Mission 1

Here's the mail servers for "starwars.com" according to "nslookup":

```
PS C:\Users\Justin> nslookup -type=mx starwars.com
Server:  cdns01.comcast.net
Address:  2001:558:feed::1

Non-authoritative answer:
starwars.com    MX preference = 10, mail exchanger = aspmx2.googlemail.com
starwars.com    MX preference = 10, mail exchanger = aspmx3.googlemail.com
starwars.com    MX preference = 5, mail exchanger = alt1.aspx.l.google.com
starwars.com    MX preference = 5, mail exchanger = alt2.aspmx.l.google.com
starwars.com    MX preference = 1, mail exchanger = aspmx.l.google.com

aspmx2.googlemail.com   internet address = 173.194.77.27
aspmx3.googlemail.com   internet address = 173.194.219.27
```

It looks like the Resistance isn't receiving emails because their MX servers aren't correct. Their MX servers should be the following:

- Primary: "aspmx.l.google.com"
- Secondary: "alt2.aspmx.l.google.com" or "alt1.aspx.l.google.com"

Mission 2

Here's the "SPF" record for "theforce.net" according to "nslookup":

```
PS C:\Users\Justin> nslookup -type=txt theforce.net
Server:  cdns01.comcast.net
Address:  2001:558:feed::1

Non-authoritative answer:
theforce.net    text =

        "google-site-verification=ycgY7mtk2oUZMagcffhFL_Qaf8Lc9tMRkZZSuig0d6w"
theforce.net    text =

        "v=spf1 a mx mx:smtp.secureserver.net include:aspmx.googlemail.com ip4:104.156.250.80 ip4:45.63.15.159 ip4:45.63.4.215"
theforce.net    text =

        "google-site-verification=XTU_We07Cux-6WCSOItl0c_WS29hzo92jPE341ckbOQ"
```

The “Force” emails are going to spam because the IP address “45.23.176.21” doesn’t exist in the “SPF” record above. A corrected DNS record would include “ip4:45.23.176.21” in the “SPF” record. I would assume that “theforce.net” would also want to remove the old IP address from the above “SPF” record.

Mission 3

Here’s the “CNAME” record(s) for “theforce.net”:

```
PS C:\Users\Justin> nslookup -type=cname www.theforce.net
Server:  cdns01.comcast.net
Address:  2001:558:feed::1

Non-authoritative answer:
www.theforce.net        canonical name = theforce.net
```

“resistance.theforce.net” isn’t redirecting to “theforce.net” because that redirect doesn’t exist as a “CNAME” record. To get that redirect to work, “resistance.theforce.net” redirecting to “canonical name = theforce.net” simply needs to be added as a “CNAME” record.

Mission 4

Here’s the “NS” records for “princessleia.site”:

```
PS C:\Users\Justin> nslookup -type=ns princessleia.site
Server:  cdns01.comcast.net
Address:  2001:558:feed::1

Non-authoritative answer:
princessleia.site       nameserver = ns26.domaincontrol.com
princessleia.site       nameserver = ns25.domaincontrol.com

ns25.domaincontrol.com  internet address = 97.74.102.13
ns25.domaincontrol.com  AAAA IPv6 address = 2603:5:2161::d
ns26.domaincontrol.com  internet address = 173.201.70.13
ns26.domaincontrol.com  AAAA IPv6 address = 2603:5:2261::d
```

In order to make sure this doesn't happen again, we would simply need to add a "NS" record that adds the backup server "ns2.galaxybackup.com".

Mission 5

The guaranteed way to determine the OSPF is to use "https://www.analyticssteps.com/blogs/dijkstras-algorithm-shortest-path-algorithm". Using that algorithm, Here is the OSPF for Batuu to Jedha:

```
justin@DESKTOP-4THSC21:~/python_programming/djikstra$ python main.py
Distance from Batuu to Jedha is: 23.00
Path from Batuu to Jedha is: Batuu -> D -> C -> E -> F -> J -> I -> L -> Q -> T -> V -> Jedha
```

I implemented Dijkstra's algorithm in Python using the following code:

```
class Dijkstra:

    def __init__(self, vertices, graph):
        self.vertices = vertices  # ("A", "B", "C" ...)
        self.graph = graph  # {"A": {"B": 1}, "B": {"A": 3, "C": 5} ...}

    def find_route(self, start, end):
        unvisited = {n: float("inf") for n in self.vertices}
        unvisited[start] = 0  # set start vertex to 0
        visited = {}  # list of all visited nodes
        parents = {}  # predecessors
        while unvisited:
            min_vertex = min(unvisited, key=unvisited.get)  # get smallest distance
            for neighbour, _ in self.graph.get(min_vertex, {}).items():
                if neighbour in visited:
                    continue
                 new_distance = unvisited[min_vertex] + self.graph[min_vertex].get(neighbour,
float("inf"))
                if new_distance < unvisited[neighbour]:
                    unvisited[neighbour] = new_distance
                    parents[neighbour] = min_vertex
            visited[min_vertex] = unvisited[min_vertex]
            unvisited.pop(min_vertex)
            if min_vertex == end:
                break
        return parents, visited

    @staticmethod
```

```
    def generate_path(parents, start, end):
        path = [end]
        while True:
            key = parents[path[0]]
            path.insert(0, key)
            if key == start:
                break
        return path
```

I got the above code from the following Stack Overflow post: https://stackoverflow.com/questions/22897209/dijkstras-algorithm-in-python/61078380#61078380. I then used the above code using the following data:

```
input_vertices_without_N =
['Batuu','B','C','D','E','F','G','H','I','J','K','L','M','O','P','Q','R','S','T','U','V','Jed
ha']

input_graph_without_N = {
    'Batuu': {'B': 4, 'C': 6, 'D': 1},
    'B': {'Batuu': 4, 'C': 5, 'E': 8},
    'C': {'Batuu': 6, 'B': 5, 'E': 1, 'F': 6, 'G': 6, 'D': 2},
    'D': {'Batuu': 1, 'C': 2, 'G': 8, 'O': 15},
    'E': {'B': 8, 'C': 1, 'F': 1, 'I': 5, 'H': 5},
    'F': {'C': 6, 'E': 1, 'I': 8, 'J': 1, 'G': 5},
    'G': {'F': 5, 'K': 8, 'O': 5, 'D': 8, 'C': 6},
    'H': {'E': 5, 'I': 5, 'L': 7},
    'I': {'E': 5, 'H': 5, 'L': 6, 'M': 3, 'J': 1, 'F': 8},
    'J': {'F': 1, 'I': 1, 'M': 7, 'K': 2},
    'K': {'J': 2, 'O': 9, 'G': 8},
    'L': {'H': 7, 'I': 6, 'M': 6, 'Q': 4, 'P': 9},
    'M': {'I': 3, 'L': 6, 'Q': 14, 'J': 7},
    'O': {'D': 15, 'G': 5, 'K': 9, 'R': 1, 'S': 11},
    'P': {'L': 9, 'Q': 99, 'T': 9},
    'Q': {'M': 14, 'L': 4, 'P': 99, 'T': 2, 'U': 8, 'R': 2},
    'R': {'O': 1, 'Q': 2, 'U': 8, 'V': 12, 'S': 8},
    'S': {'O': 11, 'R': 8, 'Jedha': 8},
    'T': {'P': 9, 'Q': 2, 'U': 7, 'V': 2},
    'U': {'T': 7, 'V': 3, 'R': 8, 'Q': 8},
    'V': {'Jedha': 2, 'R': 12, 'U': 3, 'T': 2},
    'Jedha': {'S': 8, 'V': 2}
}
```

I then called the script using the following code:

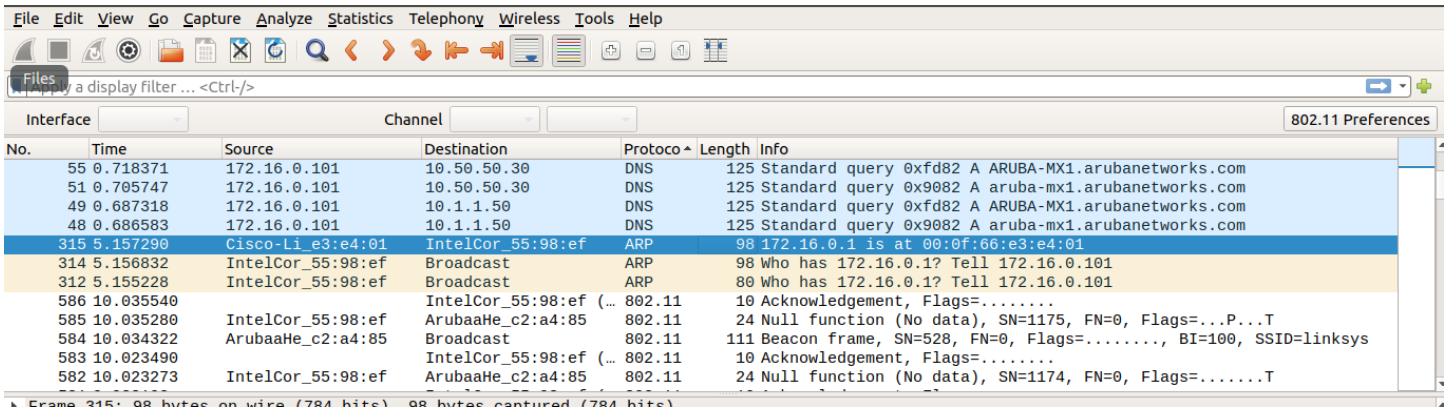
```
start_vertex = 'Batuu'
end_vertex= 'Jedha'
dijkstra = djikstra.Dijkstra(input_vertices_without_N, input_graph_without_N)
p, v = dijkstra.find_route(start_vertex, end_vertex)
print("Distance from %s to %s is: %.2f" % (start_vertex, end_vertex, v[end_vertex]))
se = dijkstra.generate_path(p, start_vertex, end_vertex)
print("Path from %s to %s is: %s" % (start_vertex, end_vertex, " -> ".join(se)))
```

Mission 6

Found the WPA key using “aircrack-ng”:

```
sysadmin@UbuntuDesktop:~$ aircrack-ng -w /usr/share/wordlists/rockyou.txt Darksi
de.pcap -l key.key
Opening Darkside.pcap
Read 586 packets.

   #  BSSID              ESSID                     Encryption

   1  00:0B:86:C2:A4:85  linksys                   WPA (1 handshake)

Choosing first network as target.

Opening Darkside.pcap
Reading packets, please wait...

                                Aircrack-ng 1.2 rc4

      [00:00:00] 2288/7120714 keys tested (2375.99 k/s)

      Time left: 49 minutes, 57 seconds                          0.03%

                          KEY FOUND! [ dictionary ]


      Master Key     : 5D F9 20 B5 48 1E D7 05 38 DD 5F D0 24 23 D7 E2
                       52 22 05 FE EE BB 97 4C AD 08 A5 2B 56 13 ED E2

      Transient Key  : 1B 7B 26 96 03 F0 6C 6C D4 03 AA F6 AC E2 81 FC
                       55 15 9A AF BB 3B 5A A8 69 05 13 73 5C 1C EC E0
                       A2 15 4A E0 99 6F A9 5B 21 1D A1 8E 85 FD 96 49
                       5F B4 97 85 67 33 87 B9 DA 97 97 AA C7 82 8F 52

      EAPOL HMAC     : 6D 45 F3 53 8E AD 8E CA 55 98 C2 60 EE FE 6F 51
```

Here’s a Wireshark screenshot showing the decrypted wireless packets:



Here are the MAC and IP addresses for hosts that I saw in the ARP packets:

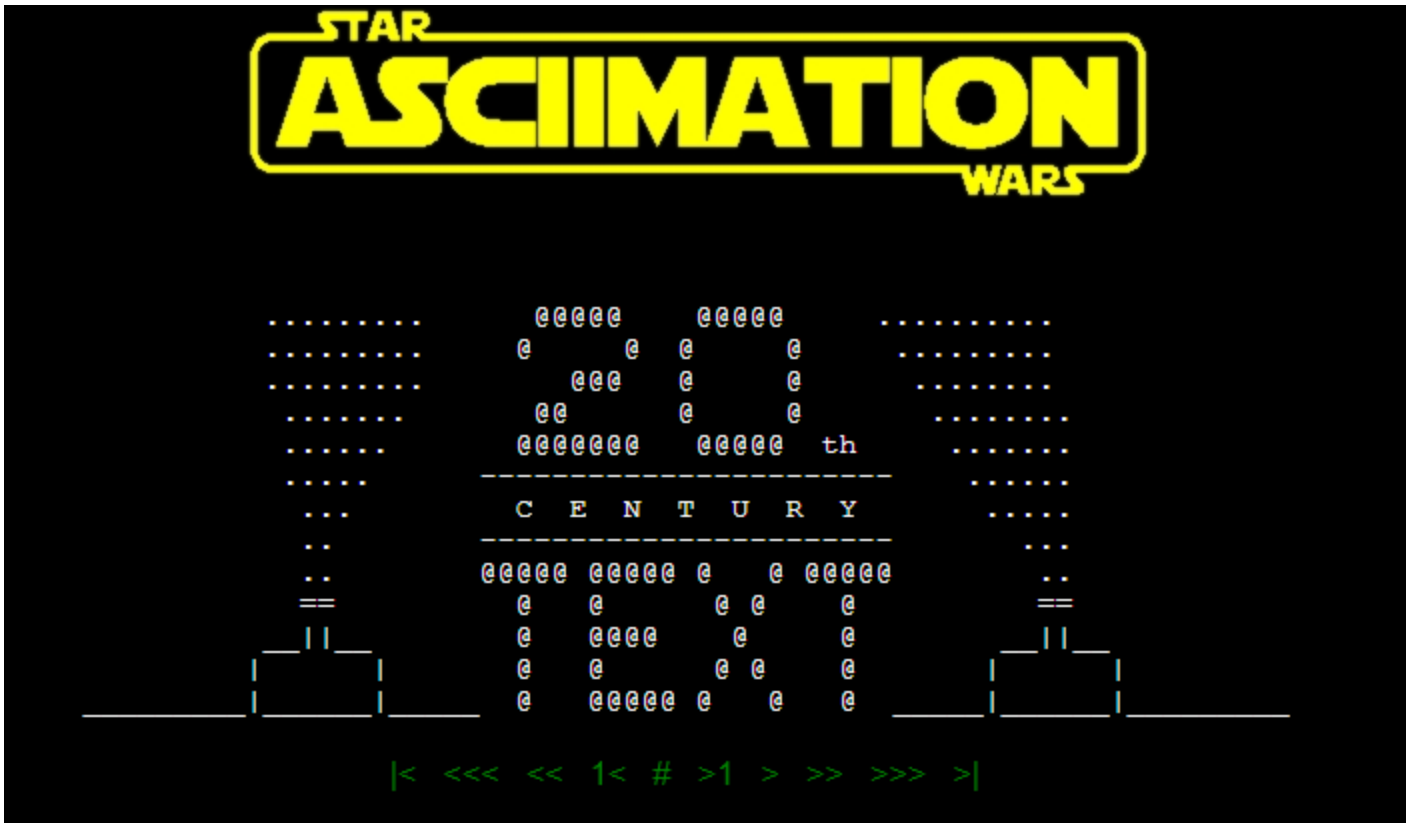
- MAC: Cisco-Li_e3:e4:01 (00:0f:66:e3:e4:01)
    - IP: 172.16.0.1
- MAC: IntelCor_55:98:ef (00:13:ce:55:98:ef)
    - IP: 172.16.0.101

Mission 7

Found the following message in the “TXT” record for “princessleia.site”:

```
PS C:\Users\Justin> nslookup -type=txt princessleia.site
Server:  cdns01.comcast.net
Address:  2001:558:feed::1

Non-authoritative answer:
princessleia.site       text =

        "Run the following in a command line: telnet towel.blinkenlights.nl or as a backup access in a browser: www.asciimation.co.nz"
```

I wasn’t able to access “towel.blinkenlights.nl” via Telnet. So, I accessed the website instead:


STAR
ASCIIMATION
WARS
C E N T U R Y
|< <<< << 1< # >1 > >> >>> >|