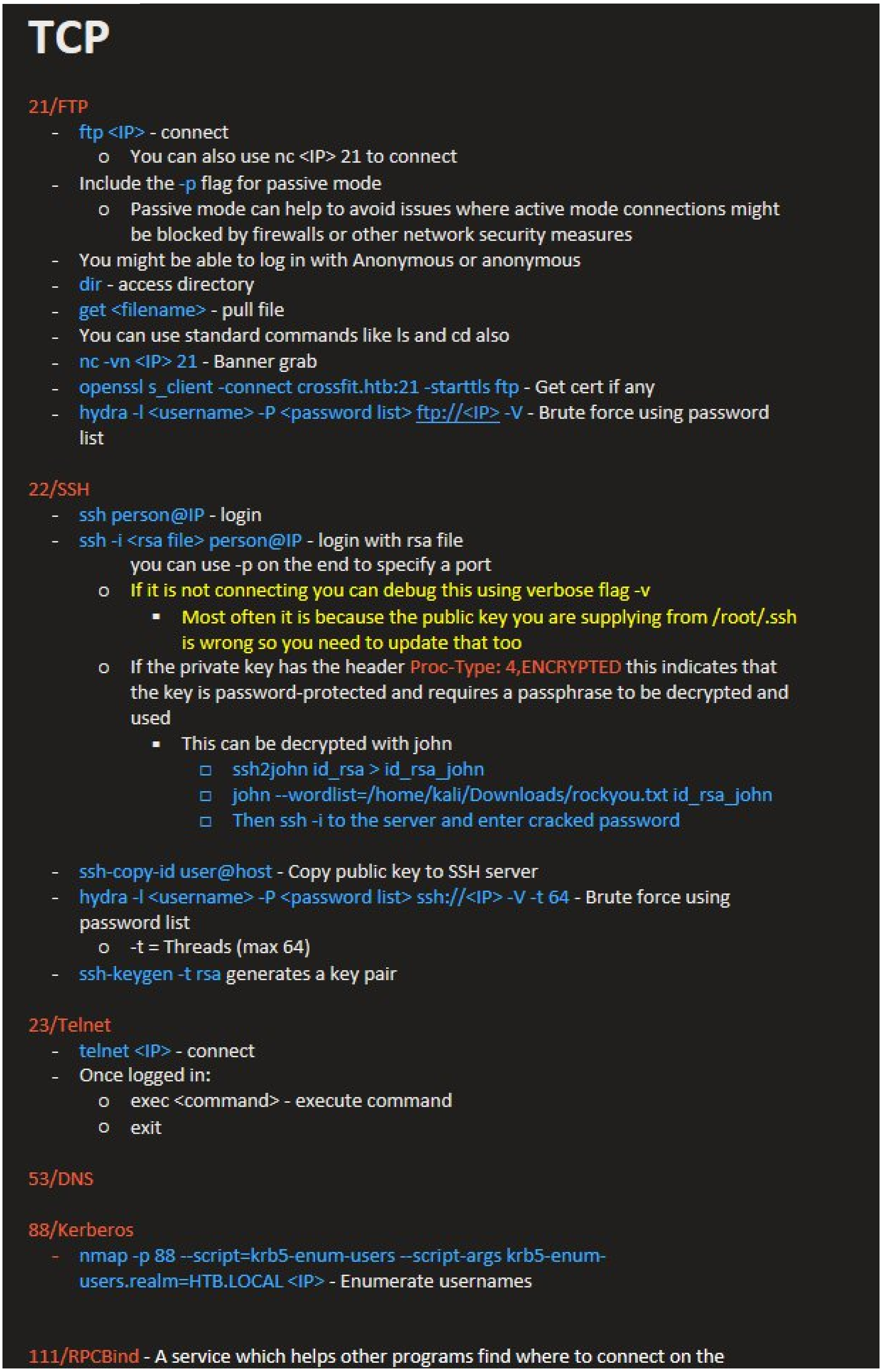
Ports



TCP

21/FTP

- ftp <IP> - connect
  - You can also use nc <IP> 21 to connect
- Include the -p flag for passive mode
  - Passive mode can help to avoid issues where active mode connections might be blocked by firewalls or other network security measures
- You might be able to log in with Anonymous or anonymous
- dir - access directory
- get <filename> - pull file
- You can use standard commands like ls and cd also
- nc -vn <IP> 21 - Banner grab
- openssl s_client -connect crossfit.htb:21 -starttls ftp - Get cert if any
- hydra -l <username> -P <password list> ftp://<IP> -V - Brute force using password list

22/SSH

- ssh person@IP - login
- ssh -i <rsa file> person@IP - login with rsa file
  you can use -p on the end to specify a port
  - If it is not connecting you can debug this using verbose flag -v
    - Most often it is because the public key you are supplying from /root/.ssh is wrong so you need to update that too
  - If the private key has the header Proc-Type: 4,ENCRYPTED this indicates that the key is password-protected and requires a passphrase to be decrypted and used
    - This can be decrypted with john
      - ssh2john id_rsa > id_rsa_john
      - john --wordlist=/home/kali/Downloads/rockyou.txt id_rsa_john
      - Then ssh -i to the server and enter cracked password

- ssh-copy-id user@host - Copy public key to SSH server
- hydra -l <username> -P <password list> ssh://<IP> -V -t 64 - Brute force using password list
  - -t = Threads (max 64)
- ssh-keygen -t rsa generates a key pair

23/Telnet

- telnet <IP> - connect
- Once logged in:
  - exec <command> - execute command
  - exit

53/DNS

88/Kerberos

- nmap -p 88 --script=krb5-enum-users --script-args krb5-enum-users.realm=HTB.LOCAL <IP> - Enumerate usernames

111/RPCBind - A service which helps other programs find where to connect on the network as services can be located on different ports, this keeps things organised.

- For example, if a program is offering a file-sharing service, it tells rpcbind "I'm listening on port 2049 for file-sharing requests."
- `rpcinfo -p <ip>` - See services
- `showmount -e <ip>` - See the mounted NFS services
- nmap enumeration:
  - `nmap -p 111 --script=nfs-ls,nfs-statfs,nfs-showmount <IP>`
- `sudo mkdir -p <save path>` - Create mount point
  - `mount -t nfs <ip>:<share path> <save path>` - Save an NFS drive on your new mount
  - `cp -r <mount directory> <new path>` - Save mount locally by copying it
- `umount <path>` - Unmount NFS share

RPCBind info

- **nfs-ls:** Lists files and directories on an NFS share. It helps in identifying what files and directories are available on the NFS server.
- **nfs-statfs:** Retrieves filesystem statistics from the NFS server. It provides information about the available space, the total size, and other filesystem-related details.
- **nfs-showmount:** Shows which directories are shared by the NFS server. It helps in identifying the exported file systems and their details.

- **RPC:** Handles communication between the client and the NFS server. It makes sure the client's requests (like opening or reading a file) reach the NFS server.
  - **RPC** is the underlying mechanism that enables NFS to perform remote file operations.
  - **rpcbind:** Keeps track of which address the NFS service is using, so when the client requests to use NFS, rpcbind tells it where to find the NFS service.
- **NFS (Network File System):** A protocol that allows a computer to access files on another computer over a network as if they were on the local machine. It facilitates file sharing and is commonly used in Unix and Linux environments
  - **NFS Server:** Runs a service that manages file requests.
  - When you create an NFS mount, you aren't just making a copy of the files, you are accessing the file system itself and interacting with it as if it were local. This is also why is slower interacting with it

How It All Fits Together

1. **Client Request:**
   - **Action:** The client wants to access data.txt from the NFS server.
   - **RPC Call:** The client sends an RPC request specifying the action (e.g., read data.txt) and the NFS program number to the server.
2. **Finding the Service:**
   - **rpcbind on Server:** The server has rpcbind running, which listens on port 111. It receives the RPC request from the client and looks up the NFS program number.
   - **Translation:** rpcbind translates this program number into the specific port where the NFS service is listening.
3. **Communication:**
   - **RPC Communication:** The client receives the address from rpcbind and sends the RPC request to the NFS service at that address.
   - **NFS Server:** The NFS service on the server receives the request and performs the action (e.g., reading data.txt), then sends the response back to the client.

135/MSRPC - Allow a program to request a service from another computer's program

- rpcclient -U "" -N <target_ip> - Enumerate info from msrpc
- rpcdump.py @<target_ip> - Impacket tool used for enumerating msrpc

139 & 445/SMB

- SMB uses both ports due to its historical development, with port 139 being associated with NetBIOS over TCP/IP and port 445 with direct SMB over TCP/IP without NetBIOS
  - Modern implementations typically use port 445
- smbclient -L //<IP> - lists shares
  - You don't have to supply password
- smbclient //<IP>/<drive> - access drive/share
- *get <filename>* - pull file
- Pull directory
  - prompt
    recurse
    mget <dir>
- smbget -r smb://<IP>/share - Pull entire smb share
- put <file> - upload file
- Administrator is a standard account on Windows
  - Try tack -U "Administrator" then blank password
- Tack -N for no password
- Nmap enum
  - nmap -p 445 --script=smb-enum-shares.nse,smb-enum-users.nse <IP>
- Execute system commands with !
  - EG !cat prep-prod.txt

443/HTTPS

873/rsync (cmd line utility for transferring files between comp and drive on unix)

- rsync --list-only <IP>:: - Show shares/files
- rsync --list-only <IP>::<foldername> - Navigate into folders
- rsync -v <remoteIP>::<filepath> <destination> - Transfer file from remote to your local
- rsync -avh /<source> <remoteIP>::<filepath> - Transfer file from local to remote

1883/MQTT

- MQTT (Message Queuing Telemetry Transport) is a lightweight, publish-subscribe network protocol that transports messages between devices. It is commonly used in machine-to-machine (M2M) or "Internet of Things" (IoT) contexts where a small code footprint is required, and network bandwidth is at a premium.

3306/mysql

- Cheatsheet: https://devhints.io/mysql
- You can sometimes use root to login without a password
- Login: mysql -u <username> -h <hostname>
  - Internal login:
    - mysql -h 127.0.0.1 -P 3306 -u <username> -p
- Show the databases: SHOW DATABASES;
- Select database: use <dbname>;
- Show tables: show tables;
- See what's inside a table: SELECT * FROM <table>;
- Show fields: SHOW FIELDS FROM <table>;
  - FIELDS = COLUMNS

- See what's inside a field: SELECT * FROM <field>;

3389/Ms-WBT-Server (used for RDP to Microsoft machines)

- On xfreerdp you can use Administrator as the username and not use a password
- xfreerdp /u:ADMINISTRATOR /v:<IP>

5000/upnp - Universal plug and play

This typically indicates that Universal Plug and Play (UPnP) services are running on the target system. UPnP is a network protocol suite that allows devices to discover each other's presence on a network and establish functional services for data sharing, communications, and entertainment

5432/PSQL - Postgresql server

PSQL - CMD line utility for accessing postgresql DBs

- Note: may require local port forwarding if only accessible internally
- Sign in: psql -h <localhost or IP> -U <username>
  - In our case because we have tunnelled already we can use localhost
- Commands:
  \l - list databases in DB
  \c <database name> - connect to database
  \dt - list tables in DB
- You can execute SQL command to extract data from the table name
  - EG SELECT * FROM flag;
  - It may take a couple tries if its saying the terminal is not fully functional

5672/amqp

- AMQP (Advanced Message Queuing Protocol). This protocol is used for message-oriented middleware, providing a robust, efficient, and flexible method for communication between applications.

6379/Redis

- In-memory database
- Redis-cli -h <IP> - connect
- INFO - information/stats
- select <index> - selects database
- keys * - show all keys
- get <key> - Show key

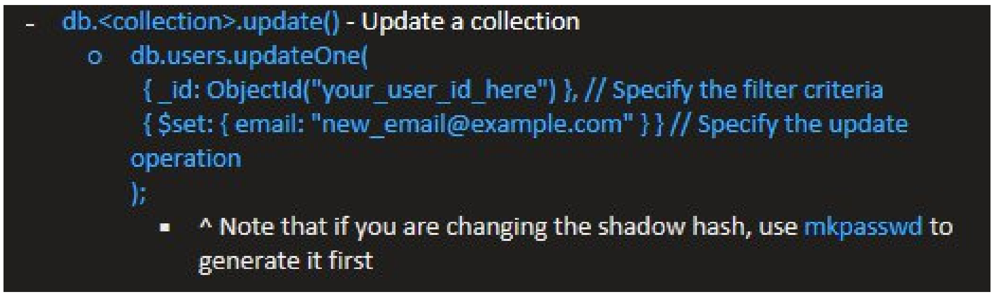
80/8080/Http

- May need to tack on a ':8080' or ':80' on the end of the URL

8443/SSL

27017/27117/mongodb - JSON-like, non-sql database

- mongo --port <mongo port> - Connect to mongodb
- show dbs - Show databases
  - 'ace' is the name of default DB names for UniFi apps
- use <dbname> - Switch to that database
- show collections - Shows all info in that database
  - In the admin collection I found /shadow hashes of users
- db.<collection>.find() - Show what's inside a collection
  - Append .pretty() to make it easier to read
- db.<collection>.update() - Update a collection



- db.<collection>.update() - Update a collection
    - db.users.updateOne(
      { _id: ObjectId("your_user_id_here") }, // Specify the filter criteria
      { $set: { email: "new_email@example.com" } } // Specify the update operation
      );
        - ^ Note that if you are changing the shadow hash, use mkpasswd to generate it first

UDP

69/TFTP (Trivial File Protocol)

- A simple version of FTP with no user authentication
- /var/lib/tftpboot/ - Default system folder
- tftp <IP> - Connect to this IP
- put <file> - Upload file

List

- 20/21 - FTP
- 21/990 - FTPS
- 22 - SSH
- 23 - Telnet
- 25 - SMTP
- 49 - Tacacs
- 53 - DNS
- 88 - Kerberos
- 110 - POP3 (unencrypted)
- 123 UDP - NTP
- 139/445 - SMB
- 161 - SNMP
- 1701 - L2TP
- 3389 - RDP
- 389 - LDAP
- 443 - HTTPs
- 587 - Secure SMTP
- 636 - LDAPS
- 993 - IMAP (TLS/SSL)
- 995 - POP3 (TLS/SSL)
- 1812 UDP - RADIUS