Enumerating Large Networks

Assignment and Goals

- Find all open ports on an external class B network
  - 65,534 Possible IP Addresses
  - 65,536 Possible Ports
- Enumerate all services
  - OS and Service Detection
- Find everything
- Accurate results (no false positives)

Step One

- High speed scan to identify live hosts and open ports
  - 65,534 x 65,536 is a big number (4,294,836,224)
- Tools like NMAP or other traditional port scanners are too slow
- MASSCAN!

What is MASSCAN?

- https://github.com/robertdavidgraham/masscan
- “masscan is an Internet-scale port scanner, useful for large scale surveys of the Internet, or of internal networks“
- masscan does this by using asynchronous transmission meaning that it doesn‘t wait for responses or a tcp handshake to be established- it simply blasts out SYN packets and makes a note of any addresses that reply

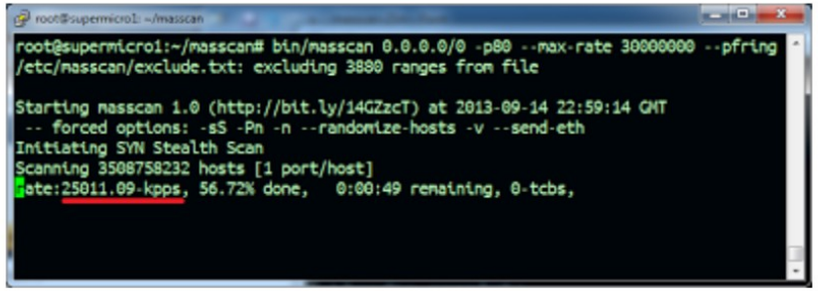
MASSCAN

Errata Security

Advanced persistent cybersecurity

Saturday, September 14, 2013

Masscan: the entire Internet in 3 minutes



- So we'll be done before lunch?
- Well, not quite…
- This is scanning one port using the PF_RING driver at a rate of 25 million packets/second

MASSCAN limitations and considerations

- Using EC2 virtual machines w/enhanced networking
  - can‘t use PF_RING driver so the best we could hope for is 2.5 million packets/second and we‘re using virtual hardware so it‘ll be far less than that
- Internet connection will be a bottleneck
- Target network and host limitations
  - I like my job so I don‘t want to cause any outages by DOS
- Time to test...

MASSCAN Testing

- Used .conf files with varying rates

```
# TARGET SELECTION
range = 192.168.0.0/16
ports = 0-99
rate = 2000
# OUTPUT/REPORTING SETTINGS
output = filename scan_p0-99_rate2000.xml
```

- 500, 1000, 2000 packets/second all had more than 20,000 active ip/port combinations
- 3000 packets/second had less than 1000 active ip/port combinations
- 2000 packets/second it is- time to scan everthing...

MASSCAN all ports

- Used .conf file again

```
# TARGET SELECTION
range = 192.168.0.0/16
ports = 0-65535
rate = 2000
# OUTPUT/REPORTING SETTINGS
output = filename scan_allports_rate2000.xml
```

- 65,534 x 65,536 is still a big number (4,294,836,224)
- This scan took over a month and resulted in a 30 GB XML file with 150,595,212 lines
- Excel definitely can‘t import a 30 GB XML file and even if it could, you can only have 1,048,576 lines in a worksheet- now what?

MASSCAN results

- Python and MySQL to the rescue

masscan_xml_to_csv.py

```
1  replacements = {'<host endtime="':'', '"><address addr="':',', '" addrtype="ipv
2
3  with open('masscan.xml') as infile, open('masscanout.csv', 'w') as outfile:
4      for line in infile:
5          for src, target in replacements.iteritems():
6              line = line.replace(src, target)
7          outfile.write(line)
```

- The 30 GB XML file is now a 9 GB CSV file

MySQL

```
mysql> CREATE TABLE masscan.masscanout (timestamp VARCHAR(20),ip VARCHAR(15),portid  mediumint(8),state VARCHAR(10),reason VARCHAR(255),reasonttl VARCHAR(15));

mysql> show columns from masscanout;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| timestamp | varchar(20)  | YES  |     | NULL    |       |
| ip        | varchar(15)  | YES  |     | NULL    |       |
| portid    | mediumint(8) | YES  |     | NULL    |       |
| state     | varchar(10)  | YES  |     | NULL    |       |
| reason    | varchar(255) | YES  |     | NULL    |       |
| reasonttl | varchar(15)  | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+

mysqlimport  --ignore-lines=5 --fields-terminated-by=, --columns='timestamp,ip,portid,state,reason,reasonttl' --local -u root -p masscan /home/tr3nt/Desktop/masscan/masscanout.csv



mysql> SELECT ip from masscanout where portid = '80' into outfile 'port80.txt';
3 minutes to run

mysql> ALTER TABLE masscanout ADD KEY (portid);

mysql> SELECT ip from masscanout where portid = '80' into outfile 'port80.txt';
14 seconds after adding index
```

- All the masscan data is now in a database and can be queried

MASSCAN lessons learned

- Split up the work if you can- running one EC2 host for four weeks costs about the same as running four hosts for one week
  - MASSCAN includes shard functionality (--shards <x>/<y>)
- Investigate using something other than XML
  - MASSCAN can also output binary, grepable, list, or JSON files

MASSCAN analysis

- 65,520 ports were identified as being in use with at least one host responding and an average of about 2300 hosts responding on each
- 5479 hosts were identified with at least one active port
- All together there are 150,595,207 ip/port combinations
- Unfortunately there are tons of false positives
- Still don‘t know anything about what‘s running on these ports

NMAP

- Now it‘s time to use NMAP for service detection on everything we found open
- 150,595,207 is a slightly smaller number but NMAP is slow so we should split it up
- Query the database in order to split it by port

```
1
2  #script to take list of ports from a file and query mysql database
3  import MySQLdb
4
5  db = MySQLdb.connect("localhost", "root", "notapassword", "masscan")
6  cursor = db.cursor()
7  with open('allports.txt') as infile:
8      for line in infile:
9          line = line.strip ()
10         query = """SELECT ip from masscanout where portid = '%s' into outfile 'port%s.txt'""" % (line,line)
11         lines = cursor.execute(query)
12 db.close()
13
```

NMAP

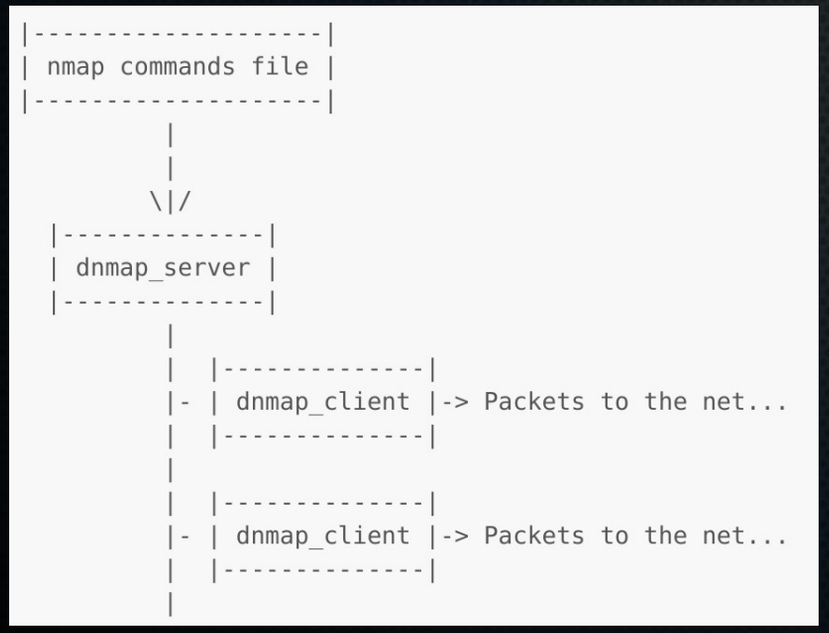
- Now I‘ve got 65,200 files (lists of ip addresses)
- Generate NMAP commands

```
1
2  #script to take list of ports from a file and output shell script with nmap commands
3  with open('allports.txt') as infile, open('nmapcommands.sh', 'w') as outfile:
4      for line in infile:
5          line = line.strip ()
6          newline = 'nmap -Pn -p %s -sV -open -iL port%s.txt -oA port%sresults\n' % (line,line,line)
7          outfile.write(newline)
```

- Using one Amazon EC2 host, I can do about 900 scans a day
- Too slow- let‘s distrubute it with DNMAP

DNMAP

- sourceforge.net/projects/dnmap
- dnmap is a distributed nmap framwork written in python



1- Put some nmap commands on a file like commands.txt
2- Start the dnmap_server
./dnmap_server -f commands.txt
3- Start any number of clients
./dnmap_client -s <server-ip> -a <alias>

The server will start to give nmap commands to the clients and results will be stored on both sides.

DNMAP results

- Using 8 EC2 hosts, I was able to complete 65,200 scans in a little more than a week
- Now we‘ve got 65,200 files to get into a database
- NMAPDB- github.com/argp/nmapdb - “nmapdb parses nmap's XML output files and inserts them into a SQLite database“

NMAPDB

- Create the database and then import an nmap file

```
python2 nmapdb.py -c nmapdb.sql -d nmapverificationscans.db ./xml_results/port1results.xml
```

- Now repeat 65,200 times or...
- script it

```
2  #script to take list of ports from a file and output shell script with nmapdb commands
3  with open('allports.txt') as infile, open('nmaptosqlitecommands.sh', 'w') as outfile:
4      for line in infile:
5          line = line.strip ()
6          newline1 = 'python2 nmapdb.py -c nmapdb.sql -d nmapverificationscans.db ./xml_results/port%sresults.xml\n' % (line)
7          newline2 = 'mv ./xml_results/port%sresults.xml ./processed_results/\n' % (line)
8          outfile.write(newline1)
9          outfile.write(newline2)
```

```
3  python2 nmapdb.py -c nmapdb.sql -d nmapverificationscans.db ./xml_results/port2results.xml
4  mv ./xml_results/port2results.xml ./processed_results/
5  python2 nmapdb.py -c nmapdb.sql -d nmapverificationscans.db ./xml_results/port3results.xml
6  mv ./xml_results/port3results.xml ./processed_results/
7  python2 nmapdb.py -c nmapdb.sql -d nmapverificationscans.db ./xml_results/port4results.xml
8  mv ./xml_results/port4results.xml ./processed_results/
```

Final Results

- 3,841,612 active ip/ports with a large number showing tcpwrapped
  - tcpwrapped in nmap means a full TCP handshake was completed but the connection was closed without receiving any data. It could be that the server on these ports is not allowing our scan host to tallk to it or that a load balancer or firewall is intercepting the connections
- After eliminating the tcpwrapped connections, there were 1876 ip/ports to further analyze
- Why was there a huge number or false positives?
  - Project is still ongoing but the best guess right now is that F5 load balancers are set up with wild card rules and are setting up connections when they don‘t need to be

Lessons Learned and Future Improvements

- Managing all the data during large scans is one of the the hardest parts
  - Over 110 GB of scan results and analysis documents were collected during the project
  - Keep lots of notes, journals of commands that were run, etc.
- Improve the scripts used and standardize the way things are done to streamline the process
  - Not a programmer so lots of things were quick and dirty
  - Could use one database with different tables

Questions or Discussion?

trentkb@gmail.com