

INTRO TO NETWORK TRAFFIC ANALYSIS

CHEAT SHEET

Keep in mind, unless you are utilizing root, sudo privileges will be required to execute any applications that need to bind a network interface or set it into promiscuous mode.

Nomachine Connection Information

Target IP == 10.129.43.4
Username == htb-student
Password == HTB_@cademy_stdnt!

Tcpdump

Prints the tcpdump and libpcap version strings then exits:

```
tcpdump --version
```

Prints the help and usage information: **tcpdump -h**

Prints a list of usable network interfaces from which tcpdump can capture: **tcpdump -D**



INTRO TO NETWORK TRAFFIC ANALYSIS

CHEAT SHEET

Executes tcpdump and utilizes the interface specified to capture on:
`tcpdump -i (interface name or #)`

Runs a capture on the specified interface and writes the output to a file:
`tcpdump -i (int) -w file.pcap`

TCPDump will read the output from a specified file:
`tcpdump -r file.pcap`

TCPDump will utilize the capture traffic from a live capture or a file and set stdout as line-buffered. We can then utilize pipe (|) to send that output to other tools such as grep to look for strings or specific patterns:
`tcpdump -r/-w file.pcap -l \| grep 'string'`

TCPDump will start a capture on the interface specified at (int) and will only capture traffic originating from or destined to the IP address or hostname specified after host:

`tcpdump -i (int) host (ip)`

Will filter the capture for anything sourcing from or destined to port (#) and discard the rest:
`tcpdump -i (int) port (#)`

Will filter the capture for any protocol traffic matching the (#). For example, (6) would filter for any TCP traffic and discard the rest:

`tcpdump -i (int) proto (#)`

Will filter the capture for any protocol traffic matching the (#). For example, (6) would filter for any TCP traffic and discard the rest:

`tcpdump -i (int) proto (#)`



INTRO TO NETWORK TRAFFIC ANALYSIS

CHEAT
SHEET

Will utilize a protocols common name to filter the traffic captured.
TCP/UDP/ICMP as examples:

```
tcpdump -i (int) (proto name)
```

Tcpdump Common Switches and Filters

Will display any interfaces available to capture from: **D**

Selects an interface to capture from ex. -i eth0: **i**

Do not resolve hostnames: **n**

Do not resolve hostnames or well-known ports: **nn**

Will grab the ethernet header along with upper-layer data: **e**

Show Contents of packets in hex and ASCII: **X**

Same as X, but will also specify ethernet headers (like using Xe): **XX**

Increase the verbosity of output shown and saved: **v**, **vv**, **vvv**

Grab a specific number of packets, then quit the program: **c**

Defines how much of a packet to grab: **s**

Change relative sequence numbers in the capture display to absolute
sequence numbers (13248765839 instead of 101): **S**



INTRO TO NETWORK TRAFFIC ANALYSIS

CHEAT SHEET

Print less protocol information: **q**

Read from a file: **r file.pcap**

Write into a file: **w file.pcap**

Host will filter visible traffic to show anything involving the designated host. Bi-directional: **host**

src and dest are modifiers. We can use them to designate a source or destination host or port: **src / dest**

net will show us any traffic sourcing from or destined to the network designated. It uses / notation: **net**

Will filter for a specific protocol type (ether, TCP, UDP, and ICMP as examples): **proto**

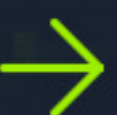
Port is bi-directional. It will show any traffic with the specified port as the source or destination: **port**

Portrange allows us to specify a range of ports (0-1024): **portrange**

Less and greater can be used to look for a packet or protocol option of a specific size: **less / greater "< >"**

And && can be used to concatenate two different filters together. for example, src host AND port: **and / &&**

Or allows for a match on either of two conditions. It does not have to meet both. It can be tricky: **or**



INTRO TO NETWORK TRAFFIC ANALYSIS

CHEAT SHEET

Not is a modifier saying anything but x. For example, not UDP: **not**

TShark

Prints the help menu: **tshark -h**

List available interfaces to capture from: **tshark -D**

Capture on a selected interface. Replace (int) with the interface name or number: **tshark -i (int)**

Apply a filter with (-f) looking for a specific host while utilizing tshark:
tshark -i eth0 -f "host (ip)"

Will display any interfaces available to capture from and then exit out:
D

Will list the Link-layer mediums you can capture from and then exit out (ethernet as an example): **L**

Choose an interface to capture from (-i eth0): **i**

Packet filter in libpcap syntax. Used during capture: **f**

Grab a specific number of packets, then quit the program. Defines a stop condition: **c**

Defines an autostop condition. It can be after a duration, specific file size, or after a certain number of packets: **a**



INTRO TO NETWORK TRAFFIC ANALYSIS

CHEAT
SHEET

Read from a file: **r** (**pcap-file**)

Write into a file using the pcapng format: **W** (**pcap-file**)

Will print the packet summary while writing into a file (-W): **P**

Will add Hex and ASCII output into the capture: **x**

See the help menu: **h**

WireShark

Capture only traffic pertaining to a certain host: **host** **x.x.x.x**

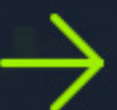
Capture traffic to or from a specific network (using slash notation to specify the mask): **net** **x.x.x.x/24**

Using **src** or **dst** net will only capture traffic sourcing from the specified network or destined to the target network:
src/dst net **x.x.x.x/24**

Will filter out all traffic except the port you specify: **port** **#**

Will capture everything except the variable specified ex. not port 80:
not

AND will concatenate your specified ports ex. host 192.168.1.1 and port 80: **and**



INTRO TO NETWORK TRAFFIC ANALYSIS

CHEAT SHEET

Portrange will grab traffic from all ports within the range only:
`portrange x-x`

These filters will only grab traffic from specified protocol headers:
`ip / ether / tcp`

Grabs a specific type of traffic. one to one, one to many, or one to all:
`broadcast / multicast / unicast`

Capture only traffic pertaining to a certain host. This is an OR statement: `ip.addr == x.x.x.x`

Capture traffic pertaining to a specific network. This is an OR statement: `ip.addr == x.x.x.x/24`

Capture traffic to or from a specific host:
`ip.src/dst == x.x.x.x`

filter traffic by a specific protocol. There are many more options:
`dns / tcp / ftp / arp / ip`

Filter by a specific tcp port: `tcp.port == x`

will capture everything except the port specified:
`src.port / dst.port ==x`

AND will concatenate, OR will find either of two options, NOT will exclude your input option: `and / or / not`



INTRO TO NETWORK TRAFFIC ANALYSIS

CHEAT SHEET

Allows us to follow a tcp session in which we captured the entire stream. Replace (#) with the session to reassemble:

```
tcp.stream eq #
```

Will filter for any traffic matching the http protocol: **http**

This filter will display any packet with a jpeg image file:

```
http && image-jif
```

Filters for the ftp protocol: **ftp**

Will filter for any control commands sent over ftp control channel:

```
ftp.request.command
```

Will show any objects transferred over ftp: **ftp-data**

Misc Commands

Sudo will run the command that proceeds it with elevated privileges:

```
sudo *
```

Utilizes which to determine if (application) is installed on the host.

Replace the application with what you are looking for ex. which

```
tcpdump: which (application)
```

Uses elevated privileges to install an application package if it does not exist on the host. ex. sudo apt install wireshark:

```
sudo apt install (application)
```



INTRO TO NETWORK TRAFFIC ANALYSIS

CHEAT
SHEET

Displays the manual pages for an application. ex. man tcpdump:
`man (application)`

Misc Commands

FTP-Data | Data channel for passing FTP files: **20**

FTP-Command | Control channel for issuing commands to an FTP server: **21**

SSH | Secure Shell Service port. Provides secure remote communications: **22**

Telnet | Telnet service provides cleartext communications between hosts: **23**

SMTP | Simple Mail Transfer protocol. Utilized for email transmissions between servers: **25**

DNS | Domain Name Services. Provides name resolution with multiple protocols: **53**

TFTP | Trivial File Transfer Protocol. A lightweight, minimal-function transfer protocol: **69**

HTTP | HyperText Transfer Protocol. Provides dynamic web services: **80**

Kerberos | Providing cryptographic network authentication: **88**



INTRO TO NETWORK TRAFFIC ANALYSIS

CHEAT SHEET

POP3 | Mail service utilized by clients to retrieve email from a server: **110**

RPC | Remote Procedure Call. Remote service for managing network file systems: **111**

SFTP | SSH File Transfer Protocol. An extension of SSH providing secure and reliable FTP services: **115**

NTP | Network Time Protocol. Provides timing and sync services for network devices: **123**

Netbios-NS | Local network name resolution: **137**

Netbios-SSN | Provides session services for data transfer. Services like SMB can utilize it: **139**

BGP | Border Gateway Protocol. BGP is a protocol for exchanging routing info with autonomous systems worldwide: **179**

LDAP | Lightweight Directory Access Protocol. System agnostic authentication and authorization services: **389**

HTTPS | HyperText Transfer Protocol Secure. An extension of HTTP utilizing SSL/TLS for encrypting the communications: **443**

SMB | Server Message Block. SMB allows for the sharing of services, files, networking ports, and printers between hosts.: **445**