Methodology

Howard Network Topology Project

1. IP Scanning & Route Tracing Approach

We used Scapy to do it. The core code snippet is as the following:

```
109  def ping(ip):
110    if ip in topology.flags: return
111    if ip in topology.ips: return
112
113    r, _ = traceroute(ip, l4 = UDP(dport = 33434), verbose = 0)
114
115    if len(r) == 0:
116      topology.add_flag(ip)
117      return
118
119    prev = r[0][0].src
120    topology.add_ip(prev)
121
122    found = False
123
124    for _, x in r:
125      topology.add_ip(x.src)
126      topology.add_link(prev, x.src)
127
128      if x.src == ip: found = True
129      prev = x.src
```

It does not block for invalid IP as well.

2. Multi-Threading and Resuming Function for Effective Scan

We utilized multi-threading to ensure a large-scale scanning. Implementation of resuming from the interrupted point was essential. We added this function in our script.

3. Scanning Strategy

Even with multithreading, investigation of the entire range is surely a time-consuming process.

Therefore, we had to find an efficient scanning strategy.

Let's say the IP v4 address is w.x.y.z. For Howard, w=10 or w=138, x=238.

At each location of the campus, we first examined only IPs with z=1.

Since only x and y are variable, about 65536 IPs must be investigated.

It took one team member about 11 hours (MacBook) to finish this.

Next, we assumed that the IP ranges where z=1 was found are promising candidates and conducted a re-examination of them.
In other words, if 10.33.30.1 was discovered, the 10.33.30.2-10.33.30.254 area was reinvestigated.
In this way, one team member worked in one location for about 20 hours.

4. Visualization

Total 144,607 IPs and 182,383 links were discovered in our work. We didn't include IPs not in 10.x.y.z or 138.238.y.z range.
To visualize the topology efficiently and beautifully, we employed some approaches for extraction and pruning.
First of all, we calculated the distribution of node linkages.

# of Linkage	# of Nodes
0	5,677
1	104,513
2	26,157
3	7,462
4	688
5	49
6	13
7	3
8	3
9	2
10+	40

Distribution of node linkages

(It shows that 688 IPs are connected to four other IPs. This table might not be the correct report, since It is only based on our discoveries.)

As see in the above table, 5,677 nodes are isolated, and 104,513(about 72.3%) nodes are just terminals which is connected to only one other node. If we include those nodes in our visualization, it could look poor. That's why we pruned those IPs.

We used NetworkX and Matplotlib for drawing graphs. We generated four visualizations.

(1) Internal Topology

Internal IPs except for those with less than three connections.

(2) Public Topology

Internal IPs except for those with less than three connections.

(3) Bridge Topology

Insight of connections from internal IPs to external IPs. This diagram only involves internal-public or public-internal connections.

(4) Gate Topology

Only IPs with more than six connections. We thought that they could be considered as gateways or local hubs.

5. Naming Rule

To prevent from confusing, we set a naming rule for each team member who is scanning at a certain place. For example, if Morgan completed a scan job at Burr Gym, she would have the following two files:

Morgan_BurrGym_ip.txt

Morgan_BurrGym_link.txt

The first file contains all IPs found, while the second file contains all link pairs.

Each member merged his/her scan results in the following files:

[Name]_total_ip.txt

[Name]_total_link.txt

For example, Morgan will have the following merged files:

Morgan_total_ip.txt

Morgan_total_link.txt

The team's overall results are the following two files:

total_ip.txt

total_link.txt