1 Network Address Identification

To initially understand NCSU's network, I was on campus and ran an `ipconfig /all` in my Windows terminal. From there, it was determined that Dynamic Host Configuration Protocol (DHCP) server my computer was accessing for client-server relations had the IP address of `152.1.14.78` (note that this address is not static).

From there, I searched the address in censys.io. It was determined that when selecting the Autonomous Server Name of 'NCSU' would yield a significant amount more about NCSUs network. Using the censys.io API, I submitted a search query for

```
autonomous_system.name='NCSU'
```

From there, I got 100 results about hosts within the NCSU network. Of the 100 hosts, there were 3 unique CIDR blocks being used with consistent Network Names, Autonomous System Numbers (ASNs) and Autonomous System Names (AS-Names) shown in the table below.

	IPv4 Block 1	IPv4 Block 2	IPv4 Block 3
CIDR Block	152.1.0.0/16	152.7.0.0/16	152.14.0.0/16
Network Name	NCSU	NCSU	NCSU
ASN	11442	11442	11442
AS-Name	NCSU	NCSU	NCSU

Table 1: Network Attributes

Quite frankly, I do not know the difference between Network Name and AS-Name, but from the censys.io API I used the Autonomous System description as Network Name, and the AS-Name corresponds properly. If it means Network Name as in the name of the network when connecting, it should all be **ncsu** or **ncsu-guest**, as that is what our wireless networks are called.

Figure 6 in the appendix shows the Python notebook used to parse the JSON output from the censys.io API request. As seen, there are 3 unique CIDR blocks (highlighted by the red box, called `bgp_prefix` in the API), and only 1 unique value for the Autonomous System Name, Number, and Description (used as Network Name).

It should be noted, that starting on Parts 3 and 4, it seems that `152.46.0.0/X` is another CIDR block, where `X` is likely 16. Using Shodin and looking through the domain `ncsu.edu`, there were many IPs using `152.46.X.X` addresses which is how I came to this conclusion.

2 Network Summary

Using censys.io, I had to get more detailed queries than the initial ones I made for Section 1. Using their API, I set up a Python script identical to the one above but I expanded the search by calling the subroutine `view_all()` on my query object. This expanded each query significantly, to the point where the first (and only) time I called it with 100 search results, it used up 100 of my 250 queries. This was severely limiting. Since I had already used more than 150 queries at this point, it was useless to call more due to the fact that I would simply get repeated results. Therefore, all the data below comes from only 100 unique IPs, which I hope will be sufficient.

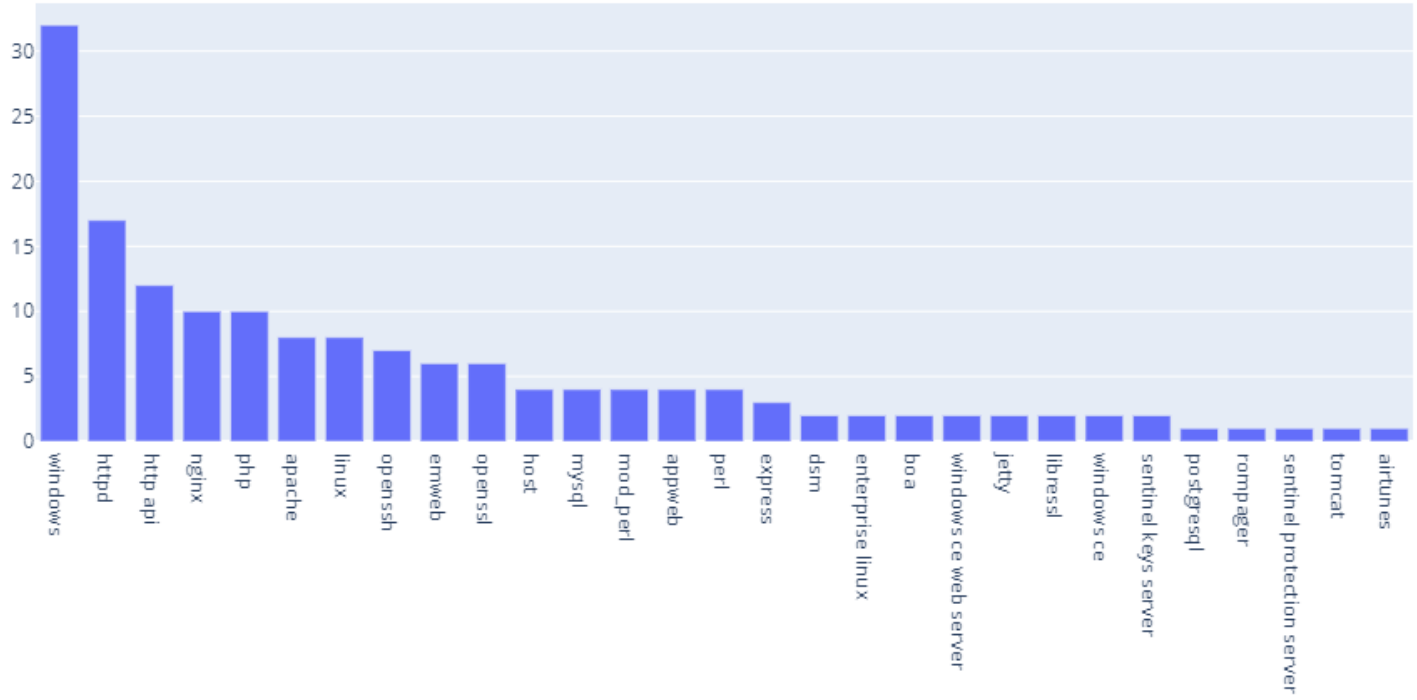

Figure 1: Hosts by OS, Server, and Protocol

Above is a bar graph showing a list of software products (operating systems, servers, and protocols) which are used to host the 100 IPs within the query. ***NOTE:*** *That for each IP address, multiple results can come back, therefore the total counts will equal greater than 100.* A manual count comes out to

Host Type	**Number of Types**	**Total Quantity**
OS	6	48
Server	17	58
Protocol	6	54

Table 2: Breakdown of Hosts

Where the Number of Types indicates the unique count of the x axis of Figure 1, and Total Quantity sums up their counts.

Some other quantifiable features I could extract and view as graphs were counts of OS's used as well as what protocols were being used on what ports (between TCP and UDP). Here are those results

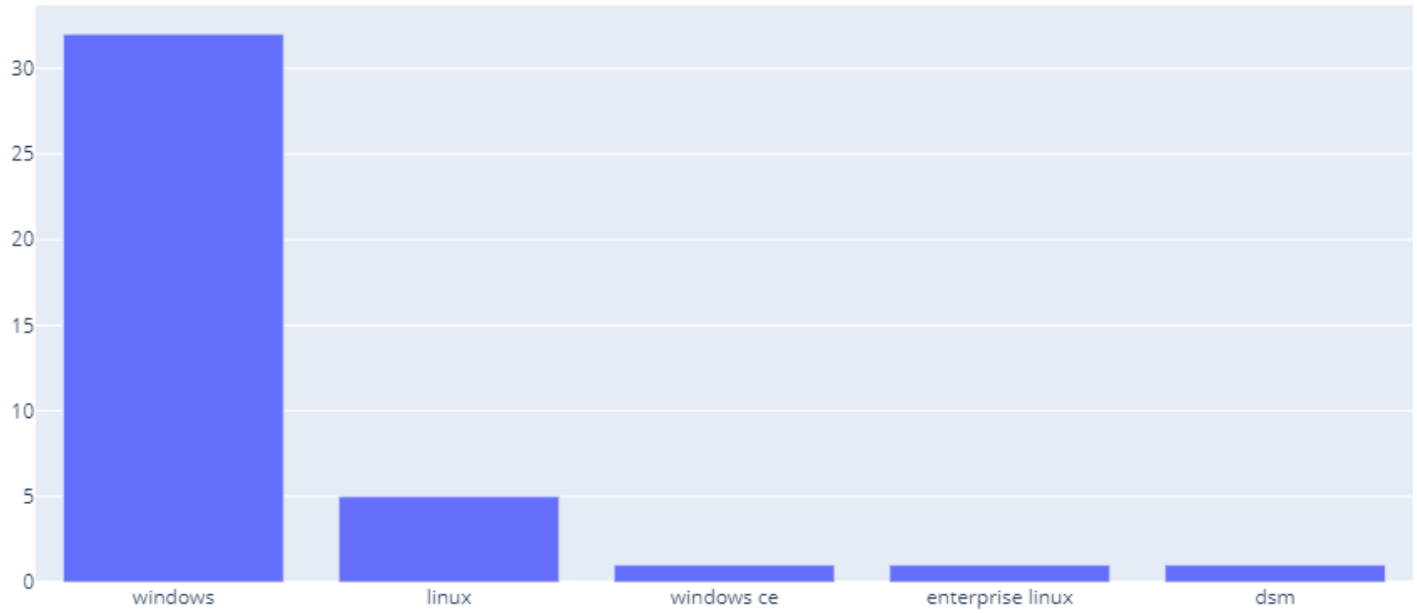

Figure 2: List of Operating Systems

These counts may seem identical to those above, but they are actually different. This can be seen in the Appendix at the end. These counts were extracted from direct queries at what operating system was used on the server end, whereas the other quantities were from what system of things the server is being hosted on. E.g., there could be a host which runs an OS and has the server run via some software or protocol, so these would be double counted. This is unlikely though, since there are in fact much less in the OS query than the Host query.

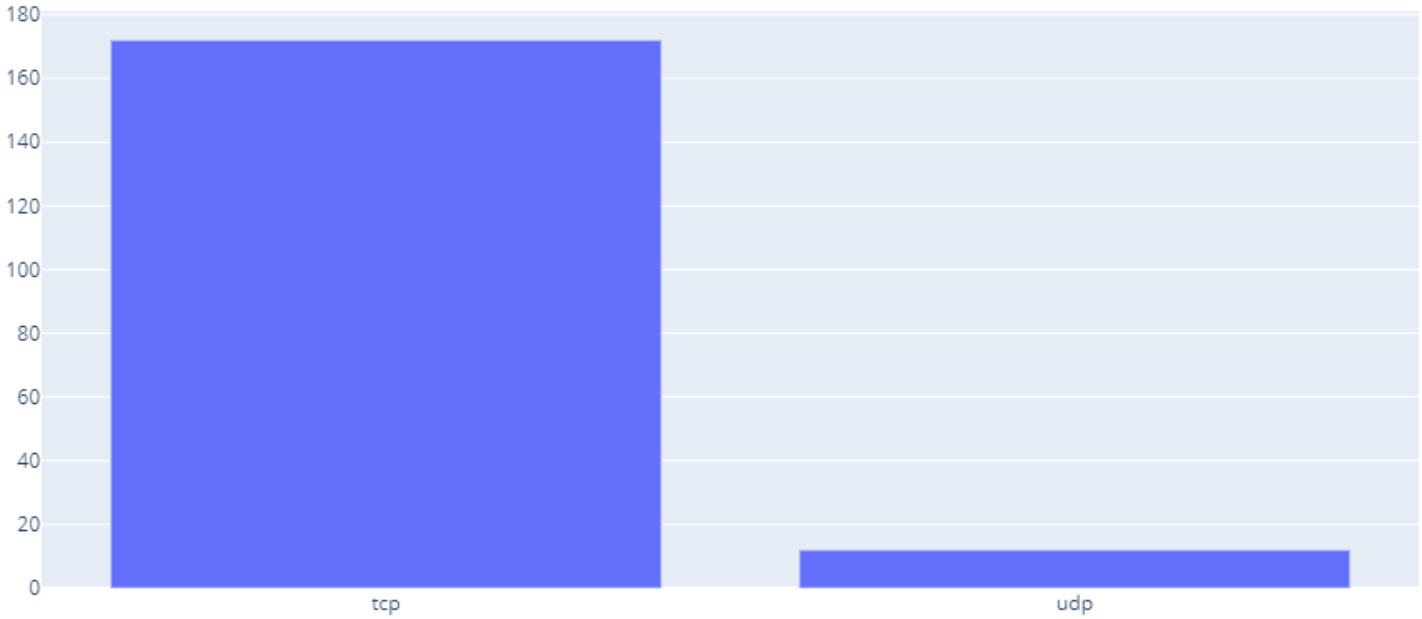

Figure 3: TCP/UDP Protocol Counts

Both protocols work on top of the network layer which IP operates on, and they are both protocols which are used to send packets back and forth between client and server. There are significantly more devices which use TCP since it was designed to be reliable because of error-checking, where as UDP is less commonly used because of it's lack of error checking. I am assuming the devices/systems on NCSUs network which use UDP are in close proximity to one another and probably have wired connections, like larger servers. This can be used to reduce latency, but is unreliable if an error or bit flip occurs.

The raw quantities for the 3 figures above are all in the Appendix at the end

3 Interesting Security Findings

Using Shodan, there were a couple IPs which said there were vulnerabilities. The two I will list are:

1. `152.14.92.108`, or highwind.csc.ncsu.edu
2. `152.14.136.33`, or staff2.lib.ncsu.edu

The links to the Shodin synopses are found **here for highwind.csc** and **here for staff2.lib**. The way I found these was simply searching through Shodin and randomly selecting IPs until

I found some that vulnerabilities. I did not know how to perform this task efficiently, as I did not use the Shodin API for this part - only the web browser search function. I honestly do not understand most of the CVE's listed, but there seem to be a lot for both. Here are 3 examples for each

1. `152.14.92.108`, or highwind.csc.ncsu.edu
   (a) *CVE-2019-0220* A vulnerability found in **Apache HTTP Server 2.4.0 to 2.4.38** that has to do with regular expressions and the parsing of the URL
   (b) *CVE-2020-1927* A vulnerability found in **Apache HTTP Server 2.4.0 to 2.4.41** where a function call is meant to be self-referential, but can be accidentally called to an unexpected URL within the request URL.
   (c) *CVE-2015-6563* A vulnerability found in **sshd in OpenSSH before 7.0** where non-OpenBSD platforms accept extraneous username data in certain requests, which allows for local users to perform impersonation attacks.
2. `152.14.136.33`, or staff2.lib.ncsu.edu
   (a) *CVE-2016-2177* A vulnerability found in **OpenSSL through 1.0.2h** which uses incorrect pointer arithmetic for heap-buffer boundary checks. Remote attackers can use this for denial-of-service attacks.
   (b) *CVE-2017-3167* A vulnerability found in **Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26** where the call of a certain function by third-party modules could cause an authentication phase to be bypassed.
   (c) *CVE-2010-2068* A vulnerability found in **Apache HTTP Server 2.2.9 through 2.2.15, 2.3.4-alpha, and 2.3.5-alpha on Windows, NetWare, and OS/2** where the system may not properly detect time-outs, allowing attackers to potentially receive sensitive responses intended for a different client.

By briefly looking through all this, it seems most of the CVE's for staff2.lib.ncsu.edu correspond to denial-of-service attacks, due to poor programming and bug checking. The CVE's for highwind.csc.ncsu.edu mainly have to do with private information being accessed, due to poor network programming and considering security cases.

4 Identifying External Shadow IT

I thought I was not going to find anything, as I looked for over 2 days for an instance of 'Shadow IT' without finding one. But I think I got lucky because I found one. (If there are many more of these, then maybe I am just bad at looking for them/using Shodin).

There was a URL with the CNAME of `ararat.ces.ncsu.edu` I saw on Shodin. I used the terminal to ping the address and received the response seen in Figure 4

Once I saw this, I realised this IP does not fit into the CIDR blocks discovered in Section 1. I ran this IP through Shodin again to find it has a different ASN than NCSU's 11442, which can be seen in Figure 5. This has an ASN of 14618, which shows there are DNS servers pointing this `ncsu.edu` domain toward a server not within NCSU's CIDR blocks, apparently located in Ashburn, VA. However, I am not entirely sure if this constitutes as Shadow IT, because it is using Amazon AWS and I know many of NCSUs systems use Amazons services. Regardless, this is all I could find for this section.

```
C:\WINDOWS\system32>ping ararat.ces.ncsu.edu

Pinging ararat.ces.ncsu.edu [184.72.209.226] with 32 bytes of data:
Reply from 184.72.209.226: bytes=32 time=33ms TTL=47
Reply from 184.72.209.226: bytes=32 time=29ms TTL=47
Reply from 184.72.209.226: bytes=32 time=33ms TTL=47
Reply from 184.72.209.226: bytes=32 time=31ms TTL=47

Ping statistics for 184.72.209.226:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 29ms, Maximum = 33ms, Average = 31ms

C:\WINDOWS\system32>_
```

Figure 4: Q4 Shadow IT IP

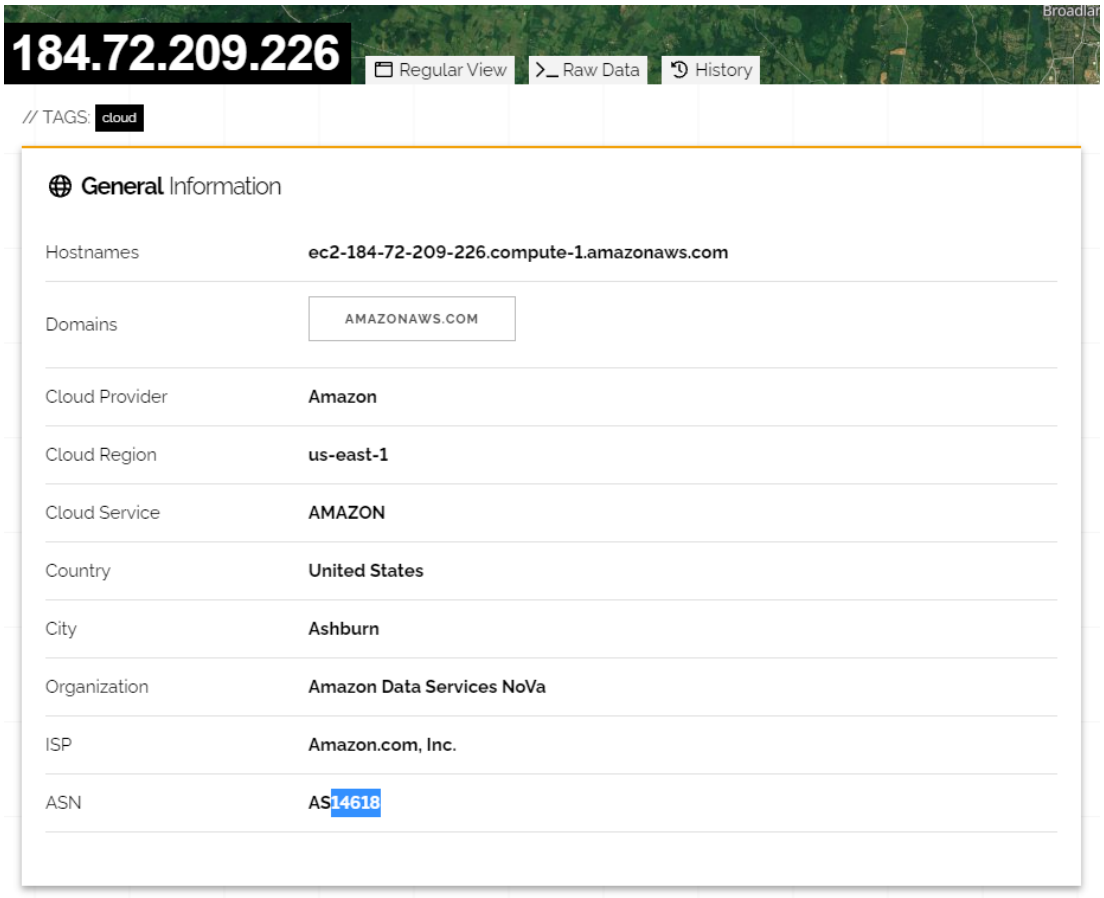

Figure 5: Q4 Shadow IT ASN on Shodin

5 Impact of IPv6

Intuitively, IPv6 should severely stunt the use of these tools. The number of addresses possible are magnitudes greater than the number of ATOMS on planet earth. With traditional computing, a full scan of the IPv6 internet will likely never be done. I was reading about the Opte Project, which is an on-going project used to visualize the internet, which is only possible due to the tiny amount of IPv4 addresses that can easily be accessed and evaluated. Similar, censys.io and Shodin will have a difficult time scanning the IPv6 internet to discover security flaws. Since IPs constantly change, this will be difficult. However, if there are standards like CIDR also implemented in IPv6, then it will be easier to scan given a certain corporation or government agency's IPv6 CIDR block(s) are made public.

6 Appendix

I used Jupyter Notebook for censys.io API and the API JSON output parsing. There is a figure below showing this, but I will not include the listing here. Rather, I will simply compress the Notebook environment and upload it to the Moodle.

1. `{'windows': 32, 'httpd': 17, 'http api': 12, 'nginx': 10, 'php': 10, 'apache': 8, 'linux': 8, 'openssh': 7, 'emweb': 6, 'openssl': 6, 'host': 4, 'mysql': 4, 'mod_perl': 4, 'appweb': 4, 'perl': 4, 'express': 3, 'dsm': 2, 'enterprise linux': 2, 'boa': 2, 'windows ce web server': 2, 'jetty': 2, 'libressl': 2, 'windows ce': 2, 'sentinel keys server': 2, 'postgresql': 1, 'rompager': 1, 'sentinel protection server': 1, 'tomcat': 1, 'airtunes': 1}`

2. `{'windows': 32, 'linux': 5, 'windows ce': 1, 'enterprise linux': 1, 'dsm': 1}`

3. `{'tcp': 172, 'udp': 12}`

In [7]:
```
import json
import numpy as np
```

In [2]:
```
# load the file into 'data'
file = open("censys_hosts_100.json")
data = json.load(file)
```

In [3]:
```
# allocate space and fill with readable/indexable data
ips = []
ports = [[]]
asns = []
descriptions = []
cidrs = []
names = []

idx = 0
for ip_idx in data[0]:
    ips.append(ip_idx["ip"])

    if idx: ports.append([])
    for services in ip_idx["services"]:
        ports[idx].append(services["port"])

    asns.append(ip_idx["autonomous_system"]["asn"])
    descriptions.append(ip_idx["autonomous_system"]["description"])
    cidrs.append(ip_idx["autonomous_system"]["bgp_prefix"])
    names.append(ip_idx["autonomous_system"]["name"])

    idx += 1
```

In [8]:
```
np.unique(cidrs)
```

Out[8]:
```
array(['152.1.0.0/16', '152.14.0.0/16', '152.7.0.0/16'], dtype='<U13')
```

In [15]:
```
print(np.unique(descriptions))
print(np.unique(asns))
print(np.unique(names))
```

```
['NCSU']
[11442]
['NCSU']
```

Figure 6: Q1 Script