Lab 4 Report

Jason Tolbert

The Pennsylvania State University

IST 894-001: Capstone Experience

Dr. Michael Bartolacci, Instructor

February 23rd, 2025

Table of Contents

General Overview

This lab introduces participants to 1) reconnaissance, and 2) scripting in the context of penetration testing.

The first half of the lab is a video introduction to reconnaissance. In the context of cybersecurity, reconnaissance is the process of gathering information about a target system — like the software and services it runs — in order to identify potential vulnerabilities that can be used to launch a cyberattack (Roy et al., 2022). Reconnaissance is a necessary first step in most kinds of cyberattacks.

The reconnaissance videos also discuss footprinting. Footprinting a more structured and targeted form of reconnaissance that specifically aims to create an identifying profile of the target system (Sh et al., 2020). You have almost certainly been subject to footprinting yourself before, even if you don't know it — many websites, for example, collect information such as your browser version, operating system, screen resolution, time zone, and more to create a unique profile they can use to track you even if you aren't signed into an account or are using private browsing mode (Kaur et al., 2017).

Participants are shown several methods of conducting reconnaissance and footprinting. Some, such as analyzing domain registration information, examining historical versions of websites via the Internet Archive's Wayback Machine "Google hacking"[1], are easily accessible to anyone with a web browser. Others employ specialized, automated, tools, that can only be used from the command line.

The second half of the lab is a hands-on cyber range that has participants explore the role of scripting in penetration testing. Participants use Python and Bash — two of the most common languages used in penetration testing automation (Seidl & Chapple, 2022) to write basic scripts that identify open ports on a target system and determine whether a given target system is online.

[1] The practice of using advanced Google Search operators to discover sensitive or hidden information on websites that Google Search has indexed. The English Wikipedia article on Google Search contains a non-exhaustive list of such operators.

Technical Overview

This lab introduces participants to the concepts and applications of reconnaissance and footprinting. It also covers applications of Python and Bash scripting in penetration testing.

The first section of the lab, which covers reconnaissance and footprinting, teaches participants several methods of gathering intelligence on a target system, focused primarily on servers accessible over the web. Participants are taught intelligence gathering that directly interacts with the target server (e.g., downloading local copies of a website with wget for later analysis), as well as ones do not interact with the target server at all (e.g., WHOIS lookups, DNS queries, analysis of Wayback Machine-archived historical version of a website).

Participants are also introduced to specialized tools specifically designed to conduct reconnaissance. The lab primarily focuses on OSINT platform Matelgo and its ability to reveal connections between software, files, organizations, and individuals. Additionally covered are the Recon-ng reconnaissance framework and the Discover collection of penetration testing scripts.

The second phase of the lab has participants write and execute Python and Bash scripts to streamline penetration testing activities. Participants use Python's *socket* networking interface to build a port scanner, then write a basic Bash script that informs the user of whether a given host is reachable.

References

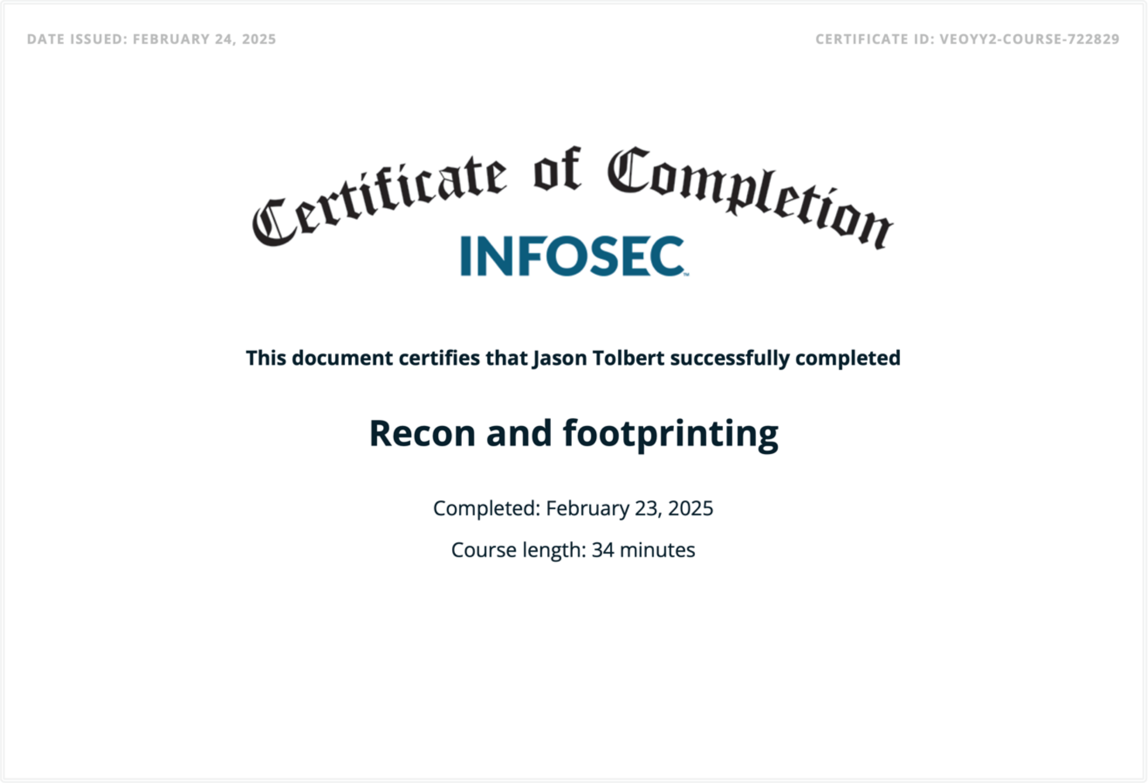
Kaur, N., Azam, S., Kannoorpatti, K., Yeo, K. C., & Shanmugam, B. (2017). Browser fingerprinting as user tracking technology. *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, 103–111. https://doi.org/10.1109/ISCO.2017.7855963

Roy, S., Sharmin, N., Acosta, J. C., Kiekintveld, C., & Laszka, A. (2022). *Survey and taxonomy of adversarial reconnaissance techniques* (No. arXiv:2105.04749). arXiv. https://doi.org/10.48550/arXiv.2105.04749

Seidl, D., & Chapple, M. (2022). Scripting for penetration testing. In *CompTIA PenTest+ Study Guide: Exam PT0-002* (pp. 429–484). CompTIA PenTest+ Study Guide: Exam PT0-002. Wiley. https://ieeexplore.ieee.org/document/9953462

Sh, S., Kumar, N. S., Rao, K., & Rao, B. (2020). Footprinting: Techniques, tools and countermeasures for footprinting. *Journal of Critical Reviews*, *7*, 2019–2025. https://doi.org/10.31838/jcr.07.11.311

Screenshots



DATE ISSUED: FEBRUARY 24, 2025

CERTIFICATE ID: VEOYY2-COURSE-722829

Certificate of Completion

INFOSEC

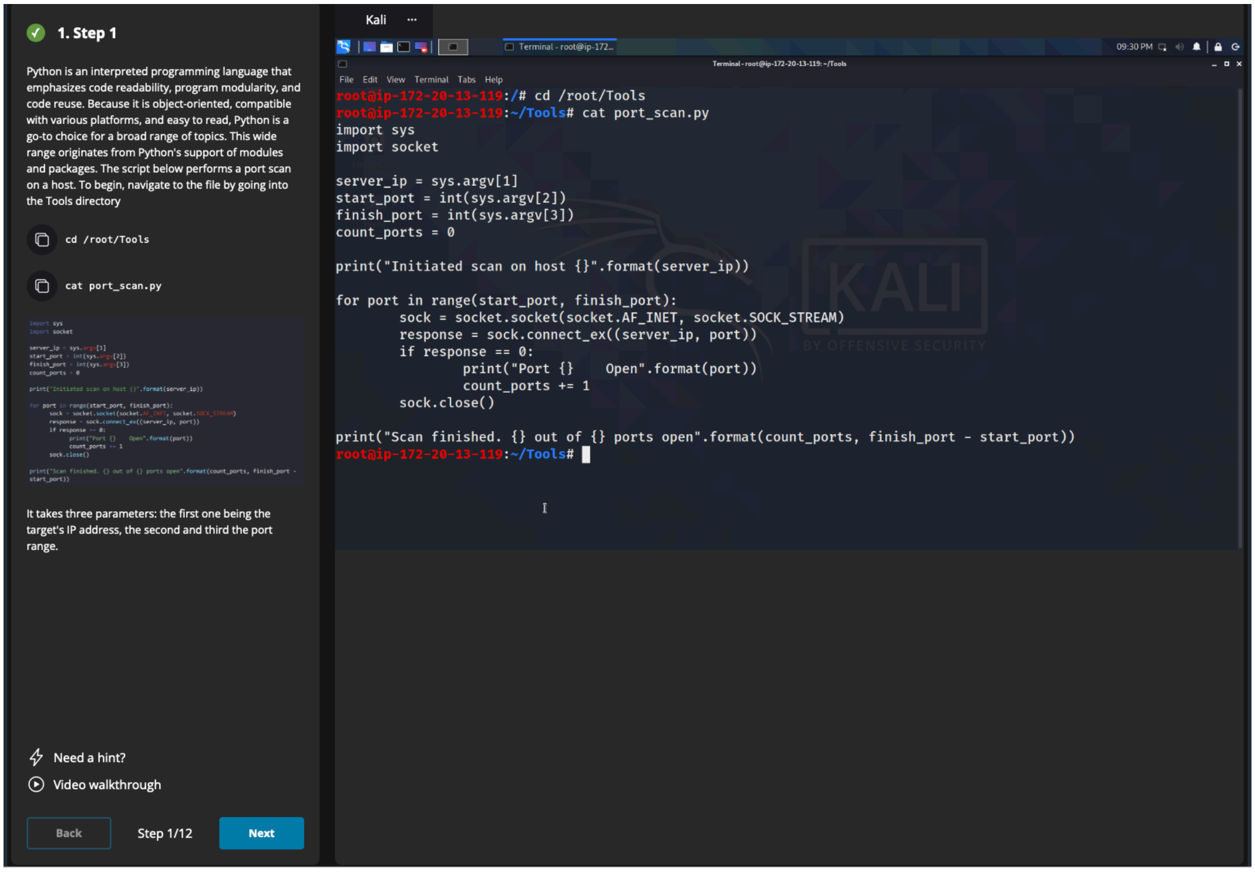
This document certifies that Jason Tolbert successfully completed

Recon and footprinting

Completed: February 23, 2025

Course length: 34 minutes

Figure 1. Certificate of completion for the "Recon and footprinting" course.



1. Step 1

Python is an interpreted programming language that emphasizes code readability, program modularity, and code reuse. Because it is object-oriented, compatible with various platforms, and easy to read, Python is a go-to choice for a broad range of topics. This wide range originates from Python's support of modules and packages. The script below performs a port scan on a host. To begin, navigate to the file by going into the Tools directory

```
cd /root/Tools
```

```
cat port_scan.py
```

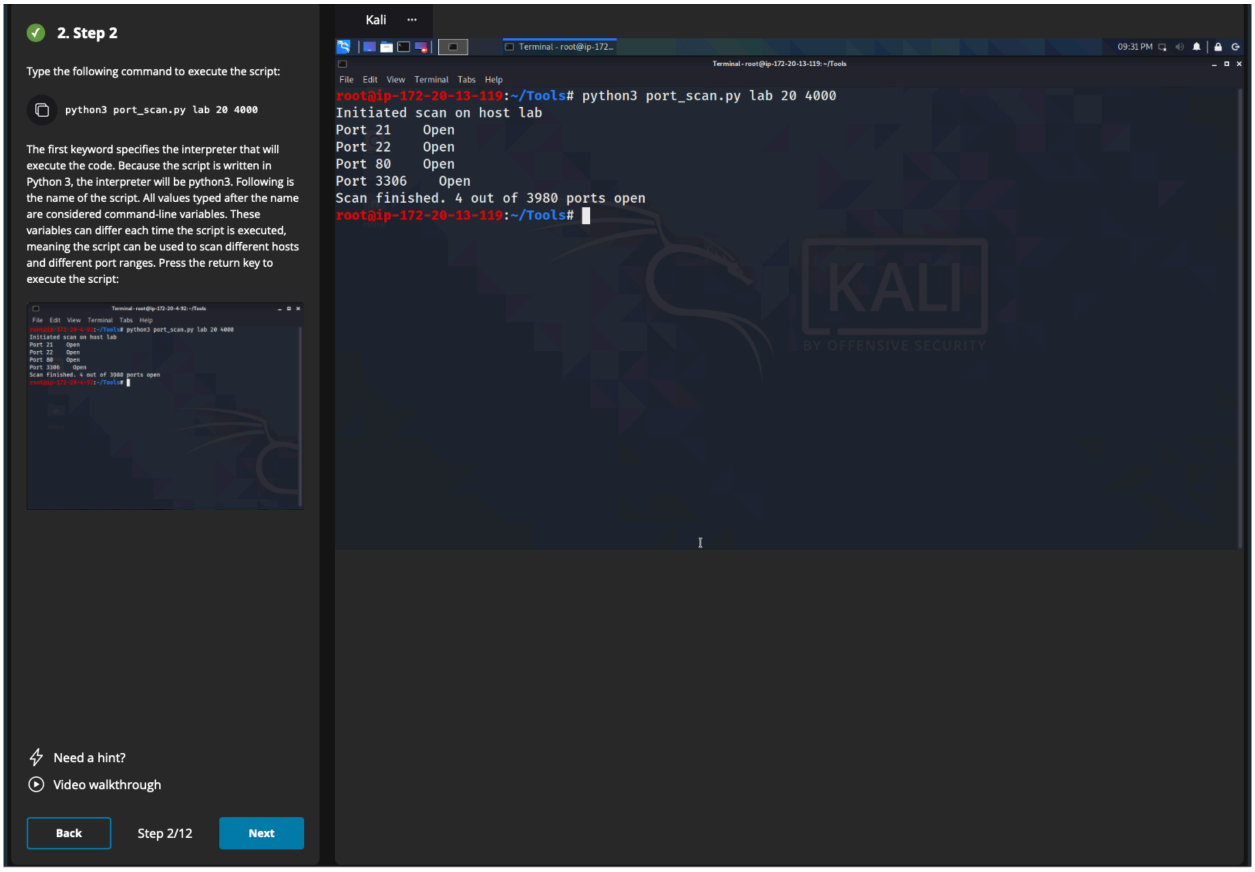
```
import sys
import socket

server_ip = sys.argv[1]
start_port = int(sys.argv[2])
finish_port = int(sys.argv[3])
count_ports = 0

print("Initiated scan on host {}".format(server_ip))

for port in range(start_port, finish_port):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        response = sock.connect_ex((server_ip, port))
        if response == 0:
                print("Port {}    Open".format(port))
                count_ports += 1
        sock.close()

print("Scan finished. {} out of {} ports open".format(count_ports, finish_port -
start_port))
```

It takes three parameters: the first one being the target's IP address, the second and third the port range.

Need a hint?
Video walkthrough
Back
Step 1/12
Next
Terminal - root@ip-172...
09:30 PM
Terminal - root@ip-172-20-13-119: ~/Tools
File Edit View Terminal Tabs Help

```
root@ip-172-20-13-119:/# cd /root/Tools
root@ip-172-20-13-119:~/Tools# cat port_scan.py
import sys
import socket

server_ip = sys.argv[1]
start_port = int(sys.argv[2])
finish_port = int(sys.argv[3])
count_ports = 0

print("Initiated scan on host {}".format(server_ip))

for port in range(start_port, finish_port):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        response = sock.connect_ex((server_ip, port))
        if response == 0:
                print("Port {}    Open".format(port))
                count_ports += 1
        sock.close()

print("Scan finished. {} out of {} ports open".format(count_ports, finish_port - start_port))
root@ip-172-20-13-119:~/Tools# 
```

Figure 2. Reading port_scan.py.



2. Step 2

Type the following command to execute the script:

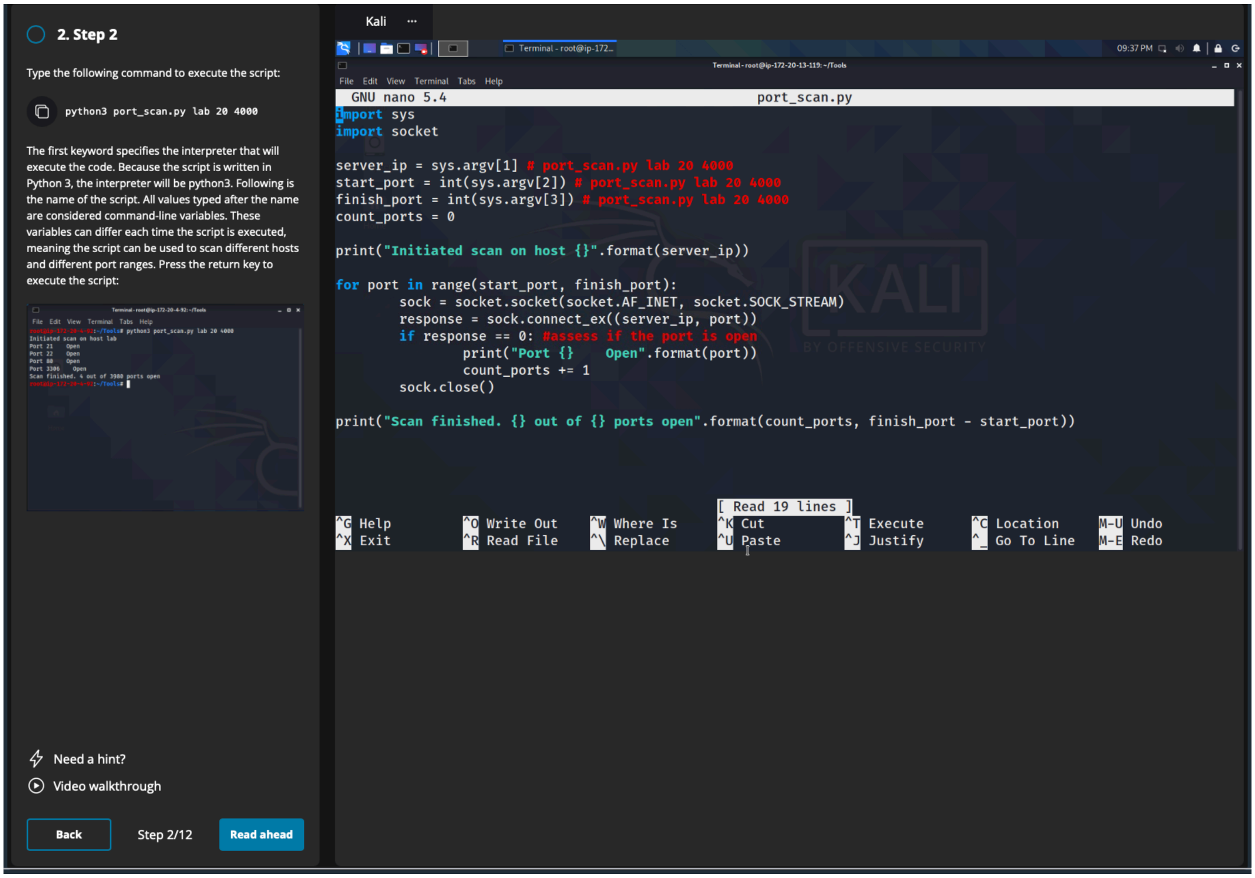
```
python3 port_scan.py lab 20 4000
```

The first keyword specifies the interpreter that will execute the code. Because the script is written in Python 3, the interpreter will be python3. Following is the name of the script. All values typed after the name are considered command-line variables. These variables can differ each time the script is executed, meaning the script can be used to scan different hosts and different port ranges. Press the return key to execute the script:

```
root@ip-172-20-13-119:~/Tools# python3 port_scan.py lab 20 4000
Initiated scan on host lab
Port 21    Open
Port 22    Open
Port 80    Open
Port 3306    Open
Scan finished. 4 out of 3980 ports open
root@ip-172-20-13-119:~/Tools# 
```

Terminal - root@ip-172...
09:31 PM
Terminal - root@ip-172-20-13-119: ~/Tools
File Edit View Terminal Tabs Help
KALI
BY OFFENSIVE SECURITY
Need a hint?
Video walkthrough
Back
Step 2/12
Next

Figure 3. Executing port_scan.py with the parameters "lab", 20, and 4000.



Kali
Terminal - root@ip-172...
09:37 PM
Terminal - root@ip-172-20-13-119: ~/Tools
File Edit View Terminal Tabs Help

2. Step 2

Type the following command to execute the script:

```
python3 port_scan.py lab 20 4000
```

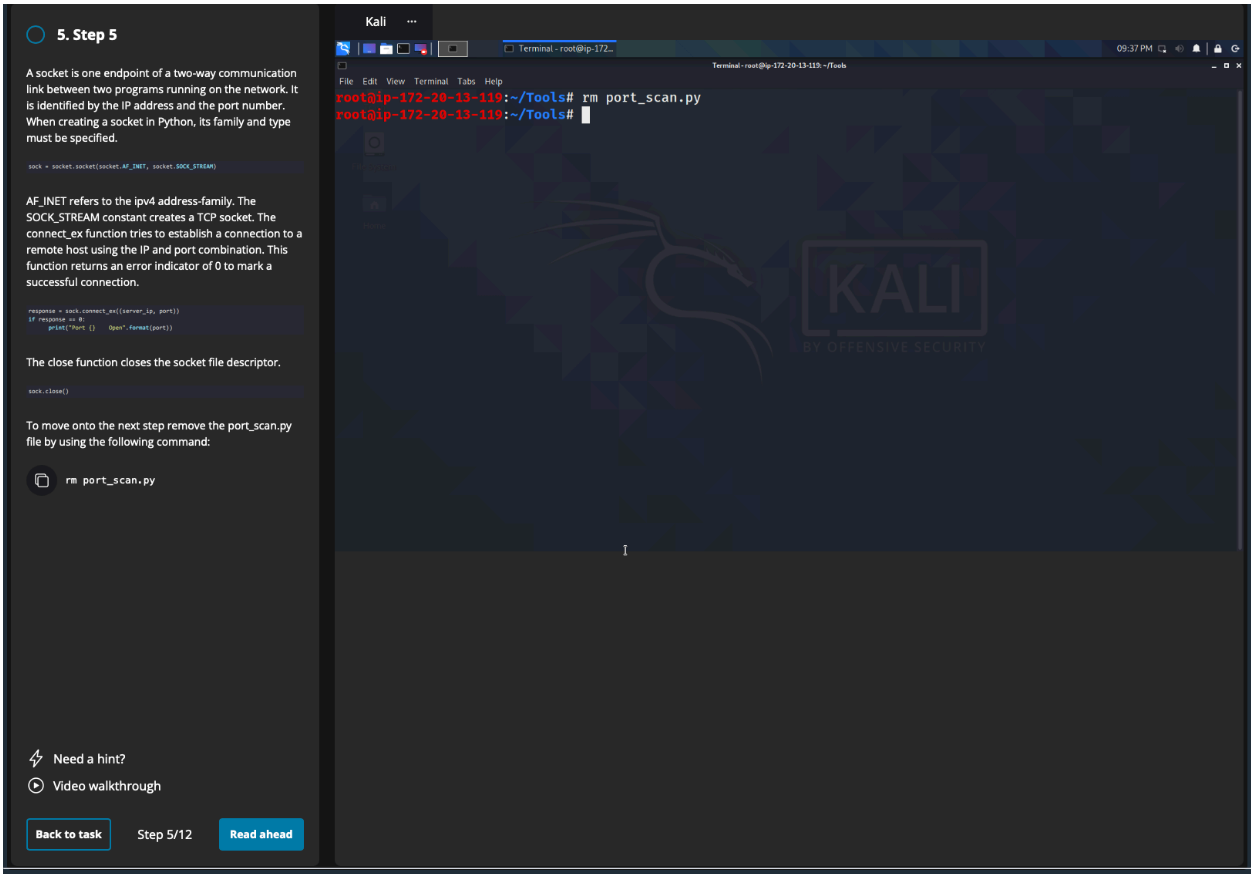
The first keyword specifies the interpreter that will execute the code. Because the script is written in Python 3, the interpreter will be python3. Following is the name of the script. All values typed after the name are considered command-line variables. These variables can differ each time the script is executed, meaning the script can be used to scan different hosts and different port ranges. Press the return key to execute the script:

Terminal - root@ip-172-20-4-92: ~/Tools
File Edit View Terminal Tabs Help

```
root@ip-172-20-4-92:~/Tools# python3 port_scan.py lab 20 4000
Initiated scan on host lab
Port 21    Open
Port 22    Open
Port 80    Open
Port 3306    Open
Scan finished. 4 out of 3980 ports open
root@ip-172-20-4-92:~/Tools# 
```

```
  GNU nano 5.4                                          port_scan.py
import sys
import socket

server_ip = sys.argv[1] # port_scan.py lab 20 4000
start_port = int(sys.argv[2]) # port_scan.py lab 20 4000
finish_port = int(sys.argv[3]) # port_scan.py lab 20 4000
count_ports = 0

print("Initiated scan on host {}".format(server_ip))

for port in range(start_port, finish_port):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        response = sock.connect_ex((server_ip, port))
        if response == 0: #assess if the port is open
                print("Port {}    Open".format(port))
                count_ports += 1
        sock.close()

print("Scan finished. {} out of {} ports open".format(count_ports, finish_port - start_port))



                                                [ Read 19 lines ]
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^_ Go To Line    M-E Redo
```

Figure 4. Adding comments to port_scan.py.



5. Step 5

A socket is one endpoint of a two-way communication link between two programs running on the network. It is identified by the IP address and the port number. When creating a socket in Python, its family and type must be specified.

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

AF_INET refers to the ipv4 address-family. The SOCK_STREAM constant creates a TCP socket. The connect_ex function tries to establish a connection to a remote host using the IP and port combination. This function returns an error indicator of 0 to mark a successful connection.

```
response = sock.connect_ex((server_ip, port))
if response == 0:
      print("Port {}    Open".format(port))
```

The close function closes the socket file descriptor.

```
sock.close()
```

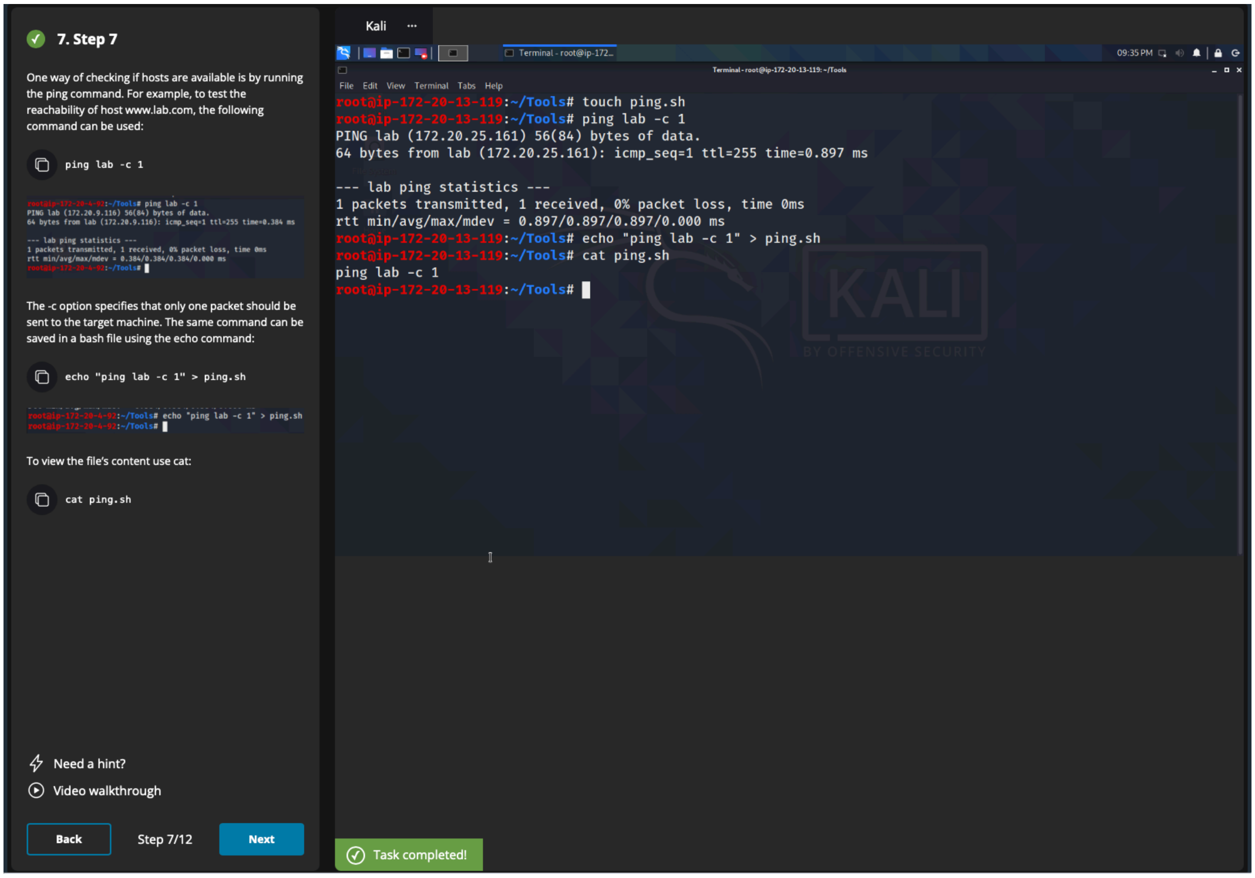
To move onto the next step remove the port_scan.py file by using the following command:

```
rm port_scan.py
```

```
root@ip-172-20-13-119:~/Tools# rm port_scan.py
root@ip-172-20-13-119:~/Tools# 
```

Kali
Terminal - root@ip-172...
09:37 PM
Terminal - root@ip-172-20-13-119: ~/Tools
File Edit View Terminal Tabs Help
KALI
BY OFFENSIVE SECURITY
Need a hint?
Video walkthrough
Back to task
Step 5/12
Read ahead

Figure 5. Removing port_scan.py.



7. Step 7

One way of checking if hosts are available is by running the ping command. For example, to test the reachability of host www.lab.com, the following command can be used:

```
ping lab -c 1
```

```
root@ip-172-20-4-92:~/Tools# ping lab -c 1
PING lab (172.20.9.116) 56(84) bytes of data.
64 bytes from lab (172.20.9.116): icmp_seq=1 ttl=255 time=0.384 ms

--- lab ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.384/0.384/0.384/0.000 ms
root@ip-172-20-4-92:~/Tools#
```

The -c option specifies that only one packet should be sent to the target machine. The same command can be saved in a bash file using the echo command:

```
echo "ping lab -c 1" > ping.sh
```

```
root@ip-172-20-4-92:~/Tools# echo "ping lab -c 1" > ping.sh
root@ip-172-20-4-92:~/Tools#
```

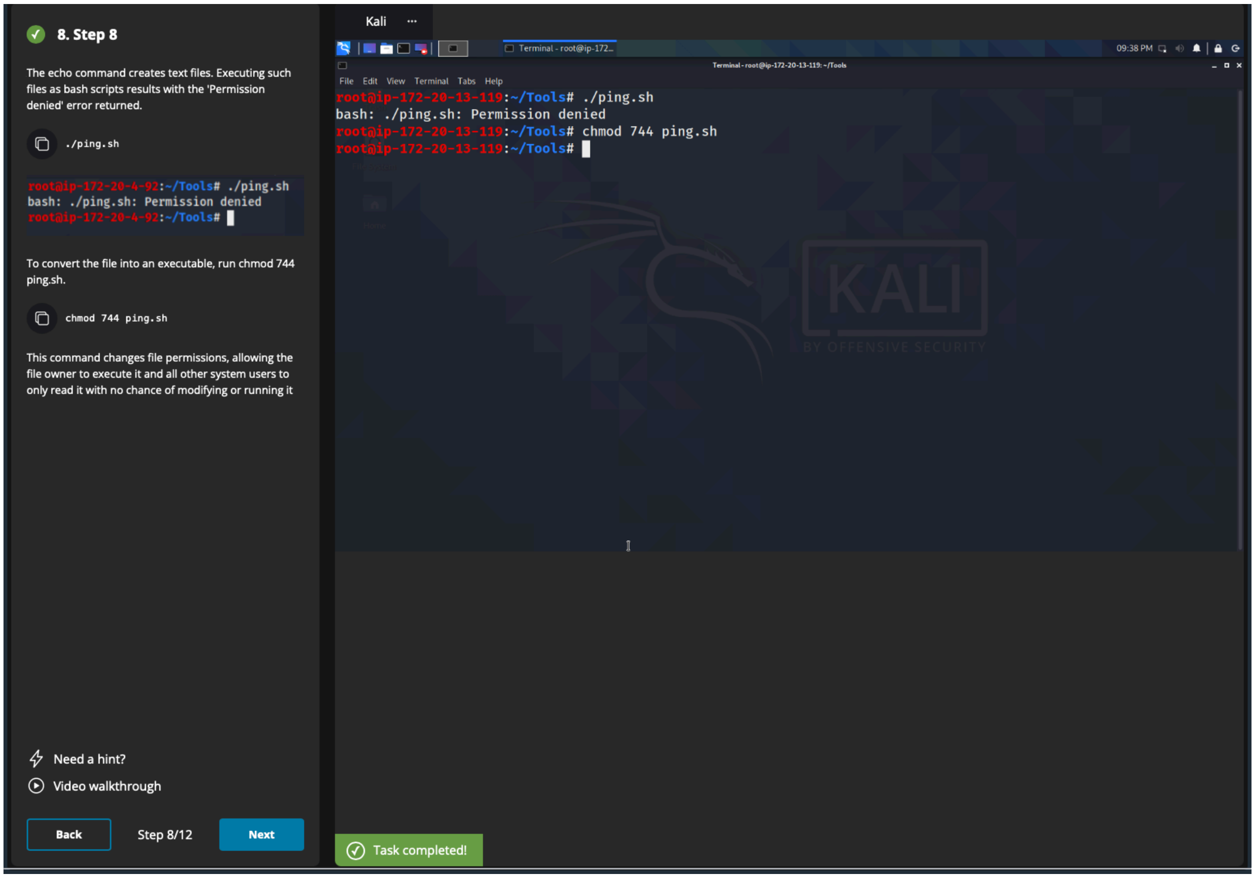
To view the file's content use cat:

```
cat ping.sh
```

```
root@ip-172-20-13-119:~/Tools# touch ping.sh
root@ip-172-20-13-119:~/Tools# ping lab -c 1
PING lab (172.20.25.161) 56(84) bytes of data.
64 bytes from lab (172.20.25.161): icmp_seq=1 ttl=255 time=0.897 ms

--- lab ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.897/0.897/0.897/0.000 ms
root@ip-172-20-13-119:~/Tools# echo "ping lab -c 1" > ping.sh
root@ip-172-20-13-119:~/Tools# cat ping.sh
ping lab -c 1
root@ip-172-20-13-119:~/Tools#
```

Kali
Terminal - root@ip-172...
09:35 PM
Terminal - root@ip-172-20-13-119: ~/Tools
File Edit View Terminal Tabs Help
Need a hint?
Video walkthrough
Back
Step 7/12
Next
Task completed!

Figure 6. Echoing text to ping.sh and reading the modified file.



8. Step 8

The echo command creates text files. Executing such files as bash scripts results with the 'Permission denied' error returned.

```
./ping.sh
```

```
root@ip-172-20-4-92:~/Tools# ./ping.sh
bash: ./ping.sh: Permission denied
root@ip-172-20-4-92:~/Tools#
```

To convert the file into an executable, run chmod 744 ping.sh.

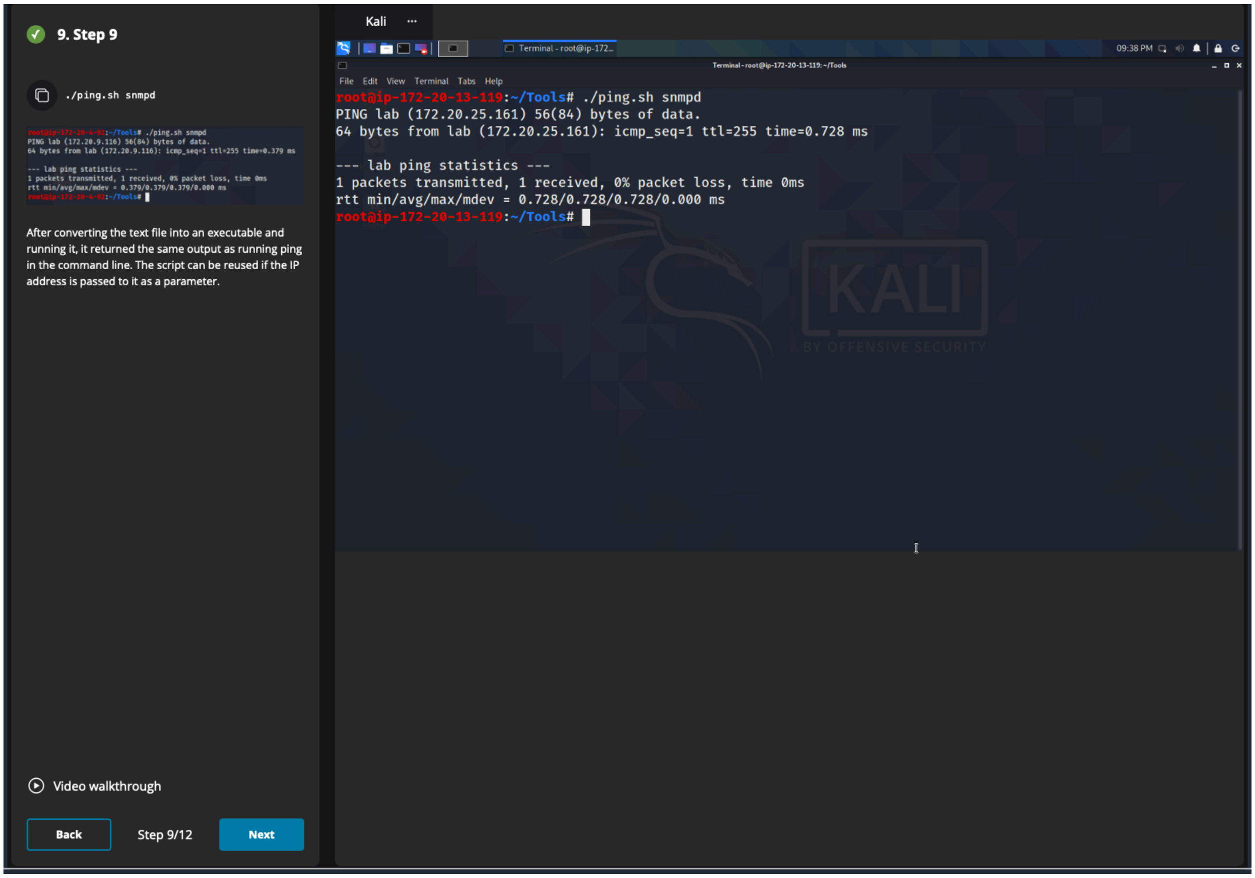
```
chmod 744 ping.sh
```

This command changes file permissions, allowing the file owner to execute it and all other system users to only read it with no chance of modifying or running it

```
root@ip-172-20-13-119:~/Tools# ./ping.sh
bash: ./ping.sh: Permission denied
root@ip-172-20-13-119:~/Tools# chmod 744 ping.sh
root@ip-172-20-13-119:~/Tools#
```

Kali
Terminal - root@ip-172...
09:38 PM
Terminal - root@ip-172-20-13-119: ~/Tools
File Edit View Terminal Tabs Help
Need a hint?
Video walkthrough
Back
Step 8/12
Next
Task completed!

Figure 7. Making ping.sh executable.



9. Step 9

```
./ping.sh snmpd
```

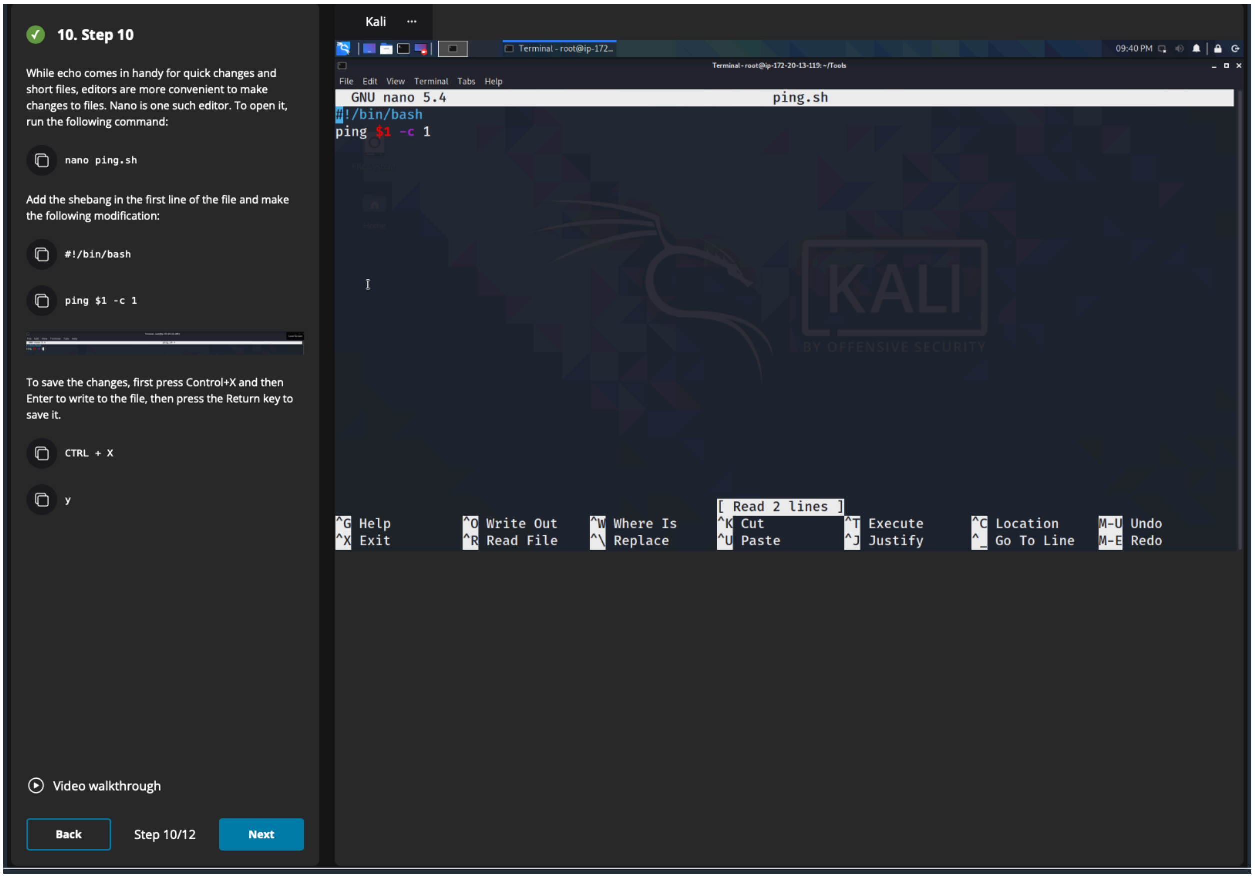
```
root@ip-172-20-4-92:~/Tools# ./ping.sh snmpd
PING lab (172.20.9.116) 56(84) bytes of data.
64 bytes from lab (172.20.9.116): icmp_seq=1 ttl=255 time=0.379 ms

--- lab ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.379/0.379/0.379/0.000 ms
root@ip-172-20-4-92:~/Tools# 
```

After converting the text file into an executable and running it, it returned the same output as running ping in the command line. The script can be reused if the IP address is passed to it as a parameter.

Video walkthrough
Back
Step 9/12
Next
Kali
Terminal - root@ip-172...
09:38 PM
Terminal - root@ip-172-20-13-119: ~/Tools
File Edit View Terminal Tabs Help

```
root@ip-172-20-13-119:~/Tools# ./ping.sh snmpd
PING lab (172.20.25.161) 56(84) bytes of data.
64 bytes from lab (172.20.25.161): icmp_seq=1 ttl=255 time=0.728 ms

--- lab ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.728/0.728/0.728/0.000 ms
root@ip-172-20-13-119:~/Tools# 
```

Figure 8. Executing ping.sh with the argument "snmpd". It has no effect since the script is hardcoded to ping the host “lab”.



10. Step 10

While echo comes in handy for quick changes and short files, editors are more convenient to make changes to files. Nano is one such editor. To open it, run the following command:

```
nano ping.sh
```

Add the shebang in the first line of the file and make the following modification:

```
#!/bin/bash
```

```
ping $1 -c 1
```

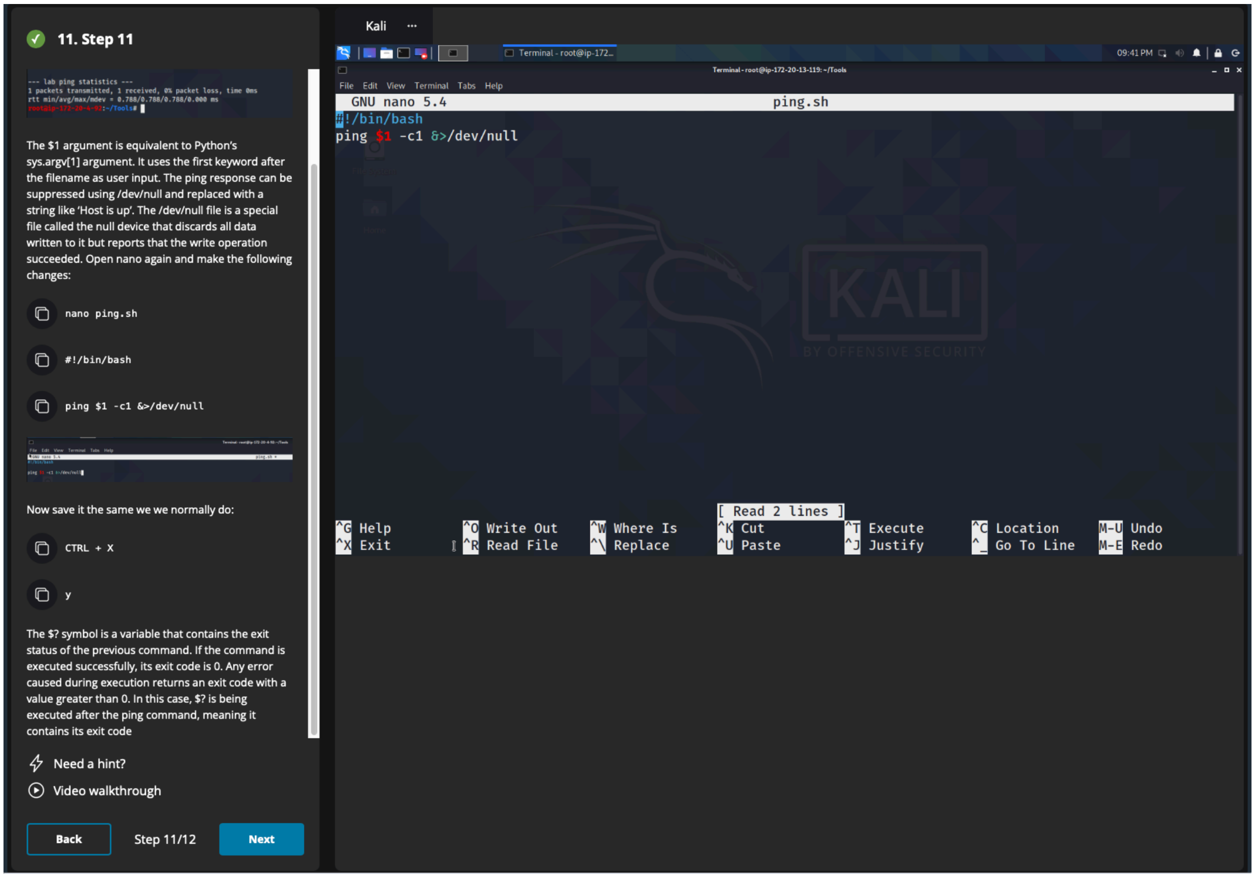
To save the changes, first press Control+X and then Enter to write to the file, then press the Return key to save it.

```
CTRL + X
```

```
y
```

```
  GNU nano 5.4                                ping.sh
#!/bin/bash
ping $1 -c 1





                              [ Read 2 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo
```

Figure 9. Modifying ping.sh to accept an arbitrary argument for the host.



11. Step 11

```
--- lab ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.788/0.788/0.788/0.000 ms
root@ip-172-20-4-92:~/Tools#
```

The $1 argument is equivalent to Python's sys.argv[1] argument. It uses the first keyword after the filename as user input. The ping response can be suppressed using /dev/null and replaced with a string like 'Host is up'. The /dev/null file is a special file called the null device that discards all data written to it but reports that the write operation succeeded. Open nano again and make the following changes:

```
nano ping.sh
```

```
#!/bin/bash
```

```
ping $1 -c1 &>/dev/null
```

Now save it the same we we normally do:

```
CTRL + X
```

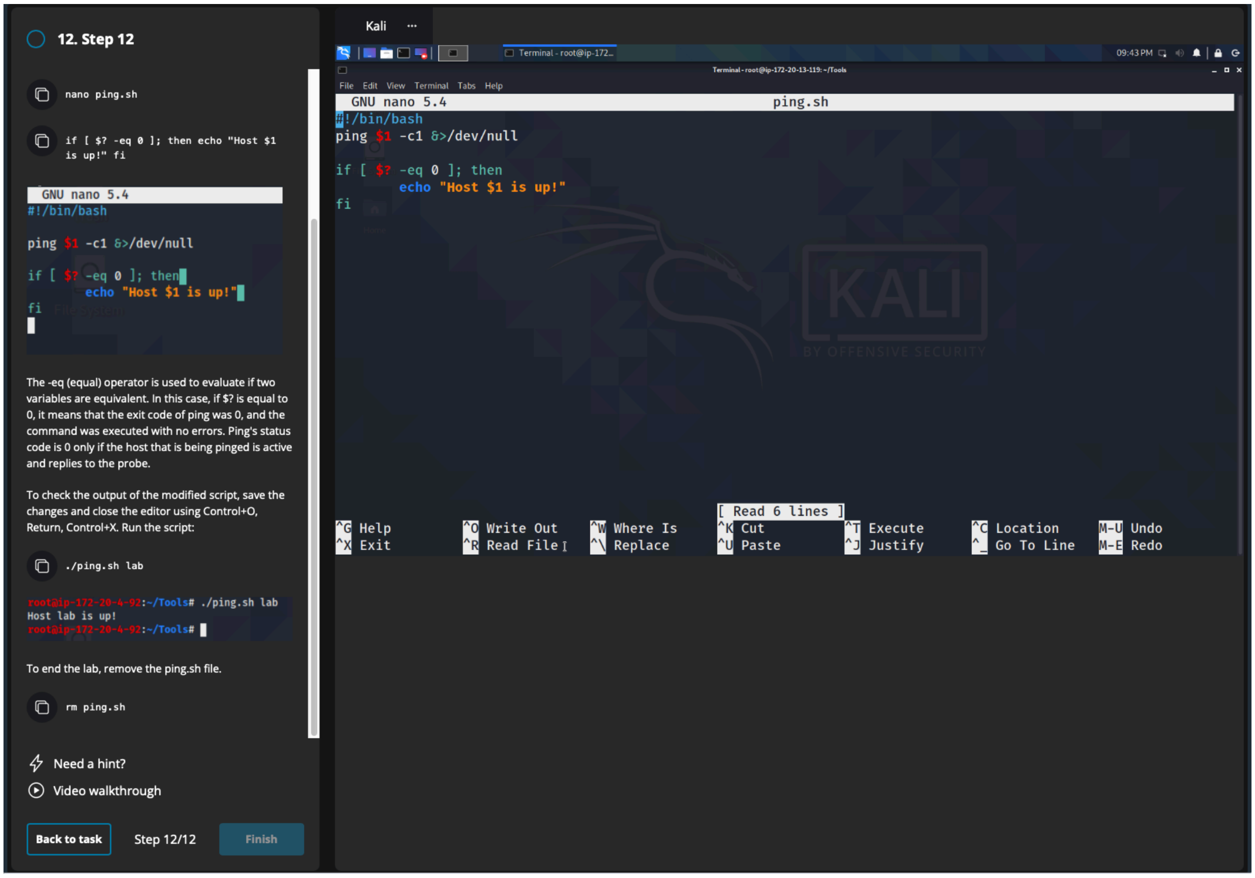
```
y
```

The $? symbol is a variable that contains the exit status of the previous command. If the command is executed successfully, its exit code is 0. Any error caused during execution returns an exit code with a value greater than 0. In this case, $? is being executed after the ping command, meaning it contains its exit code

Need a hint?
Video walkthrough
Back
Step 11/12
Next
Terminal - root@ip-172-20-13-119: ~/Tools
File Edit View Terminal Tabs Help

```
  GNU nano 5.4                                        ping.sh
#!/bin/bash
ping $1 -c1 &>/dev/null




                                        [ Read 2 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo
```

Figure 10. Modifying ping.sh to suppress its output.



12. Step 12

```
nano ping.sh
```

```
if [ $? -eq 0 ]; then echo "Host $1 is up!" fi
```

```
 GNU nano 5.4
#!/bin/bash

ping $1 -c1 &>/dev/null

if [ $? -eq 0 ]; then
        echo "Host $1 is up!"
fi
```

The -eq (equal) operator is used to evaluate if two variables are equivalent. In this case, if $? is equal to 0, it means that the exit code of ping was 0, and the command was executed with no errors. Ping's status code is 0 only if the host that is being pinged is active and replies to the probe.

To check the output of the modified script, save the changes and close the editor using Control+O, Return, Control+X. Run the script:

```
./ping.sh lab
```

```
root@ip-172-20-4-92:~/Tools# ./ping.sh lab
Host lab is up!
root@ip-172-20-4-92:~/Tools#
```

To end the lab, remove the ping.sh file.

```
rm ping.sh
```

Need a hint?
Video walkthrough
Back to task
Step 12/12
Finish
Kali
Terminal - root@ip-172-20-13-119: ~/Tools
09:43 PM
File Edit View Terminal Tabs Help

```
 GNU nano 5.4                                         ping.sh
#!/bin/bash
ping $1 -c1 &>/dev/null

if [ $? -eq 0 ]; then
        echo "Host $1 is up!"
fi





                                                [ Read 6 lines ]
^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^_ Go To Line   M-E Redo
```

Figure 11. Editing ping.sh to output "Host X is up!" if X is reachable, where X is a host supplied via argument.