Connecting to our Ubuntu node: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html>

Log in to EC2 instance:

#1: us-east: 18.188.229.17 (terminated)

ssh -i confundo1.pem [ubuntu@ec2-18-188-229-17.us-east-2.compute.amazonaws.com](mailto:ubuntu@ec2-18-188-229-17.us-east-2.compute.amazonaws.com)

#2: us-west: 34.218.45.230

ssh -i confundo2.pem ubuntu@ec2-34-218-45-230.us-west-2.compute.amazonaws.com

#3: north-asia: 13.125.227.165 (terminated)

ssh -i confundo3.pem [ubuntu@ec2-13-125-227-165.ap-northeast-2.compute.amazonaws.com](mailto:ubuntu@ec2-13-125-227-165.ap-northeast-2.compute.amazonaws.com)

#4: eu: 18.130.60.154 (terminated)

ssh -i confundo4.pem ubuntu@ec2-18-130-60-154.eu-west-2.compute.amazonaws.com

Bitcoin core directory: /home/ubuntu/confundo/bitcoin-core

#5: south america: 18.231.151.223 (terminated)

ssh -i confundo5.pem ubuntu@ec2-18-231-151-223.sa-east-1.compute.amazonaws.com

Bitcoin core directory: /home/ubuntu/confundo/bitcoin-core

#6: singapore: 137.132.83.82: ask Muoi for creating an account for you

ssh acc@137.132.83.82

Bitcoin core directory: /home/muoi/confundo/bitcoin-core

#7: us-east: 18.221.141.16 (terminated)

ssh -i confundo7.pem [ubuntu@ec2-18-221-141-16.us-east-2.compute.amazonaws.com](mailto:ubuntu@ec2-18-221-141-16.us-east-2.compute.amazonaws.com)

Bitcoin core directory: /home/ubuntu/confundo/bitcoin-0.17.0

#8: us-west: 34.215.105.138 (terminated)

ssh -i confundo8.pem [ubuntu@ec2-34-215-105-138.us-west-2.compute.amazonaws.com](mailto:ubuntu@ec2-34-215-105-138.us-west-2.compute.amazonaws.com)

Bitcoin core directory: /home/ubuntu/confundo/bitcoin-core

scp -i confundo3.pem [ubuntu@ec2-13-125-227-165.ap-northeast-2.compute.amazonaws.com](mailto:ubuntu@ec2-13-125-227-165.ap-northeast-2.compute.amazonaws.com):/home/ubuntu/.bitcoin/debug.log ./confundo\_ec2/debug.log3

scp -i confundo4.pem [ubuntu@ec2-18-130-60-154.eu-west-2.compute.amazonaws.com](mailto:ubuntu@ec2-18-130-60-154.eu-west-2.compute.amazonaws.com):/home/ubuntu/.bitcoin/debug.log ./confundo\_ec2/debug.log4

scp -i confundo5.pem [ubuntu@ec2-18-231-151-223.sa-east-1.compute.amazonaws.com](mailto:ubuntu@ec2-18-231-151-223.sa-east-1.compute.amazonaws.com)::/home/ubuntu/.bitcoin/debug.log ./confundo\_ec2/debug.log5

scp -i confundo7.pem [ubuntu@ec2-18-221-141-16.us-east-2.compute.amazonaws.com](mailto:ubuntu@ec2-18-221-141-16.us-east-2.compute.amazonaws.com):/home/ubuntu/.bitcoin/debug.log ./confundo\_ec2/debug.log7

scp -i confundo8.pem [ubuntu@ec2-34-215-105-138.us-west-2.compute.amazonaws.com](mailto:ubuntu@ec2-34-215-105-138.us-west-2.compute.amazonaws.com):/home/ubuntu/.bitcoin/debug.log ./confundo\_ec2/debug.log8

**Pem file take from Dropbox folder**:

Confundo-attack/EC2/

chmod 400 confundo-x.pem

**Harvest the debug logs:**

EC2 on home folder: ~/.bitcoin/debug.log

On lab node: home/moon/.bitcoin/debug.log

**Notes:**

Do not terminate the EC2 instances, otherwise all data will be gone.

Enable ccache for faster compilation: <https://stackoverflow.com/questions/13929514/how-to-enable-ccache-on-linux>

Already installed on the EC2 instance #1.

**Directory of Original debug.log and peers.dat of Confundo4:**

ubuntu@ec2-18-130-60-154.eu-west-2.compute.amazonaws.com:/home/ubuntu/confundo4\_first\_debug\_peers/

**Directory of Original debug.log and peers.dat of Confundo5:**

ubuntu@ec2-18-231-151-223.sa-east-1.compute.amazonaws.com:/home/ubuntu/confundo5\_first\_debug\_peers/

**Directory of Original debug.log and peers.dat of Confundo6:**

moon@137.132.83.82:/home/moon/confundo6\_first\_debug\_peers/

confundo4:

IPs from: incoming = 1977776, outgoing = 435801

start = 1547617325, end = 1550042520, dur = 2425195

52 incoming connections, 8 outgoing connections

rate: incoming = 1977776/2425195/52 = 0.01568 IP/connection/second,

0.022038 outgoing = 435801/2425195/8 = 0.02246

0.02246 \* 120 \* 8 + 0.01568 \* 120 \* 52

**Awk Codes for our debug.log:**

‘sudo apt-get install gawk’: to use mktime

**\*Awk Code 1 for new debug.log : continuous status of new table with IP address and slot #\* (Work on NEW debug.log only)**

Code 1: cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if($3 == "NEW++"){print $1, $8, $22, $25, "add";}else if($2 == "Add\_," && $3 =="NEW--"){print $1, $5, $18, $21, "delete";}else if($2 == "MakeTried," && $3 == "NEW--"){print $1, $6, $16, $19, "delete";}else if($3 == "NEW++TRIED--"){print $1, $8, $24, $27, "add";}}' | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}' > new\_table\_IP\_status

Code 1: Muoi

Code 1: cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if($3 == "NEW++"){print $8;}else if($2 == "Add\_," && $3 =="NEW--"){print $5;}else if($2 == "MakeTried," && $3 == "NEW--"){print $6;}else if($3 == "NEW++TRIED--"){print $8;}else if($3 == "FAILED"){print $6;}}' > all\_heard\_ip.txt

Format:

/ Time / IP / Bucket Number of New table / Slot Number of New table / "add" or "delete" /

Example:

1544736341 142.93.246.239:8333 508 52 delete

1544736342 183.95.169.179 671 56 add (edited)

**\*Awk Code 2 for new debug.log : number of filled slot - time\* (New Table) (Work on NEW debug.log only)**

Code 2: cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if($3 == "NEW++"){print $1, $8, $22, $25, "add";}else if($2 == "Add\_," && $3 =="NEW--"){print $1, $5, $18, $21, "delete";}else if($2 == "MakeTried," && $3 == "NEW--"){print $1, $6, $16, $19, "delete";}else if($3 == "NEW++TRIED--"){print $1, $8, $24, $27, "add";}}' | awk -F' ' '{if($5 == "add"){a++; print $1, a;}else if($5 == "delete"){a--; print $1, a;}}' | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}' > new\_table\_no\_of\_filledslot

Format:

/ Time / number of filled slot in New table /

Example:

1544736341 22821

1544736342 22822

1544736343 22823

**\*Awk Code 3 for new debug.log : continuous status of tried table with IP and slot\* (Work on NEW debug.log only)**

Code 3: cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if($2 == "MakeTried," && $3 == "TRIED++"){print $1, $5, $12, $15, "add";}else if($2 == "MakeTried," && $3 == "NEW++TRIED--"){print $1, $8, $14, $17, "delete";}}' | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}' > tried\_table\_IP\_status

Format:

/ Time / IP / Bucket Number of Tried table / Slot Number of Tried table / "add" or "delete" /

Example:

1544736341 47.96.132.187 128 4 add

1544736342 173.249.18.164 224 30 add

**\*Awk Code 4 for new debug.log : number of filled slot - time ( tried table )\* (Work on NEW debug.log only)**

Code 4: cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if($2 == "MakeTried," && $3 == "TRIED++"){print $1, $5, $12, $15, "add";}else if($2 == "MakeTried," && $3 == "NEW++TRIED--"){print $1, $8, $14, $17, "delete";}}' | awk -F' ' '{if($5 == "add"){a++; print $1, a;}else if($5 == "delete"){a--; print $1, a;}}' | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}' > tried\_table\_no\_of\_filledslot

Format:

/ Time / number of filled slot in Tried table /

Example:

1544736341 25

1544736342 26

1544736343 27

**\*Awk Code 5 for new debug.log : number of Migration to Tried (only count plus from Code 4) (Work on NEW debug.log only)**

Code 5: cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if($3 == "TRIED++"){a++; print $1, a;}}' | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}' > number\_of\_Migration\_to\_tried

Format:

/ UnixTime / number of Migration from New to Tried /

Example:

1544734628 76

1544734656 77

**\*Awk Code 6 for new debug.log : status of collision\* (Work on NEW debug.log only)**

Code 6: cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if(s == 0){b=0; c=0; d=0; s=1;}if($4 == "collides"){p[$3]=$7; a=10; b++;}else if($7 == "ignored" && p[$3] == $12 && a==9){c++; print $1, "ignored-collision", "inserting ip:", $3, "existing ip:", $12;}else if($8 == "terrible," && p[$14] == $5 && a==9){d++; print $1, "replaced-collision", "inserting ip:", $14, "existing ip:", $5;}a--;}' | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}' > collision\_status

Format:

/ UnixTime / Collision type / inserting ip / existing ip /

Example:

1547617335 ignored-collision inserting ip: 223.89.195.116 existing ip: 208.94.243.45

1549004268 replaced-collision inserting ip: 216.154.54.60 existing ip: 5.138.73.19

**\*Awk Code 7 for new debug.log : incoming number of IP(per 1min)\* (Work on NEW debug.log only)**

Code 7: cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}'| awk -F' ' '{if($4 == "collides"){a++; print $1, a;}else if($3 =="NEW++"){a++; print $1, a;}}'| awk -F' ' '{if(a[start] == 0){a[start] = $1; print $0;}else if($1 == a[start]){print $0;}else if($1 > a[start]){b = $1 - a[start]; for(i=1;i<b;i++){$1 = (a[start]+i); print $1, $2-1;} print (a[start]+b), $2;} a[start] = $1;}' | awk -F' ' '{if(a[start] == 0){a[start] = $1; v=0;}else if($1 - a[start] == **60**){r=$2-v; v=$2;}else if($1 - a[start] == **60** +1){print $1-1, r; a[start] = $1 - 1;}}' > Incoming\_IP\_per\_min

Format:

/UnixTime/number of incoming IP/

Example:

1544752561 3159

1544752621 1974

1544752681 1974

1544752741 1000

1544752801 9

1544752861 3

1544752921 14

1544752981 20

**Bash Code to check whether the table status data from AWK code match with the actual printed table of live node**

moon@137.132.83.82:/home/moon/compare\_bash\_function/check\_equal\_value.sh

**\*Awk Code 8 for new debug.log : analyze each addr msg\*(Work on NEW debug.log only)**

Code 8:

cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if(start == 0){a=1; start=1;}else if($4=="Receiving" && c[a] != 0 && d[a] != 0){start =2; if(b[a] == 0){b[a]=0;}if(f[a]==0){f[a]=0;}if(g[a]==0){g[a]=0;}if(g[a]-f[a]+b[a] == e[a]){h[a]="true";}else{h[a]="false";}if(((g[a]-f[a]+b[a])-e[a])<1){print d[a], c[a], e[a], b[a], g[a], f[a], h[a];} a++; split($7, first, ":"); if(length(first[3])==0 && length(first[4])==0 && length(first[5])==0 && length(first[6])==0 && length(first[7])==0 && length(first[8])==0){c[a] = first[1];}else{c[a]=(first[1]":"first[2]":"first[3]":"first[4]":"first[5]":"first[6]":"first[7]":"first[8]);} d[a]=$1; e[a]=$12;}else if($2=="Add\_," && $3=="NEW++"){b[a]++;}else if($2=="Add\_," && $3=="NEW--"){f[a]++;}else if($2=="Add\_," && $4 == "collides"){g[a]++;}else if($2=="Add\_," && $9 == "same"){g[a]++;}else if($4 == "Receiving" && start == 1){split($7, first, ":"); if(length(first[3])==0 && length(first[4])==0 && length(first[5])==0 && length(first[6])==0 && length(first[7])==0 && length(first[8])==0){c[a] = first[1];}else{c[a]=(first[1]":"first[2]":"first[3]":"first[4]":"first[5]":"first[6]":"first[7]":"first[8]);}d[a]=$1; e[a]=$12;}}' | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}' > analyze\_each\_addr\_msg

Format:  
Unix Time / IP / Number of IP in ADDR / Number of inserted IP / Number of collision / Number of eviction / whether the number matches

Example:

1547617837 3.0.70.221 1000 886 113 0 false

1547617848 3.0.70.221 1 1 0 0 true

1547617868 66.188.249.136 1 0 0 0 false

**\*Awk Code 9 for new debug.log: outgoing connection incoming connection addr msg analysis(raw data)\* (Work on NEW debug.log only)**

Code 9:

cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if($5=="outgoing"){split($7, ip, "="); outgoing[ip[2]]=1; print $1, "outgoing\_connection\_made", ip[2];}else if($5=="incoming"){split($7, cip, "="); incoming[cip[2]]=1; print $1, "incoming\_connection\_made", cip[2];}else if($3=="GETADDR," && $4=="Sending"){split($7, gip, ","); getaddr[gip[1]]=1;}else if($3=="ADDR," && $4=="Receiving"){split($7, aip, ":"); if(getaddr[aip[1]":"aip[2]]==0){getaddr[aip[1]":"aip[2]]=0;} if(outgoing[aip[1]":"aip[2]] == 1){addr[aip[1]":"aip[2]]++; addrip[aip[1]":"aip[2]] = $9 + addrip[aip[1]":"aip[2]]; print $1, "outgoing", aip[1]":"aip[2], addr[aip[1]":"aip[2]], addrip[aip[1]":"aip[2]], getaddr[aip[1]":"aip[2]];getaddr[aip[1]":"aip[2]] = 0;}else if(incoming[aip[1]":"aip[2]] == 1){addr2[aip[1]":"aip[2]]++; addrip2[aip[1]":"aip[2]] = $9 + addrip2[aip[1]":"aip[2]]; print $1, "incoming", aip[1]":"aip[2], addr2[aip[1]":"aip[2]], addrip2[aip[1]":"aip[2]], getaddr[aip[1]":"aip[2]];getaddr[aip[1]":"aip[2]] = 0;}}}' | sort -k3,3 -k4n > outgoing\_incoming\_connection\_addr\_msg\_analysis

Example: 4 formats

First format:

2019-01-03T01:21:14Z incoming\_connection\_made 1.132.106.80:1313

->This shows the node had successfully made incoming connection to 1.132.106.80:1313

Second format:

2019-01-03T01:35:40Z incoming 1.132.106.80:1313 14 35 1

-> / Time / type of connection / IP address / number of ADDR msg / total number of IPs / whether the GETADDR msg was sent (1: Sent 0: not Sent)/

Third format:

2018-12-21T12:31:28Z outgoing\_connection\_made 193.77.34.106:8333

-> This shows the node had successfully made outgoing connection to 193.77.34.106:8333

Fourth format:

2018-12-21T15:33:09Z outgoing 193.77.34.106:8333 184 1397 1

->/ Time / type of connection / IP address / number of ADDR msg / total number of IPs / whether the GETADDR msg was sent (1: Sent 0: not Sent)/

**\*Awk Code 10 for new debug.log: outgoing connection incoming connection addr msg analysis(filtered data)\*(May take some time to run) (Work on NEW debug.log only)**

\*Warning: This code should be ran after the code 9 is done, and also adding “final” to the end of the output file of code 9!!!

Recommended Command:

**echo final >> file\_name.txt**

Code 10:

cat file\_name\_from\_code9.txt | awk -F' ' '{if($2 == "incoming\_connection\_made"){k++;}else if($2 == "outgoing\_connection\_made"){o++;}else if($2 == "outgoing"){oaddr[$3]=$4; oip[$3]=$5;}else if($2 == "incoming"){iaddr[$3]=$4; iip[$3]=$5;}else if($1 == "final"){sumaddr=0; sumip=0; for(j in oaddr){sumaddr+=oaddr[j]; sumip+=oip[j]; print "outgoing: ", j, "number\_of\_addr: ",oaddr[j], "number\_of\_total\_ip: ", oip[j];} sumaddr2=0; sumip2=0; for(y in iaddr){sumaddr2+=iaddr[y]; sumip2+=iip[y]; print "incoming: ", y, "number\_of\_addr: ",iaddr[y], "number\_of\_total\_ip: ", iip[y];}}}' > addr\_msg\_per\_IP

Format:

/ connection type / IP address / cumulative number of addr msg sent / cumulative total number of ip sent /

Example:

outgoing: 193.77.34.106:8333 number\_of\_addr: 32 number\_of\_total\_ip: 231

Incoming: 1.132.106.80:1313 number\_of\_addr: 245 number\_of\_total\_ip: 1236

**\*Awk Code 11 for new debug.log: outgoing connection incoming connection addr msg analysis(Evaluated data)\*(May take some time to run) (Work on NEW debug.log only)**

\*Warning: This code should be ran after the code 9 is done, and also adding “final” to the end of the output file of code 9!!!

Recommended Command:

**echo final >> file\_name.txt**

Code 11:

cat file\_name\_from\_code9.txt | awk -F' ' '{if($2 == "incoming\_connection\_made"){k++;}else if($2 == "outgoing\_connection\_made"){o++;}else if($2 == "outgoing"){oaddr[$3]=$4; oip[$3]=$5;}else if($2 == "incoming"){iaddr[$3]=$4; iip[$3]=$5;}else if($1 == "final"){sumaddr=0; sumip=0; for(j in oaddr){sumaddr+=oaddr[j]; sumip+=oip[j];} sumaddr2=0; sumip2=0; for(y in iaddr){sumaddr2+=iaddr[y]; sumip2+=iip[y];} print "number\_of\_outgoing:", o, "total\_number\_of\_ADDR\_msg:", sumaddr, "total\_number\_of\_IP:", sumip; print "number\_of\_incoming:", k, "total\_number\_of\_ADDR\_msg:", sumaddr2, "total\_number\_of\_IP:", sumip2;}}' > total\_numebr\_of\_ADDR-IP\_for\_each\_type\_of\_connection

Example:

number\_of\_outgoing: 135 total\_number\_of\_ADDR\_msg: 171423 total\_number\_of\_IP: 426685

number\_of\_incoming: 43793 total\_number\_of\_ADDR\_msg: 628339 total\_number\_of\_IP: 1817726

/ number of connection in each type / total number of ADDR in each type / total number of IP from each type /

**\*Awk Code 12 for new debug.log: incoming connection outgoing connection addr msg analysis(filtered data) with number of inserted slots (Work on NEW debug.log only)**

Before we run the code, we have to merge the result from code 8 with code 10.

**First, add a line to the result of code 8.**

**Command:**

***echo final >> file\_name\_code\_8.txt***

**Second, merge the result of code 8 and code 10 together with following code.**

**Command:**

***cat file\_name\_code\_8.txt file\_name\_code\_10.txt > file\_name\_code\_12.txt***

**Third, run the code.**

Code 12:

cat code12\_sub.txt | awk -F' ' '{if($1 == "final"){start=1;}else if(start==0){sum[$2]+=$4;}else if(start == 1){split($2, ip, ":"); if($1 =="outgoing:" || $1 == "incoming"){print $0, "number\_of\_ip\_actually\_inserted:",sum[ip[1]];}}}' > addr\_msg\_per\_IP\_detailed\_version

Example:

outgoing: 13.251.13.18:8333 number\_of\_addr: 2 number\_of\_total\_ip: 1001 number\_of\_ip\_actually\_inserted: 140

outgoing: 88.99.187.170:8333 number\_of\_addr: 2 number\_of\_total\_ip: 1001 number\_of\_ip\_actually\_inserted: 113

/ connection type where ADDR came from / IP address / cumulative number of addr msg sent / cumulative number of ip sent / cumulative number\_of\_ip\_actually\_inserted /

**\*Awk Code 13: distinguish IP in NEW table(type: DNS, outgoing, incoming) (Work on NEW debug.log only)**

Wanrning: this code can only work with the combiantion of debug.log and the result from code 10. There are few steps for this.

**First, add a line to the result of code 10**

**Command:**

***echo final >> file\_name\_code\_10.txt***

**Second, merge the debug.log and the result of code 10 together with following code.**

**Command:**

***cat file\_name\_code\_10.txt debug.log > file\_name\_code\_13-1.txt***

**Third, run the code to the previous result file:**

Code 13-1:

cat file\_name\_code\_13.txt | awk -F' ' '{if($1 == "final"){start=1;}else if(start==0){if($1 == "outgoing:"){result[$2] = "outgoing";}else if($1 == "incoming:"){result[$2] = "incoming";}}else if(start == 1){if($5 == "ADDR"){split($7,ip,":"); print $0, result[ip[1]":"ip[2]];}else if($3 == "NEW++"){print $0;}else if($5 == "DNS"){print $0;}}}' | awk -F' ' '{if($16 == "incoming"){split($7, ip, ":"); state="incoming"; previous\_source = ip[1]":"ip[2];}else if($5 == "DNS"){state ="DNS";}else if($16 == "outgoing"){split($7, ip, ":"); state="outgoing"; previous\_source = ip[1]":"ip[2];}else if($3== "NEW++" ){print state, $8, " ", previous\_source;}}' > inserted\_ip\_state.txt

**Fourth, add a line to the result of code 13:**

**Command:**

***echo final >> inserted\_ip\_state.txt***

**Fifth, merge previous result with the printed New table:**

**Command:**

***cat file\_name\_code\_10.txt NT9.16:05:34.904.txt > file\_name\_code\_13-2.txt***

**Sixth, run this code to the merged file:**

Code 13-2:

cat 13-2result.txt | awk -F' ' '{if($1 == "final"){start=1;}else if(start==0){if($1 == "outgoing"){result[$2] = "outgoing";}else if($1 == "incoming"){result[$2] = "incoming";}}else if($1 == “DNS”){result[$2] = "DNS";}else if(start == 1){print $0, result[$1];}}' > source\_type\_of\_IP\_in\_New\_table

**Example:**

**206.189.88.148 outgoing**

**31.173.240.52 incoming**

**/ IP / Type it was from /**

**\*Awk Code 14: real time number of outgoing and incoming connection (Only work for Newest debug.log)**

**Code 14:**

cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}'| awk -F' ' '{if($5 == "outgoing"){split($7, ip, "="); out[ip[2]]=1; print $1, ip[2], "o-connected", "outgoing:", $17, "incoming:", $12;}else if($5 == "incoming"){split($7, ip, "="); inc[ip[2]]=1; print $1, ip[2], "i-connected", "outgoing:", $17, "incoming:", $12;}else if($6 == "disconnected"){split($3, dip, ","); if(out[dip[1]]==1){print $1, dip[1], "o-disconnected", "outgoing:", $16, "incoming:", $11;}else if(inc[dip[1]]==1){print $1, dip[1], "i-disconnected", "outgoing:", $16, "incoming:", $11;}}}' > number\_of\_incoming\_outgoing\_connection.txt

Example:

2019-02-01T07:26:41Z 37.228.129.29:36516 i-connected outgoing: 8 incoming: 51

2019-02-01T07:26:42Z 37.228.129.29:36516 i-disconnected outgoing: 8 incoming: 51

2019-02-01T07:26:42Z 188.166.69.73:32897 i-disconnected outgoing: 8 incoming: 49

Time / IP / change in connection (i : incoming, o : outgoing) / number of outgoing connection / number of incoming connection

**\*Awk Code 15: nNow-nTime code (Work for all peers.dat) (nNow Differ by TIME!!!!)**

**Code 15:**

First get ready for the two table files from peers.dat of the live node.

1. **Two table: NN and TT**

Second run this code to debug.log and get the result.

Code: cat debug.log | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}' | awk -F' ' '{if($3 == "NEW++" && $2 == "Add\_,"){print $8, $1}}' > code15-1

Third add one line to “code15-1”

***echo final >>* code15-1**

Third **merge** bitpeers **result with the printed table(choose one that you want to see the result):**

***cat* code15-1 *NT14.17:50:45.040.txt > file\_name\_code\_15.txt***

Fourth run the code:

cat file\_name\_code\_15.txt | awk -F' ' '{if($1 == "final"){start = 1;}else if(start == 0){ time[$1] = $2;}else if(start == 1){print $1, 1550111773-time[$1];}}' > C4\_how\_old\_is\_the\_IP\_in\_New\_table

Format:

/ IP / nNow-nTime /

Example:

1.189.234.226 2321262

104.231.138.32 2323110

125.119.99.93 2250920

BELOW CODE IS NOT ACCURATE

**~~\*Awk Code 16: : IP new/tried nNow nTime nLastSuccess nAttempts (Work for all peers.dat) (nNow Differ by TIME!!!!)~~**

**~~Code 16:~~**

**~~DOWNLOAD BITPEER! Follow the instruction in below website~~**

[**~~https://bitcoin.stackexchange.com/questions/75324/how-to-dump-peers-dat-file-to-a-plain-text-file~~**](https://bitcoin.stackexchange.com/questions/75324/how-to-dump-peers-dat-file-to-a-plain-text-file)

~~First get ready for the peers.dat and get two files from it.~~

1. **~~bitpeers~~** ~~--filepath ./peers.dat --format text > peers.txt~~
2. **~~Two table: NN and TT~~**

~~Second add one line to TT~~

***~~echo final >> TT14.17:50:45.040.txt~~***

~~Third~~ **~~merge~~** ~~bitpeers~~ **~~result with the printed Tried table:~~**

***~~cat TT14.17:50:45.040.txt~~*** ~~peers.txt~~ ***~~> file\_name\_code\_15.txt~~***

~~Fourth run the code:~~

~~cat file\_name\_code\_15.txt | awk -F' ' '{if($1 == "final"){start = 1;}else if(start == 0){ip[$1]=1;}else if(start == 1){if($1 == "Time:"){time = $2;}else if($1 == "IP:"){split($2, sip, ":");}else if($1 == "LastSuccess:"){LastSuccess = $2;}else if($1 == "Attempts:"){Attempts=$2; if(length(sip[3])==0 && length(sip[4])==0 && length(sip[5])==0 && length(sip[6])==0 && length(sip[7])==0 && length(sip[8])==0){if(ip[sip[1]]==1){print sip[1], "TRIED",~~ **~~1550111773~~**~~, time, LastSuccess, Attempts;}else{print sip[1], "NEW",~~ **~~1550111773~~**~~, time, LastSuccess, Attempts;}}else{nip = (sip[1]":"sip[2]":"sip[3]":"sip[4]":"sip[5]":"sip[6]":"sip[7]":"sip[8]);if(ip[nip]==1){print nip, "TRIED",~~ **~~1550111773~~**~~, time, LastSuccess, Attempts;}else{print nip, "NEW",~~ **~~1550111773~~**~~, time, LastSuccess, Attempts;}}}}}' > every\_detail\_of\_IP\_in\_New\_Tried\_table~~

~~Example:~~

**~~IP / new/tried / nNow / nTime / nLastSuccess / nAttempts~~**

**\*Awk Code 17: : length of connection (Work for all Debug.log)**

**First:**

**echo final >> debug.log**

**Code 17:**

cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}' | awk -F' ' '{if($1!="final"){current\_time = $1;}if($6 == "connection:"){split($7, ip, "="); con[ip[2]]=1; contime[ip[2]] = $1; repeat\_time[ip[2]]=0; if($5 == "outgoing"){contype[ip[2]]="outgoing";}else if($5 == "incoming"){contype[ip[2]]="incoming";}}else if($6 == "disconnected"){split($3, dip, ","); if(con[dip[1]]==1){repeat\_time[dip[1]]++; print dip[1], contype[dip[1]], "connected\_time:", contime[dip[1]], "disconnected\_time:", $1, "length\_of\_connection:", $1-contime[dip[1]], repeat\_time[dip[1]]; }}else if($2 == "final"){for(i in repeat\_time){if(repeat\_time[i]==0){print i, contype[i], "connected\_time:", contime[i], "disconnected\_time:", current\_time, "length\_of\_connection:", current\_time-contime[i], repeat\_time[i];}}}}' | sort -k1,1 -k9nr | awk -F' ' '{if(not\_first[$1"\_"$4]==1){}else if(not\_first[$1"\_"$4]==0){rep\_count[$1"\_"$4]++; if(rep\_count[$1"\_"$4]>1){not\_first[$1"\_"$4]=1;}else{print $0;} }}' > length\_of\_connection

Example:

116.203.55.105:54849 incoming connected\_time: 1547494442 disconnected\_time: 1547494443 length\_of\_connection: 1

136.243.139.96:8448 incoming connected\_time: 1547494454 disconnected\_time: 1547494469 length\_of\_connection: 15

IP / Type of Connection / timestamp when the connection is made / timestamp when the connection is disconnected / Length of Connection

**\*Awk Code 18: code to make all the lines to have timestamp for debug.log**

**Code 18:**

cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}'

**\*Awk Code 19: type of ADDR per day(Code 22 is better code, so USE CODE 22)  
Code19:**

cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if($5=="outgoing"){split($7, ip, "="); outgoing[ip[2]]=1; addrc[ip[2]]=1;}else if($5=="incoming"){split($7, cip, "="); incoming[cip[2]]=1; addrc[cip[2]]=1;}else if($3=="GETADDR," && $4=="Sending"){split($7, gip, ","); if(length(getaddr[gip[1]]) == 0){getaddr[gip[1]]=1;}else{getaddr[gip[1]]++;}}else if($6 == "disconnected"){split($3, jip, ","); getaddr[jip[1]]=0; addrc[jip[1]]=0;}else if($3=="ADDR," && $4=="Receiving"){split($7, aip, ":"); if(getaddr[aip[1]":"aip[2]]==0){getaddr[aip[1]":"aip[2]]=0;} if(outgoing[aip[1]":"aip[2]] == 1 && addrc[aip[1]":"aip[2]] == 1){addr[aip[1]":"aip[2]]++; addrip[aip[1]":"aip[2]] = $9 + addrip[aip[1]":"aip[2]]; print $1, "outgoing", aip[1]":"aip[2], addr[aip[1]":"aip[2]], addrip[aip[1]":"aip[2]], getaddr[aip[1]":"aip[2]];}else if(incoming[aip[1]":"aip[2]] == 1 && addrc[aip[1]":"aip[2]] == 1){addr2[aip[1]":"aip[2]]++; addrip2[aip[1]":"aip[2]] = $9 + addrip2[aip[1]":"aip[2]]; print $1, "incoming", aip[1]":"aip[2], addr2[aip[1]":"aip[2]], addrip2[aip[1]":"aip[2]], getaddr[aip[1]":"aip[2]];}}}' | awk -F' ' '{split($1, date, "T"); split(date[1], day, "-"); if(start==0){current\_day = day[3]; print "-------------------------------------------------------------------------"; print " \*\*\*\*\*\*\*", date[1], "\*\*\*\*\*\*\* "; print $0; start=1;}else if(current\_day == day[3]){print $0;}else if(current\_day == day[3] - 1){print "-------------------------------------------------------------------------"; print " \*\*\*\*\*\*\*", date[1], "\*\*\*\*\*\*\* "; print $0; current\_day = day[3];}}' > addr\_analysis\_per\_day

Example:

-------------------------------------------------------------------------

\*\*\*\*\*\*\* 2019-01-16 \*\*\*\*\*\*\*

2019-01-16T05:53:58Z outgoing 131.188.40.191:8333 1 1 1

2019-01-16T05:53:58Z outgoing 131.188.40.191:8333 1 1 1

Time / Type of Connection / IP / number of ADDR sent by the IP / number of IP inside ADDR / whether the GETADDR msg was sent (1: Sent 0: not Sent)/

**\*Awk Code 20: evaluate whether incoming and outgoing accept incoming connection or not(Work for both old and new version of debug.log unless there is enough superset data in raichu for the specific date)**

**First, run this code to get the result.**

**Code20:**

cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if($5=="outgoing"){split($7, ip, "="); outgoing[ip[2]]=1; addrc[ip[2]]=1; print $1, "outgoing\_connection\_made", ip[2];}else if($5=="incoming"){split($7, cip, "="); incoming[cip[2]]=1; addrc[cip[2]]=1; print $1, "incoming\_connection\_made", cip[2];}}' | awk -F' ' '{split($3, sip, ":"); if(length(sip[3])==0 && length(sip[4])==0 && length(sip[5])==0 && length(sip[6])==0 && length(sip[7])==0 && length(sip[8])==0){print $1, $2, sip[1];}else{nip = (sip[1]":"sip[2]":"sip[3]":"sip[4]":"sip[5]":"sip[6]":"sip[7]":"sip[8]); print $1, $2, nip;}}' > Code20\_result.txt

**Second copy per\_day\_superset\_of\_btc\_ipaddr directory from raichu,**

scp -r moon@raichu.d2.comp.nus.edu.sg:/data/BGP\_LOG/bitnodes\_data/per\_day\_superset\_of\_btc\_ipaddr ./

**Third run the py file,**

python check\_acceptance\_of\_incoming\_connection.py confundo1

**Python code:**

import sys

confundo\_name = sys.argv[1]

code20\_path\_dir = './' + confundo\_name + '/Code20\_result.txt'

live\_node\_path\_dir\_sub = './per\_day\_superset\_of\_btc\_ipaddr/'

result\_path\_dir = './' + confundo\_name + '/connection\_check\_acceptance\_of\_incoming'

incoming = 0

f1 = open(code20\_path\_dir, 'r')

for line in f1:

space\_divide = line.split()

print space\_divide[0] + space\_divide[1] + space\_divide[2]

IP = space\_divide[2]

time\_divide = space\_divide[0].split("T")

date\_divide = time\_divide[0].split("-")

live\_node\_path\_dir\_origin = live\_node\_path\_dir\_sub + str(date\_divide[0]) + "-" + str(date\_divide[1]) + "/"

f2 = open(live\_node\_path\_dir\_origin + "superset\_" + str(date\_divide[0]) + str(date\_divide[1]) + str(date\_divide[2]) + '.txt', 'r')

for line2 in f2:

IP2 = line2.split()

if IP == IP2[0] :

print IP + "==" + IP2[0]

incoming = 1

print incoming

if incoming == 1 :

with open(result\_path\_dir + ".txt", 'a') as file\_out:

file\_out.write(space\_divide[0] + " " + space\_divide[1] + " " + space\_divide[2] + " 1\n")

elif incoming == 0 :

with open(result\_path\_dir + ".txt", 'a') as file\_out:

file\_out.write(space\_divide[0] + " " + space\_divide[1] + " " + space\_divide[2] + " 0\n")

incoming = 0

print incoming

**Example:**

2018-11-18T15:46:23Z outgoing\_connection\_made 192.95.29.22 1

2018-11-18T15:46:25Z incoming\_connection\_made 162.218.65.238 0

Time Stamp / Type of Connection / IP / whether it receive incoming connection

**\*Awk Code 21: feeler connection evaluation code (Work for both old and new version of debug.log)**

**Code21:**

cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if(start == 0){time\_limit=2; start = 1;}if($3 == "Making" && $4 == "feeler" && time\_limit == 1){print previous, 0; previous=$0; ip[$7]=$0; repeat[$7] = 3;}else if($3 == "Making" && $4 == "feeler" && time\_limit == 2){previous=$0; ip[$7]=$0; time\_limit=1; repeat[$7] = 3;}else if($5 == "being" && $6 == "disconnected"){split($3, sip, ","); if(length(ip[sip[1]])!=0 && repeat[sip[1]] == 3){print ip[sip[1]], 1; time\_limit=2; repeat[sip[1]] = 1;}}}' | awk -F' ' '{print $1, "Feeler", $7, $8;}' > Feeler\_connection\_analysis

Example:

2018-12-06T16:36:01Z Feeler 35.196.103.175:8333 0

2018-12-06T16:36:12Z Feeler 46.172.90.10:8333 0

2018-12-06T16:36:21Z Feeler 45.50.46.221:8333 1

2018-12-18T19:02:55Z Feeler 142.93.141.201:8333 1

/ Time / "Feeler" / IP / whether it is Successful(0: unsuccessful, 1: successful)

**\*Awk Code 22: type of ADDR per day(deleting cumulative of # of IP in ADDR and also deleting the total ADDR sent by the IP, also one solicited ADDR per GETADDR) (Only work for Newest Debug.log C4, C5, C6, C7, C8)**

**Code22:**

cat debug.log | awk -F' ' '{split($1, filt, "0"); if(filt[1] == 2){current\_time = $1; print $0;}else if(filt[1] != 2){print current\_time, $0;}}' | awk -F' ' '{if($5=="outgoing"){split($7, ip, "=");outgoing[ip[2]]=1;addrc[ip[2]]=1;}else if($5=="incoming"){split($7, cip, "=");incoming[cip[2]]=1;addrc[cip[2]]=1;}else if($3=="GETADDR," && $4=="Sending"){split($7, gip, ","); getaddr[gip[1]] = 1;response\_limit[gip[1]] = 1;}else if($6 =="disconnected"){split($3, jip, ",");addrc[jip[1]]=0;response\_limit[jip[1]] = 0;}else if($3=="ADDR," && $4=="Receiving"){split($7, aip, ":");if(outgoing[aip[1]":"aip[2]] == 1 && addrc[aip[1]":"aip[2]] == 1){addrip[aip[1]":"aip[2]] = $9;if(response\_limit[aip[1]":"aip[2]] <= 0){print $1, "outgoing", aip[1]":"aip[2], addrip[aip[1]":"aip[2]], 0;}else if(response\_limit[aip[1]":"aip[2]] > 0 && getaddr[aip[1]":"aip[2]] == 1){print $1, "outgoing", aip[1]":"aip[2], addrip[aip[1]":"aip[2]], 1;}}else if(incoming[aip[1]":"aip[2]] == 1 && addrc[aip[1]":"aip[2]] == 1){addrip[aip[1]":"aip[2]] = $9;if(response\_limit[aip[1]":"aip[2]] <= 0){print $1, "incoming", aip[1]":"aip[2], addrip[aip[1]":"aip[2]], 0;}else if(response\_limit[aip[1]":"aip[2]] > 0 && getaddr[aip[1]":"aip[2]] == 1){print $1, "incoming", aip[1]":"aip[2], addrip[aip[1]":"aip[2]], 1;}}response\_limit[aip[1]":"aip[2]] = response\_limit[aip[1]":"aip[2]] - 1;}}' | awk -F' ' '{split($1, date, "T"); if(start==0){current\_day = date[1]; print "-------------------------------------------------------------------------"; print " \*\*\*\*\*\*\*", date[1], "\*\*\*\*\*\*\* "; print $0; start=1;}else if(current\_day == date[1]){print $0;}else if(current\_day != date[1]){print "-------------------------------------------------------------------------"; print " \*\*\*\*\*\*\*", date[1], "\*\*\*\*\*\*\* "; print $0; current\_day = date[1];}}' > addr\_analysis\_per\_day.txt

Example:

-------------------------------------------------------------------------

\*\*\*\*\*\*\* 2019-01-16 \*\*\*\*\*\*\*

2019-01-16T05:53:58Z outgoing 131.188.40.191:8333 1 1

2019-01-16T05:54:10Z outgoing 173.249.17.103:8333 1000 0

Format:

Time / Type of Connection / IP / number of IP inside the ADDR / whether the GETADDR msg was sent (1: Sent 0: not Sent)/

**\*Awk Code 23: the number of unsolicited/solicited addr messages received per day (Only work for Newest Debug.log C4, C5, C6, C7, C8)**

Add “final” to the result of Code 22

echo final >> addr\_analysis\_per\_day.txt

Code:

awk -F' ' '{if($5==1){s++;}else if($5==0){u++;} if(length($2)==0){i++; if(i >= 2){print s, u; s=0; u=0;}}}' > shorten\_C\_addr\_analysis\_per\_day.txt

Format:

/ number of solicited / number of unsolicited /

112 672314

**\*Awk Code 24: Code to get number of live IP in New / Tried (Only work for Newest Debug.log C4, C5, C6, C7, C8)**

First, download the lasted peers.dat of the live node, and remember the timestamp of the time you’ve downloaded the peers.dat.

Second, goto

[**https://bitnodes.earn.com/api/v1/snapshots/**](https://bitnodes.earn.com/api/v1/snapshots/)

And scrawl ip from the snapshot (nearest timestamp that you remember from last step) and save the file. Let’s yes the file name is “live\_ip\_list”.

Third, open the VM(with the source for printing out the table).

Fourth, replace the peers.dat of VM into or live node peers.dat.

Fifth, run the bitcoin-core

Sixth, stop the bitcoin-core as long as you see first two files are generated inside /home/nameoftheVM/table/

Seventh, add “final” on the last line of “live\_ip\_list”

**echo final >> live\_ip\_list**

Eighth, merge the “live\_ip\_list” file and one of the table file(if you want to see live IP in New then select NT file. Verse versa) together in this sequence.

**cat live\_ip\_list NT > pre-result**

Ninth, run this code to the “pre-result” file.

**Code:**

cat pre-result | awk -F' ' '{if($1=="final"){start=1;}else if(start==0){re[$1]=1;}else if(start==1){if(re[$1]==1){a++; print $1, a;}}}' > list\_of\_live\_IP

Format:

Live IP / cumulative number of live IP

Example:

80.229.14.115 7181

2001:41d0:203:1d30:: 7182

47.98.47.181 7183

2001:8d8:978:9700::a9:7851 7184

67.180.158.179 7185

**\*Awk Code 25 for old debug.log : number of filled slot - time ( tried table )\* (Work on OLD debug.log only)**

Code 25: cat debug.log | awk -F'Z ' '{split($1, adate, "T"); split(adate[1], date, "-"); split(adate[2], time, ":"); datetime = (date[1]" "date[2]" "date[3]" "time[1]" "time[2]" "time[3]" 0"); $1=mktime(datetime)+28800; print $0;}' | awk -F' ' '{if($2 == "MakeTried,"){print $1, $8;}}' > tried\_table\_no\_of\_filledslot

Format:

/ Time / number of filled slot in Tried table /

Example:

1544736341 25

1544736342 26

1544736343 27