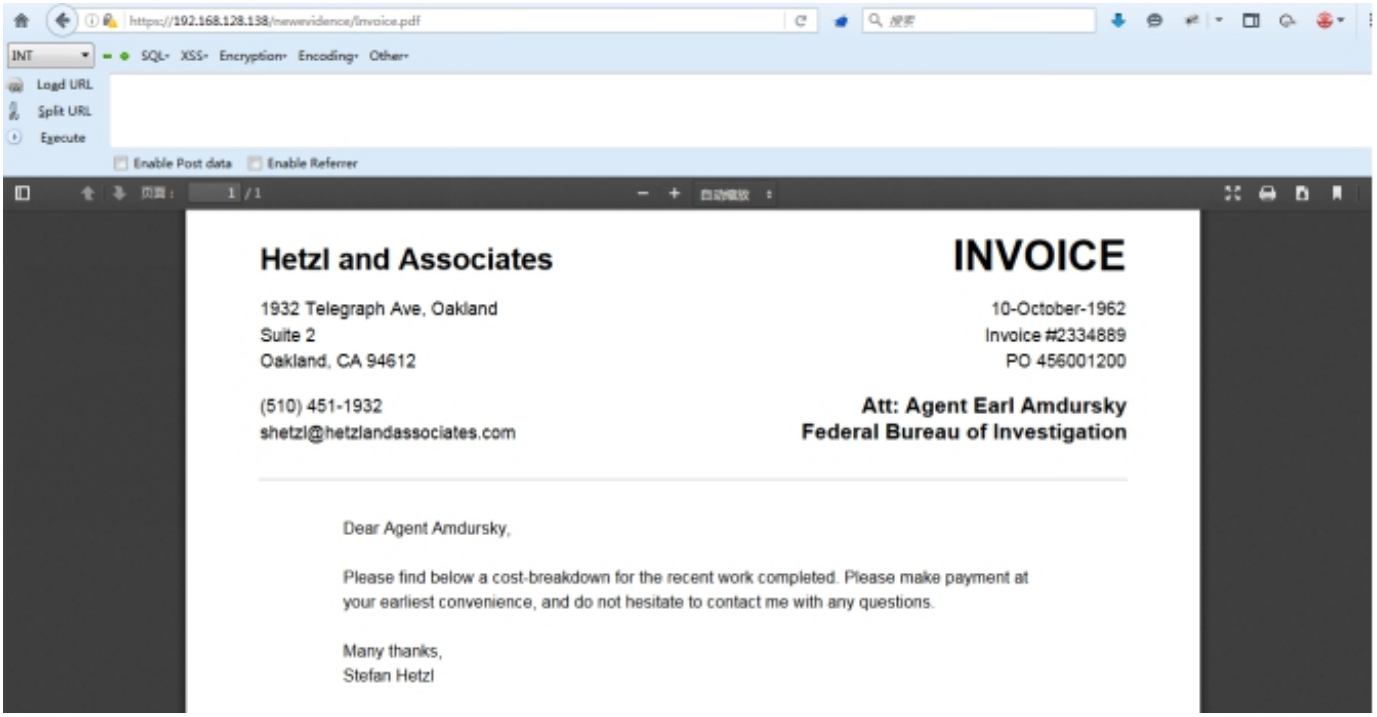


找到了第五个flag:

flag{117c240d49f54096413dd64280399ea9}

解密后的线索是： panam

访问 两个文件image.jpg和Invoice.pdf



把图片下载下来，打开图片发现第六个flag

flag{d1e5146b1719287313 85eb7ea38c37b8}

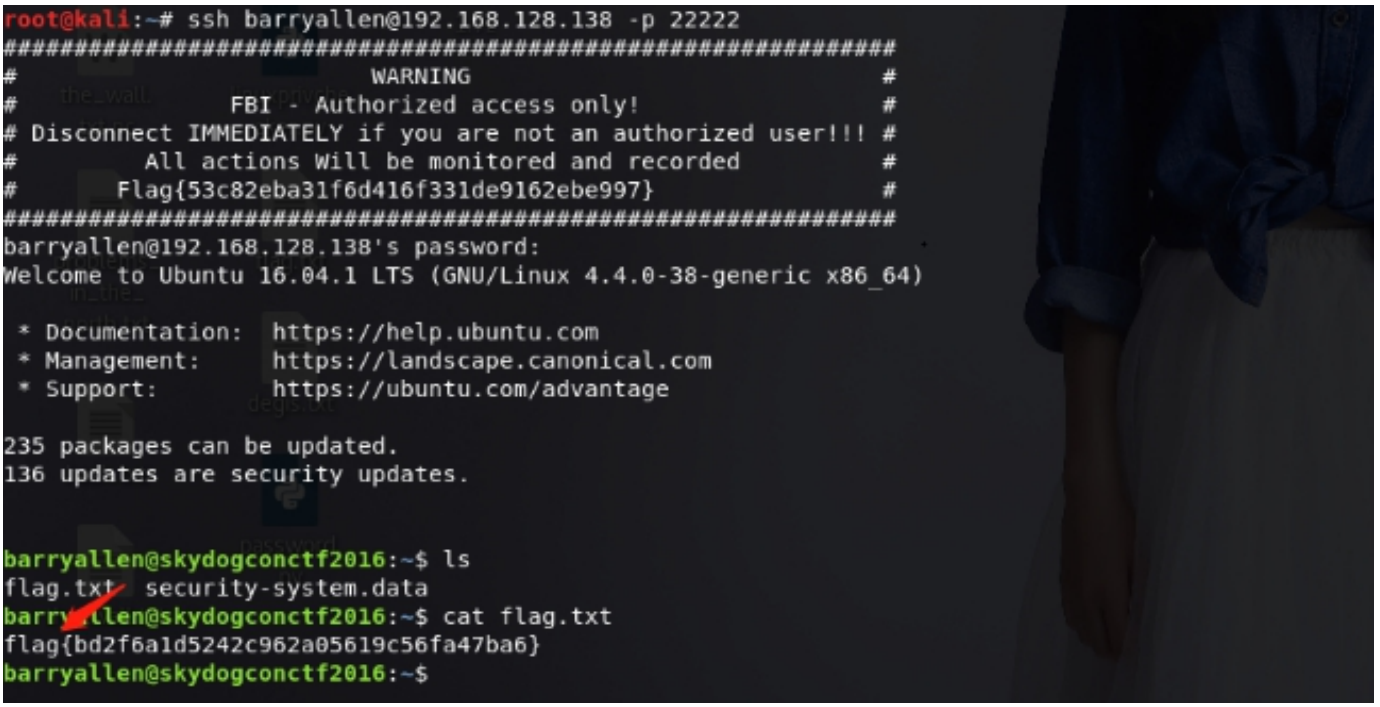
解密后的线索是： ILoveFrance

```
root@Intrusion-Exploitation:~/Desktop/SKycon# steghide --info image.jpg
"image.jpg":
  format: jpeg
  capacity: 230.1 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "flag.txt":
    size: 71.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes
root@Intrusion-Exploitation:~/Desktop/SKycon# steghide extract -sf flag.txt
Enter passphrase:
steghide: could not open the file "flag.txt".
root@Intrusion-Exploitation:~/Desktop/SKycon# steghide extract -sf image.jpg
Enter passphrase:
wrote extracted data to "flag.txt".
root@Intrusion-Exploitation:~/Desktop/SKycon# ls
flag.txt  image.jpg  Invoice.pdf
root@Intrusion-Exploitation:~/Desktop/SKycon# cat flag.txt
flag{d1e5146b1719287313385eb7ea38c37b8}
=ILoveFrance

clue=iheartbrenda
root@Intrusion-Exploitation:~/Desktop/SKycon#
```

这是一个奇怪的线索。为什么弗兰克大喊“我是最快的人活着！”？这听起来很奇怪， 我谷歌这句话，果然是超级英雄barry.allen；又名Flash。

现在我做了一堆“barry.allen”和“flash”的不同组合，现在使用我发现的任何凭据的唯一地方就是SSH。所以我尝试使用密码“iheartbrenda”的“barry.allen”，但这不起作用。接下来，我尝试使用“barryallen”和“iheartbrenda”作为密码，登陆成功。



成功拿到flag，flag{bd2f6a1d5242c962a05619c56fa47ba6}

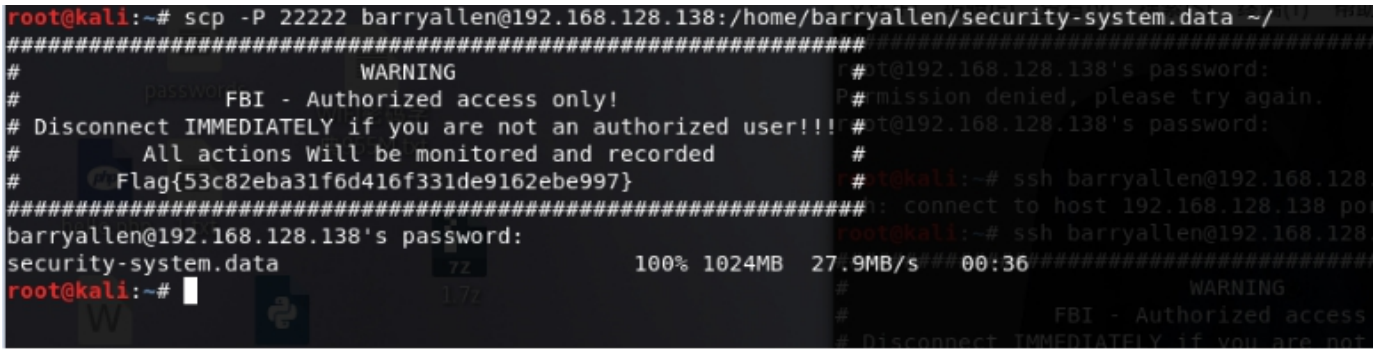
并且有一个名为“security-system.data”的非常大的文件。

解密flag的线索： theflash

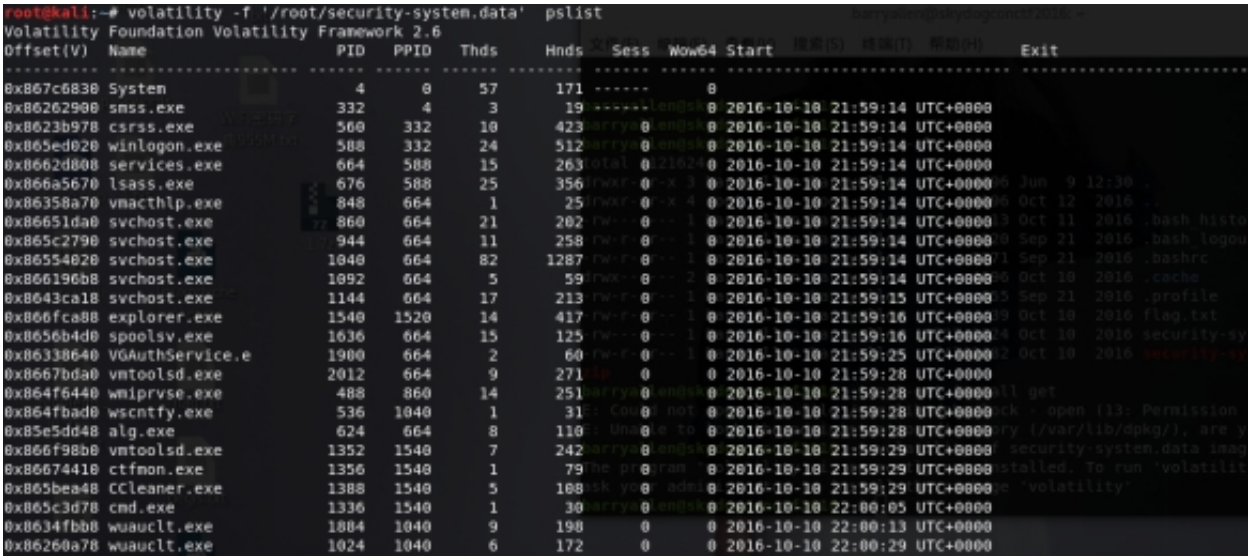
现在我可以通过SSH访问Barry Allen帐户，我开始仔细查看主目录中的security-system.data文件。

在kali执行如下命令，将文件下载下来

```
scp -P 22222 barryallen@192.168.128.138:/home/barryallen/security-system.data ~/
```



我将文件下载到Kali，查看它是什么类型的文件。该文件命令显示它是一个zip文件，所以我运行命令



现在security-system.data显示为简单的数据。在文件上运行字符串我看到很多提及内存的内容，所以我在想它是一台机器的内存映像。下一步是使用volatility来查看文件

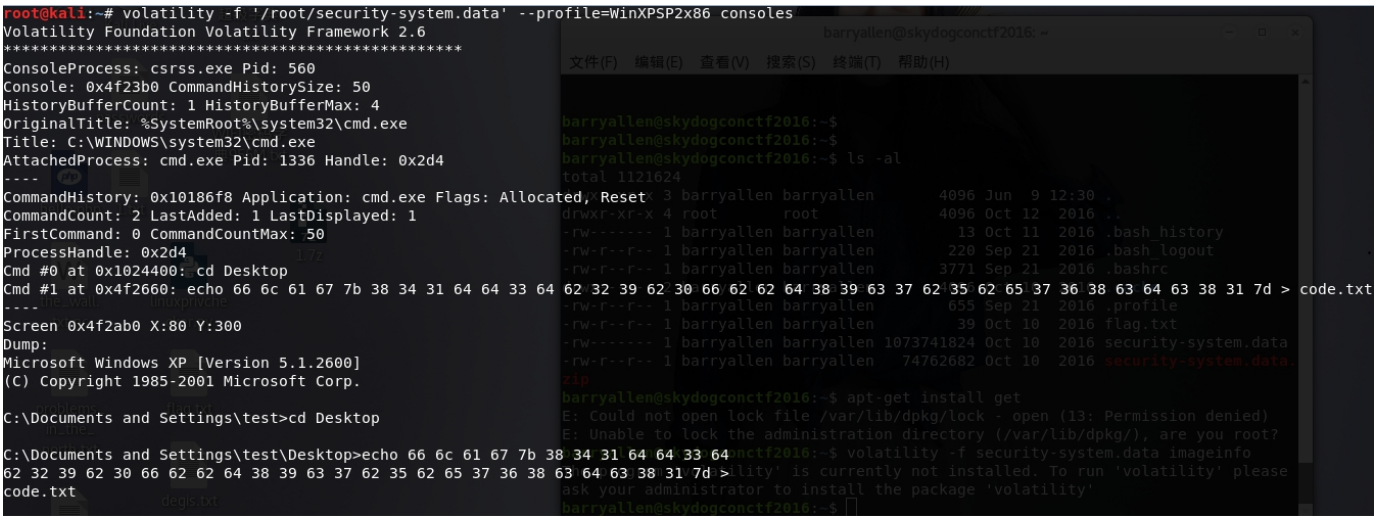
```
root@kali:~# volatility -f '/root/security-system.data' imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/root/security-system.data)
                      PAE type : PAE
                           DTB : 0x33e000L
                          KDBG : 0x80545b60L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2016-10-10 22:00:50 UTC+0000
     Image local date and time : 2016-10-10 18:00:50 -0400
```

确实能够显示一些有趣的信息。我继续使用volatility进一步挖掘。

```
root@kali:~# volatility -f '/root/security-system.data' --profile=WinXPSP2x86 iehistory
Volatility Foundation Volatility Framework 2.6
**************************************************
Process: 1540 explorer.exe
Cache type "DEST" at 0x159d0f
Last modified: 2016-10-10 18:00:41 UTC+0000
Last accessed: 2016-10-10 22:00:42 UTC+0000
URL: test@file:///C:/Documents%20and%20Settings/test/Desktop/code.txt
```

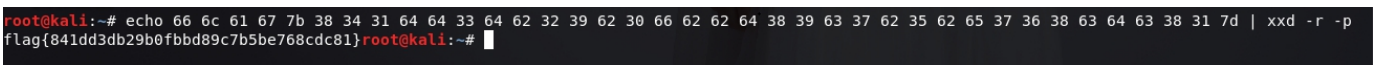
我可以在桌面上看到一个名为code.txt的文件的引用，它是对我们线索的直接引用。

最后我能够抓住一些图像，但只有一个有可见的东西显示空的code.txt，但没有别的。我的下一步是查看是否有任何内容输入到控制台中。



真棒！我可以看到code.txt是通过在文件中回显hex来在桌面上创建的。有时间看看十六进制表示什么，所以我再次运行xxd命令。

```
echo 66 6c 61 67 7b 38 34 31 64 64 33 64 62 32 39 62 30 66 62 62 64 38 39 63 37 62 35 62 65 37 36 38 63 64 63 38 31 7d | xxd -r -p
```



最后的flag：flag{841dd3db29b0fbbd89c7b5be768cdc81}

解密： Twolittlemice