ez_mem&usb writeup

原创

Warning 于 2020-03-17 22:49:54 发布 270 收藏
分类专栏： 杂项

本文链接：https://blog.csdn.net/destiny1507/article/details/104933089

知识点：

- 内存取证（volatility使用）
- usb流量取证

解题过程：

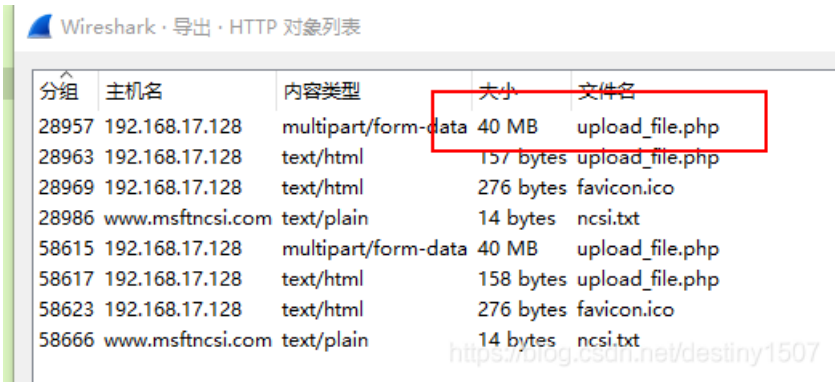
我们拿到手的是一个pcap包：

看到流量包首先使用wireshark打开，随便追踪一个数据流就会发现这里面的交互中有一个压缩包：

```
-----------------------------154555628677
Content-Disposition: form-data; name="file"; filename="data.zip"
Content-Type: application/octet-stream

PK.........XP..$ -Sm.....    ...data.vmem.].\S..?
.QQR..V[\ik..u....@.j+...ZA.%....^.mp...
4fs......ns..l..M.Q;qk.............l..-z....y...o....wr..
{.s...so.If].....J*DZ....s.u}..u...L..X...u.......+.i.....3.Z...O.....C.
4..'.U})D.G..zw-..w$....dz....Bw+dZv.._..
1W_8yCCrN.;..........ED.m....Pb....e.g...H....../.......Y'c.....?.w1-....
...S...F+...4..<vDC.Xv.......>...h...K..X
.,.E.;&...[.l.v..o..._.....m.{#........'B......(rEXRR..............  ..
{...1.......  .vi..lc.........u........Z......_.....)..\....
1S...m...'...k..z...=........z)....R...&v.0..      ..m?.._v.i.\y.
8W..&.g.T.q`.}....&@=..o....}o..f../....+f{t..Yw.c..>.......
9...K.......:...3...o...........g..\....zd}.<a..
3.Q..n....hO6h..........k.;,..;...MD..t../......i....'.
.x.r....U.....ZK.P!T.T.
..=.1L.....KYE....Zk.E.\..z....W.....O_w..Mz*.).W..._......
```

所以我们先导出http对象看一下，发现有一个php特别大，直觉告诉我们肯定有问题，很有可能这里面就包含着我们想要的压缩包：



保存upload_file.php之后，使用binwalk查看，果然里面含有压缩包。使用foremost分离后得到一个vmem文件

（但是很奇怪的是，我使用winhex手动对pcap文件进行分离总是失败，使用foremost分离pcap也不行，会报出压缩包数据格式损坏，但是对upload_file.php分离就可以。难道是因为pcap中还是加密的数据吗？不懂，希望有大佬可以帮忙解释一下）

得到data.vmem之后，很明显考察方向是内存取证。内存取证一般使用volatility，很强大的工具，Kali中有自带。

首先查看一下进程树：

```
root@warning:~/桌面/CTF# volatility -f data.vmem pstree
Volatility Foundation Volatility Framework 2.6
Name                                                  Pid   PPid   Thds   Hnds T
ime
-------------------------------------------------- ------ ------ ------ ------ -
---
 0x80ea2660:System                                      4      0     52    231 1
970-01-01 00:00:00 UTC+0000
. 0xff57fc28:smss.exe                                372      4      3     19 2
020-02-23 13:17:13 UTC+0000
.. 0xff435020:winlogon.exe                           492    372     20    501 2
020-02-23 13:17:13 UTC+0000
... 0xff46b020:lsass.exe                             680    492     19    309 2
020-02-23 13:17:14 UTC+0000
... 0xff445020:services.exe                          668    492     16    253 2
020-02-23 13:17:14 UTC+0000
.... 0xff416b10:svchost.exe                         1024    668     44    939 2
020-02-23 13:17:14 UTC+0000
.... 0xff493568:svchost.exe                          848    668     14    189 2
020-02-23 13:17:14 UTC+0000
..... 0xff4afda0:wmiprvse.exe                        540    848     13    242 2
020-02-23 13:17:41 UTC+0000
.... 0x80dac020:svchost.exe                         1072    668      4     57 2
020-02-23 13:17:14 UTC+0000
```


没发现有什么特别的内容，我们可以再看一下cmd的历史操作：

```
root@warning:~/桌面/CTF# volatility -f data.vmem cmdscan
Volatility Foundation Volatility Framework 2.6
**************************************************
CommandProcess: csrss.exe Pid: 464
CommandHistory: 0x556bb8 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 2 LastAdded: 1 LastDisplayed: 1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x504
Cmd #0 @ 0x3609ea0: passwd:weak_auth_top100
Cmd #1 @ 0x5576d0: start wireshark
Cmd #13 @ 0x9f009f: ??
Cmd #41 @ 0x9f003f: ?\?????????
root@warning:~/桌面/CTF#
```


在我们能看到的内容里，发现有一个密码passwd:weak_auth_top100，然后打开了wireshark，可以知道它捕获了流量。一般来说passwd都是很关键的信息，现在我们还不知道有什么用，暂且搁置。

接下来可以扫描一下内存里面的文件：

```
root@warning:~/桌面/CTF# volatility -f data.vmem filescan | grep flag
Volatility Foundation Volatility Framework 2.6
0x0000000001155f90      1      0 R--rwd \Device\HarddiskVolume1\Documents and Se
ttings\Administrator\flag.img
root@warning:~/桌面/CTF#
```

发现了一个flag.img，既然跟flag有关，那么就先导出来看看吧：

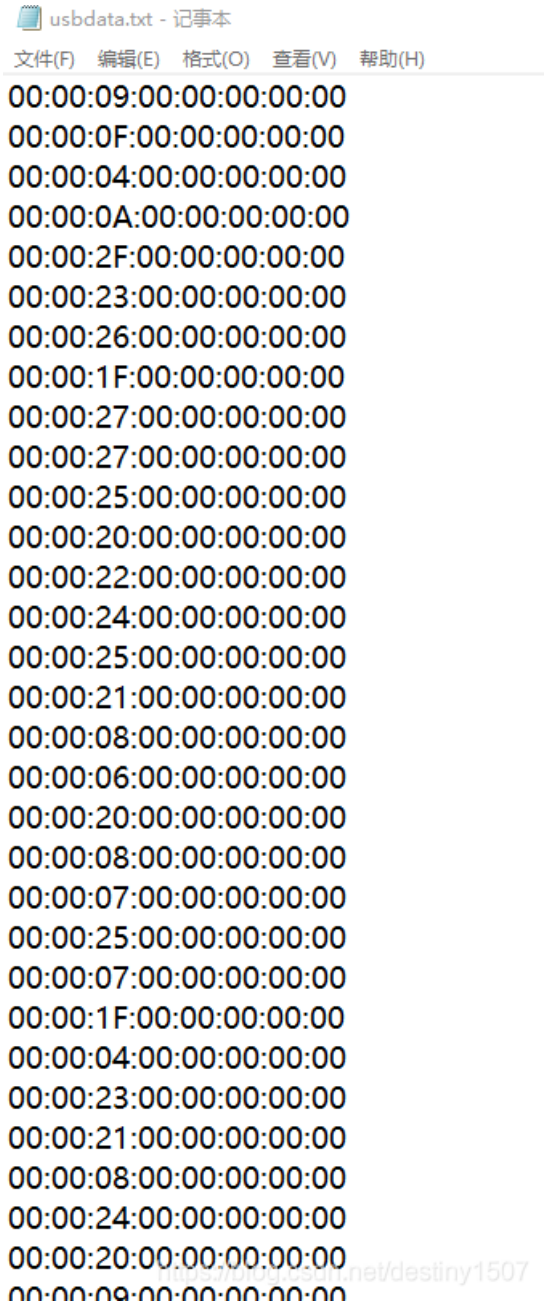
```
root@warning:~/桌面/CTF# volatility -f data.vmem dumpfiles -Q 0x0000000001155f90
 -D ./
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x01155f90   None   \Device\HarddiskVolume1\Documents and Sett
ings\Administrator\flag.img
root@warning:~/桌面/CTF#
```

这时候会发现你的文件夹中多了一个.dat文件。接下来我们可以对这个文件进行分析。

同样binwalk看完之后发现有一个zip文件，我们使用foremost分离，提取出一个压缩包。

解压时用到了我们之前在cmd中找到的密码。

压缩包内是一个txt文件，内容如下：



结合题目，可以想到这是一个usb流量数据包。这里的数据都集中在了第三键上，说明是一个键盘流量。对照分析表，可以写出代码解码。

> 说明：
>
> 1. 键盘数据包的数据长度为8个字节，击键信息集中在第3个字节，每次key stroke都会产生一个keyboard event usb packet。
>
> 2. 鼠标数据包的数据长度为4个字节，第一个字节代表按键，当取0x00时，代表没有按键、为0x01时，代表按左键，为0x02时，代表当前按键为右键。第二个字节可以看成是一个signed byte类型，其最高位为符号位，当这个值为正时，代表鼠标水平右移多少像素，为负时，代表水平左移多少像素。第三个字节与第二字节类似，代表垂直上下移动的偏移。

分析表：

Table 12: Keyboard/Keypad Page

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
0	00	Reserved (no event indicated)[9]	N/A	√	√	√	4/101/104
1	01	Keyboard ErrorRollOver[9]	N/A	√	√	√	4/101/104
2	02	Keyboard POSTFail[9]	N/A	√	√	√	4/101/104
3	03	Keyboard ErrorUndefined[9]	N/A	√	√	√	4/101/104
4	04	Keyboard a and A[4]	31	√	√	√	4/101/104
5	05	Keyboard b and B	50	√	√	√	4/101/104
6	06	Keyboard c and C[4]	48	√	√	√	4/101/104
7	07	Keyboard d and D	33	√	√	√	4/101/104
8	08	Keyboard e and E	19	√	√	√	4/101/104
9	09	Keyboard f and F	34	√	√	√	4/101/104
10	0A	Keyboard g and G	35	√	√	√	4/101/104
11	0B	Keyboard h and H	36	√	√	√	4/101/104
12	0C	Keyboard i and I	24	√	√	√	4/101/104
13	0D	Keyboard j and J	37	√	√	√	4/101/104
14	0E	Keyboard k and K	38	√	√	√	4/101/104
15	0F	Keyboard l and L	39	√	√	√	4/101/104
16	10	Keyboard m and M[4]	52	√	√	√	4/101/104
17	11	Keyboard n and N	51	√	√	√	4/101/104
18	12	Keyboard o and O[4]	25	√	√	√	4/101/104
19	13	Keyboard p and P[4]	26	√	√	√	4/101/104
20	14	Keyboard q and Q[4]	17	√	√	√	4/101/104

代码：

```
# coding:utf-8
import sys
import os

usb_codes = {
    0x04: "aA", 0x05: "bB", 0x06: "cC", 0x07: "dD", 0x08: "eE", 0x09: "fF",
    0x0A: "gG", 0x0B: "hH", 0x0C: "iI", 0x0D: "jJ", 0x0E: "kK", 0x0F: "lL",
    0x10: "mM", 0x11: "nN", 0x12: "oO", 0x13: "pP", 0x14: "qQ", 0x15: "rR",
    0x16: "sS", 0x17: "tT", 0x18: "uU", 0x19: "vV", 0x1A: "wW", 0x1B: "xX",
    0x1C: "yY", 0x1D: "zZ", 0x1E: "1!", 0x1F: "2@", 0x20: "3#", 0x21: "4$",
    0x22: "5%", 0x23: "6^", 0x24: "7&", 0x25: "8*", 0x26: "9(", 0x27: "0)",
    0x2C: "  ", 0x2D: "-_", 0x2E: "=+", 0x2F: "[{", 0x30: "]}", 0x32: "#~",
    0x33: ";:", 0x34: "'\"", 0x36: ",<", 0x37: ".>", 0x4f: ">", 0x50: "<"
}


def code2chr(filepath):
    lines = []
    pos = 0
    for x in open(filepath, "r").readlines():
        code = int(x[6:8], 16)  # 即第三个字节
        if code == 0:
            continue
        # newline or down arrow - move down
        if code == 0x51 or code == 0x28:
            pos += 1
            continue
        # up arrow - move up
        if code == 0x52:
            pos -= 1
            continue

        # select the character based on the Shift key
        while len(lines) <= pos:
            lines.append("")
        if code in range(4, 81):
            if int(x[0:2], 16) == 2:
                lines[pos] += usb_codes[code][1]
            else:
                lines[pos] += usb_codes[code][0]

    for x in lines:
        print(x)


if __name__ == "__main__":
    code2chr('E://CTF练习/杂项/18e4c103d4de4f07b33a42cb1f0eaa1d/00000122/usbdata.txt')
```

关于**USB**流量：

USB 接口是目前最为通用的外设接口之一，通过监听该接口的流量，可以得到很多有意思的东西，例如键盘击键，鼠标移动与点击，存储设备的明文传输通信、USB 无线网卡网络传输内容等。使用wireshark就可以捕获。

USB有不同的规格，比较常见的有以下三种：

> USB USRT：这种方式下，设备只是简单的将USB用于接受和发射数据，除此之外就再没有其他通讯功能了。
>
> USB HID：人性化的接口。这一类通讯适用于交互式，有这种功能的设备有：键盘，鼠标，游戏手柄和数字显示设备。
>
> USB Memory：数据存储。External HDD, thumb drive / flash drive,等都是这一类的。

可以使用Wireshark分析USB流量，USB协议的数据部分在Leftover Capture Data域之中，在Mac和Linux下可以用tshark命令可以将 leftover capture data单独提取出来，命令如下：

```
tshark -r example.pcap -T fields -e usb.capdata
```

在Windows下去tshark.exe的目录下输入上面的命令即可。