Analysis of authentication events and graphs using Python

Randy Heiland and Von Welch
CACR, Indiana University, Bloomington, Indiana

Center for Applied Cybersecurity Research
http://cacr.iu.edu/

CENTER FOR TRUSTWORTHY SCIENTIFIC CYBERINFRASTRUCTURE
http://trustedci.org/

Task: Analyze an authentication (authN) dataset with 708M authN events over a nine month period. An event is anonymized and represented as:

<time>, <U#>, <C#>

where <time> is in secs, offset from the start, <U#> is a User ID, and <C#> is a Computer ID.

Approach:
Install Anaconda Python (3.x) which includes:
pandas, NetworkX, matplotlib, IPython modules.

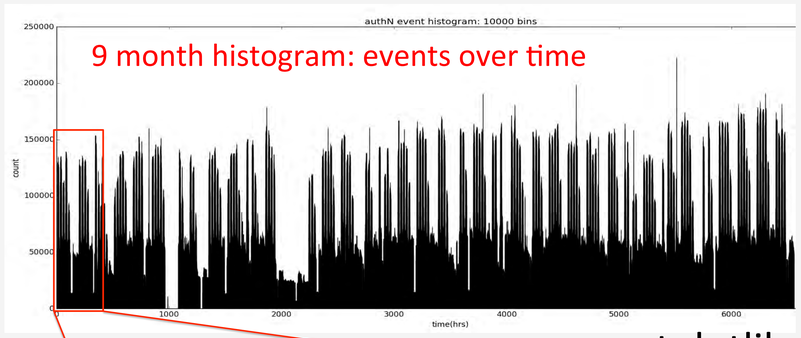
Download original data, graph, scripts:
http://csr.lanl.gov/data/auth/
http://trustedci.org/data
https://github.com/rheiland/authpy

Step 1: Use the pandas module to read the entire dataset, in chunks, and generate two files with:
① only time values
② the bipartite graph (as an adjacency list)

Step 2: Sanity check: read time values; plot a histogram, looking for a temporal pattern.

Step 3: Sanity check: read the graph, verify it's bipartite, and check # of Users and Computers.

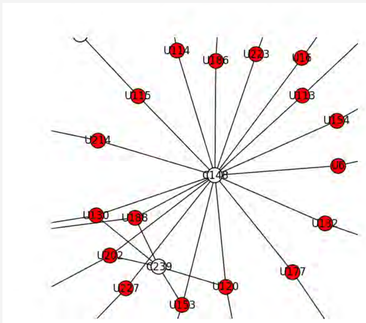
Step 4: Perform additional analysis/visualization of the graph, e.g. looking for hubs, connected components, etc.



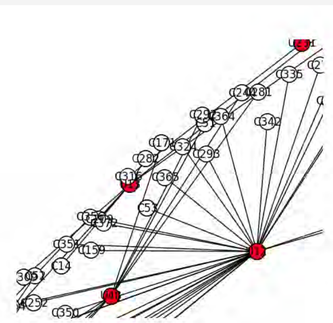
Graph G

```
import networkx as nx
from networkx.algorithms import bipartite

graph_filename = 'auth_graph_adjlist.dat'
G = nx.read_adjlist(graph_filename)

print('# nodes = '+str(len(G.nodes())))
print('# edges = '+str(len(G.edges())))

# nodes = 33644
# edges = 312283

"""Since G is bipartite, extract the 2 types
of nodes (users & computers)"""
cnodes,unodes = bipartite.sets(G)

print('  # users     = '+str(len(unodes)))
print('  # computers = '+str(len(cnodes)))

  # users     = 11364
  # computers = 22280
```

IPython Notebook



Computer hub: C148



User hub: U12

```
# Extract Hubs
node,deg=max(G.degree_iter(), key=itemgetter(1))
if node[0] == 'U':    # 'C' for computer hubs
   print(node, str(deg), end=', ')
   G.remove_node(node)
```

U8060 6724, U4075 2068, U8488 1344, …
C4692 9490, C11893 9453, C4691 9321, …

```
# Connected Components
for c in nx.connected_components(G):
   print('{0:d}: {1:d}'.format(num_conn, len(c)))
   num_conn += 1
```

30 connected components: one very large; others only 2 or 3 nodes.

Acknowledgements: LANL for providing the authN dataset. The Python community.
The NSF (OCI-1234408) for supporting this work.

A. Hagberg, A. Kent, N. Lemons, and J. Neil. Credential hopping in authentication graphs. In 2014 International Conference on Signal-Image Technology Internet-Based Systems. IEEE Computer Society, Nov. 2014.

A. D. Kent, L. M. Liebrock, and J. C. Neil. Authentication graphs: Analyzing user behavior within an enterprise network. Computers & Security, 48:150 – 166, 2015.