ANALYSIS OF AUTHENTICATION EVENTS AND GRAPHS USING PYTHON

Randy Heiland and Von Welch

SIAM Workshop on Network Science 2015
May 16-17 · Snowbird, UT

Summary

Discerning meaningful information from network log files is an ongoing challenge in cybersecurity. We demonstrate techniques for analyzing a large log of authentication events and associated graphs. Our approach is instructional and exploratory, using Python modules and tools.

Background

Cybersecurity analysts deal with a broad range of hardware, software, and data in their attempt to protect systems (and the users of those systems). As in other scientific fields, *big data* is a common theme. Monitoring and analyzing dynamic, real-time traffic on an enterprise network, e.g. a university, government lab, or large corporation, is one example. Analyzing a static, historical log of activity, looking for for suspicious patterns, is another. For example, users need to authenticate themselves to computers, via username/password, SSH keys, etc., before gaining access. By analyzing a historical log of authentication (authN) events on a computer, it may be possible to detect fraudulent activity, e.g. using stolen credentials.

The tools that are available to analyze cybersecurity-related data can be quite specialized or more general-purpose. For the work presented here, we will analyze a large authN dataset using the general-purpose Python programming language, together with some specialized modules for 1) reading the data, 2) generating and analyzing graphs, and 3) visualizing results. Our presentation offers a hands-on approach that could be adopted to either a classroom setting or an analyst's toolbox.

Authentication Data

Los Alamos National Laboratory (LANL) provides a large, publicly available authN dataset [1] that we use in this presentation. It consists of 708,304,516 successful authN events from users to computers over a period of nine months. The events are time-stamped with a resolution of 1 second and the ASCII dataset has been anonymized to consist of just three values per row: $\langle time \rangle, \langle U\# \rangle, \langle C\# \rangle$, where time is in seconds (integer offset from beginning) and the next two values are unique (integer-suffixed) identifiers for users and computers.

Python

Python is an interpreted, high-level programming language that is well-known to many science communities. It is easy to learn and provides many modules from a built-in standard library. For this work, we used the Anaconda Python distribution that also includes many useful 3rd party modules. IPython is a popular Python shell for improved interactivity. When used together with the matplotlib plotting library, a researcher can also generate valuable visual results. Finally, the IPython Notebook is a web-based environment that allows for analyses to be easily shared and reproduced. (Note: all the software used in our presentation is free and the scripts can be found at github.com/rheiland/authpy).

Analysis

We demonstrate some early results and highlight snippets of our Python scripts. The first obstacle to overcome was the sheer size of the authN dataset. One of our goals was to be able to perform the analysis on a desktop computer with a modest amount of memory. Thanks to the pandas Python module, it is possible to read and process the dataset in chunks. Therefore, our first script reads, in chunks, the entire dataset, extracts the time values and constructs a (global, static) bipartite graph of users and computers. The time values are written to a compact binary file. The graph is written to a compact adjacency list formatted file. Initially, we discard duplicate (at different times) authN events between a user and a computer. Later, we incorporate them and build a graph with weighted edges.

The time values for the entire dataset are displayed using a histogram (Figure 1). By using matplotlib's interactive zooming and panning features, we can explore the timeline more closely, looking for both normal patterns (Figure 2) and abnormal activity.

We use the NetworkX Python module to analyze the

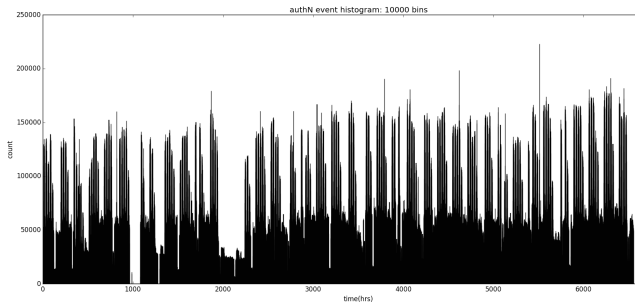

Figure 1: Histogram of authentication events' times.

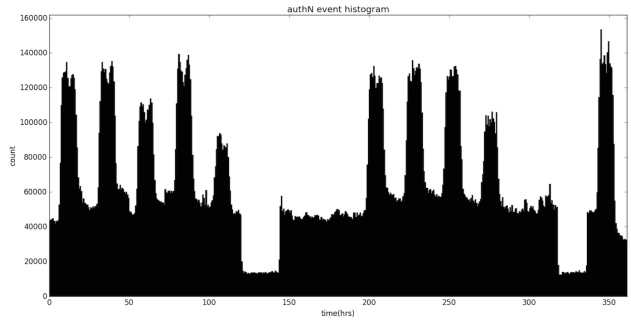

Figure 2: Zoom of a two week cycle.

user-computer bipartite graph. One of the first things we can easily check is how many total nodes (users and computers) and how many edges are in the graph:

```
G = nx.read_adjlist(graph_filename)
len(G.nodes())  # nodes= 33644
len(G.edges())  # edges= 312269
```

Next, an analyst would want to know how many distinct users and computers:

```
cnodes,unodes = bipartite.sets(G)
# -->  Computers= 22282, Users= 11362
```

As is the case in many graph applications, knowing which nodes are hubs, i.e. nodes with lots of edges, might be insightful (Figure 3). This is quite easy to do with NetworkX and Python (note: hubs with subtraction):

```
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, ...
```

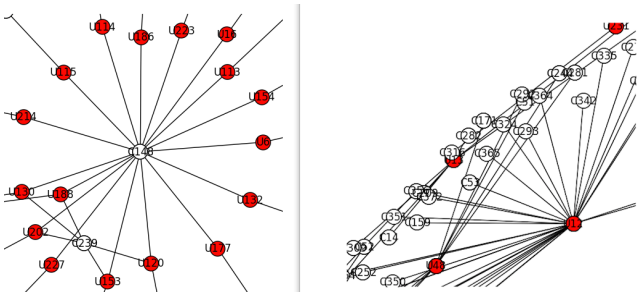

Figure 3: Two hubs: computer and user.

So it appears that user U8060 authenticated to over 6000 computers during the nine month period. If an analyst had access to more information, e.g. if U8060 was a system administrator, it might be possible to draw better conclusions.

Another graph theoretic concept that may be useful for analyzing the authN graph is to find all connected components, i.e. subgraphs in which it is possible to travel between any two nodes via edges. As Kent et al. [2] have discussed, increasing the number of connected components might make it more difficult for malicious actors to steal credentials. The following snippet reveals that the authN graph has 30 connected components: one that is very large (33582 nodes) and the remainder having either just 2 or 3 nodes.

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

Rather than analyze just the static, global graph, we are currently exploring the analysis of dynamic, temporal graphs from the dataset.

Acknowledgements

We are grateful to LANL for providing the authentication dataset and acknowledge the greater Python community for supporting the underlying software used here. We thank the National Science Foundation (grant 1234408) for supporting this work.

References

[1] 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 (SITIS)*. IEEE Computer Society, Nov. 2014.

[2] 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.