Identifying Rogue/Nefarious Applications

D. Lapsley, R. Walsh and W. T. Strayer

A security equipment vendor was recently describing their Network Intrusion Detection System. Of particular interest was a new technology the product implemented for detecting botnets. When asked how this technology worked, the vendor embarked on a lengthy explanation, the crux of which, was that the botnet detection algorithm used a port-based scheme that looked for TCP connections on port 6667.

While port-based application detection schemes certainly have their uses and their advantages (they are easy to implement and easy to understand), for detecting rogue/nefarious[1] applications, they are sadly lacking. This is not a new problem, and much work has been done in this area. We offer a contribution towards rogue application identification, while providing a framework that can be expanded for the identification of more applications. Our initial motivation for this work was the detection of botnets.

Several techniques have been developed to automatically identify (and often classify) various types of communication streams. Some use clues from the traffic content. Dewes *et al.* [1] propose a scheme for identifying chat traffic that relies on a combination of discriminating criteria, including service port number, packet size distribution, and packet content. Sen *et al.* [5] use a signature-based scheme to discern traffic produced by several well-known P2P applications by identifying particular characteristics in the syntax of packet contents exchanged as part of the operation of the particular P2P applications.

Other flow classification approaches focus on the use of statistical techniques to characterize and classify traffic streams. Roughan *et al.* [4] use traffic classification for the purpose of identifying four major classes of service: interactive, bulk data transfer, streaming, and transactional. They investigate the effectiveness of using packet size and flow duration characteristics, and simple classification schemes were observed to produce very accurate traffic flow classification.

In a similar approach, Moore and Zuev [3] apply variants of the Naive Bayesian classification scheme to classify flows into 10 distinct application groups. The authors also search through the various traffic characteristics to identify those that are most effective at discriminating among the various traffic flow classes. By also identifying highly correlated traffic flow charac-teristics, this search is also effective in pruning the number of traffic flow characteristics used to discriminate among traffic flows. Highly correlated characteristics provide comparable and, often, redundant information about the traffic flows. Thus, in many cases it suffices to use only one of the correlated characteristics to discriminate among traffic flows.

These techniques all operate on the same fundamental principle: identify characteristics application traffic features (through statistical means or through machine learning), train on sample data, and then use these characteristics to classify applications. The system we propose builds on this principle. We use standard classification and machine learning techniques to identify flows that are of interest as candidate rogue application flows. We then use additional techniques that look at the spatial and temporal similarity between flows and the topology of the graph formed by these flows to identify rogue application flows.

Here we present a system for aggregating monitored traffic looking for evidence of botnets so that botnet-based attacks can be attributed to a botnet controller. We adopt a proactive approach by

[1]**dlapsley:** We will simply refer to them collectively as “rogue” applications.

identifying hosts that are likely part of a botnet. Yet, this is not the same as definitively identifying the botnet and its controller; there is always uncertainty until there is an actual attack.

Our approach follows a set of increasingly more complex analyzers, filtering out unlikely flows along each step, so that the most computationally intensive analysis is done on a dramatically reduced traffic set. First, individual flows are subjected to a series of filters and classifiers to eliminate as many of the flows as possible, while being somewhat conservative so that botnet flows are not likely to be eliminated. Next, the flows are correlated with each other, looking for groups of flows that may be related by being part of the same botnet. The result is a group of flows that are most likely part of one or more botnets. Finally, the topological information in the flows are examined for the presence of a common communication hub.

In order to generate traffic that was representative of real botnet traffic, we implemented a benign bot based on the "Kaiten" bot, a widespread bot that has readily downloadable source code. The Kaiten bot was implemented in C using approximately 1000 lines of code. We reverse engineered the Kaiten software and then reimplemented it. In order to obtain traces of actual botnet traffic, we constructed a botnet test facility. This facility involved a simple setup modeled along the lines of a chat-based botnet architecture. Our setup involved an IRC server, a code server, 13 zombies, and an attacker. A host within BBN's network was used as the victim. We then used this test facility to obtain actual traces of the communications between the various botnet entities while the botnet was in operation. This data was then integrated with a 4 month traffic trace from the Dartmouth campus collected under their CRAWDAD project [2]. The data were then passed into our system to see whether or not our system could identify the ground truth botnet flows from the background Dartmouth traffic.

Our system performs gross, simple filtering to reduce the amount of data that will be subjected to more computationally intensive algorithms. Once the data has been filtered, the flows are classified using machine learning techniques, then the flows that are in the "chat" class are correlated to find clusters of flows that share similar timing and packet size characteristics. The cluster is then analyzed to try to identify the botnet controller host.

Our experiment with Dartmouth campus data, starting with nearly 9 million flows augmented with traffic traces from a benign botnet, shows that the ground truth botnet C2 flows can indeed survive the data reduction and correlation to be identified as a cluster. These results show that the method is promising. This method is also nicely suited for real-time analysis of traffic data. The filtering stage requires very simple logic to cull the data set down by a factor of 37. This culling of the data, especially when done in real time, allows much more time for more complex algorithms later in the pipe, namely the machine learning classifiers and the correlation.

Detecting botnet activity is presently labor intensive and largely *ad hoc*. Our pipelined botnet C2 detection system shows that it is possible to comb through packet traces, even in real time, to extract evidence of tight command and control activity.

While it has been suggested that botnet controllers will migrate from IRC as their preferred C2 infrastructure, the abstract model of tight central control represented by IRC is very efficient and will likely survive for quite some time. Future work will consider a system that detects very large, high volume data sets for evidence of tight botnet C2 activity. Future work will also investigate the extension of our system to include more application classes.

References

[1] Christian Dewes, Arne Wichmann, and Anja Feldmann. An analysis of internet chat systems. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 51–64, New York, NY, USA, 2003. ACM Press.

[2] David Kotz and Tristan Henderson. CRAWDAD: A Community Resource for Archiving Wireless Data at Dartmouth. *IEEE Pervasive Computing*, 4(4), oct-dec 2006.

[3] Andrew W. Moore and Denis Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 50–60, New York, NY, USA, 2005. ACM Press.

[4] Matthew Roughan, Subhabrata Sen, Oliver Spatscheck, and Nick Duffield. Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 135–148, New York, NY, USA, 2004. ACM Press.

[5] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 512–521, New York, NY, USA, 2004. ACM Press.