Memory Analysis of Password Managers

Shantanu Jha, Vishal Wadhwani

CY 5130 : Computer Systems Security

12/14/2019

Abstract

This paper highlights the observations made from the memory analysis of Lastpass Password Manager(PM). Here, we test the PMs on Windows and Linux platforms. We have provided detailed analysis of Lastpass PMs working on both the platforms. We describe the major differences in the way the processes are spawned for each PM and how does it reduce or increase the attack surface. We also describe the future scope of this work.

Introduction

At the highest level, attacks on password managers can be classified into two types : 1. Attacks generated from outside the system, 2. Attacks generated from inside the system.

PMs use various strategies to perform an autofill on a webpage. Autofill can be automatic, manual or hybrid. The choice of which kind of autofill should be used on a particular webpage is made after evaluating various parameters. For instance, an automatic autofill can only be performed on HTTPS webpages. The autofill functionality can be exploited by an attacker who is in control of the users network by injecting invisible iframes into the requested web page by a user. We have mentioned this example just to highlight the first category of attacks. But, we majorly focus on the second category where the attacks are generated from an infected system. Narrowly speaking, we will focus on analysing systems memory for any unsafe practices.

Jeffery Goldberg, 1Password's Chief Defender Against the Dark Arts, said, an attacker who is in a position to exploit the information in memory is already in a very powerful position. No password manager (or anything else) can promise to run securely on a compromised computer. We agree with the statement, but, our only motive is to reduce the attack surface of the currently available system.

Tools Used

Linux : strace, gdb, process structure using (/proc/$pid/maps)

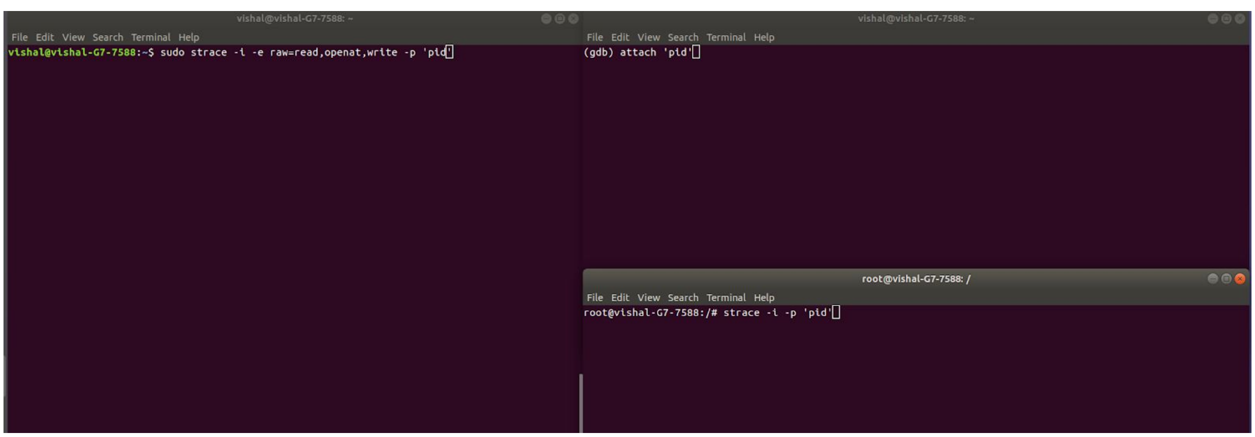
Windows : Interactive Disassembler (IDA), WinDbg, Volatility, Task Manager

Linux

Lastpass

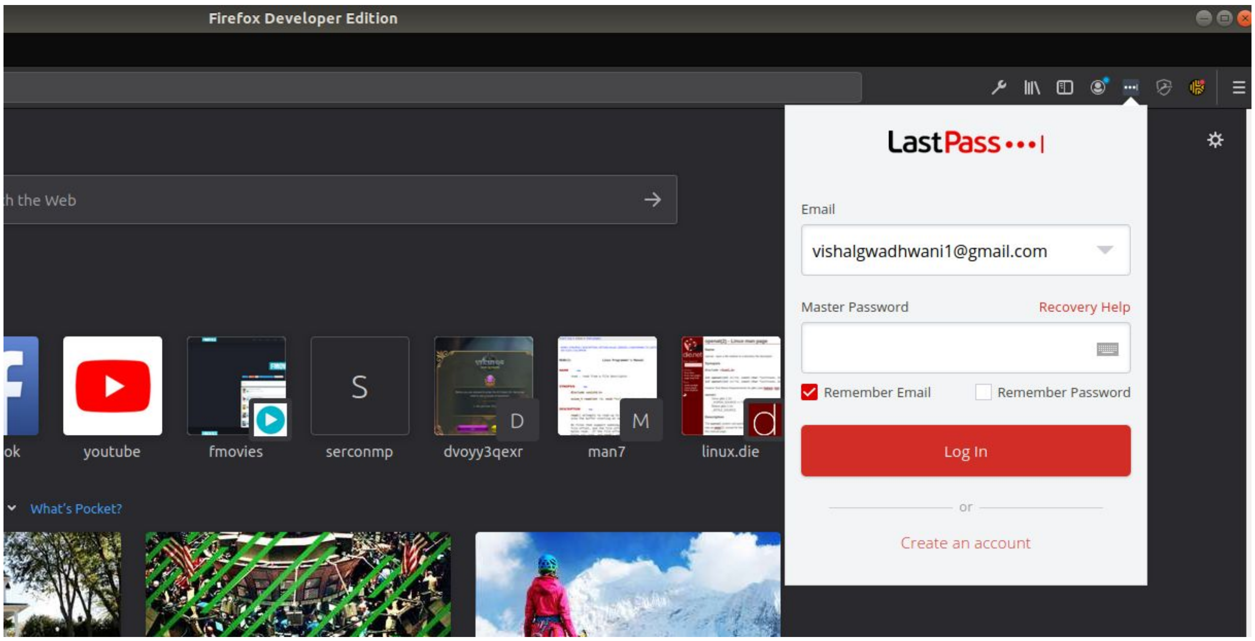
Observation 1 : Lastpass's linux process sets up multiple buffers in memory for different purposes. Everytime it has to achieve the same purpose, it rewrites the data in the buffer instead of clearing it out and then writing into it.

1. Debugging setup :
   a. Trace for system call with raw memory details
   b. Gdb attached to the process for checking memory
   c. Trace for system call with runtime memory details

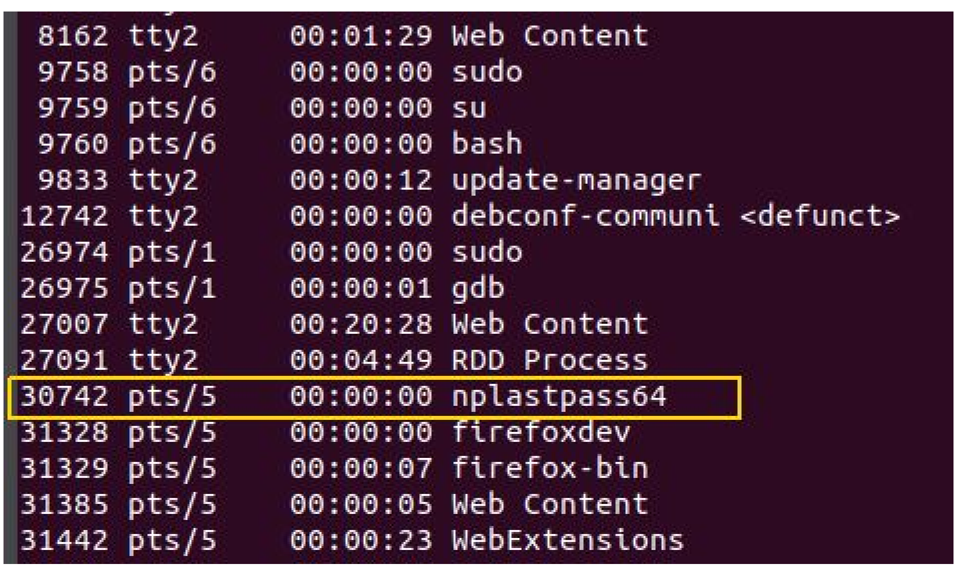


2. Start by checking the initial commands provided to the process by fd=0. (This can be done by starting the browser and a new lastpass out of extension process spawns. Don't login at this moment.)

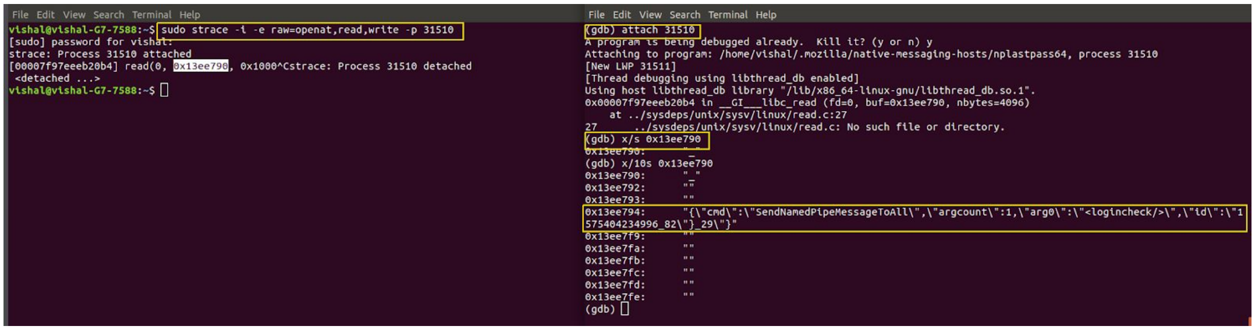
    a. Browser opened and lastpass isn't logged in.



    b. New out of extension process is spawned.



c. Initiating the debugging setup with the latest pid i.e. in our case 31510.



Observation 2 : The Basic process communication happens with the following command structure.

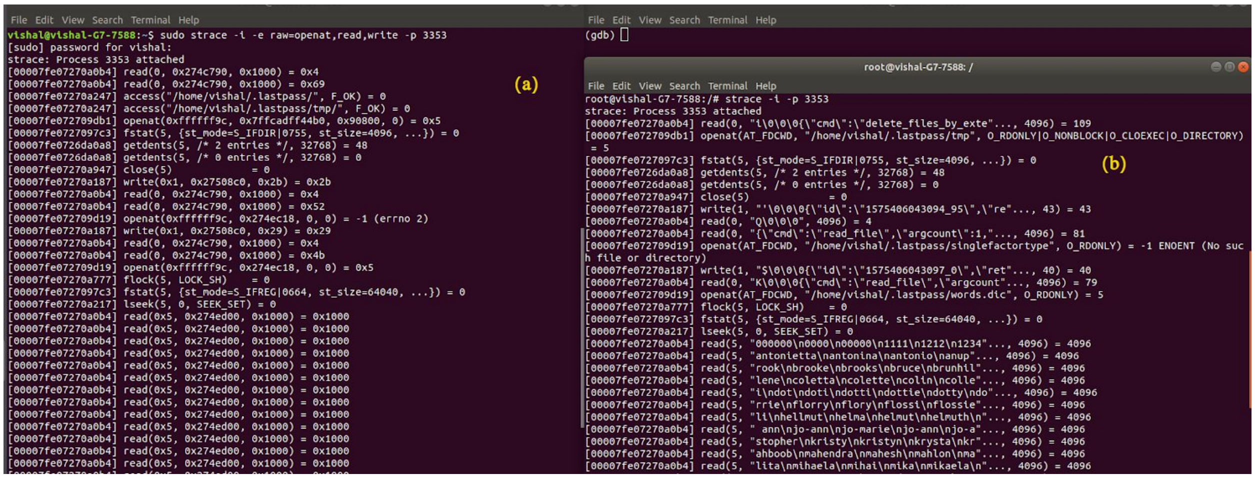
“{\"cmd\":\"SendNamedPipeMessageToAll\",\"argcount\":1,\"arg0\":\"<logincheck/>\",\"id\":\"1575404234996_82\"}_29\"}”

(Highlighted command gives us the idea that the process is performing a login check. Now since we have not entered the password, this is the first message that we could extract from the memory buffer whose offset was obtained from the processes system call trace.)

3. Now, we log in with the legitimate password and logout and analyse the system calls trace and buffer memory after that.

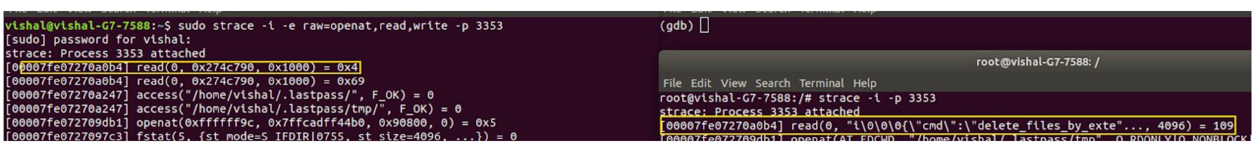
    a. Raw Trace

    b. Trace with data in the buffer at runtime.

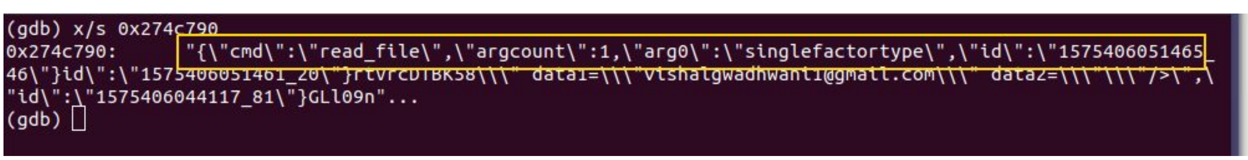


Figures (a) and (b) are similar traces. The only difference is the raw trace contains the physical addresses of the buffers whereas the normal trace contains the data in buffer at runtime.

    c. As per observation 1, Laspass overwrites buffers. Proof of that is shown below.



Buffer values during runtime

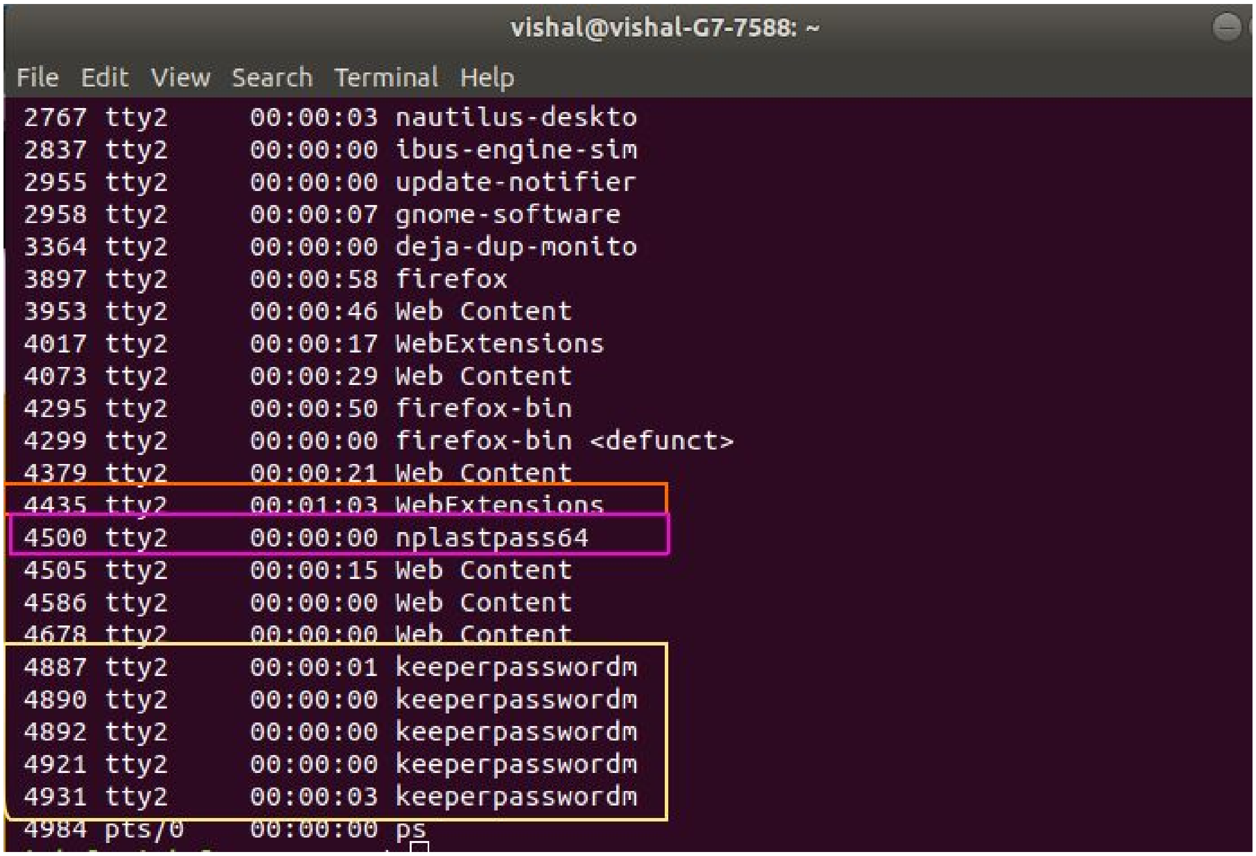


Buffer values currently in memory

Observation 3 : Username is passed in plain text.

    d. Scanning through the trace.

Observation 4 : The contents of this file are altered at every login.



Dashlane : Highlighted in orange, dashlane runs in the webextension process created by firefox.

Lastpass : HIghlighted in pink, lastpass extension has a separate process unlike dashlane.

Out-of-process Extensions : The actual extension code will run in a content process. The main process will load a process script into the extension process and the two processes will communicate using the process message manager (IPC mechanism).

Keeper : Since we installed a standalone software, Keeper is bound to have it's own process. What's interesting is it spawns 4 different processes when run. Similar to how out-of-process extensions communicate with the main process script in the WebExtension process, keeper also has two processes that handle communication and one process that performs actual task.

Future Scope

Based on the observations, the work can be extended to the analysis of web extensions where we can check if a malicious extension can somehow access the memory of another extension that runs in the WebExtension process. At a high level, we can check if there is a security disadvantage of running web extensions in one process.

Windows

We started with LastPass desktop application on Windows. We reviewed the LastPass security model and architecture. This served as a starting point of the analysis and also compare the findings with their security guarantees.

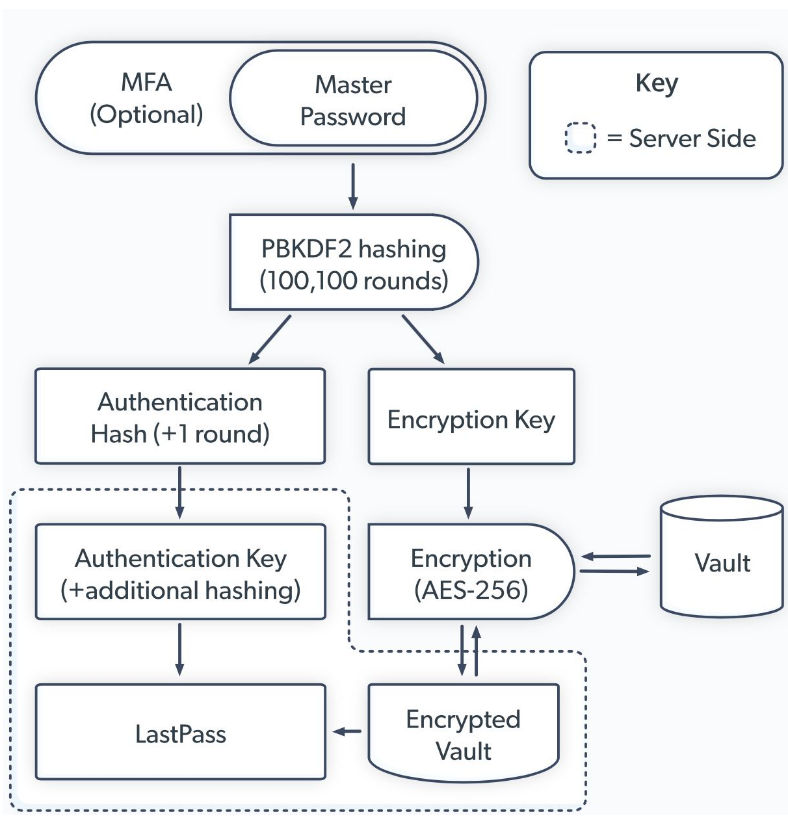
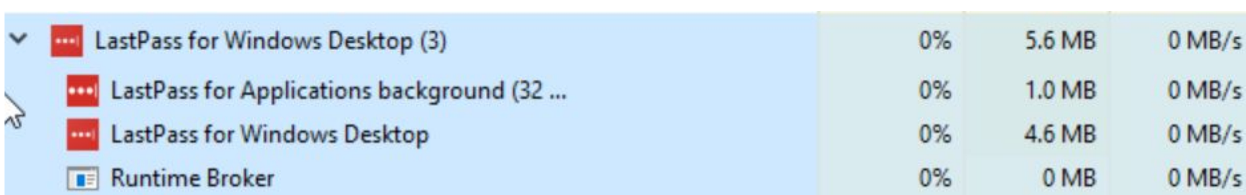


Image taken from https://www.lastpass.com/enterprise/security

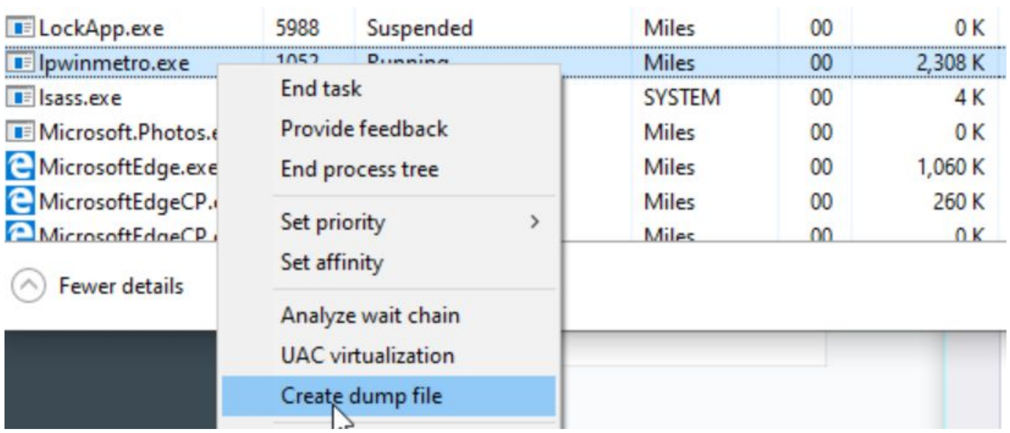
We see that the master password is used to generate an encryption key. They are using AES-256 to encrypt and decrypt the vault. This key along with Authentication Key is used to show passwords on the LastPass app.

First we looked at the processes created by LastPass when it starts up. There are 3 processes created as seen below in the Task Manager. We are interested in the first two in that list of 3 processes. LastPass for Windows Desktop is the process corresponding to the desktop application. There is a background process which does a lot of behind the scenes operations. These operations include creating guarded memory regions, this is used to avoid memory dumping. It allocates memory with PAGE_GUARD access rights. It reads the computer name, cryptographic machine GUID and query kernel debugger information.



We used WinDbg which is a windows debugging tool by Microsoft. The aim was to use the tool to understand the program flow when we did certain operations on the application. Unfortunately, WinDbg attached to LastPass didn't give much useful information. For every operation that we executed on the application, like adding a new password, copying passwords, editing passwords, etc., LastPass made calls from statemanger.cpp and stateatom.cpp which used kernelbase.dll to execute the requested operations. Moreover, the memory contents were obfuscated.

Then we decided that we are going to analyze memory dump files after executing operations. Even though the background process is supposed to have these memory regions protected, getting the dump file for the process was very easy.



We analyzed this dump file using Interactive Disassembler (IDA) as a binary file. We weren't concerned about the memory headers and windows specific dump file structure. We wanted to read the memory contents and reading it as a binary file was appropriate. We made a list of operations and what we want to look for after execution. We looked at dump files in these following scenarios:

1. After startup. We haven't logged in yet
2. After logging in. We dump the process immediately after logging in with the master password.
3. After saving a new password for an account.
4. After copying the username only.
5. After copying the password only.
6. After logging out of LastPass.
7. Can we get some cryptographic information from the memory.

Observations

Observation 1:

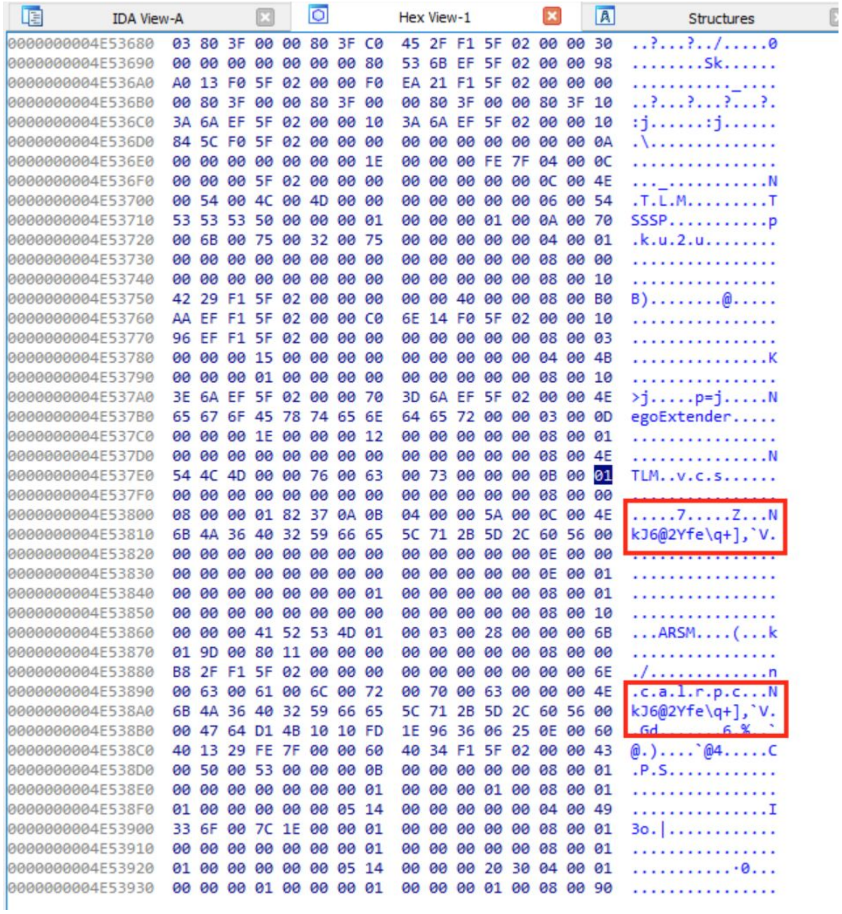
After starting up LastPass Windows App (not logged in yet), we only see the DLLs load in memory with their paths and other logistics like time zone, language etc. We don't expect to find anywhere here. But, we find a key named **pwdeckey.** This can't be related to the password vault key because we haven't entered the master password yet. As we saw before, LastPass uses the master password to generate the symmetric key for encryption and decryption. So, this key seems to be something else. Regardless, a key is kept in memory in plaintext.

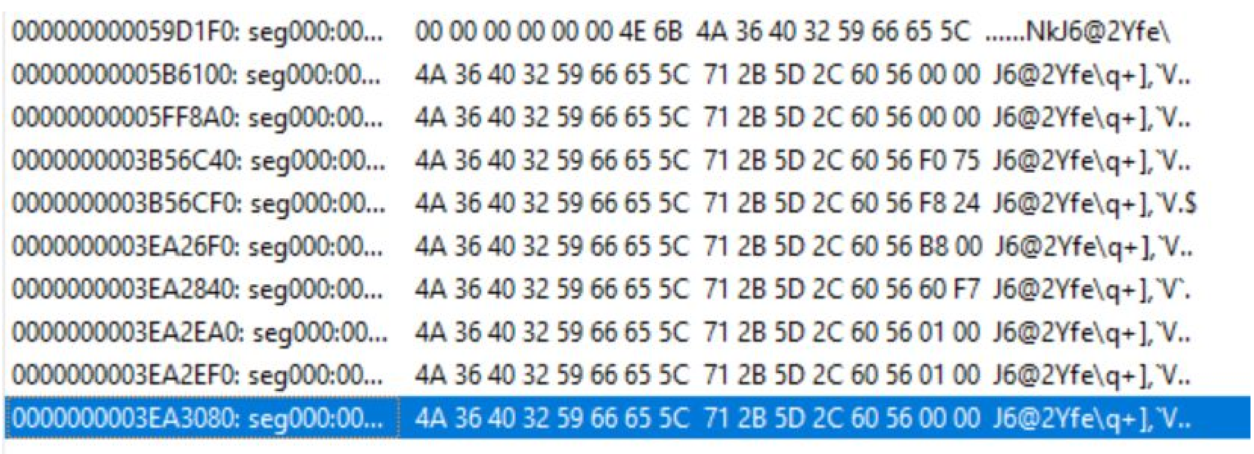
```
0000000000961560  63 6C 69 65 6E 74 5F 65  6E 63 3D 22 31 22 20 61  client_enc="1"·a
0000000000961570  63 63 74 73 5F 76 65 72  73 69 6F 6E 3D 22 31 35  ccts_version="15
0000000000961580  22 20 70 77 64 65 63 6B  65 79 3D 22 65 65 63 31  "·pwdeckey="eec1
0000000000961590  36 36 34 30 64 36 37 61  64 35 64 34 36 35 37 35  6640d67ad5d46575
00000000009615A0  38 63 38 39 65 31 65 35  39 62 61 30 66 38 32 34  8c89e1e59ba0f824
00000000009615B0  64 63 35 36 34 37 38 34  35 61 64 63 38 32 64 66  dc5647845adc82df
00000000009615C0  34 62 66 64 35 38 33 63  31 39 31 33 22 0A 20 20  4bfd583c1913".··
00000000009615D0  20 20 20 20 20 20 20 20  20 20 20 20 20 20 68 69  ··············hi
00000000009615E0  68 3D 22 30 22 20 67 65  6E 68 3D 22 30 22 20 61  h="0"·genh="0"·a
00000000009615F0  64 64 68 3D 22 30 22 0A  20 20 20 20 20 20 20 20  ddh="0".········
```

Observation 2:

Next, we logged in with the master password and immediately dumped the process. The aim was to locate the master password in memory. If we did find it, we try to see if it is in obfuscated form or in plain text. As seen in the screenshots below, we saw the master password in plaintext. Additionally, we found the password in memory in plaintext 10 times.

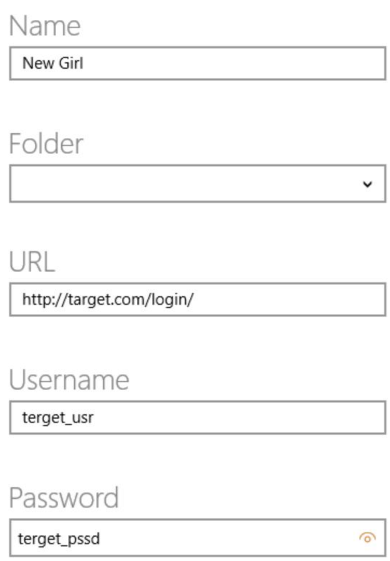
We looked for the above **pwdeckey** after logging in. The keys were still present. We were not able to get to the root of the use of they key.

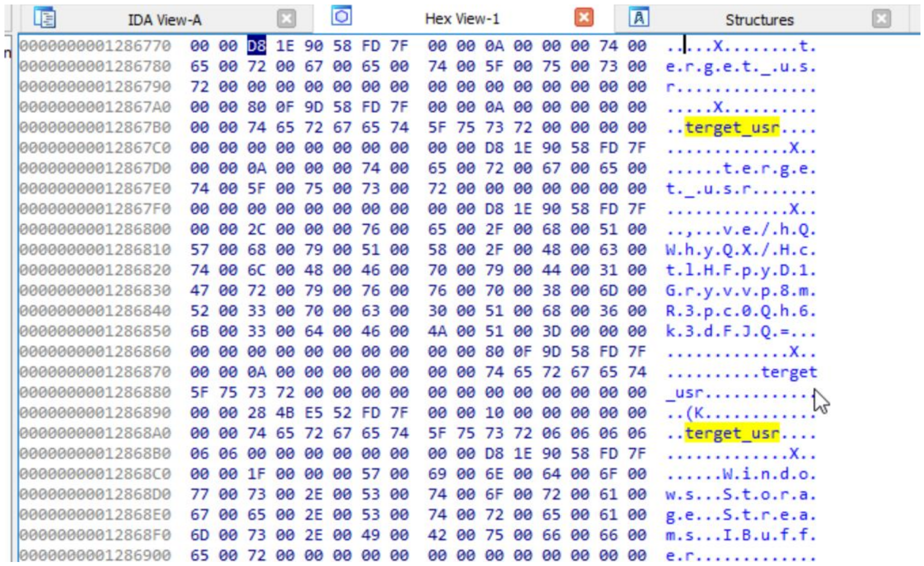

IDA View-A
Hex View-1
Structures
0000000004E53680 03 80 3F 00 00 80 3F C0 45 2F F1 5F 02 00 00 30 ..?...?../.....0
0000000004E53690 00 00 00 00 00 00 00 80 53 6B EF 5F 02 00 00 98 ....Sk......
0000000004E536A0 A0 13 F0 5F 02 00 00 F0 EA 21 F1 5F 02 00 00 00 ..._.....!._....
0000000004E536B0 00 80 3F 00 00 80 3F 00 00 80 3F 00 00 80 3F 10 ..?...?...?...?.
0000000004E536C0 3A 6A EF 5F 02 00 00 10 3A 6A EF 5F 02 00 00 10 :j......:j......
0000000004E536D0 84 5C F0 5F 02 00 00 00 00 00 00 00 00 00 00 0A .\..............
0000000004E536E0 00 00 00 00 00 00 00 1E 00 00 00 FE 7F 04 00 0C ....
0000000004E536F0 00 00 00 5F 02 00 00 00 00 00 00 00 00 0C 00 4E ..._...........N
0000000004E53700 00 54 00 4C 00 4D 00 00 00 00 00 00 00 06 00 54 .T.L.M.........T
0000000004E53710 53 53 53 50 00 00 00 01 00 00 00 01 00 0A 00 70 SSSP...........p
0000000004E53720 00 6B 00 75 00 32 00 75 00 00 00 00 00 04 00 01 .k.u.2.u........
0000000004E53730 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 00 ....
0000000004E53740 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 10 ....
0000000004E53750 42 29 F1 5F 02 00 00 00 00 00 40 00 00 08 00 B0 B)........@.....
0000000004E53760 AA EF F1 5F 02 00 00 C0 6E 14 F0 5F 02 00 00 10 ....
0000000004E53770 96 EF F1 5F 02 00 00 00 00 00 00 00 00 08 00 03 ....
0000000004E53780 00 00 00 15 00 00 00 00 00 00 00 00 00 04 00 4B ....K
0000000004E53790 00 00 00 01 00 00 00 00 00 00 00 00 00 08 00 10 ....
0000000004E537A0 3E 6A EF 5F 02 00 00 70 3D 6A EF 5F 02 00 00 4E >j.....p=j.....N
0000000004E537B0 65 67 6F 45 78 74 65 6E 64 65 72 00 00 03 00 0D egoExtender.....
0000000004E537C0 00 00 00 1E 00 00 00 12 00 00 00 00 00 08 00 01 ....
0000000004E537D0 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 4E ....N
0000000004E537E0 54 4C 4D 00 00 76 00 63 00 73 00 00 00 0B 00 01 TLM..v.c.s......
0000000004E537F0 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 00 ....
0000000004E53800 08 00 00 01 82 37 0A 0B 04 00 00 5A 00 0C 00 4E .....7.....Z...N
0000000004E53810 6B 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 00 kJ6@2Yfe\q+],`V.
0000000004E53820 00 00 00 00 00 00 00 00 00 00 00 00 00 0E 00 00 ....
0000000004E53830 00 00 00 00 00 00 00 00 00 00 00 00 00 0E 00 01 ....
0000000004E53840 00 00 00 00 00 00 00 01 00 00 00 00 00 08 00 01 ....
0000000004E53850 00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 10 ....
0000000004E53860 00 00 00 41 52 53 4D 01 00 03 00 28 00 00 00 6B ...ARSM....(...k
0000000004E53870 01 9D 00 80 11 00 00 00 00 00 00 00 00 08 00 00 ....
0000000004E53880 B8 2F F1 5F 02 00 00 00 00 00 00 00 00 00 00 6E ./.............n
0000000004E53890 00 63 00 61 00 6C 00 72 00 70 00 63 00 00 00 4E .c.a.l.r.p.c...N
0000000004E538A0 6B 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 00 kJ6@2Yfe\q+],`V.
0000000004E538B0 00 47 64 D1 4B 10 10 FD 1E 96 36 06 25 0E 00 60 .Gd.......6.%..`
0000000004E538C0 40 13 29 FE 7F 00 00 60 40 34 F1 5F 02 00 00 43 @.)....`@4.....C
0000000004E538D0 00 50 00 53 00 00 00 0B 00 00 00 00 00 08 00 01 .P.S............
0000000004E538E0 00 00 00 00 00 00 00 01 00 00 00 01 00 08 00 01 ....
0000000004E538F0 01 00 00 00 00 00 05 14 00 00 00 00 00 04 00 49 I
0000000004E53900 33 6F 00 7C 1E 00 00 01 00 00 00 00 00 08 00 01 3o.|............
0000000004E53910 00 00 00 00 00 00 00 01 00 00 00 00 00 08 00 01 ....
0000000004E53920 01 00 00 00 00 00 05 14 00 00 00 20 30 04 00 01 .... 0...
0000000004E53930 00 00 00 01 00 00 00 01 00 00 00 01 00 08 00 90 ....

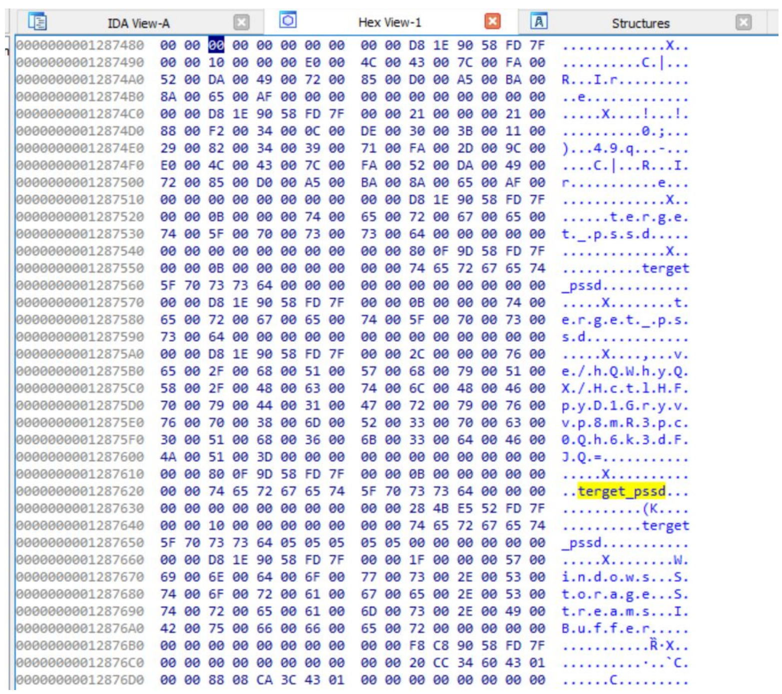

00000000059D1F0: seg000:00... 00 00 00 00 00 00 4E 6B 4A 36 40 32 59 66 65 5C ......NkJ6@2Yfe\
00000000005B6100: seg000:00... 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 00 00 J6@2Yfe\q+],`V..
00000000005FF8A0: seg000:00... 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 00 00 J6@2Yfe\q+],`V..
0000000003B56C40: seg000:00... 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 F0 75 J6@2Yfe\q+],`V..
0000000003B56CF0: seg000:00... 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 F8 24 J6@2Yfe\q+],`V.$
0000000003EA26F0: seg000:00... 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 B8 00 J6@2Yfe\q+],`V..
0000000003EA2840: seg000:00... 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 60 F7 J6@2Yfe\q+],`V`.
0000000003EA2EA0: seg000:00... 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 01 00 J6@2Yfe\q+],`V..
0000000003EA2EF0: seg000:00... 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 01 00 J6@2Yfe\q+],`V..
0000000003EA3080: seg000:00... 4A 36 40 32 59 66 65 5C 71 2B 5D 2C 60 56 00 00 J6@2Yfe\q+],`V..

Observation 3:

Saving new passwords for an account can be seen on the memory dump as well in clear text. Furthermore, the username was also present. Interestingly, the URL was not found in memory in plaintext. After repeatedly doing this for many passwords and analyzing the memory dump, we noticed that all account credentials, usernames and passwords, were located within an estimable bound of memory. LastPass is using a memory region for decrypting account credentials repeatedly. This made our search narrower and thus faster.







Observation 4:

Here, we just copy the username and check memory. We made an interesting observation here. Once we copied the username for an account, we could find the usernames for all accounts in memory in plaintext.

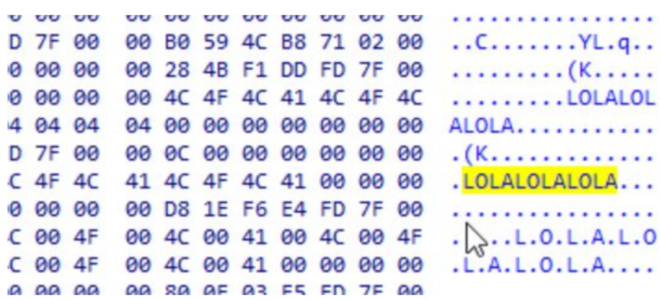
Observation 5:

Here, we copy the password for an account and check the dump. After the previous results, we expected the password to be there in plaintext. Our expectation was correct, the password in plaintext in memory. Furthermore, the corresponding username was also found and interestingly all usernames were found like we observed previously.

Note: We restarted the system and did the operation again to make sure the memory was completely flushed.

All usernames were present in memory in plaintext. This could imply that the entire vault is in memory and only the passwords are encrypted and as we copy the passwords they get decrypted in memory.

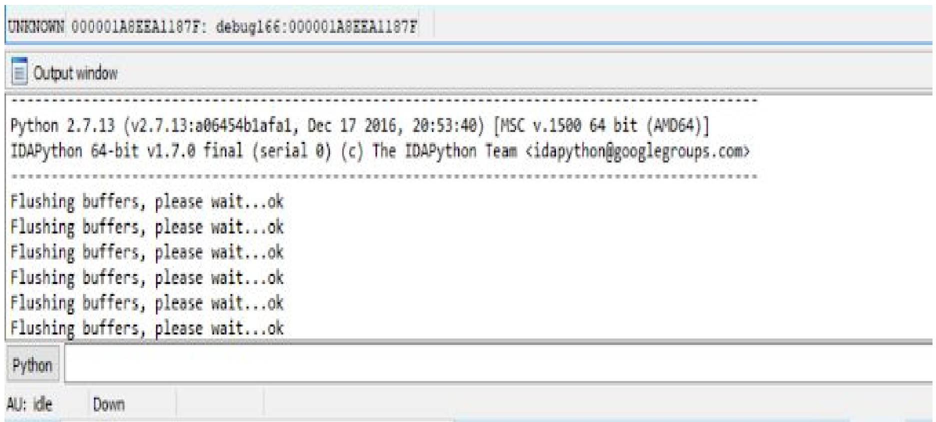
In the image below, we can see the account password for a test account in clear text.



Observation 6:

We log out of the password manager and analyze the memory dump. First we immediately dumped the process memory after logging out. All the usernames and passwords from our previous operations were present. It is noteworthy that the master password was in the memory in plaintext at all times.

Next, we kept the LastPass Desktop App in logged out state but running for 40 minutes and analyzed the memory dump again. We found all the usernames and passwords from our previous operations present just like we found above. This is odd because when LastPass process was attached to WinDbg, the console showed that LastPass is constantly flushing buffers but it doesn't seem like it.



Observation 7:

We got few but significant cryptographic information. We know LastPass is using AES 256 encryption but we found that it has been using the Cipher Block Chaining encryption method (CBC) and they are hashing using SHA 256.

Concluding Remarks

Looking at the working of Lastpass purely from a systems perspective, the web extension follows fairly secure practices whereas the standalone software fails obfuscate memory contents successfully. If LastPass software is in a locked state, then in a given time frame unless the memory location is overwritten with the latest contents, we can extract the master password and any other password copied for usage from the processes memory dump. Keeper's software on the other hand spawns four separate processes unlike the LastPass's single process. Separating the tasks in the form of processes reduces the attack surface. With this project, we had a first hand look at how Firefox handles web extensions and this work can be extended to understanding the attack surface of Firefox web extensions.

References

Kristine Amari, *Techniques and Tools forRecovering and Analyzing Data from Volatile Memory* https://www.sans.org/reading-room/whitepapers/forensics/techniques-tools-recovering-analyzing-data-volatile-memory-33049

Password Managers: Under the Hood of Secrets Management https://www.ise.io/casestudies/password-manager-hacking/

Andrew Case, *Linux Memory Analysis Workshop –Session 1* https://media.blackhat.com/bh-us-11/Case/BH_US_11_Case_Linux_Slides.pdf (Linux Memory Analysis Workshop)

Rupali Sharma, *Fiddling with Linux Processes in Memory* https://www.linux.com/tutorials/fiddling-linux-processes-memory/

Martin Brinkmann*, Firefox 63: Linux out-of-process extensions* https://www.ghacks.net/2018/08/02/firefox-63-linux-out-of-process-extensions/

BackgroundProcess.exe Analysis Report https://hybrid-analysis.com/sample/a27d19086d0ca3de93906f61b0f5b746f51600485f7d43de00e688c82feba884?environmentId=120

Viewing and Editing Memory in WinDbg - Windows drivers. (2017)
https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/memory-windo

Security | *LastPass,* https://www.lastpass.com/enterprise/security

Even the Lastpass will be gone, deal with it!
https://www.blackhat.com/docs/eu-15/materials/eu-15-Vigo-Even-The-Lastpass-Will-Be-Stolen-deal-with-it.pdf