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
~65K connections (A Class-B network)
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??
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?
To state the obvious: Block reconnaissance at earliest
Various Strategies to block scanners

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.
“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
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

It's OK to have false negative

It's 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
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_Suspicious_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 i.e. all hosts move up or down an IP and may be even hostnames change
- Signal a router to throttle an IP address - Tarpit ??
- Let’s 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