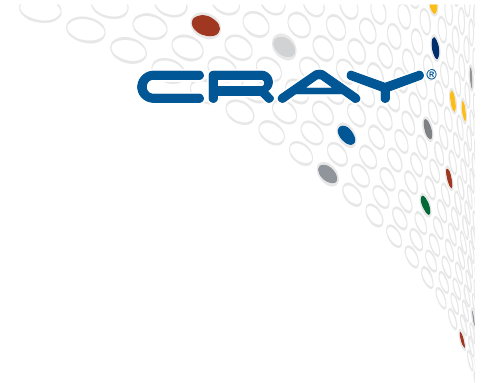


Graph analysis and computer network security challenges

Eric Dull



Agenda

- What is SCinet
- What analytic questions were we answering
- How we applied graphs to answer these questions
- Places to start exploring graphs

What is SCinet

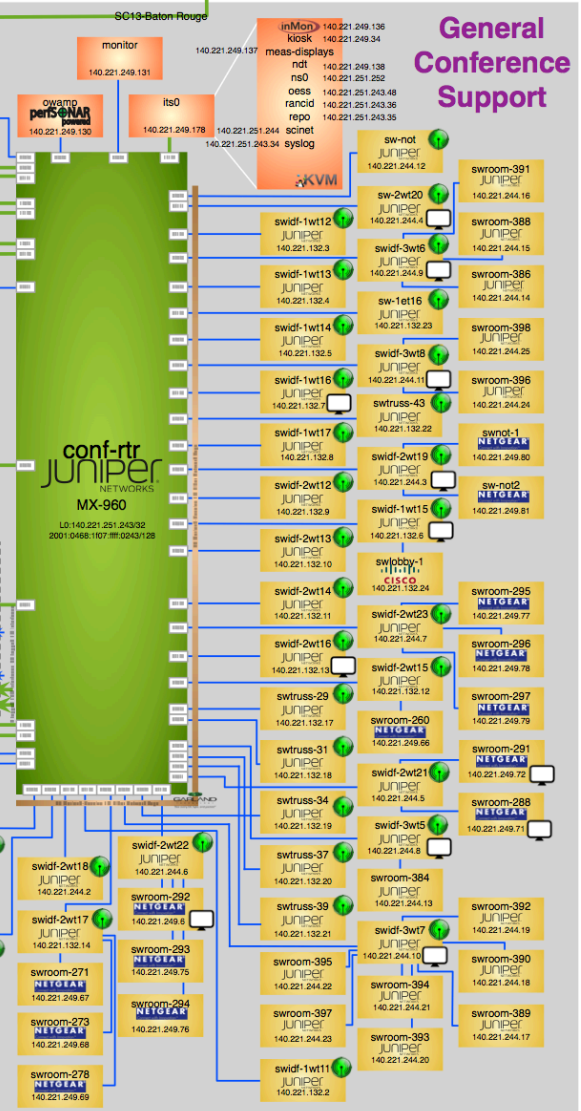
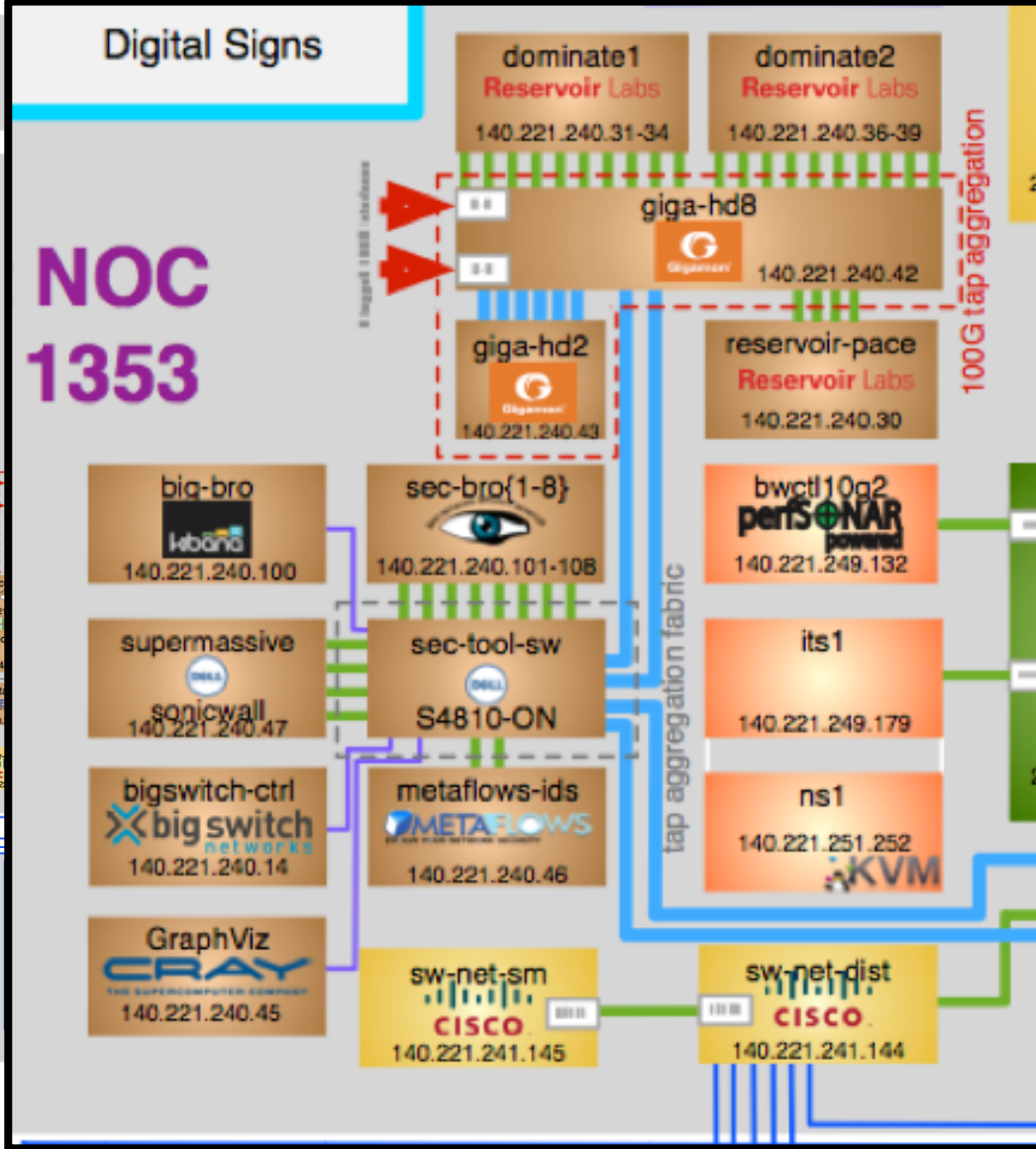
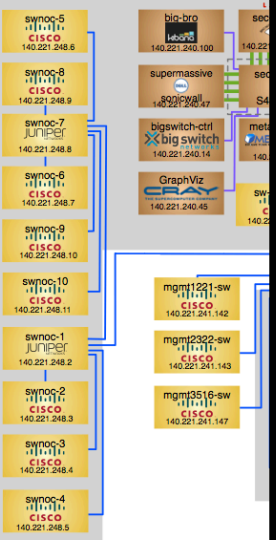
- /17 publicly routed network
- Network supporting the SC technical conference and exhibit hall
- 10,972 devices on the network
- 1.2 Tb/s onto the show floor
- 296 Gb/s under BRO observation
- Set-up to teardown – about 10 days
- Rebuilt/reused every year



Network Architecture V5.5
 Linda Winkler
 Argonne National Laboratory
 Nov. 16-21, 2014

- 10/100/1000-TX
- 1 Gigabit Ethernet
- 10 Gigabit Ethernet
- 40 Gigabit Ethernet
- 100 Gigabit Ethernet
- Dense Wave Division Multiplexing
- Wi-Fi Access Point
- Digital Signs

SCinet NOC Booth 1353



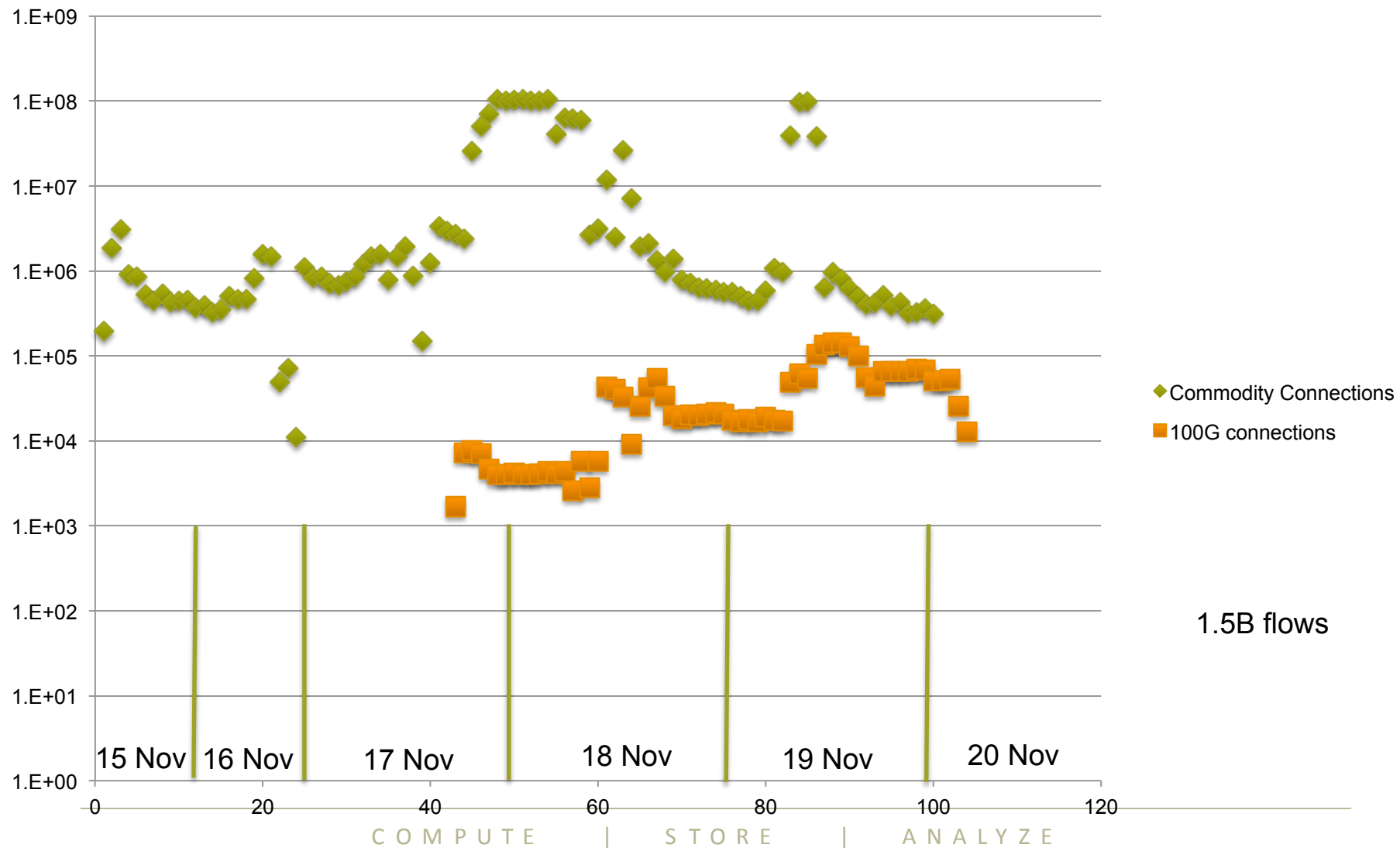
Total generated triples



BRO Log type	Lines	Triples per line	Triples
files	13,432,704	10	134,327,040
syslog	1,085,812	10	10,858,120
notice	380,842	10	3,808,420
http	12,133,443	25	303,336,075
ssh	2,093,004	10	20,930,040
dhcp	986,072	10	9,860,720
weird	49,789,135	5	248,945,675
conn	1,487,430,036	12	17,849,160,432

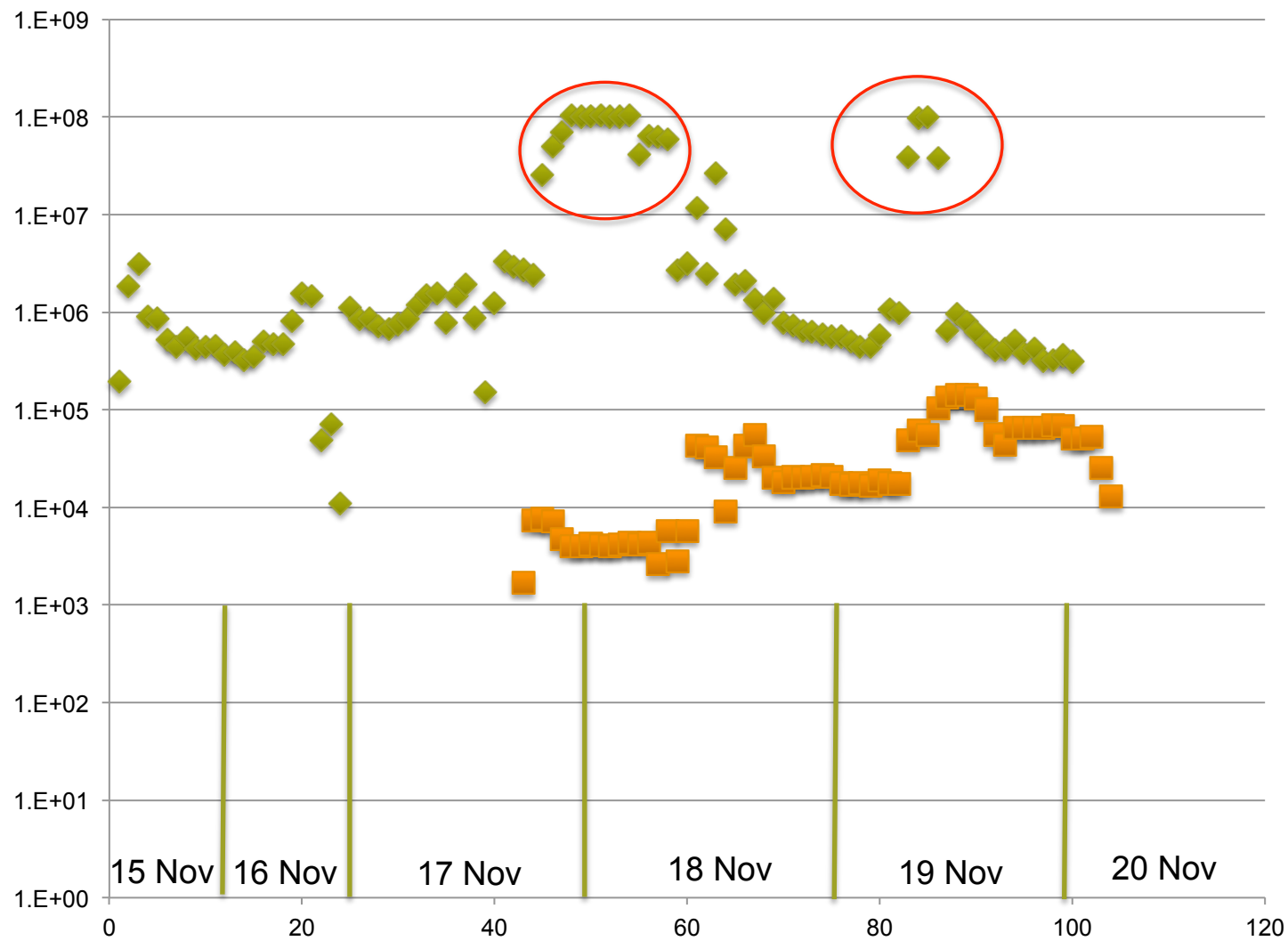


Flow counts





Flow counts



SYN flood
1.3B flow

◆ Commodity Connections
■ 100G connections

1.5B flows

COMPUTE | STORE | ANALYZE

Analytic charge

- Find outbound scanning or attacking
- Help identify groups of infected systems from C2 and download activities
- “Perform the next hop” of analysis. Use graphs to ease automated analysis
- Find new DNS and DHCP servers as they appear on the network

Applicable graph operations

- **Search – “Find SSH connection networks”**
 - IP address based search
 - Port and volume based search
 - IDS alert based search
 - 1,2, or 3 hop
- **Jaccard Scoring – “which is the likely C2 channel for IPs downloading from this port?”**
- **Betweenness Centrality – “Which IP address in this network should be considered first when cleaning up an infection network?”**

Search example: SSH chain alerting hosts – X > 10K response bytes

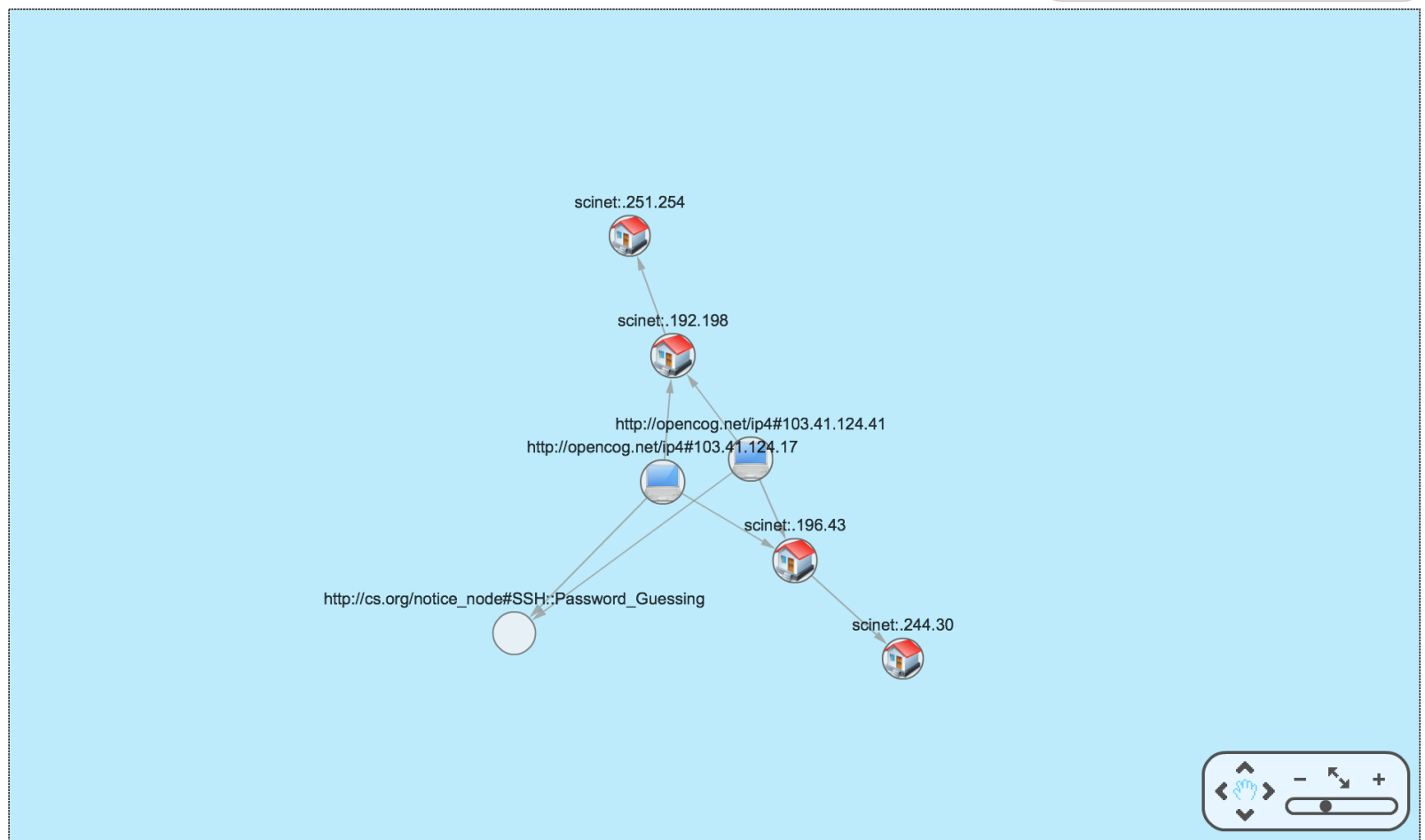


Urika SC14 SCinet NetSec

Run Queries View Queries Graph Options Querying

ssh chain

Search in graph



Copyright © 2013 YarcData | Cray Inc.

Pleasanton, CA | Seattle, WA | [Contact YarcData](#)

COMPUTE | STORE | ANALYZE

Search example: SSH connection chain SPARQL query

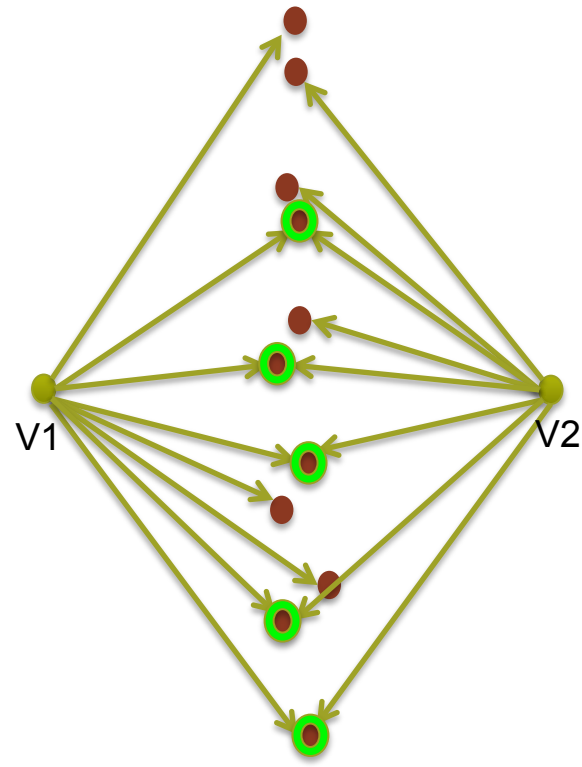
```
CONSTRUCT{
  ?ap_addr <http://cs.org/p/hasNoticeNote> <http://cs.org/notice_node#SSH::Password_Guessing>.
  ?ap_addr <urn:p/hasSSH> ?internal_addr.
  ?internal_addr <urn:p/hasSSH> ?a_addr.
}
{
  SELECT distinct ?internal_addr ?ap_addr ?a_addr
  WHERE
  {
    ?uid4 <http://opencog.net/p/destinationAddress> ?a_addr.
    ?uid4 <http://opencog.net/p/sourceAddress> ?internal_addr.
    ?uid4 <http://opencog.net/p/hasProtocol> <http://opencog.net/proto#tcp>.
    ?uid4 <http://opencog.net/p/destinationPort> <http://opencog.net/port#22> .
    ?uid4 <http://cs.org/p/hasRespBytes> ?rbytes1.
    FILTER(?rbytes1 > 10000)
  }
  {
    SELECT distinct ?internal_addr ?ap_addr
    WHERE
    {
      ?uid <http://cs.org/p/hasNoticeNote> <http://cs.org/notice_node#SSH::Password_Guessing>.
      ?uid <http://cs.org/p/hasNoticeMsg> ?msg.
      ?uid <http://cs.org/p/hasOrigAddr> ?ap_addr.
      ?uid4 <http://opencog.net/p/sourceAddress> ?ap_addr.
      ?uid4 <http://opencog.net/p/destinationAddress> ?internal_addr.
      ?uid4 <http://opencog.net/p/destinationPort> <http://opencog.net/port#22>.
      ?uid4 <http://cs.org/p/hasRespBytes> ?rbytes1.
      FILTER(?rbytes1 > 20900)
    }
  }
  LIMIT 1000
}
}
```

Jaccard example: math and SPARQL implementation

```

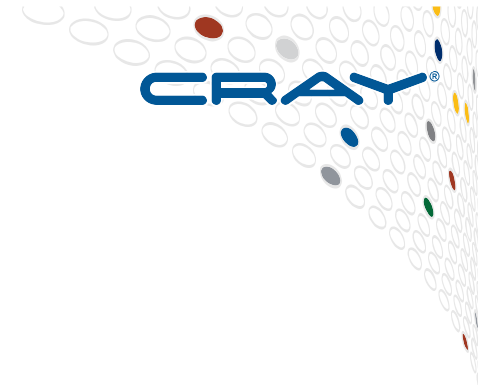
SELECT ?proto ?port ?client_count ?big_client_count
WHERE
{
  {
    SELECT ?proto ?port (count(distinct ?ap_addr) as ?big_client_count)
    WHERE
    {
      ?uid3 <http://opencog.net/p/sourceAddress> ?ap_addr.
      ?uid3 <http://opencog.net/p/destinationAddress> ?dest_addr2 .
      ?uid3 <http://opencog.net/p/destinationPort> ?port .
      ?uid3 <http://opencog.net/p/hasProtocol> ?proto .
      ?uid3 <http://cs.org/p/hasRespBytes> ?rbytes2.
    }
    GROUP BY ?proto ?port
  }
  {
    SELECT ?proto ?port (count(distinct ?ap_addr) as ?client_count)
    WHERE
    {
      ?uid3 <http://opencog.net/p/sourceAddress> ?ap_addr.
      ?uid3 <http://opencog.net/p/destinationAddress> ?dest_addr2 .
      ?uid3 <http://opencog.net/p/destinationPort> ?port .
      ?uid3 <http://opencog.net/p/hasProtocol> ?proto .
      ?uid3 <http://cs.org/p/hasRespBytes> ?rbytes2.
      FILTER(?rbytes2 > 0)
      ?uid4 <http://opencog.net/p/sourceAddress> ?ap_addr.
      ?uid4 <http://opencog.net/p/destinationAddress> ?dest_addr .
      ?uid4 <http://opencog.net/p/destinationPort> <http://opencog.net/port#9162>.
      ?uid4 <http://cs.org/p/hasRespBytes> ?rbytes1.
      FILTER(?rbytes1 > 0)
    }
    GROUP BY ?proto ?port
    HAVING (?client_count > 1)
  }
}
ORDER BY DESC(?client_count)

```



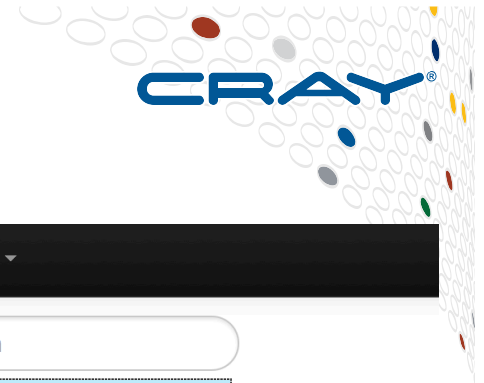
Definition: $|V1 \cap V2| / |V1 \cup V2|$

Jaccard example: SSH password forced C2 channel candidates



?proto	?port	?client_coun	?big_client_count
<http://opencog.net/proto#udp>	<http://opencog.net/port#26161>	2	6 0.33333333
<http://opencog.net/proto#udp>	<http://opencog.net/port#9162>	2	7 0.28571429
<http://opencog.net/proto#tcp>	<http://opencog.net/port#9162>	5	18 0.27777778
<http://opencog.net/proto#tcp>	<http://opencog.net/port#7668>	4	20 0.2
<http://opencog.net/proto#tcp>	<http://opencog.net/port#17471>	2	12 0.16666667
<http://opencog.net/proto#tcp>	<http://opencog.net/port#30261>	2	12 0.16666667
<http://opencog.net/proto#tcp>	<http://opencog.net/port#28761>	2	13 0.15384615
<http://opencog.net/proto#tcp>	<http://opencog.net/port#15770>	2	13 0.15384615
<http://opencog.net/proto#tcp>	<http://opencog.net/port#12081>	2	13 0.15384615
<http://opencog.net/proto#tcp>	<http://opencog.net/port#6218>	2	14 0.14285714
<http://opencog.net/proto#tcp>	<http://opencog.net/port#18675>	2	14 0.14285714
<http://opencog.net/proto#tcp>	<http://opencog.net/port#19244>	2	14 0.14285714
<http://opencog.net/proto#tcp>	<http://opencog.net/port#32712>	2	16 0.125
<http://opencog.net/proto#tcp>	<http://opencog.net/port#7168>	2	16 0.125
<http://opencog.net/proto#tcp>	<http://opencog.net/port#10090>	2	16 0.125
<http://opencog.net/proto#tcp>	<http://opencog.net/port#30556>	2	16 0.125

Jaccard example: ports 7668 and 9162 visualization

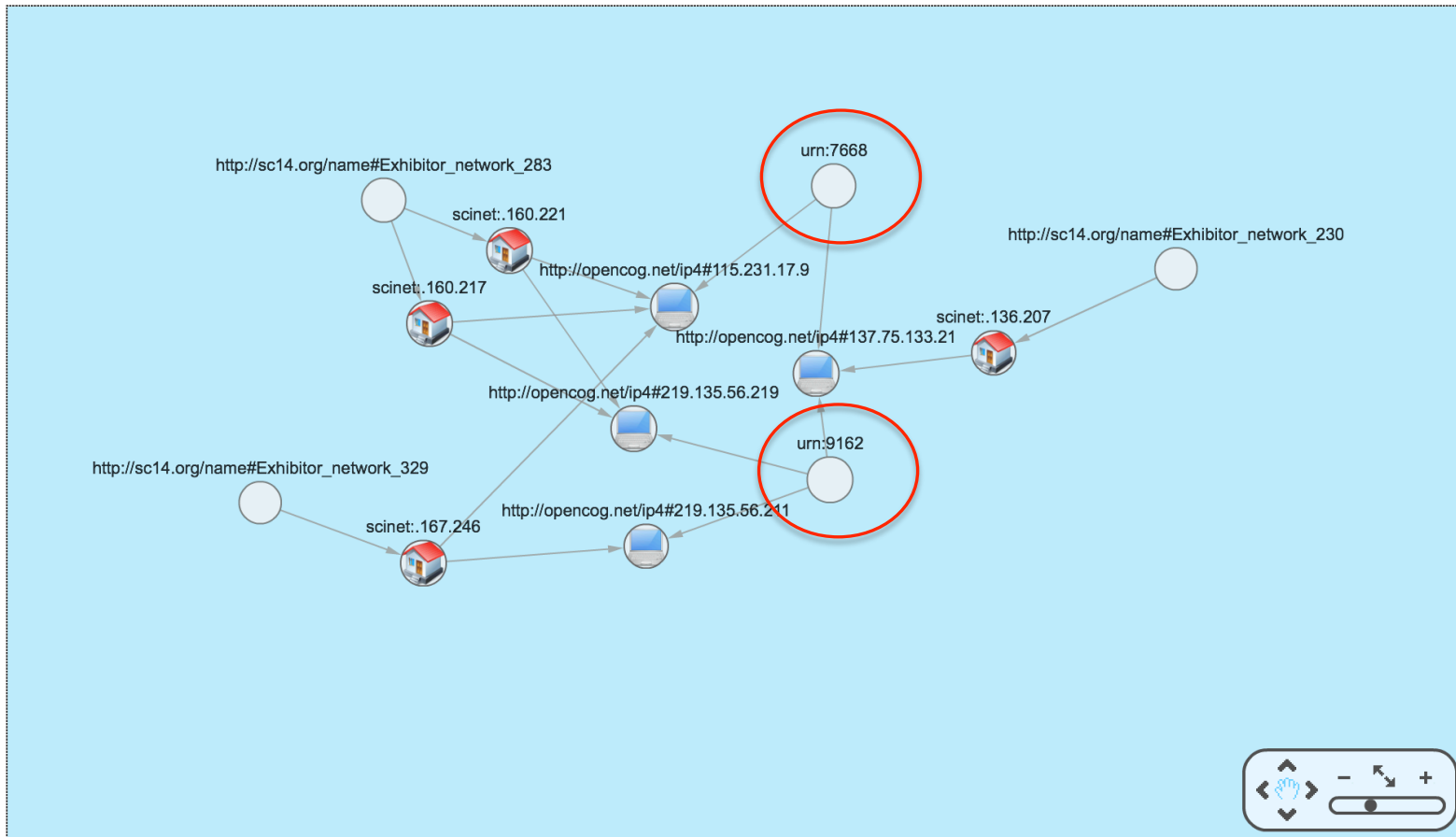


Urika YarcData SC14 SCinet
NetSec

Run Queries ▾ View Queries ▾ Graph ▾ Options ▾ Connected: discover. ▾

Port 7668

Search in graph



Copyright © 2013 YarcData | Cray Inc.

Pleasanton, CA | Seattle, WA | [Contact YarcData](#)

COMPUTE | STORE | ANALYZE

Betweenness example: pseudo-math and SPARQL implementation

```

SELECT ?vertices ?scores
WHERE
{
  CONSTRUCT{ #<urn:SSH_forcer> <urn:p/HasMember> ?src_addr.
  ?src_addr <urn:p/hasSSH> ?dest_addr.
  ?dest_addr <urn:p/hasSSH> ?dest_addr2
  }
  WHERE
  {
    SELECT distinct ?src_addr ?dest_addr ?dest_addr2
    WHERE
    { ?booth2 a <http://sc14.org/class#SCinet_subnet> .
      ?booth2 <http://opencog.net/hasMember> ?dest_addr .
      ?uid3 <http://opencog.net/p/sourceAddress> ?dest_addr .
      ?uid3 <http://opencog.net/p/destinationAddress> ?dest_addr2 .
      ?uid3 <http://opencog.net/p/hasProtocol> <http://opencog.net/proto#tcp> .
      ?uid3 <http://opencog.net/p/destinationPort> <http://opencog.net/port#22> .
      ?uid3 <http://opencog.net/p/start> ?start_time2.
      ?uid3 <http://cs.org/p/hasRespBytes> ?rbytes2.
      FILTER (?rbytes2 > 12000)
      FILTER (?start_time < ?start_time2)
      OPTIONAL
    }
  }
  SELECT ?src_addr ?dest_addr ?start_time
  { #?src_addr a <http://sc14.org/class#SSHattacker>.
    ?uid <http://cs.org/p/hasNoticeNote> <http://cs.org/notice_node#SSH::Password_Guessing>.
    ?uid <http://cs.org/p/hasNoticeMsg> ?msg.
    ?uid <http://cs.org/p/hasOrigAddr> ?src_addr.

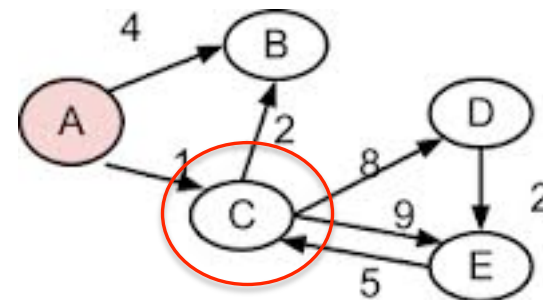
    ?uid3 <http://opencog.net/p/sourceAddress> ?src_addr .
    ?uid3 <http://opencog.net/p/destinationAddress> ?dest_addr .
    ?uid3 <http://opencog.net/p/hasProtocol> <http://opencog.net/proto#tcp>.
    ?uid3 <http://opencog.net/p/destinationPort> <http://opencog.net/port#22> .
    ?uid3 <http://opencog.net/p/start> ?start_time.
    ?uid3 <http://cs.org/p/hasRespBytes> ?rbytes2.
    FILTER(?rbytes2 > 12000)
  }
  LIMIT 500
}
}
!INVOKE yd:graphAlgorithm.betweenness centrality (.5,1)
PRODUCING ?vertices ?scores
}
ORDER BY DESC(?scores)

```

How to compute Betweenness centrality (All-pairs shortest-path)

- From every node, compute the shortest path(s) to every other node
- For every node, count the number of shortest paths that go through it
- For every edge, count the number of shortest paths that go through it
- Divide the shortest path counts by the total number of shortest paths to generate centrality scores

The nodes and edges with the highest centrality scores are most central





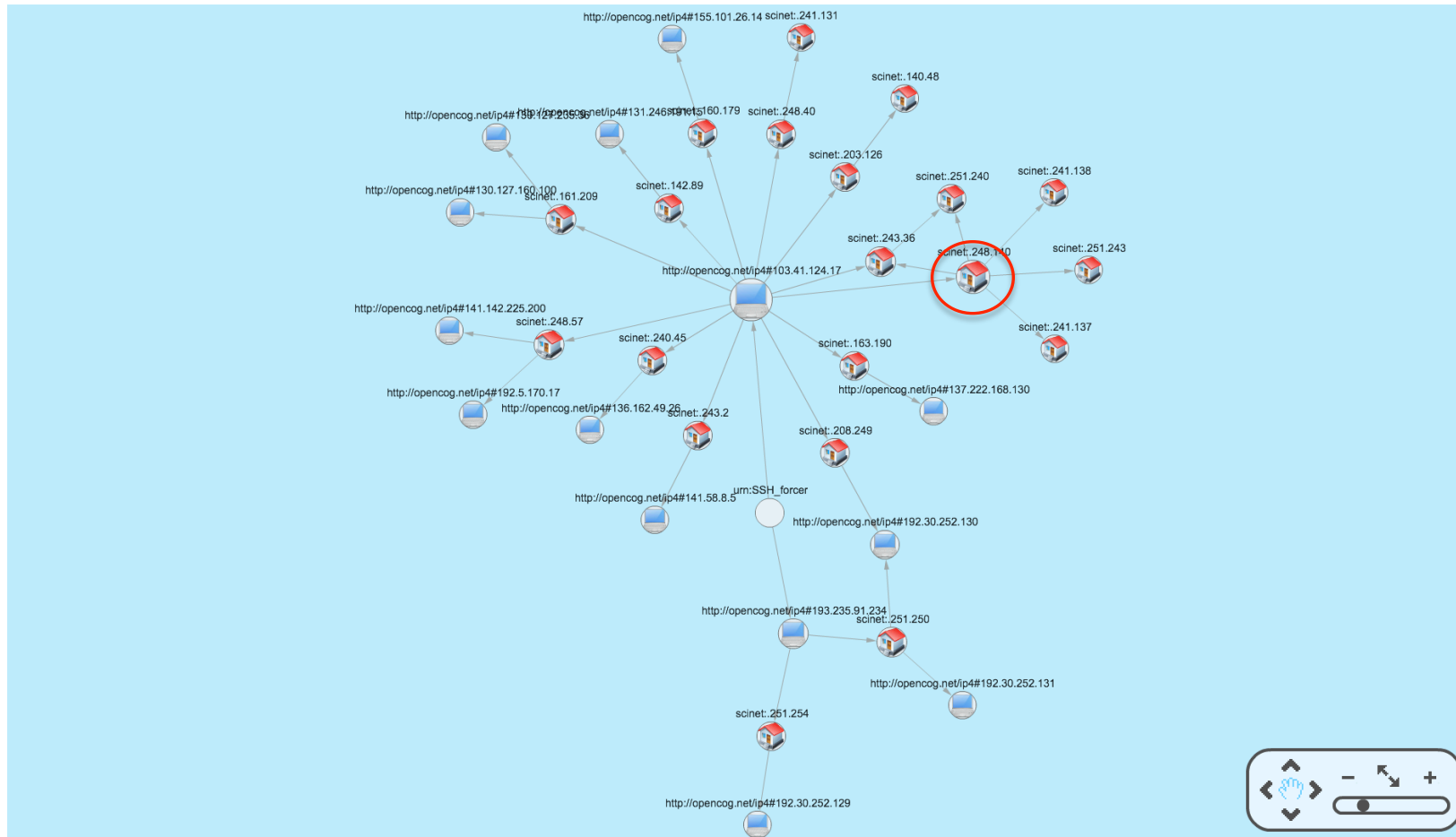
Betweenness example: SSH / Internal / ?



Run Queries ▾ View Queries ▾ Graph ▾ Options ▾ Connected: discover. ▾

SSH Internal ?

Search in graph





Betweenness example: Centrality results

?vertices

<http://opencog.net/ip4#A.B.248.140>

<http://opencog.net/ip4#A.B.248.57>
<http://opencog.net/ip4#A.B.161.209>
<http://opencog.net/ip4#A.B.160.131>
<http://opencog.net/ip4#A.B.243.2>
<http://opencog.net/ip4#A.B.203.126>
<http://opencog.net/ip4#A.B.248.40>
<http://opencog.net/ip4#A.B.163.190>
<http://opencog.net/ip4#A.B.240.45>
<http://opencog.net/ip4#A.B.142.81>
<http://opencog.net/ip4#A.B.243.36>

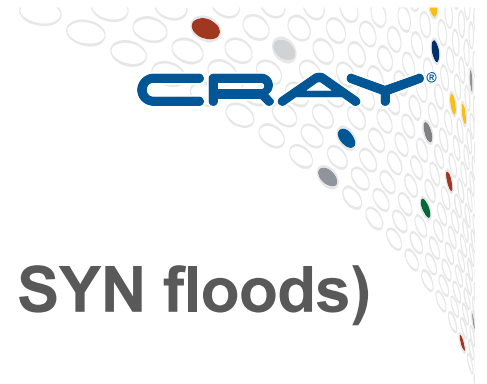
?scores

1^^<http://www.w3.org/2001/XMLSchema#double>
0.5217391304347826^^<http://www.w3.org/2001/XMLSchema#double>
0.5217391304347826^^<http://www.w3.org/2001/XMLSchema#double>
0.2608695652173913^^<http://www.w3.org/2001/XMLSchema#double>
0.2608695652173913^^<http://www.w3.org/2001/XMLSchema#double>
0.2608695652173913^^<http://www.w3.org/2001/XMLSchema#double>
0.2608695652173913^^<http://www.w3.org/2001/XMLSchema#double>
0.2608695652173913^^<http://www.w3.org/2001/XMLSchema#double>
0.2608695652173913^^<http://www.w3.org/2001/XMLSchema#double>
0.2608695652173913^^<http://www.w3.org/2001/XMLSchema#double>
0.04347826086956522^^<http://www.w3.org/2001/XMLSchema#double>

Available Algorithms



- **Search / neighborhood identification and extraction**
 - Pattern-matching / subgraph isomorphism: (**Core functionality**)
 - Cybersecurity application: Context and search, data exfiltration, beaconing, attack identification
- **Community detection**
 - Modularity: (**Fall2013 release**)
 - Relaxed clique
 - Cybersecurity application: Botnet detection and server hierarchy mapping
- **Similarity scoring**
 - Jaccard scoring: (**Example code available**)
 - Cybersecurity: Infrastructure mapping, botnet detection, client / server mapping
- **Path finding**
 - Shortest path: (**Summer2014 release**), S-T connectivity: : (**Fall2014 release**)
 - Cybersecurity application: Identify likely paths for information flow between nodes
- **Key node / edge identification**
 - Betweenness centrality: (**Summer2014 release**)
 - Cybersecurity application: find the vulnerable points in network configurations
- **Anomaly identification and clustering**
 - Cybersecurity application: Unknown-unknown identification
 - Cybersecurity application: BadRank (**Summer2014 release**): finds likely worst actors by association with known bad actors, a la PageRank



Successes

- Identified outbound scanning behaviors (and SYN floods)
- Identified candidate external C2 hosts
- Identified candidate internal infected hosts based on port usage
- Identified candidate C2 ports using Jaccard scoring
- Identified the first place to start cleaning up the XX SSH client chain (if we chose to do that. We turned off the network instead)

Places to start your own graph analysis journey



- **Literature**

- Chapter 13 of Network Security through Data Analysis by Michael Collins
- Mark Newman's publications (<http://www-personal.umich.edu/~mejn/pubs.html>)
- Linked by Barabasi

- **Available FOSS tools**

- Gephi
- Cytoscape
- Apache Jena
- MTGL (<https://software.sandia.gov/trac/mtgl>)

