Find My Pass - HackTM CTF Quals 2020

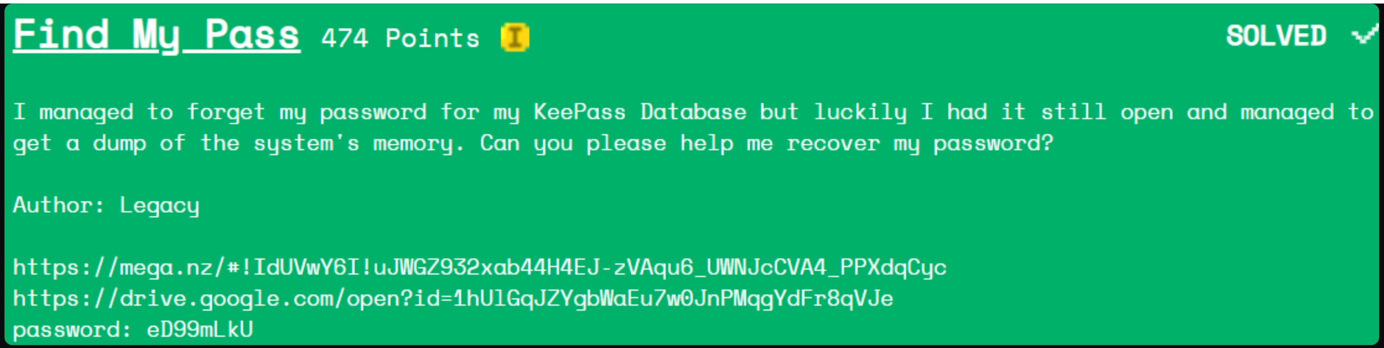
stuxn3t 2020-02-09 Forensics / Memory 1028

tl;dr

- Memory dump analysis using Volatility.
- Extracting Keepass Master Password from the memory.
- Extracting flag from ZIP archive attached in the Keepass database.

Challenge points: 474
No. of solves: 24
Solved by: stuxn3t

Challenge Description



Find My Pass 474 Points

SOLVED

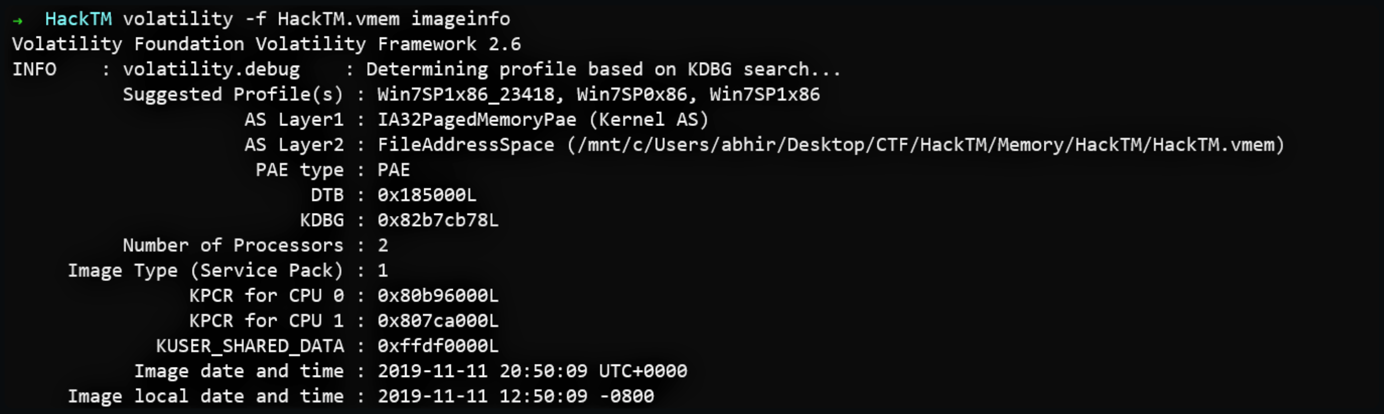
```
I managed to forget my password for my KeePass Database but luckily I had it still open and managed to
get a dump of the system's memory. Can you please help me recover my password?

Author: Legacy

https://mega.nz/#!IdUVwY6I!uJWGZ932xab44H4EJ-zVAqu6_UWNJcCVA4_PPXdqCyc
https://drive.google.com/open?id=1hUlGqJZYgbWaEu7w0JnPMqqYdFr8qVJe
password: eD99mLkU
```

The challenge file can be downloaded from Google-Drive or Mega-Drive.

Initial Analysis

The challenge description tells us that the user lost his Keepass master password. So we need to retrieve the password from the memory and also it is provided that the database is also open when the memory dump was taken.

First, we need to find what OS his system was using. For this, I used the `imageinfo` plugin.

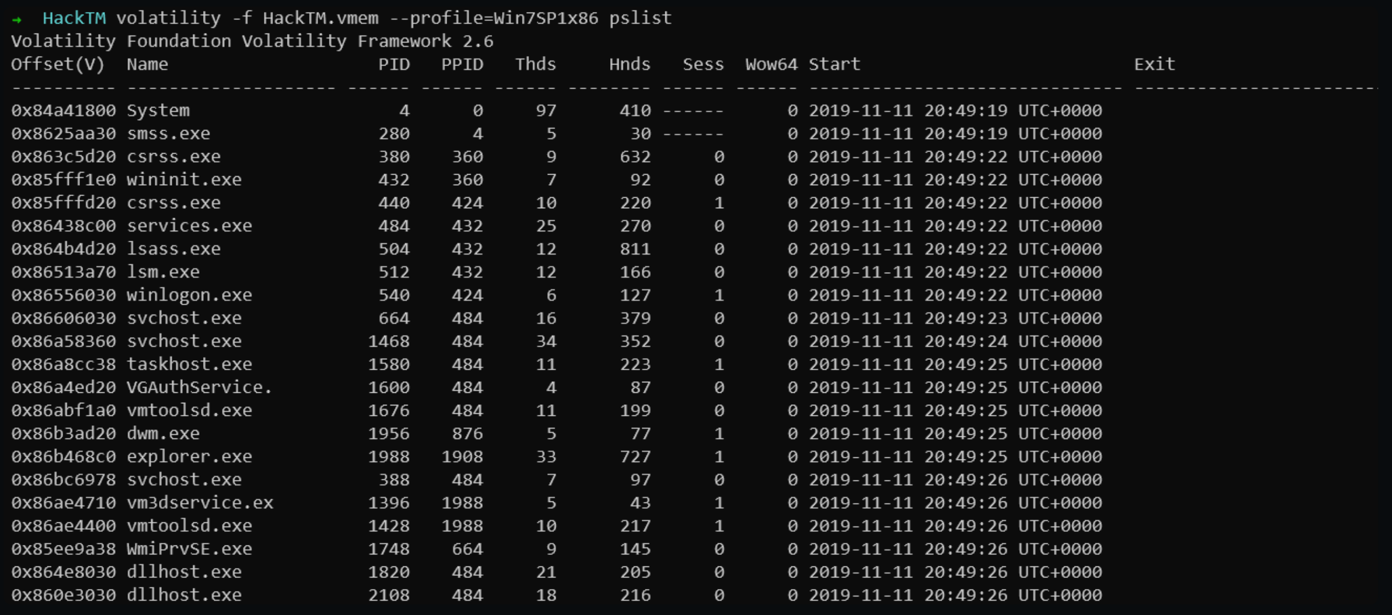


```
→ HackTM volatility -f HackTM.vmem imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/mnt/c/Users/abhir/Desktop/CTF/HackTM/Memory/HackTM/HackTM.vmem)
                      PAE type : PAE
                           DTB : 0x185000L
                          KDBG : 0x82b7cb78L
          Number of Processors : 2
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0x80b96000L
                KPCR for CPU 1 : 0x807ca000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2019-11-11 20:50:09 UTC+0000
     Image local date and time : 2019-11-11 12:50:09 -0800
```

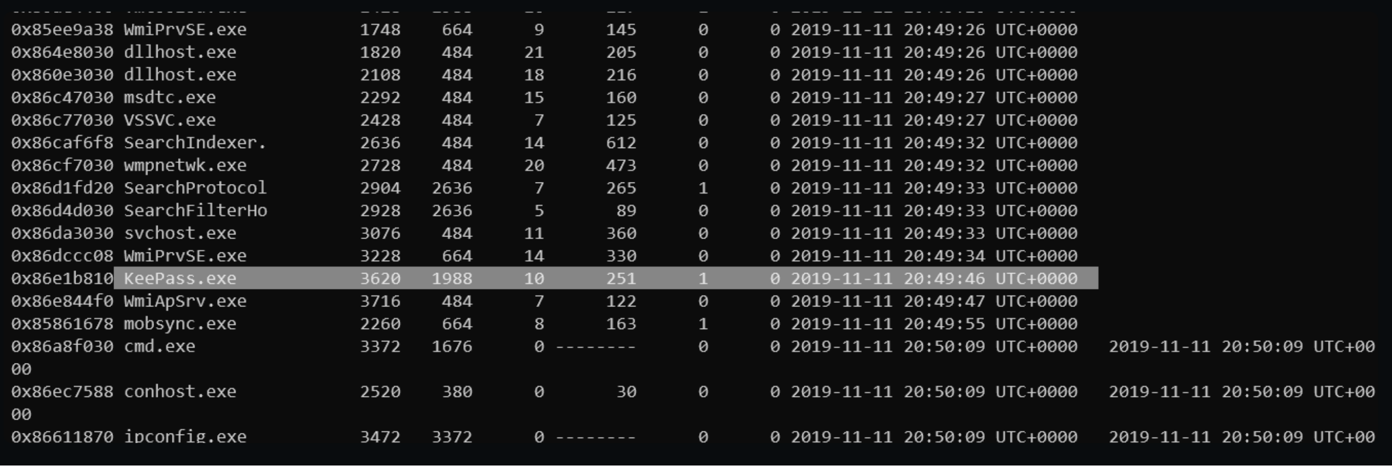
Table of Contents

I chose the profile `Win7SP1x86`.

Since the description provides us with initial information stating that the Keepass database is open, one can easily expect that `Keepass.exe` will be running in the list of running processes.



```
→  HackTM volatility -f HackTM.vmem --profile=Win7SP1x86 pslist
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
---------- -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0x84a41800 System                    4      0     97      410 ------      0 2019-11-11 20:49:19 UTC+0000
0x8625aa30 smss.exe                280      4      5       30 ------      0 2019-11-11 20:49:19 UTC+0000
0x863c5d20 csrss.exe               380    360      9      632      0      0 2019-11-11 20:49:22 UTC+0000
0x85fff1e0 wininit.exe             432    360      7       92      0      0 2019-11-11 20:49:22 UTC+0000
0x85fffd20 csrss.exe               440    424     10      220      1      0 2019-11-11 20:49:22 UTC+0000
0x86438c00 services.exe            484    432     25      270      0      0 2019-11-11 20:49:22 UTC+0000
0x864b4d20 lsass.exe               504    432     12      811      0      0 2019-11-11 20:49:22 UTC+0000
0x86513a70 lsm.exe                 512    432     12      166      0      0 2019-11-11 20:49:22 UTC+0000
0x86556030 winlogon.exe            540    424      6      127      1      0 2019-11-11 20:49:22 UTC+0000
0x86606030 svchost.exe             664    484     16      379      0      0 2019-11-11 20:49:23 UTC+0000
0x86a58360 svchost.exe            1468    484     34      352      0      0 2019-11-11 20:49:24 UTC+0000
0x86a8cc38 taskhost.exe           1580    484     11      223      1      0 2019-11-11 20:49:25 UTC+0000
0x86a4ed20 VGAuthService.         1600    484      4       87      0      0 2019-11-11 20:49:25 UTC+0000
0x86abf1a0 vmtoolsd.exe           1676    484     11      199      0      0 2019-11-11 20:49:25 UTC+0000
0x86b3ad20 dwm.exe                1956    876      5       77      1      0 2019-11-11 20:49:25 UTC+0000
0x86b468c0 explorer.exe           1988   1908     33      727      1      0 2019-11-11 20:49:25 UTC+0000
0x86bc6978 svchost.exe             388    484      7       97      0      0 2019-11-11 20:49:26 UTC+0000
0x86ae4710 vm3dservice.ex         1396   1988      5       43      1      0 2019-11-11 20:49:26 UTC+0000
0x86ae4400 vmtoolsd.exe           1428   1988     10      217      1      0 2019-11-11 20:49:26 UTC+0000
0x85ee9a38 WmiPrvSE.exe           1748    664      9      145      0      0 2019-11-11 20:49:26 UTC+0000
0x864e8030 dllhost.exe            1820    484     21      205      0      0 2019-11-11 20:49:26 UTC+0000
0x860e3030 dllhost.exe            2108    484     18      216      0      0 2019-11-11 20:49:26 UTC+0000
```

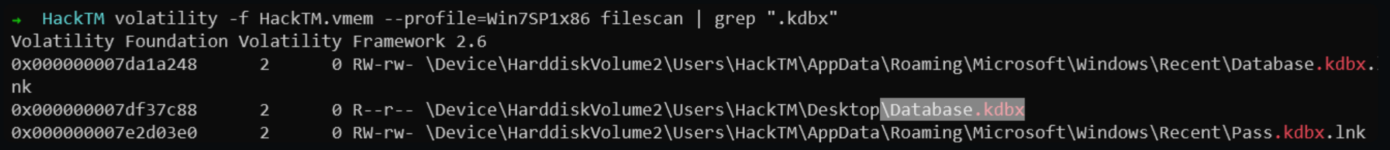


```
0x85ee9a38 WmiPrvSE.exe          1748    664      9      145      0      0 2019-11-11 20:49:26 UTC+0000
0x864e8030 dllhost.exe           1820    484     21      205      0      0 2019-11-11 20:49:26 UTC+0000
0x860e3030 dllhost.exe           2108    484     18      216      0      0 2019-11-11 20:49:26 UTC+0000
0x86c47030 msdtc.exe             2292    484     15      160      0      0 2019-11-11 20:49:27 UTC+0000
0x86c77030 VSSVC.exe             2428    484      7      125      0      0 2019-11-11 20:49:27 UTC+0000
0x86caf6f8 SearchIndexer.        2636    484     14      612      0      0 2019-11-11 20:49:32 UTC+0000
0x86cf7030 wmpnetwk.exe          2728    484     20      473      0      0 2019-11-11 20:49:32 UTC+0000
0x86d1fd20 SearchProtocol        2904   2636      7      265      1      0 2019-11-11 20:49:33 UTC+0000
0x86d4d030 SearchFilterHo        2928   2636      5       89      0      0 2019-11-11 20:49:33 UTC+0000
0x86da3030 svchost.exe           3076    484     11      360      0      0 2019-11-11 20:49:33 UTC+0000
0x86dccc08 WmiPrvSE.exe          3228    664     14      330      0      0 2019-11-11 20:49:34 UTC+0000
0x86e1b810 KeePass.exe           3620   1988     10      251      1      0 2019-11-11 20:49:46 UTC+0000
0x86e844f0 WmiApSrv.exe          3716    484      7      122      0      0 2019-11-11 20:49:47 UTC+0000
0x85861678 mobsync.exe           2260    664      8      163      1      0 2019-11-11 20:49:55 UTC+0000
0x86a8f030 cmd.exe               3372   1676      0 --------      0      0 2019-11-11 20:50:09 UTC+0000   2019-11-11 20:50:09 UTC+00
00
0x86ec7588 conhost.exe           2520    380      0       30      0      0 2019-11-11 20:50:09 UTC+0000   2019-11-11 20:50:09 UTC+00
00
0x86611870 ipconfig.exe          3472   3372      0 --------      0      0 2019-11-11 20:50:09 UTC+0000   2019-11-11 20:50:09 UTC+00
```

So my thought process was correct and so the next step would be to extract the database file from the memory.

Extracting the Keepass database

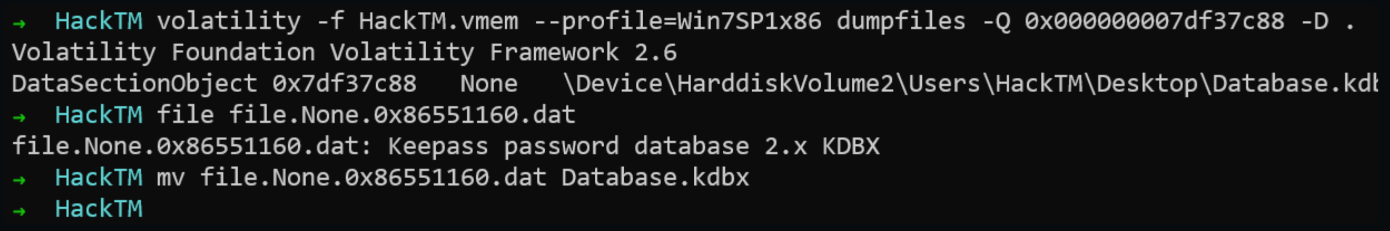
All KeePass database files have the extension `.kdbx`. So I used the `filescan` plugin to get the offset of the database.



```
→  HackTM volatility -f HackTM.vmem --profile=Win7SP1x86 filescan | grep ".kdbx"
Volatility Foundation Volatility Framework 2.6
0x000000007da1a248      2      0 RW-rw- \Device\HarddiskVolume2\Users\HackTM\AppData\Roaming\Microsoft\Windows\Recent\Database.kdbx.
nk
0x000000007df37c88      2      0 R--r-- \Device\HarddiskVolume2\Users\HackTM\Desktop\Database.kdbx
0x000000007e2d03e0      2      0 RW-rw- \Device\HarddiskVolume2\Users\HackTM\AppData\Roaming\Microsoft\Windows\Recent\Pass.kdbx.lnk
```

From the image above, we can see that the file `database.kdbx` is present at the offset `0x000000007df37c88`.

We can use the `dumpfiles` plugin to extract the file out.



```
→  HackTM volatility -f HackTM.vmem --profile=Win7SP1x86 dumpfiles -Q 0x000000007df37c88 -D .
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x7df37c88   None   \Device\HarddiskVolume2\Users\HackTM\Desktop\Database.kdt
→  HackTM file file.None.0x86551160.dat
file.None.0x86551160.dat: Keepass password database 2.x KDBX
→  HackTM mv file.None.0x86551160.dat Database.kdbx
→  HackTM
```

When I attempted to open the database, it asked for the master password. The challenge description tells us that the user forgot his password, which means that we have to recover it.

Retrieving the Master Password

So one thing is certain. The database was “open” when the memory dump was taken. So we can also expect the master password to be loaded in the process’s memory.

So let us use the `memdump` plugin to extract the process’s memory. The process of interest, in this case, is the Keepass.exe with the `PID 3620`.

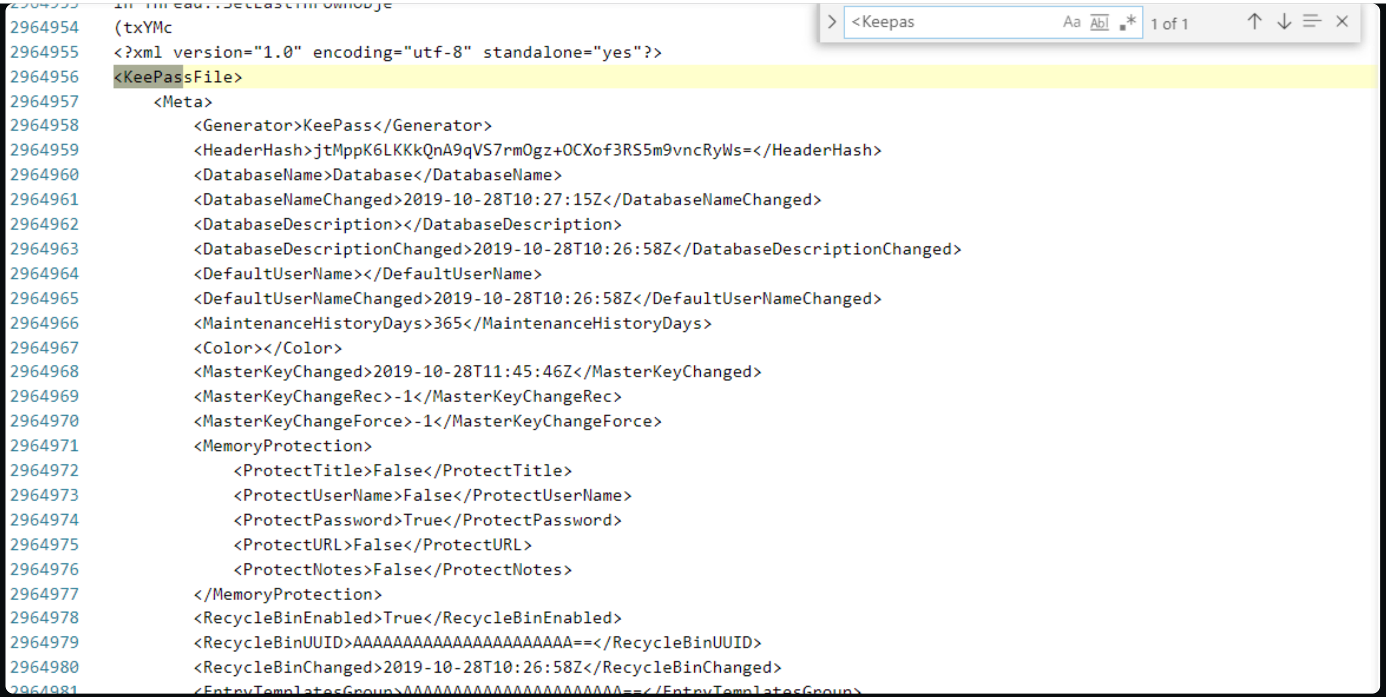
```
→  HackTM volatility -f HackTM.vmem --profile=Win7SP1x86 memdump -p 3620 -D dump
Volatility Foundation Volatility Framework 2.6
************************************************************************
Writing KeePass.exe [  3620] to 3620.dmp
→  HackTM dump/
→  dump ls
3620.dmp
→  dump
```

The password is obviously a readable ASCII character so let me extract what I need.

So for this, I extracted the readable data from the memory dump.

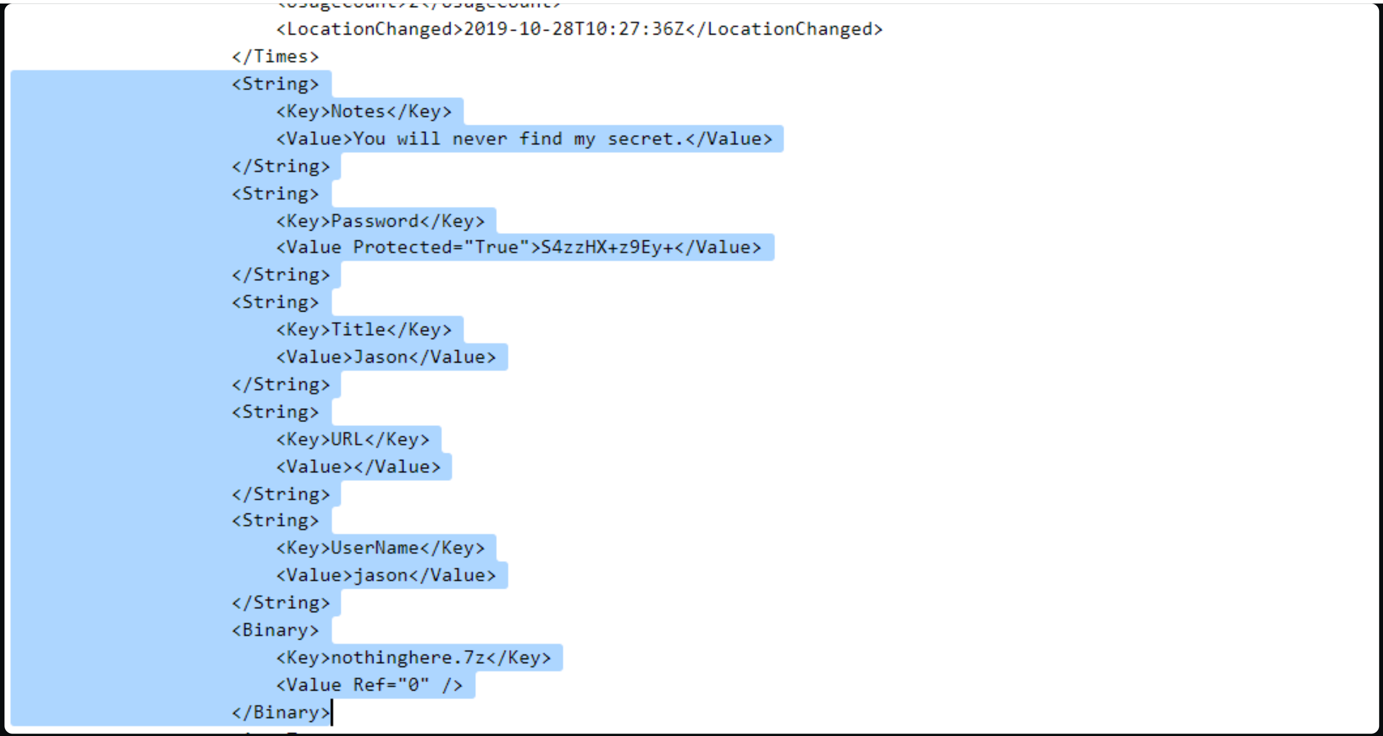
`strings -n 5 HackTM.vmem > new`

Now comes the tricky part. I now have to search where the Keepass database is loaded. It normally starts with an XML tag at the start.



```
in Thread::SetLastThrownObje
(txYMc
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<KeePassFile>
    <Meta>
        <Generator>KeePass</Generator>
        <HeaderHash>jtMppK6LKKkQnA9qVS7rmOgz+OCXof3RS5m9vncRyWs=</HeaderHash>
        <DatabaseName>Database</DatabaseName>
        <DatabaseNameChanged>2019-10-28T10:27:15Z</DatabaseNameChanged>
        <DatabaseDescription></DatabaseDescription>
        <DatabaseDescriptionChanged>2019-10-28T10:26:58Z</DatabaseDescriptionChanged>
        <DefaultUserName></DefaultUserName>
        <DefaultUserNameChanged>2019-10-28T10:26:58Z</DefaultUserNameChanged>
        <MaintenanceHistoryDays>365</MaintenanceHistoryDays>
        <Color></Color>
        <MasterKeyChanged>2019-10-28T11:45:46Z</MasterKeyChanged>
        <MasterKeyChangeRec>-1</MasterKeyChangeRec>
        <MasterKeyChangeForce>-1</MasterKeyChangeForce>
        <MemoryProtection>
            <ProtectTitle>False</ProtectTitle>
            <ProtectUserName>False</ProtectUserName>
            <ProtectPassword>True</ProtectPassword>
            <ProtectURL>False</ProtectURL>
            <ProtectNotes>False</ProtectNotes>
        </MemoryProtection>
        <RecycleBinEnabled>True</RecycleBinEnabled>
        <RecycleBinUUID>AAAAAAAAAAAAAAAAAAAAAA==</RecycleBinUUID>
        <RecycleBinChanged>2019-10-28T10:26:58Z</RecycleBinChanged>
        <EntryTemplatesGroup>AAAAAAAAAAAAAAAAAAAAAA==</EntryTemplatesGroup>
```

Going further down in the same file, I could see that an attachment (nothinghere.7z) was stored in the Keepass database. So this might be the file where our flag was stored.



```
            <LocationChanged>2019-10-28T10:27:36Z</LocationChanged>
        </Times>
        <String>
            <Key>Notes</Key>
            <Value>You will never find my secret.</Value>
        </String>
        <String>
            <Key>Password</Key>
            <Value Protected="True">S4zzHX+z9Ey+</Value>
        </String>
        <String>
            <Key>Title</Key>
            <Value>Jason</Value>
        </String>
        <String>
            <Key>URL</Key>
            <Value></Value>
        </String>
        <String>
            <Key>UserName</Key>
            <Value>jason</Value>
        </String>
        <Binary>
            <Key>nothinghere.7z</Key>
            <Value Ref="0" />
        </Binary>
```

Going further down, I noticed a weird looking string (dmVZQmdzOlUrcEBlRj87dHQ3USVBIn) which I thought to be the password. So let us try it out.

Voila! It turns out that I got the master password. Now it is the final part - Getting the flag.

Flag

As soon as I opened the database, I tried to retrieve the files present in the 7z archive. Unfortunately, the archive was password protected. At this stage, I pinged the admin asking whether brute force was needed to extract the flag. He told me that it wasn't needed and subsequently a hint was released which eventually implied that the user used the same password everywhere.

So now I tried the same password (database master password) on the archive. Yay! We successfully extracted the text file present in it and as expected, it contains the flag.

We also got the **FIRST BLOOD** in this challenge. So I was quite happy :)

FLAG: `HackTM{d14c02244b17f4f9dfc0f71ce7ab10e276a5880a05fca64d39a716bab92cda90}`

For further queries, feel free to message me on Twitter: https://twitter.com/_abhiramkumar

2021-03-05 # Windows Memory Analysis # HackTM

< Previous

Think twice before speaking once - HackTM CTF Quals 2020

HackTM

Next >

RR - HackTM CTF Quals 2020

HackTM # RAID Recovery