

## Author WU keepa-si-safe:

### 1. Chall Background:

So I wanted to create a forensics challenge for this event, and I stumbled upon an [lppsec video](#) where he breaks Keepass using [CVE-2023-32784](#).

From the NIST website:

*In KeePass 2.x before 2.54, it is possible to recover the cleartext master password from a memory dump, even when a workspace is locked or no longer running. The memory dump can be a KeePass process dump, swap file (pagefile.sys), hibernation file (hiberfil.sys), or RAM dump of the entire system. The first character cannot be recovered. In 2.54, there is different API usage and/or random string insertion for mitigation.*

The main idea is that keepass left residual bits of strings in memory whenever you typed your master password, which you could recover by searching for patterns in the memory itself. I thought it was an interesting vulnerability and started playing around with it.

I downloaded keepass on a vm, created a new Database and added a sub-password entry inside.

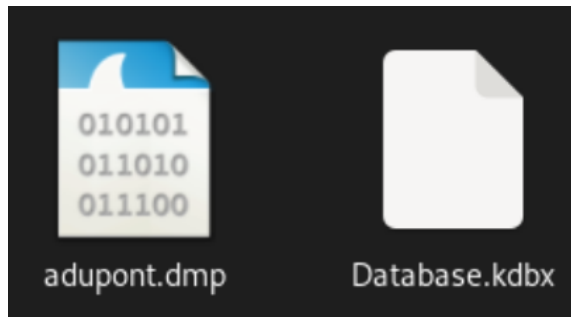
But something weird happened when running one of the few available exploits online, I had bits of my master password intertwined with the sub-password entry. Taking a closer look, I realized that although the CVE on the NIST website says that it “**is possible to recover the cleartext master password from a memory dump**”, what’s actually happening is you can recover “any password last typed in the memory dump”.

If I’d taken the time to thoroughly read the actual post from the original finder of the vuln @vdohney <https://github.com/vdohney/keepass-password-dumper>, which is very well explained, it would have been clear from the start.

Which is why I thought it would be fun to store the password to something else in the keepass db, and see if people realized that it was not the master password 😊

## 2. WRITE-UP:

### 2.1 Start



So you are given a keepass .kdbx file, which is just the keepass standard for storing a database of passwords.

Of course you could immediately look up something online such as “Keepass get master password from .dmp” and you would most likely stumble on the CVE.

As expected many people tried to bf the master-password using the one you found with the CVE (we had a lot of tickets asking why it wasn’t working :) so we ended up adding a hint to make it clearer that this was indeed not the master password.

If you didn’t think of immediately looking for a CVE, then just taking a look at the running processes 3 main ones should pop out (vol3 windows.pslist)

*** 5248	3832	thunderbird.exe	0xab0a37c0	45	-	1	False	2023-05-17 09:12:42.000000	N/A
*** 6008	5248	thunderbird.exe	0xad6d4840	15	-	1	False	2023-05-17 09:12:50.000000	N/A
*** 2896	3832	KeePass.exe	0x9d87b8c0	4	-	1	False	2023-05-17 09:13:27.000000	N/A
*** 4208	3832	powershell.exe	0xa6fef040	14	-	1	False	2023-05-17 09:14:59.000000	N/A

Thunderbird, Keepass, and powershell. The rest are all regular classic services, and hopefully you didn’t spend too much time looking at them.

Thunderbird is particularly interesting because, well, emails. If powershell is running then maybe some commands were run ? And the keepass process will come in handy later

There are actually two separate paths you could have taken to solve the chall, which both lead to the same outcome.

### 2.2 SOLVING USING THUNDERBIRD ONLY

Let’s take a closer look at thunderbird. Without looking for files in particular, you could just dump the process and have a look at the strings. There are quite a few so we’ll try to filter on some useful words.

#### 2.2.1 Thunderbirdbird-Just looking at the strings

If you tried “mail”, “gmail”, “Sent”, “Inbox”, then you would find many references to adupont.rep@gmail.com

Then grepping the strings and looking at some lines before and after  
`strings pid.5248.dmp | grep @gmail.com -A 30 -B 30`

```
Content-Type: multipart/mixed; boundary="-----XJLZoYn0bXsIJ4JsRDDa7k5Y
Message-ID: <40c10fc0-b6fb-4765-a80b-17940bece79a@gmail.com>
Date: Wed, 17 May 2023 08:25:40 GMT +0100
MIME-Version: 1.0
User-Agent: Mozilla Thunderbird
Content-Language: en-US
To: fgalthier.repadmin@gmail.com
From: Antoine Dupont <adupont.rep@gmail.com>
Subject: Recovery file for password manager
This is a multi-part message in MIME format.
-----XJLZoYn0bXsIJ4JsRDDa7k5Y
Content-Type: text/plain; charset=UTF-8; format=flowed
Content-Transfer-Encoding: 7bit
Hey Fabien,
Here is the recovery file for my keepass db. For good measure I've
ciphered it using open ssl with aes cbc 256 and 2048 iterations. I'll
come to your desk to tell you the password, which I'll store securely in
the db.
Cheers
Antoine
-----XJLZoYn0bXsIJ4JsRDDa7k5Y
Content-Type: application/pdf; name="savecipher.pdf"
Content-Disposition: attachment; filename="savecipher.pdf"
Content-Transfer-Encoding: base64
U2FsdGVkX1/JDDCaToEZSYAf+sRcFFcQaVGazRcrtiUf6ekqqw+YakTx7JieultS4vSf9XTQ
BRGVX6NXgYX7VZhBpg/xJNHvvdv7hR8z1i4Tjml06TGcYP7FeHKEk5/LcS43X9rF/XA/XwjD
f1E/d/V0nZBymg93rYm0NIeTIOVzuLp8nhJLmvu2A45XxdnP8UcoqKngS1yuprV05JVzmJiJ
kNWMjS9wj8lZYG7BGQxSSa13wikVCrSUD0nmR71TRSoddTkPz9dvodvc/VZsmoAUanlwj3TA
7Guxy2GBoxTAi1Y67FLeD8WwEMUkhi+KPBORCLd0Av2eTDL7zfe70detFi0iN8kxwv2kPbC
```

How wonderful! a pdf file base64 encoded that you can recover, and you are told exactly how it was ciphered!

And there even was a Re: to this email:

```
Content-Transfer-Encoding: 7bit
I forgot to mention - my softwares are up to date as of today. Unless
there's like a big cve or something we should be safe aha
On 17/05/2023 08:25, Antoine Dupont wrote:
> Hey Fabien,
> Here is the recovery file for my keepass db. For good measure I've
```

Now you have a ciphered pdf, you know how it was ciphered, and there's probably a CVE involved.

## 2.2.2 Thunderbird-If you have an idea where to look

Thunderbird uses profile folders on to store all the data for a given email profile.

[http://kb.mozillazine.org/Profile\\_folder\\_-\\_Thunderbird](http://kb.mozillazine.org/Profile_folder_-_Thunderbird)

It stores the sent emails in a data file called "Sent" and the received ones in "Inbox" in a sub-folder for the profile.

With that in mind:

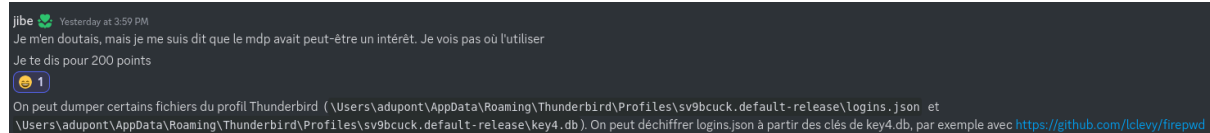
```
vol3 -f adupont.dmp windows.filescan > files.txt
cat files.txt | grep "Sent"
```

```
0xa6f04738 \Users\adupont\AppData\Roaming\Thunderbird\Profiles\sv9bcuck.default-release\Mail\pop.gmail.com\Sent 128
```

If you didn't know that, you could still funnel your way to that file anyway by looking at all the Thunderbird files from the filescan (cat files.txt | grep "Thunderbird") and figure it out.

## NB:

I know at least a team or two managed to find the password to the gmail account that was used. I thought that might happen but didn't know how it was possible, so many thanks to jibe on discord who explained it to me:



jibe Yesterday at 3:59 PM  
Je m'en doutais, mais je me suis dit que le mdp avait peut-être un intérêt. Je vois pas où l'utiliser  
Je te dis pour 200 points  
👉 1  
On peut dumper certains fichiers du profil Thunderbird (\Users\adupont\AppData\Roaming\Thunderbird\Profiles\sv9bcuck.default-release\logins.json et \Users\adupont\AppData\Roaming\Thunderbird\Profiles\sv9bcuck.default-release\key4.db). On peut déchiffrer logins.json à partir des clés de key4.db, par exemple avec <https://github.com/lclevy/firepwd>

At this point you didn't need any more info. You could just skip to the deciphering of the pdf

## 2.3 SOLVING BY CARVING OUT THE PDF

### 2.3.1 Filescan

If you started looking for files out of the ordinary in ordinary places then you may have tried to look at the files in the /Documents folder

```
vol3 -f adupont.dmp windows.filescan > files.txt
cat files.txt | grep Documents
```

```
0x90e28c68 \Users\Public\Documents\desktop.ini 128
0xa1a0a478 \Users\adupont\Documents\desktop.ini 128
0xa6f0e938 \Users\adupont\Documents 128
0xa6f0f0d0 \Users\adupont\Documents 128
0xa6f0f280 \Users\adupont\Documents\keepass 128
0xa6f0f508 \Users\adupont\Documents\keepass 128
0xa6f12370 \Users\adupont\Documents\keepass 128
0xa6f17a58 \Users\adupont\Documents\savecipher.pdf 128
0xab011950 \Users\adupont\Documents\keepass\Database.kdbx 128
```

"savecipher.pdf" ?? what could that be :')

We can carve it out from the memory

```
vol3 -f adupont.dmp windows.dumpfiles --virtaddr 0xa6f17a58
```

But we can't open it.

Taking a look at the file type you may have noticed that it was indeed ciphered (as the file name would suggest), using openssl

```
file file.0xa6f17a58.0x9d8661b8.DataSectionObject.savecipher.pdf.dat
```

```
file.0xa6f17a58.0x9d8661b8.DataSectionObject.savecipher.pdf.dat: openssl enc'd data with salted password
```

Openssl enc'd data !

Now there are still multiple ways to use openssl and to cipher the file, so we're missing some more info.

NB:I know a team tried to bruteforce every single openssl cipher type (and may have succeeded?) but the simpler way was to ask yourself **“how was it ciphered/what ciphered it”** ?

### 2.3.2 How was it ciphered ? powershell cli

If you remember at the start we saw that there was a powershell process running at the time of the dump. Maybe some commands were run ? But where to find them ? :')

A quick google search would give the answers

<https://0xdf.gitlab.io/2018/11/08/powershell-history-file.html>

**Windows stores commands in a .txt file called “ConsoleHost\_history.txt” in AppData/Roaming.**

Let's go back to our filescan:

`cat files.txt | grep “ConsoleHost_history”`

```
0x8e6a1778      \Users\adupont\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt 128
```

cat the file

```
.\tools\openssl.exe enc --aes-256-cbc --in .\save.pdf --out savecipher.pdf --iter 2048
rm .\save.pdf
rm .\savecipher.pdf
cd .\Documents\
ls
cd .\Documents\
cat .\savecipher.pdf
ls
cd .\tools\
ls
cd ..
cat .\savecipher.pdf
cd ..
cd .\adupont\Documents\
ls
cat .\savecipher.pdf
cd .\tools\
ls
```

And now we know exactly how it was ciphered!

## 2.4 Deciphering the pdf

### 2.4.1 Run the exploit

Whichever path you took, at the end of the day you needed 3 elements:

- the savecipher.pdf file
- how it was ciphered
- the key to the cipher

But we're still missing the key with which it was ciphered. Most people had already figured out at this point that [CVE-2023-32784](#) was to be used.

So let's use it:

I used: <https://github.com/dawnl3ss/CVE-2023-32784>

We need to dump the keepass process memory

```
vol3 -f adupont.dmp windows.memmap --dump --pid 2896
```

Then running the exploit gives us a list of possible passwords

```
Possible password: •ysuper*s4F3pwd78224DB
Possible password: •ysuper*s4F3pwd78224DC
Possible password: •%super*s4F3pwd78224DB
Possible password: •%super*s4F3pwd78224DC
Possible password: •'super*s4F3pwd78224DB
Possible password: •'super*s4F3pwd78224DC
```

But we're still missing the first character

### 2.4.2 Find the right password

A few intended options at this point:

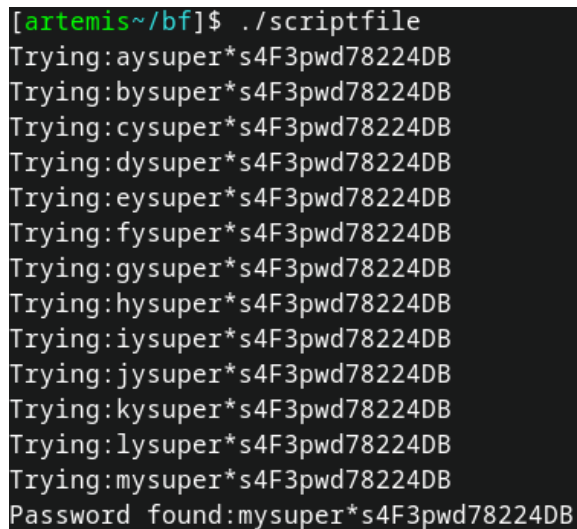
**#1:** If you tried to guess that “mysuper\*s4F3pwd...” seemed like a good candidate then congrats you've just saved yourself a lot of time

**#2:** If you're a heavy-approach kind of guy/gal, then you could bruteforce your way into finding the right password.

Here's a little shell script I wrote to do that

```
#!/bin/bash
```

```
for char in {a..z} {A..Z} {0..9}
do
    result="${char}ysuper*s4F3pwd78224DB"
    echo "Trying:$result"
    cmd=$(openssl enc -d --aes-256-cbc -in file -out plzdecipher.pdf --iter
2048 --pass pass:$result 2>&1)
    if echo "$cmd" | grep -q "bad"; then
        continue
    else
        echo "Password found:$result"
        break
    fi
done
```



```
[artemis~/bf]$ ./scriptfile
Trying:aysuper*s4F3pwd78224DB
Trying:bysuper*s4F3pwd78224DB
Trying:cysuper*s4F3pwd78224DB
Trying:dysuper*s4F3pwd78224DB
Trying:eysuper*s4F3pwd78224DB
Trying:fysuper*s4F3pwd78224DB
Trying:gysuper*s4F3pwd78224DB
Trying:hysuper*s4F3pwd78224DB
Trying:iysuper*s4F3pwd78224DB
Trying:jysuper*s4F3pwd78224DB
Trying:kysuper*s4F3pwd78224DB
Trying:lysuper*s4F3pwd78224DB
Trying:mysuper*s4F3pwd78224DB
Password found:mysuper*s4F3pwd78224DB
```

We have the password, the file is unciphered  
Open it

	<b>KeePass Emergency Sheet</b>	
17/02/2024		

**Database file:**

You should regularly create a backup of the database file (onto an independent data storage device). Backups are stored here:

**Master Key**

The master key for this database file consists of the following components:

- **Master password:**

**Instructions and General Information**

- A KeePass emergency sheet contains all important information that is required to open your database. It should be printed, filled out and stored in a secure location, where only you and possibly a few other people that you trust have access to.
- If you lose the database file or any of the master key components (or forget the composition), all data stored in the database is lost. KeePass does not have any built-in file backup functionality. There is no backdoor and no universal key that can open your database.
- The latest KeePass version can be found on the KeePass website:  
<https://keepass.info/>.

And boom ! We have the master password  
(that's a real sheet provided by keepass when creating a DB btw, I just wrote the password on it)  
All that's left is to enter the Database



```
[artemis~/keepass-chall]$ kpcli

KeePass CLI (kpcli) v4.0 is ready for operation.
Type 'help' for a description of available commands.
Type 'help <command>' for details on individual commands.

kpcli:/> open Database.kdbx
Provide the master password: *****
kpcli:/> ls
=== Groups ===
Database/
kpcli:/> cd Database/
kpcli:/Database> ls
=== Groups ===
eMail/
General/
Homebanking/
Internet/
Network/
Recycle Bin/
Windows/
=== Entries ===
0. admin pwd ssh
1. flag
2. openssl recovery pwd
kpcli:/Database> show -f flag

  Path: /Database/
Title: flag
Uname:
  Pass: THCON{p4ssw0rDS_4re_mY_favourite_things!}
  URL:
Notes:
```

Flag recovered, end of chall 😊

### 3. Conclusion

**TL;DR what to remember for future challs:**

- Actually read what a vulnerability does before spending hours trying to figure out why it's not working
- Thunderbird stores emails data in profiles under `C:\Users\<Windows user name>\AppData\Roaming\Thunderbird\Profiles\<Profile name>` and the actual data of the Sent and Received emails are in cleartext "Sent" and "Inbox" data files
- Windows stores powershell info in `$env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt`
- You can recover a mail password from Thunderbird
- Finding a needle in a haystack is though, luckily there were two needles in this challenge :')