

# Commands

Monday, October 16, 2023 5:42 PM

## ping

ping IP

ping google.com

ping -t google.com

mimic linux; doesn't stop until you cancel

ping -c google.com

linux version to mimic windows 4 ICMP echo requests

## hping

hping google.com

hping -S -p 80 google.com

-S = use TCP SYN packet

-p = port

-A = TCP ACK packets to check if ports are listening

-F = TCP FIN packets to check for open ports (stealth)

-U = UDP

## ipconfig

ipconfig

shows summary info of TCP/IP config for NIC

ipconfig /all

detailed version of default

ipconfig /displaydns

shows DNS cache and hostname to IP

ipconfig /flushdns

clear DNS cache

## ifconfig

ifconfig -a

windows /all

ifconfig eth0

shows individual NIC

ifconfig wlan0

shows wireless NIC

ifconfig eth0 promisc

enable promiscuous mode

ifconfig eth0 -promisc

remove promiscuous mode

ifconfig eth0 allmulti

view all multigroup traffic it processes

ifconfig eth0 -allmulti

remove allmulti

lo = loopback

if only loopback exists then there is an issue with other NIC

inet = ipv4

inet6 = ipv6

ether = MAC address

## ip

- ip link show
  - shows NIC and some details
- ip link set eth0 up
  - enables NIC
- ip -s link
  - shows stats of NIC
- ip link set eth0 promisc on
- ip link set eth0 promisc off
- sudo ip route add 192.168.2.0/24 via 192.168.1.1
  - Add route

## netstat

- netstat
  - all open TCP connections
- netstat -a
  - all open TCP/UDP connections
- netstat -r
  - routing table
- netstat -e
  - net stats like how many bytes received and sent
- netstat -s
  - stats for each protocol (ipv4, ipv6, TCP, ...)
- netstat -n
  - all addresses and ports in numerical order
  - can combine with -a to show all ports
  - netstat -a -n
- netstat -p
  - specify protocol
  - netstat -p tcp
  - netstat -p udp

## tracert

- increments TTL value assigned to ICMP packets
- tracert google.com
- tracert -d google.com
  - does not resolve ips to hostnames

## tracert

- linux version
- tracert google.com
- tracert -n google.com
  - same as -d for windows

## pathping

- combines ping and tracert
- finds all hops between systems and pings each one
- pathping google.com
- pathping -n google.com
  - doesn't resolve ips to hostnames

## arp

arp  
    windows = show help  
    linux = show arp cache  
arp -a  
    windows = show arp cache  
arp -a 10.0.0.1  
    shows ARP cache entry for the specified IP

head/tail  
    tail -n 20 casino.py  
    head -n 20 casino.py

grep "hello" casino.py

cat casino.py  
    cat casino.py | more  
    looks 1 page at a time

logger "Backup started"  
    adds to syslog file  
    /var/log/syslog

journalctl  
    journalctl  
        shows all logs  
    journalctl --since "1 hour ago"  
        shows log from last hour  
    journalctl --list-boots  
        shows boot logs  
    journalctl -1  
        shows previous log  
    -0 shows current  
    -2 shows 2 previous

chmod  
    r = 4  
    w = 2  
    x = 1  
    all = 7  
    none = 0

chmod 466 test.txt  
    user = read  
    group = read, write  
    other = read, write

chmod g-w test.txt  
    remove write from group  
chmod u=r test.txt  
    user only has read  
chmod o+x test.txt  
    add execute to other  
chmod u=rwx, g+r, o-rw test.txt

- user = r, w, x
- add read to group
- remove rw from other

## faillog

- faillog
  - show all failed login attempts
- faillog -u "username"
  - show logs for user

## nslookup

- nslookup
  - start nslookup to then run options
- nslookup -querytype=mx gcgapremium.com
  - manually do query type
- nslookup
  - server=
    - set server
    - similar to @ in dig
  - set type=MX
    - set query type
  - ls -d gcgapremium.com
    - do zone transfer

## dig

- dig google.com
  - show general lookup and related IP of google.com
- dig -t <> google.com
  - specify the lookup type (record type)
- dig -t A google.com
  - A record Ipv4
- dig -t AAAA google.com
  - AAAA record IPv6
- dig -t MX google.com
  - mx record
- dig -t NS google.com
  - name server
- dig -t CNAME google.com
- dig @8.8.8.8 google.com
  - specify the server you want to lookup
- dig +short google.com
  - shorten output
- dig +trace google.com
  - do a trace through the root DNS to the authoritative servers

## route

- route print -4
  - ipv4
- route print -6
  - ipv6
- route add <dest> mask <netmask> <gateway>

Linux route (deprecated)

Sudo route add -net <IP> netmask <mask> gw <gateway>

## Nmap

nmap google.com  
default scan with dns resolution

nmap -sn google.com  
don't do port scan, only do ping scan

nmap -p 443 172.217.165.14  
network scan for specified port to see if they are open

variations:

- nmap -p 1-100 172.217.165.14
- nmap -p 443,80 172.217.165.14
- nmap -p 80,443-445 172.217.165.14

nmap -sv 172.217.165.14  
service and version detection

nmap -O 172.217.165.14  
os detection

nmap -F 172.217.165.14  
fast scan, 100 ports

nmap -A 172.217.165.14  
aggressive scan  
service and version, script scanning, and OS detection

nmap --script vuln 172.217.165.14  
run a predefined nmap script on specified IP  
ex: a script that checks for common vulns

nmap -sS 172.217.165.14  
stealthy SYN scan  
never completes handshake

nmap -sF 172.217.165.14  
sends tcp FIN packet to see if ports are open  
doesn't establish connection like SYN

nmap -sA 172.217.165.14  
sends tcp ACK packet  
determine whether firewall is stateful or not to id if port is filtered by a firewall

nmap -sT 172.217.165.14  
does the full TCP handshake connect, "full open scan"

nmap -sU 172.217.165.14  
does a UDP scan; connectionless  
if UDP port is open the system responds with ICMP port unreachable  
if UDP port is closed responds with ICMP destination unreachable  
if UDP filtered by firewall, may be no response

## ssh

ssh username@hostname.com  
connect with username and password

ssh -i /path/to/private-key username@hostname.com  
connect with ssh key authentication

ssh -p 2222 username@hostname.com  
specify port number, default is 22

ssh username@hostname.com "ls -l"  
execute remote commands

ssh -J maggie@jump maggie@ca1  
connects to jump server then tcp forwards to ca1 server

keygen  
ssh-keygen -t rsa  
ssh-copy-id -i ~/.ssh/id\_rsa.pub ethan@google.com

dd

dd if=input\_file of=output\_file  
copy data from one to another  
dd if=/dev/sdX of=image\_file  
create disk image  
dd if=/dev/sdX of=/dev/sdY bs=4M  
clone one drive to another  
bs=4M is block size of 4 megabytes

memdump

memdump -b <buffer-size>  
memdump -k  
dump kernel memory instead of physical memory  
memdump -m <map\_file>  
print memory map  
memdump -p <memory\_page\_size>  
use system page size  
memdump -s <memory\_dump-size>  
dump all memory  
memdump -v  
verbose

Tcpdump

tcpdump  
start packet capture  
tcpdump -i eth0  
start packet capture on certain interface  
tcpdump -i eth0 tcp  
capture specific protocol  
tcp, udp, icmp  
tcpdump -i eth0 port 443  
capture specific port number  
tcpdump -i eth0 src 10.0.0.1  
capture packets from specific source IP address  
tcpdump -i eth0 dst 10.0.0.1  
capture packets with specific destination ip address  
tcpdump -i eth0 host 10.0.0.1  
capture packets either from or destined to host  
tcpdump -i eth0 -w capture.pcap  
write results to file  
tcpdump -r capture.pcap  
read results from file  
tcpdump -A  
display packet capture details in human readable format  
tcpdump -i eth0 -c 5  
capture specific # of packets  
tcpdump -v -w capture.pcap -i eth0 host 10.0.0.1  
-v = verbose  
-w = write to file

-I = interface  
host = to and from IP

## tcpreplay

tcpreplay -i eth0 -t capture.pcap  
replay a capture file

tcpreplay -I eth0 -t capture.pcap -l 2  
control the replay speed  
in this example it will be 2x speed

tcpreplay -I eth0 -t capture.pcap -C  
randomize packet timing

tcpreplay -I eth0 -t capture.pcap -L  
replay capture in a loop

tcpreplay -I eth0 -t capture.pcap -S  
view stats on the replay

tcpreplay -I eth0 -t capture.pcap -p 1-10  
select range of packets to replay

tcpreplay -I eth0 -t capture.pcap -M <src mac>:<dst mac>  
change source and dst mac addresses

tcpreplay -I eth0 -t capture.pcap -A <src IP>:<dst IP>  
change src and dst IP

tcpreplay -I eth0 -t capture.pcap -F "port 80"  
replay specific protocols

## curl

curl google.com  
basic GET request

curl -X POST google.com  
to do anything other than GET, use -X

curl -H "HeaderName: HeaderValue" google.com  
change headers

curl --cookie "name=value" google.com  
send and receive cookies

curl --cookie-jar cookies.txt google.com

curl -o output.txt google.com  
output to file

curl -O google.com  
output to file with host name

curl -k google.com  
allow SSL sites without certificates

curl -u username:password google.com  
specify username and pword for http auth