Program Flow

- Starts in main()
- Calls buildFiles()
  - Calls buildDns()
    - Builds DNS using powershell
  - Calls buildEmail()
    - Sets up powershell script used to send emails when data limits of blacklist violations occur in \ User*currentuser*\SWEProj\
    - Calls buildDataFiles()
      - Writes blacklist files in User*currentuser*\SWEProj\
- Defines variables
  - alldevs, d – both items in a list of network interfaces
  - inum – number of interfaces found
  - i – incrementor for a loop
  - errbuf – a character array for an error buffer
  - adhandle - this is a descriptor of an open capture instance
- Checks to see if there's an error finding network devices.
- For loop to store all the network devices in d
  - d->name is the devices name
  - d->description in a description of the device
  - increments i
- if (I == 0) print a message saying no devices found and exit the program
- Next two loops print a list of network devices
- Ask the user what network interface to use
- Loops to select the user selected network interface
- Opens selected device
  - Function is passed:
    - the name of the device in d_name
    - the size of the packet 65536(full packets)
    - the flag is set to 0 which allows us to see all packets
    - number of milliseconds wait time set to 1000 before grabbing more packets
    - pcap_rmtauth * set to NULL because we aren't using a remote machine
    - errbuf, stores the error message if there's an error
  - If this fails
    - the program exits and prints errbuf
    - free all devs
    - exits program
- New user is made then assigned to a global variable – All of the following pull information about the user of the computer and sets up folders to store email templates and blacklist info.
  - UserInfo();
  - UserInfo(pcap_if_t*);
  - void setUserName();
  - void setComputerName();

- std::string getUserName();
- std::string getComputerName();
- pcap_addr_t* usedInterfaceAddresses;
- void setIP4Address();
- void setIP6Address();
- pcap_if_t* usedInterface;
- ip_address getLocalIPAddress();
- ip_address getSubnetAddress();
- ip_address getBroadcastIPAddress()

- Blacklist is made and assigned to a global variable – All of the following create an ip4 and ip6 blacklist in the specified folder.
  - std::vector<ip_address> IPv4addresses;
  - std::vector<std::string> hostNames;
  - std::vector<ip6_address> IPv6addresses;
  - BlackList(UserInfo);
  - BlackList();
  - void generateAddresses(UserInfo);
  - bool checkBlackListIPv4(ip_address);
  - bool checkBlackListIPv6(ip6_address);
  - bool checkBlackListHostName(std::string);
- Free all devs (as they are no longer needed since packet sniffing will begin)
- pcap_loop is passed
  - adhandle
  - 0 – which means to capture an infinite amount of packets
  - packet_handler – this is where the packet is analyzed in info is pulled from the packet
    - splits the first byte of data in the packet to determine what type of packet it is using BinaryTo Decimal function
    - cout the type of packet
    - If an ip4 packet
      - Packet data is passed into an instance of the packet class
        - Ip4header is pulled from the packet and stored
        - TCPheader is pulled from the packet and stored
        - ethernetHeader is pulled from the packet and stored
        - type is set to 4
      - All the above variables, plus a hex dump of the packet header is output to the screen
      - Checks to see if the packet is from out of network. If True:
        - Adds the bytes of the packet to the total used for data limit check
        - Packet IP is checked vs the blacklist to see if there's a violation(loop runs twice for sent and received. If true:
          - Sends an email to the admin
          - Exits program
      - Else: cout >> "in network"
    - Same process for an ip6 packet

- Program runs until user breaks, or there is a data or blacklist violation