Stopping Scanners Early and Quickly

Quick questions ...

How many people here block scanners ?

How many not block scanners ?

Block other things ?

No blocking at all ?

Block everything ?

Insight into some numbers - Meaningful or Not ?

How many connections / day - 160-180M

How many LBNL IPs hit / day - Both class B’s

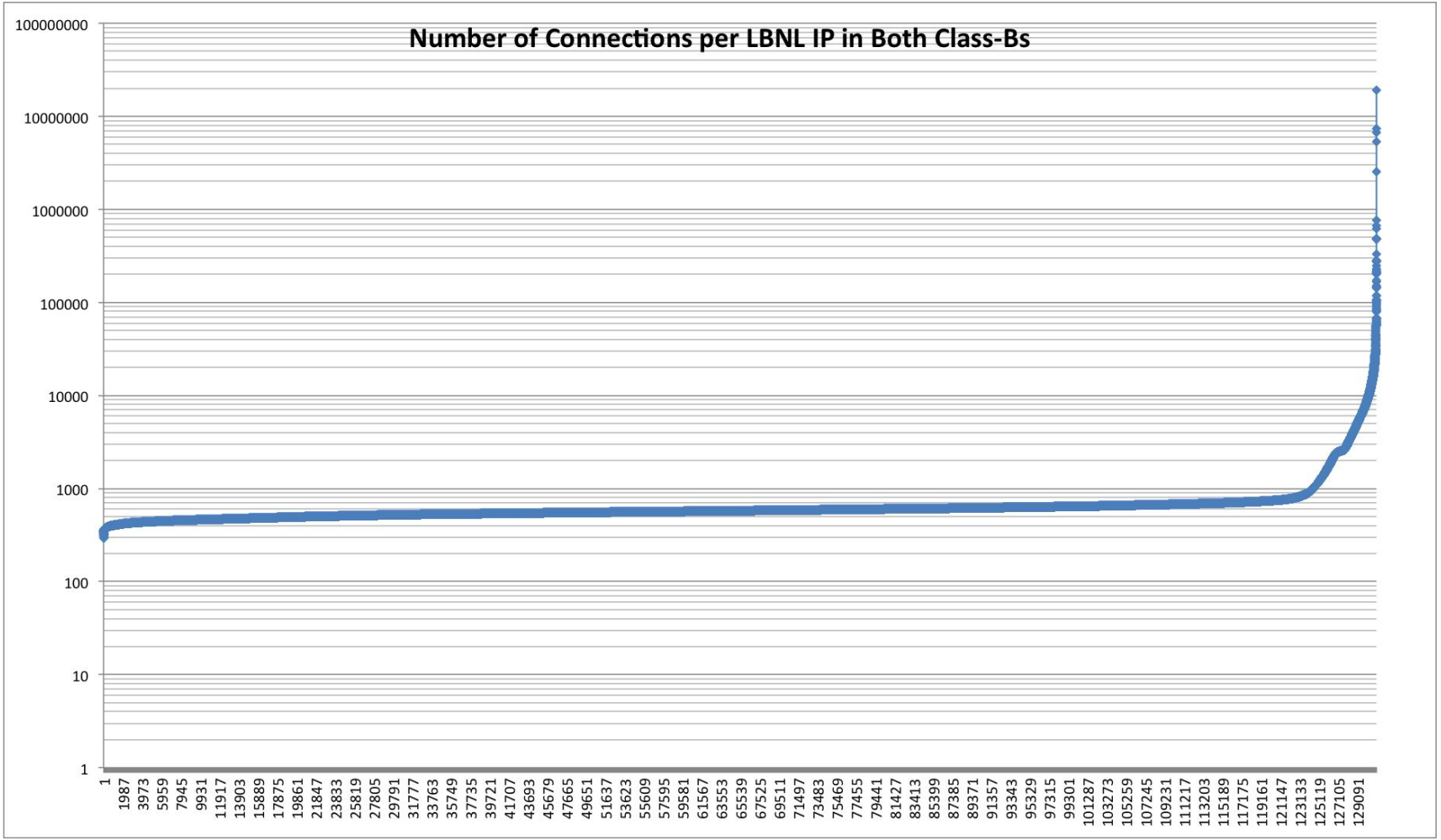
How many Avg. connections / LBNL IPs per day: ~800 for 128.3/16 and ~600 131.243/16

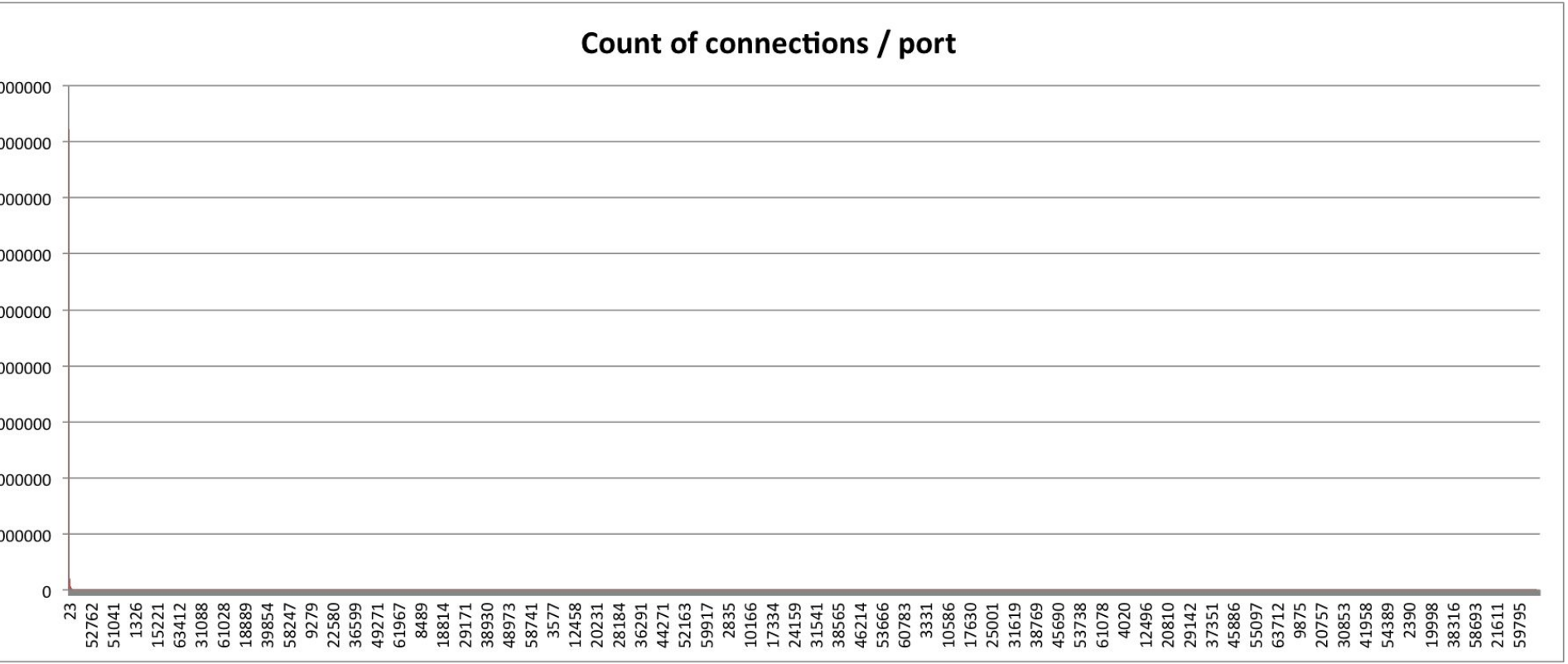
How many IPs Scan / day ~20K

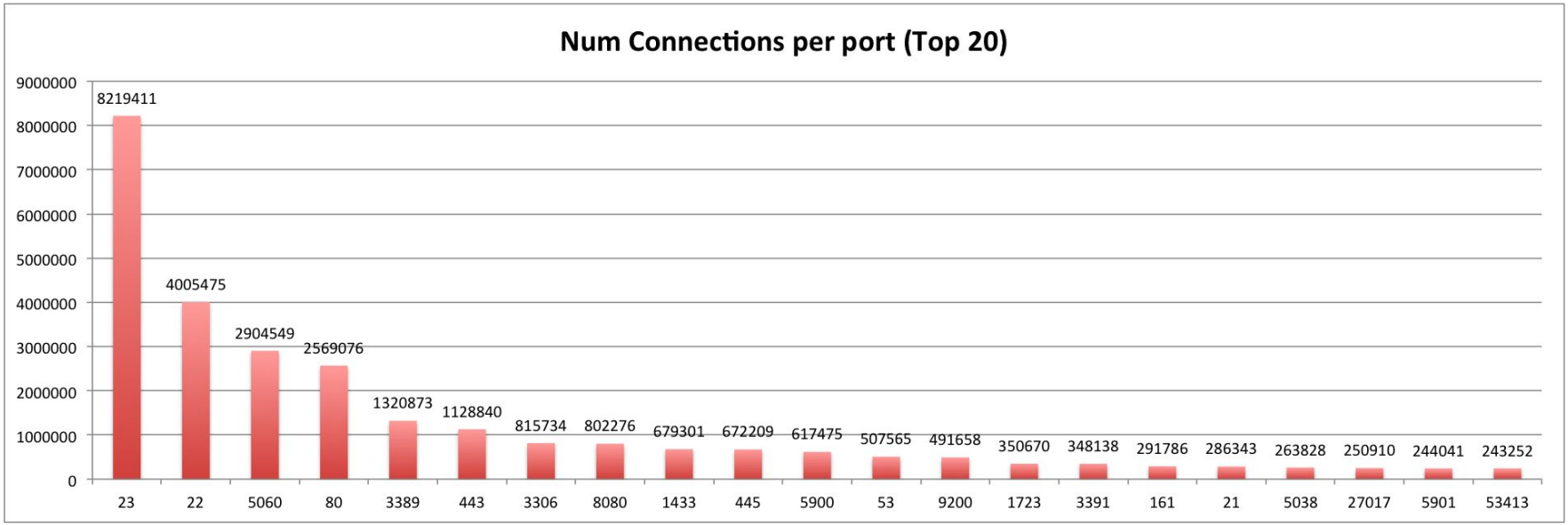
Are numbers meaningful or not ?

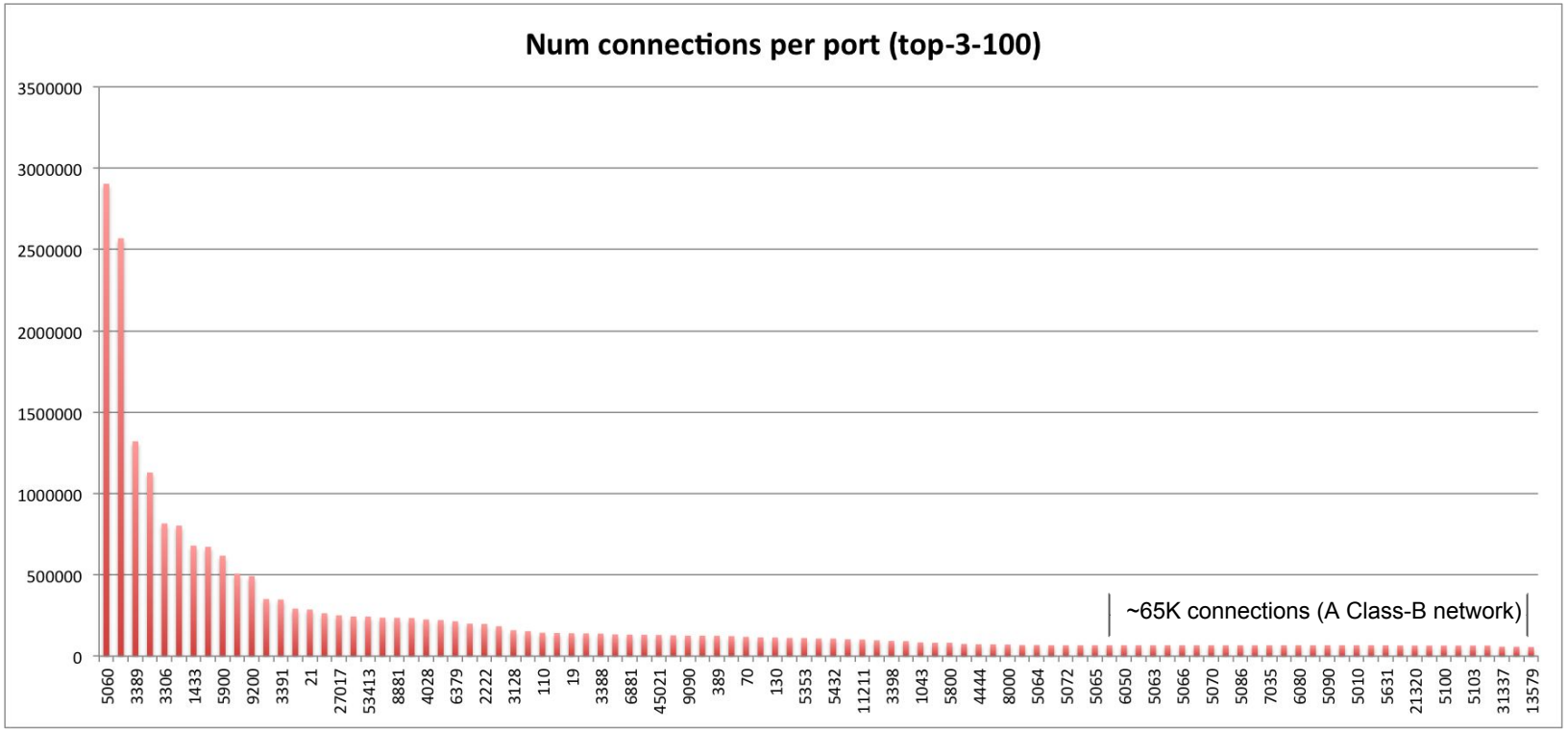
Philosophically a scan is an attribution or an intentionality problem but operationally we want to make it a measurement problem

- Partha Banerjee, LBNL

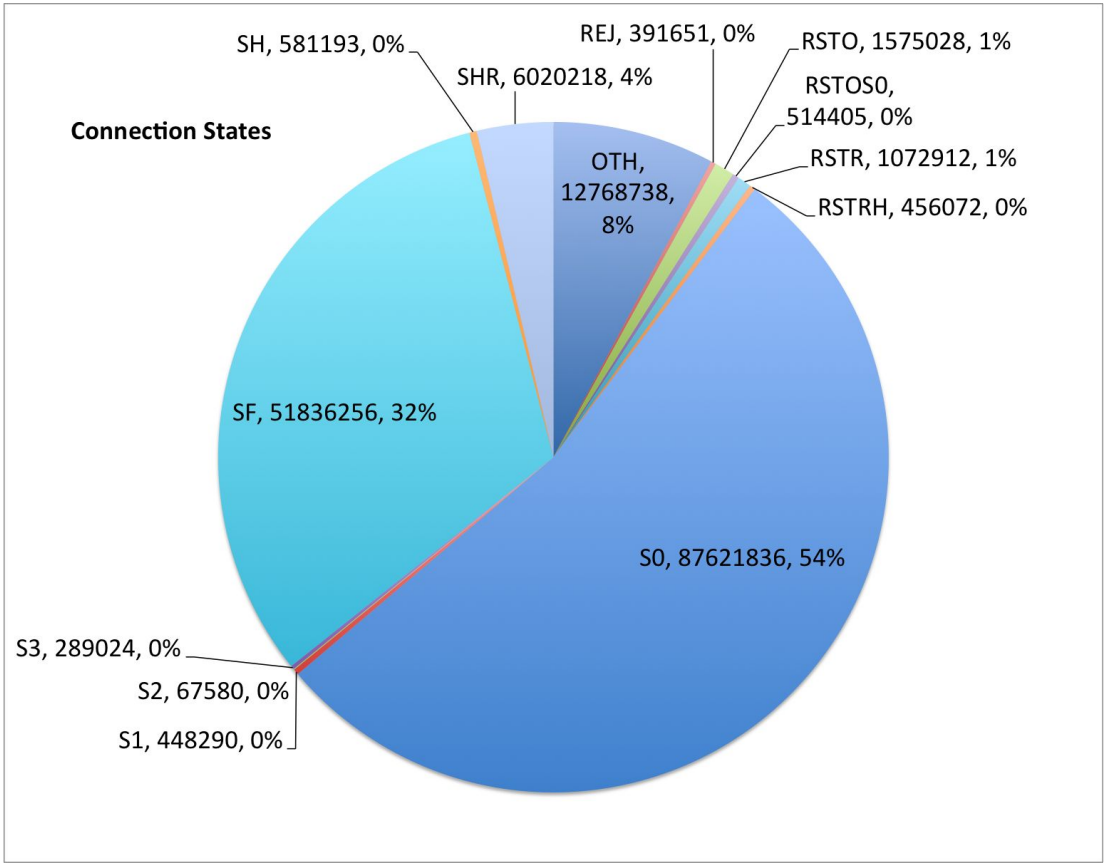

Number of Connections per LBNL IP in Both Class-Bs
100000000
10000000
1000000
100000
10000
1000
100
10
1
1
1987
3973
5959
7945
9931
11917
13903
15889
17875
19861
21847
23833
25819
27805
29791
31777
33763
35749
37735
39721
41707
43693
45679
47665
49651
51637
53623
55609
57595
59581
61567
63553
65539
67525
69511
71497
73483
75469
77455
79441
81427
83413
85399
87385
89371
91357
93343
95329
97315
99301
101287
103273
105259
107245
109231
111217
113203
115189
117175
119161
121147
123133
125119
127105
129091


Count of connections / port
000000
000000
000000
000000
000000
000000
000000
000000
0
23
52762
51041
1326
15221
63412
31088
61028
18889
39854
58247
9279
22580
36599
49271
61967
8489
18814
29171
38930
48973
58741
3577
12458
20231
28184
36291
44271
52163
59917
2835
10166
17334
24159
31541
38565
46214
53666
60783
3331
10586
17630
25001
31619
38769
45690
53738
61078
4020
12496
20810
29142
37351
45886
55097
63712
9875
20757
30853
41958
54389
2390
19998
38316
58693
21611
59795


Num Connections per port (Top 20)
9000000
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
8219411
4005475
2904549
2569076
1320873
1128840
815734
802276
679301
672209
617475
507565
491658
350670
348138
291786
286343
263828
250910
244041
243252
23
22
5060
80
3389
443
3306
8080
1433
445
5900
53
9200
1723
3391
161
21
5038
27017
5901
53413


Num connections per port (top-3-100)
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
~65K connections (A Class-B network)
5060
3389
3306
1433
5900
9200
3391
21
27017
53413
8881
4028
6379
2222
3128
110
19
3388
6881
45021
9090
389
70
130
5353
5432
11211
3398
1043
5800
4444
8000
5064
5072
5065
6050
5063
5066
5070
5086
7035
6080
5090
5010
5631
21320
5100
5103
31337
13579

Connection States in conn.log for a day of traffic on 2/29/16



S0 Connection attempt seen, no reply.
S1 Connection established, not terminated
SF Normal establishment and termination.
REJ Connection attempt rejected.
S2 Connection established and close attempt by originator seen (but no reply from responder).
S3 Connection established and close attempt by responder seen (but no reply from originator).
RSTO Connection established, originator aborted (sent a RST).
RSTR Established, responder aborted.
RSTOS0 Originator sent a SYN followed by a RST, we never saw a SYN-ACK from the responder.
RSTRH Responder sent a SYN ACK followed by a RST, we never saw a SYN from the (purported) originator.
SH Originator sent a SYN followed by a FIN, we never saw a SYN ACK from the responder (hence the connection was "half" open).
SHR Responder sent a SYN ACK followed by a FIN, we never saw a SYN from the originator.
OTH No SYN seen, just midstream traffic (a "partial connection" that was not later closed).

So the question is

There is so much (~54%) of irrelevant connections which I need to weed out !

What is the meaning of these connections ? Are these utterly useless or there is some reason into them

Why do I care to block these ?

Again, should I really care ??

TABLE IV. ATTACK PHASES

Attack Phase	Description	Incident Count
Scan Phase	Attackers try to identify vulnerable hosts and gather information about the target, e.g., services that are running.	1/1
Breach Phase	Attackers gain access to the system (e.g., using stolen or guessed credentials or by exploiting system misconfiguration (e.g., world writable files on an open share).	30/39
Penetration	Attackers exploit vulnerability (e.g., buffer overflow vulnerability) to obtain unauthorized access to the system.	9/10
Control	Attackers set up the compromised host to accept remote commands and provide reusable access (e.g., connect to command and control channel or install a backdoor).	21/23
Embedding	Attackers hide their malware and tracks by embedding the malware in the system, e.g., installing a rootkit, deleting system logs, adding ssh keys to authorized_key file, changing configuration files.	8/9
Data extraction/ modification	Attackers change or modify data in the system, e.g., deface web pages, copy database content, or steal information.	7/7
Attack-relay/ misuse	Attackers start misusing the system for personal gain, e.g., spam, DDoS using a bot, password harvesting, distributing warez, spreading virus, and phishing.	48/61

Q. How many incidents are detected at Scan Phase ?

Ans: We might not even have incidents yet.

Q. Of all the incidents we detect, for how many can we go back to and find the scan-phase that might have caused it ?

Q. How many incidents happen without any scan-phase/recon ?

***Analysis of Security Incidents from a large computing organization**

To state the obvious: Block reconnaissance at earliest

Various Strategies to block scanners

Mar 9 09:31:36 1457544696.394527 HairTrigger::AddressDropped (Site-XXX: Event connection_attempt: 104.200.29.248 to dst port 83/tcp)

Mar 9 09:31:40 1457544700.286791 Scan::KnockKnockScan 104.200.29.248 scanned a total of 3 hosts: [83/tcp] (US : 2923 miles)

Mar 9 09:32:12 1457544732.966686 TRW::TRWAddressScan 104.200.29.248 scanned a total of 4 hosts

Mar 9 09:32:59 1457544779.956911 Darknet::LandMine Scanner Darknet : 104.200.29.248 [83/tcp]

Mar 9 09:36:26 1457544986.436158 Scan::Address_Scan 104.200.29.248 scanned at least 25 unique hosts on port 83/tcp in 1m48s

Mar 9 09:42:14 1457545334.699021 OldScan::ShutdownThresh shutdown threshold reached for 104.200.29.248

Mar 9 09:42:16 1457545336.390310 OldScan::AddressScan 104.200.29.248 has scanned 101 hosts (83/tcp)

Note: This is a hand-picked example to show various strategies. This doesn't necessarily mean all scripts perform in the same fashion all the time.

Ankle-biters* - we should get rid of these

We should get rid of Ankle-biters* which are obviously noise

So that we can start paying attention to things which actually matter - rather than things that are noise

*First heard from Scott Campbell, NERSC

Scan::Address_Scan

Stock policy shipped with bro-2.4.1

Scan detection based on counters isn't sufficient enough

This Remote-IP connected to 25 remote IPs on 22/tcp (%likelyhood of scan?)

This Remote-IP connected to 5 remote IPs on 22/tcp (% likelyhood of scan ?)

This IP scanned N hosts in M minutes - hence scanner

We can leverage on quite a bit more intelligence to make a determination of a scanner

Also, more aggressive config can be rather false positive prone

HairTrigger::AddressDropped

What: Drop any connection based on intel from a remote data feed

+ve: Blocked on the very first connection_attempt

-ve: Clumsy data results in clumsy actions

Hairtrigger.bro is a pretty sleek bro policy which digests many remote feeds

1) using input-framework
2) maintains a cache of about 30-40K IPs at any given time
3) these IPs are constantly getting added and removed.
4) Smart ACLD optimizations for bulk adds and deletes

Scan::KnockKnock

Basically, this policy takes incoming remote IP connection and checks it against **table of known-services** for the LBNL IP and accesses if that's a good or bad connection.

If external IP makes 3 (or 5 or 12 depending on logics of dynamic thresholds) such failed connections, it is flagged as scanner

Policy is adaptive on how to increase and decrease its sensitivity for each scanner based on what port they are hitting and what’s the "popularity" of that port at that time.

“table of known-services”

- The advantage of a table like this is that upon observing an initial SYN sent by a remote host, one doesn't need to wait to see the response
- Notion that's basically a more refined version of using "landmine" addresses that if a remote host attempts to connect to, then it's likely a scanner since the address isn't used for anything (OR the port on that address isn’t used for anything)
- Enables a quicker decision since there's no need to wait to observe responses

Example showing dynamic thresholds

1458036937.778880 - - - - - - - - - Scan:: KnockKnockScan 204.155.30.109 scanned a total of 5 hosts: [2323/tcp] (US : 1693.38 miles) on 128.3.37.108 bro Notice::ACTION_LOG

1458036942.089409 - - - - - - - - - Scan:: KnockKnockScan 179.43.147.205 scanned a total of 4 hosts: [2323/tcp] (CH : nan miles) bro Notice::ACTION_LOG

1458036946.508650 - - - - - - - - - Scan:: KnockKnockScan 31.148.219.11 scanned a total of 3 hosts: [2323/tcp] (NL : nan miles) bro Notice::ACTION_LOG

Darknet::Landmine Scan Detection

- Policy - ingests the list of allocated subnets from a text-file using input-framework
- Any connection not in the above list is a Darknet Connection
- “N” such connections lead to a conclusion that this is a scanner
- Block the IP.

TRW::TRWAddressScan Detection

Fast Portscan Detection Using Sequential Hypothesis Testing

(http://www.icir.org/vern/papers/portscan-oak04.pdf)

- Model accesses to local IP addresses as a random walk on one of two stochastic processes, corresponding respectively to the access patterns of benign remote hosts and malicious ones
- TRW requires a much smaller number of connection attempts (4 or 5 in practice) to detect malicious activity, while also providing theoretical bounds on the low (and configurable) probabilities of missed detection and false alarms.
- TRW performs significantly faster and more accurately

OldScan::AddressScan

“Bro treats connections differently depending on their service (application protocol). For connections using a service specified in a configurable list, Bro only performs bookkeeping if the connection attempt failed (was either unanswered, or elicited a TCP RST response). For others, it considers all connections, whether or not they failed. It then tallies the number of distinct destination addresses to which such connections (attempts) were made. If the number reaches a configurable parameter N, then Bro flags the source address as a scanner. By default, Bro sets N = 100”

Exploring into the physical world - granularity of identity

So can we predict if something is a scanner based on

Subnet affinity ? - No brainer

GeoIP affinity ? - IP_A = City-C, IP_B = City-C

Should we wait for IP_B to cross a threshold if its touching the same port as IP_A

1457687793.012137 85.90.245.74 scanned a total of 3 hosts: [110/tcp] (DE : 8956.09 miles)
1457687793.012137 139.162.146.165 scanned a total of 5 hosts: [110/tcp] (DE : 8956.09 miles)
1457687793.012137 139.162.194.129 scanned a total of 4 hosts: [110/tcp] (US : 1693.38 miles)

Over fitting problem

```
1457687793.012137 85.90.245.74 scanned a total of 3 hosts: [110/tcp] (DE : 8956.09 miles)
1457687793.012137 139.162.146.165 scanned a total of 5 hosts: [110/tcp] (DE : 8956.09 miles)
1457687793.012137 139.162.194.129 scanned a total of 4 hosts: [110/tcp] (US : 1693.38 miles)
1457687793.012137 104.200.29.248 scanned a total of 7 hosts: [110/tcp] (US : nan miles)
```

1457687787.715675	Cr3JSu3PgFAdPhPuCd	85.90.245.74	44038	131.243.168.80	110	tcp	-	-	-	-	S0
1457687787.737843	CvhPj03UICzZOn2Gg3	85.90.245.74	42794	131.243.85.114	110	tcp	-	-	-	-	S0
1457687787.771505	CbwDFP3cFZQLRmJF97	139.162.194.129	60593	131.243.59.98	110	tcp	-	-	-	-	S0
1457687787.857129	CnhV3d2RjbgFIwNbE7	139.162.194.129	44238	128.3.137.107	110	tcp	-	-	-	-	S0
1457687789.516423	CwJwZP3LsBnjJ2qHCk	139.162.146.165	36591	131.243.134.222	110	tcp	-	-	-	-	S0
1457687789.735182	CrR9PD135VnOGu99Ri	139.162.146.165	45097	128.3.21.176	110	tcp	-	-	-	-	S0
1457687789.801726	C5NxTp4GqrxpTZbKk6	139.162.194.129	34038	131.243.201.1	110	tcp	-	-	-	-	S0
1457687789.907552	CuBQzI3Gx9Gk5zqn9k	139.162.146.165	33297	128.3.190.215	110	tcp	-	-	-	-	S0
1457687790.058140	CNELYYkqXirh6N4Mf	139.162.146.165	51528	128.3.101.213	110	tcp	-	-	-	-	S0
1457687790.352425	CeGFeaK74PBKw7ou3	85.90.245.74	53995	128.3.228.11	110	tcp	-	-	-	-	S0
1457687790.529582	Cf43ux1YY91gczIvni	85.90.245.74	48111	128.3.101.67	110	tcp	-	-	-	-	S0
1457687790.870907	CqlB5eHlraYSFWUdc	85.90.245.74	45821	131.243.168.71	110	tcp	-	-	-	-	S0
1457687790.965051	CgoAHG3nzeI3IzlfZj	139.162.146.165	59925	128.3.106.173	110	tcp	-	-	-	-	S0
1457687791.002899	Cvr9K849PqCAoWNDhf	85.90.245.74	53293	131.243.157.172	110	tcp	-	-	-	-	S0
1457687791.189753	CuEgUu29GCG72YokT	85.90.245.74	56960	131.243.38.80	110	tcp	-	-	-	-	S0
1457687791.311602	CeYaEE42q3j8u9hgcj	139.162.146.165	54794	131.243.207.50	110	tcp	-	-	-	-	S0
1457687791.330807	CpCmuK2svUuQcppZKf	139.162.194.129	52988	131.243.63.115	110	tcp	-	-	-	-	S0
1457687792.247252	CqTiif1wUKCSsWFQX	139.162.194.129	45832	128.3.240.210	110	tcp	-	-	-	-	S0
1457687792.387033	Cf9f8O10Mln9dLWDM5	139.162.194.129	49521	131.243.20.187	110	tcp	-	-	-	-	S0
1457687792.462379	CGz7QpGLYgEEjYLK4	139.162.194.129	48346	128.3.81.89	110	tcp	-	-	-	-	S0
1457687792.481816	CrbdvVcY9eTWtyiD1	139.162.146.165	57547	131.243.78.79	110	tcp	-	-	-	-	S0
1457687792.486815	ChAf6xJBiIrCeUH27	139.162.194.129	58300	128.3.112.126	110	tcp	-	-	-	-	S0
1457687792.648765	Cg4nSN23q8lYCYJAk2	139.162.194.129	53653	131.243.220.153	110	tcp	-	-	-	-	S0
1457687792.680243	Cz4IpW1Q9cjAi0aQ3k	139.162.194.129	44982	128.3.130.199	110	tcp	-	-	-	-	S0
1457687792.880279	CJfxDy2DfNqEgMqvSg	139.162.194.129	40245	128.3.14.185	110	tcp	-	-	-	-	S0

So the big question is

Why do we care about blocking 10th of a millisecond or 100th of a millisecond or even a few seconds ?

Why are we being picky here ?

Two reasons :

1) Can we be predictive about a potential scanner as soon as it touches us ?
2) Next slide shows story of a 3389/tcp (RDP) scanner

Can we use physical world as basis for lowering the threshold to 1 from 3 ?

Definitely use geoIP for FP suppressions

May be -

“Any scanner within ~50 mile radius needs to be vetted with a higher threshold” ?

Sensitivity and specificity

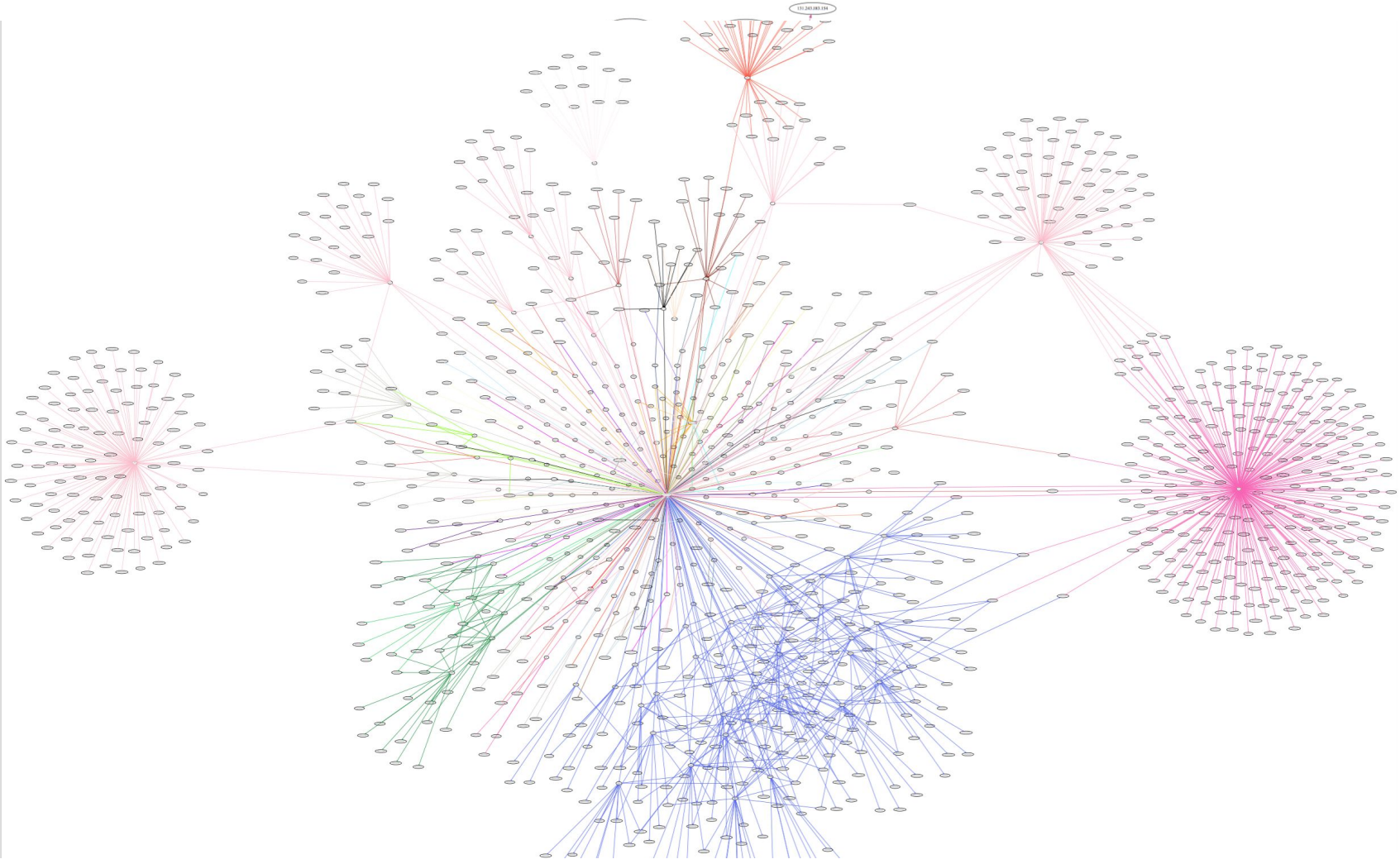
Its OK to have false negative

Its not ok to have a false positive

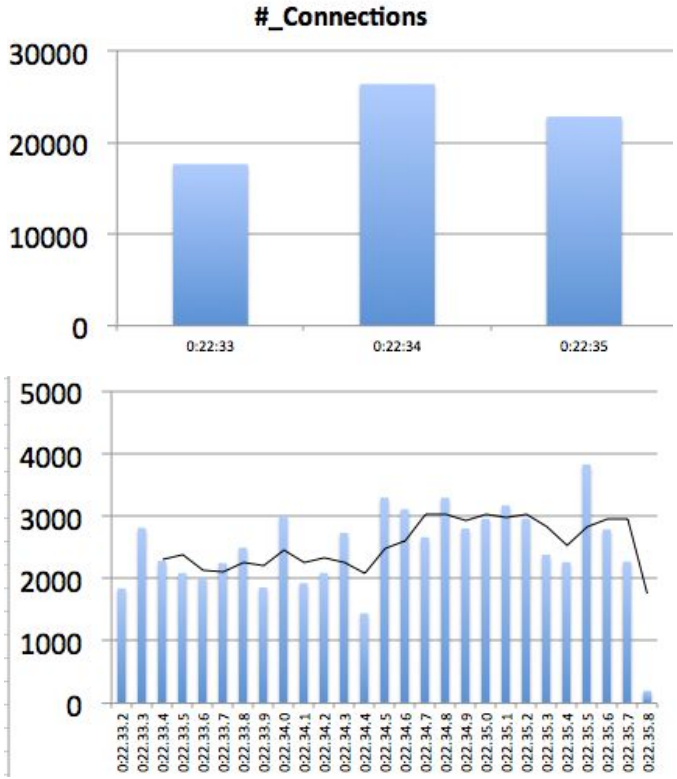
So that we can be super-aggressive in blocking quickly

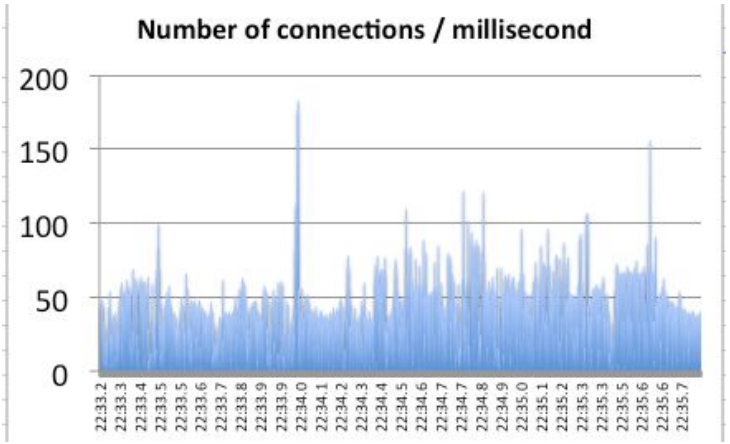
False positives hard to eliminate

- Web spiders - Well they are scanners in true sense
- Sticky configurations
  - Active directory systems
- Perf sonars systems
- Xbox games
  - “OTH” packet which are middle of connection in xbox gaming


131.243.183.184

A very fast scan - /16 in 2.59 seconds





40 conn/millisecond
30 millisecond to block
40x30=1200 hosts already scanned

Deep blocks

- Drop::AddressSeenAgain
- HTTP::HTTP_SensitiveURI
- HTTP::HTTP_Suspicous_Client_Header
- HTTP::SQL_Injection_Attacker
- HTTP::SQL_Injection_Victim
- HTTP::Sensitive_UserAgent
- Heartbleed::SSL_Heartbeat_Attack
- ICMP::ICMPAddressScan
- NTP::NTP_Monlist_Queries
- Nullroute::AddNullRoute
- SIP::SIP_403_Forbidden
- SIP::SipviciousScan
-

Future Work

- Block things which don’t matter
- Need to further Identify things which matter
- Can we create an “non-reconable” network ie all hosts move up or down an IP and may be even hostnames change
- Signal a router to throttle an IP address - Tarpit ??
- Lets look at their DNS story
  - dns is like looking into a car’s window and not pull the door knob
- What happens to known hosts services being connected
- Make Bro more knowledgeable based on nessus, nmap, syslogs fed into Bro

Questions

security@lbl.gov

asharma@lbl.gov

Bro-scripts from the talk: https://github.com/initconf/bro4pros-16