"Botnet Buster": Thwarting Botnets with Discriminative Boosted Bayesian Networks

Brian Ricks
The University of Texas at Dallas
absolutefunk@utdallas.edu

Alexander L. Hayes
The University of Texas at Dallas
alexander.hayes@utdallas.edu

Abstract

The authors present the application of learning Discriminative Boosted Bayesian Networks to the problem of detecting botnet activity on a network from the CTU-13-Dataset, and compare their results with those of traditional machine learning approaches both with and without expert knowledge. To our knowledge, this is the first application of statistical relational learning on a domain such as this.[1]

Introduction

Distributed network attacks have been a thorn in the side of the internet since its early days. In addition to denial-of-service, they are also responsible for spreading spam, malware, and even have a hand in data exfiltration and theft. In spite of much effort in the research and white hat communities, these attacks are more prevalent today than ever.

Distributed network attacks are commonly spread by the use of *botnets*, which are a network of (often) hijacked computers for the purpose of accomplishing some nefarious goal, such as flooding a web server with page requests. While reducing the frequency of these attacks is ideal, the evolution of attack strategies in response to current detection and mitigation techniques requires an approach which can generalize to newer, potential unobserved distributed attacks.

However, given the growth of distributed attacked within the past 10 years, how well do these approaches generalize to the current threat landscape? This question has implications for practically every aspect of the machine learning pipeline: data collection, feature extraction, which model to learn, and which algorithm to train the model?

We focus on the problem of learning a generalizable model from *sparse* botnet data, which often arises in the real-world. This normally occurs due to the overall time period spent collecting network traffic compared to the duration in which the botnet was active. In spite of this sparsity, careful feature engineering should help prevent data overfitting and allow for some acceptable measure of accuracy for many different off the shelf learning algorithms. We hope that careful selection of our learning algorithm will enable model generality on related data.

Our experiments use the *CTU-13-Dataset*, which is "a dataset of botnet traffic that was captured in the CTU University, Czech Republic, in 2011" (Garcia et al. 2014). This dataset comprises 13 different runs utilizing multiple disjoint botnets. Most alluring to us is the sparsity of the botnet examples; a vast majority (>97%) of the examples are from normal traffic flows. Also of importance is the use of multiple botnets over the runs which make up the datasets. Such diversity is invaluable to experiment with generality across multiple similar botnets, in which some of the botnets may not be observed during training.

The rest of the paper is structured as follows: In the related work section, we discuss related work that is both relevant to the general field of intrusion detection and to the subset of botnet detection. Also presented is work related to why domain knowledge is important in feature engineering. Next we dive into statistical relational learning and related work that is relevant to our problem definition and approach. The experimental section showcases results from the feature engineering stage, and we wrap up the paper with a thorough discussion on inherent limitations which prevented full results from our chosen approach to reach fruition.

Related Work

Traditional Botnet Detection

As the number of people and devices accessing the internet increases, the need to deal with problems such as intrusion detection, denial of service attacks, and botnets increase as well.

Many approaches have applied traditional machine learning techniques for botnet and general distributed denial-of-service classification tasks, often with good results on specific network datasets. Earlier work tended to focus on general intrusion detection tasks, often in a temporal environment (Ye 2000; Ourston et al. 2003; Joshi and Phoha 2005; Xu, Sun, and Huang 2007). In this setting, intrusions were modeled as sets of events ordered by time, with each set of events serving as a single example for either normal or abnormal behavior. Markov chains and hidden Markov networks are two popular models to learn in this setting.

[1] All code and LaTeX discussed in this paper are available online: https://github.com/batflyer/Botnet-Buster

In a non-temporal setting, examples tend to be individual events, with each event corresponding to one or more features that describe the event. For network security related intrusion tasks, these features are commonly those found in net flow data: source and destination IP address and port, number of packets comprising the flow, total bytes send or received, direction of flow, etc. Early work in this setting trained models such as mixture models (Puttini, Marrakchi, and M 2002), Naive Bayes (Bringas and Penya 2009), Bayesian networks (Kruegel et al. 2003; Xu and Shelton 2010), random forests (Zhang, Zulkernine, and Haque 2008), and decision trees (Osanaiye et al. 2016).

Early work specifically to botnet detection focused more on the *command and control (c&c)* flows (Livadas et al. 2006; Gu, Zhang, and Lee 2008; Cho et al. 2010; Bilge et al. 2012; Dietrich, Rossow, and Pohlmann 2013), which is traffic generated by a botnet for node coordination. A comparison of C4.5, naive Bayes, and Bayesian network learners to classify botnet c&c traffic from normal traffic was spotlighted in (Livadas et al. 2006). The Bayesian network results especially seemed to highlight the issue of model overfitting to a given training set when confronted with test data that may have originated from the same botnet over different runs. We imagine this problem is compounded when factoring in test data from similar botnets.

More recent work has applied a random forest learner to a large scale botnet dataset captured by UC San Diego (Singh et al. 2014). Most of their work went into the feature engineering and underlying distributed framework to handle the shear amount of data present, but the results were promising. Feature engineering specifically related to botnet c&c channels were discussed in (Alejandre, Corts, and Anaya 2017), using C4.5 as the learning algorithm. Their approach for feature engineering used a genetic algorithm and exhaustive search to select important features from a list of non-temporal features generated through aggregation of temporal flow data. Given the small feature space to search (19), and given their results for features selected, we feel that domain knowledge can reproduce such a set of important features.

One drawback to much of the previous work in this field deals with how well a trained model represents reality. For example, the dataset used in (Zhang, Zulkernine, and Haque 2008) was released in 1999, almost a decade before the paper was published. The network and threat landscape changed considerably during that time. In (Xu and Shelton 2010), the span of time between the dataset they used and publishing of their paper was 12 years. More recently, (Alejandre, Corts, and Anaya 2017) used a dataset released in 2011.

Another factor to consider is the feature engineering itself. Plug-and-play methods may do well on a single dataset with minimal domain knowledge, but what happens when applying the trained model to other datasets showcasing the same class of threats? Is the generality there? How much domain knowledge is needed to maintain a baseline of generality? (Ben-Asher and Gonzalez 2015) took a deep look into this problem and, perhaps unsurprisingly, domain experts are quite important during the feature engineering task. This suggests that a plug-and-play type approach to botnet detection may not yield the best results, but rather a combination of a domain expert and a 'model' expert might be preferable.

Statistical Relational Learning

The imbalance between the potential number of positive and negative examples makes this a potential problem to approach from the perspective of statistical relational learning (SRL). The authors apply the state-of-the-art statistical relational learning system: BoostSRL[2] based on the relational functional gradient boosting algorithm (Natarajan et al. 2015). Gradient boosted tree learners generally set up the problem in the form of learning a series of regression trees, where each tree is a relatively weak learner that fits toward correcting the error of the previous.

BoostSRL (an implementation of the *Relational Functional Gradient Boosting* algorithm) has been applied to a variety of real world domains; including from identifying Parkinson's patients, predicting the onset of postpartum depression, and recommending jobs to potential applicants (Dhami et al. 2017; Natarajan et al. 2017; Yang et al. 2017).

The authors build on the work of (Ramanan et al. 2017; Yang et al. 2014) for learning discriminative boosted Bayesian networks. Because statistical relational models may operate over data with a massive imbalance between the number of positive and negative examples, and different costs for classification of each, explicitly tweaking the cost function for this problem is essential. Usually the cost function is tweaked explicitly to set the trade-off between false negatives and false positives– since in recommendation systems the goal is to have high precision, but medical applications high recall is more desirable–but explicitly deciding on the trade-off in the botnet problem is not as obvious. If the goal is to identify bots with high precision, some may not be marked as bots under cases of uncertainty; if high recall is desired, some humans may also be labeled as bots. This point is explored by tweaking alpha and beta values in our experiments.

Experiments

We answer four questions: (1) How do standard machine learning techniques perform on this task? (2) Do relational learning techniques–notably discriminative boosted Bayesian networks–provide better generalization? (3) Can we make general observations about alpha and beta values for tweaking the cost of false positives or false negatives in practice? (4) Given the general knowledge, can we retroactively tweak the standard machine learning techniques to produce superior results? (5) What is the value of expert knowledge in such a task?

[2] https://github.com/starling-lab/BoostSRL

Results

CTU-13-Dataset		
Algorithm	Training Accuracy	Testing Accuracy
BN_CI	100.0%	0.0%
BN_CI2	99.9%	0.1%
BN_CI3	99.6%	0.4%
BN_Tabu	100.0%	0.0%
BN_Tabu2	99.9%	0.1%
BN_Tabu3	99.5%	0.5%
Random Forest	100.0%	0.0%
Naive Bayes	95.5%	4.5%

Before diving into the model training, we need to remove features of our original set which will probably not generalize. Eight Weka (Witten et al. 2016) algorithms were used to perform these baseline experiments, using all the features in our set. Since the goal is to learn a general method for detecting botnet activity, each of the thirteen CTU "tasks" were treated as a fold, and we report the average accuracy when the classifier is trained on one task and tested on another.

As reflected in these results, there are present features which are not generalizing to other runs within the dataset. Now the task becomes one of removing the 'bad' features from the set. Determining such features can be accomplished through domain knowledge and analysis of the Bayesian network structures learned from the table above.

In terms of domain knowledge, we observe that features such as source IP address will naturally overfit to a training set due to the implicit assumption this feature makes that all botnets will originate from the same set of source IP addresses. Clearly this is not true, even for the same botnet. For example, botnets which utilize source IP spoofing, even the same botnet will appear to come from different sets of source IP addresses over different experimental runs.

In analyzing the Bayesian network structures learned from the table above, we observe that given the single class tree-based networks learned, the children (feature) nodes have more influence to the class the closer they are to the class in terms of edges. So direct children will hold more influence than grandchildren. Indeed, source IP address (`SrcAddr`) was a direct child in all the models learned. This doesn't necessarily mean the feature is bad, just that it will hold a lot of influence, so we need to look at all such features that are direct children. One feature that falls into this boat, destination IP address (`DstAddr`), should also be removed given that a botnet may not always attack the same destination, and will not generalize to data collected from other network topologies. Removing destination IP address as a feature is an example of applying domain knowledge after using other techniques to point in a certain direction.

One interesting feature that usually is a direct influence, but which does not generalize at all is start time (`StartTime`). When a botnet begins its attack, relative to the start of collecting data, is very much determined by the specific dataset run. Outside of the dataset, it's practically irrelevant, but sometimes can give the illusion of being a good feature depending on exactly when the attack begins relative to whatever else is going on at that point in time. And that's the key, the probably of other events sharing the exact timestamp of the start of the attack is rare.

Other features which hold direct influence to the class, but which should generalize well, are destination port (`Dport`) and flow duration (`Dur`). For the former, botnets typically will target a specific port on a victim server for a specific protocol (say http), so destination port should generalize to most botnets of the same attack class. The same argument can be used for keeping flow duration as well.

The next step is finding an appropriate ordering of the features we keep, which is a prerequisite for using discriminative boosted Bayesian networks as a learning method. The binetflow in the CTU-13 Dataset supplies the following fourteen labels:

`StartTime`, `Dur`, `Proto`, `SrcAddr`, `Sport`, `Dir`, `DstAddr`, `Dport`, `sTos`, `dTos`, `TotPkts`, `TotBytes`, `SrcBytes`, `Label`

Of these labels, the target is the value in the `Label` column, and after eliminating the features mentioned above, the variable ordering becomes: `Dport`, `Dur`, `TotBytes`, `TotPkts`, `SrcBytes`, `Proto`, `Dir`, `Sport`.

Limitations

The code is implemented, but we do not have the results for any of the thirteen tasks yet. When a task is converted to the appropriate predicate-logic format, there are around 25 million facts and several million positive and negative examples. Regardless of how much computing power is thrown at it, BoostSRL appears to hang while reading the facts and does not recover.

There are several possible ways around this. The variable ordering we are currently using may be adapted such that we use even fewer variables, iteratively removing one variable (starting from the end deemed "least relevant") until we have something that can be computed. A more efficient representation of the facts may be possible–currently we have discretized the positive and negative examples into "bot or not", but similar discretizations may be possible for the facts may reduce the overall number of groundings that need to be reasoned about. If neither of these accomplish our goals, a more powerful learning and inference framework may need to be considered.

Conclusion

While the experiments still may need considerable work, the authors have presented a novel approach toward detecting botnet activity on a network–as far as we know, this is the first application of statistical relational learning to this domain.

Acknowledgements

The authors would like to thank Professor Sriraam Natarajan, Professor Gautam Kunapuli, and their classmates in the Spring 2018 Statistical Relational Learning Seminar (CS 7301.002) for their comments and general feedback as this paper came together.

References

Alejandre, F. V.; Corts, N. C.; and Anaya, E. A. 2017. Feature selection to detect botnets using machine learning algorithms. In *2017 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, 1–7.

Ben-Asher, N., and Gonzalez, C. 2015. Effects of cyber security knowledge on attack detection. *Computers in Human Behavior* 48:51 – 61.

Bilge, L.; Balzarotti, D.; Robertson, W.; Kirda, E.; and Kruegel, C. 2012. DISCLOSURE: Detecting botnet command and control servers through large-scale netflow analysis. In *ACSAC 2012, 28th Annual Computer Security Applications Conference, December 3-7, 2012, Orlando, Florida, USA*.

Bringas, P. G., and Penya, Y. K. 2009. Next-generation misuse and anomaly prevention system. In Filipe, J., and Cordeiro, J., eds., *Enterprise Information Systems*, 117–129. Berlin, Heidelberg: Springer Berlin Heidelberg.

Cho, C. Y.; Babi ć, D.; Shin, E. C. R.; and Song, D. 2010. Inference and analysis of formal models of botnet command and control protocols. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, 426–439. New York, NY, USA: ACM.

Dhami, D. S.; Soni, A.; Page, D.; and Natarajan, S. 2017. Identifying parkinsons patients: A functional gradient boosting approach. In *Conference on Artificial Intelligence in Medicine in Europe*, 332–337. Springer.

Dietrich, C. J.; Rossow, C.; and Pohlmann, N. 2013. Cocospot: Clustering and recognizing botnet command and control channels using traffic analysis. *Computer Networks* 57(2):475 – 486. Botnet Activity: Analysis, Detection and Shutdown.

Garcia, S.; Grill, M.; Stiborek, J.; and Zunino, A. 2014. An empirical comparison of botnet detection methods. *Computers & Security* 45:100–123.

Gu, G.; Zhang, J.; and Lee, W. 2008. Botsniffer: Detecting botnet command and control channels in network traffic. In *NDSS*.

Joshi, S. S., and Phoha, V. V. 2005. Investigating hidden markov models capabilities in anomaly detection. In *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 1*, ACM-SE 43, 98–103. New York, NY, USA: ACM.

Kruegel, C.; Mutz, D.; Robertson, W.; and Valeur, F. 2003. Bayesian event classification for intrusion detection. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, 14–23.

Livadas, C.; Walsh, R.; Lapsley, D.; and Strayer, W. T. 2006. Usilng machine learning technliques to identify botnet traffic. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, 967–974.

Natarajan, S.; Kersting, K.; Khot, T.; and Shavlik, J. 2015. *Boosted statistical relational learners: From benchmarks to data-driven medicine*. Springer.

Natarajan, S.; Prabhakar, A.; Ramanan, N.; Bagilone, A.; Siek, K.; and Connelly, K. 2017. Boosting for postpartum depression prediction. In *Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2017 IEEE/ACM International Conference on*, 232–240. IEEE.

Osanaiye, O.; Cai, H.; Choo, K.-K. R.; Dehghantanha, A.; Xu, Z.; and Dlodlo, M. 2016. Ensemble-based multi-filter feature selection method for ddos detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking* 2016(1):130.

Ourston, D.; Matzner, S.; Stump, W.; and Hopkins, B. 2003. Applications of hidden markov models to detecting multistage network attacks. In *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, 10 pp.–.

Puttini, R. S.; Marrakchi, Z.; and M, L. 2002. Bayesian classification model for real-time intrusion detection. In *In 22th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*.

Ramanan, N.; Yang, S.; Grannis, S.; and Natarajan, S. 2017. Discriminative boosted bayes networks for learning multiple cardiovascular procedures. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 870–873. IEEE.

Singh, K.; Guntuku, S. C.; Thakur, A.; and Hota, C. 2014. Big data analytics framework for peer-to-peer botnet detection using random forests. *Information Sciences* 278:488 – 497.

Witten, I. H.; Frank, E.; Hall, M. A.; and Pal, C. J. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Xu, J., and Shelton, C. R. 2010. Intrusion detection using continuous time bayesian networks. *J. Artif. Int. Res.* 39(1):745–774.

Xu, X.; Sun, Y.; and Huang, Z. 2007. Defending ddos attacks using hidden markov models and cooperative reinforcement learning. In *Proceedings of the 2007 Pacific Asia Conference on Intelligence and Security Informatics*, PAISI'07, 196–207. Berlin, Heidelberg: Springer-Verlag.

Yang, S.; Khot, T.; Kersting, K.; Kunapuli, G.; Hauser, K.; and Natarajan, S. 2014. Learning from imbalanced data in relational domains: A soft margin approach. In *Data Mining (ICDM), 2014 IEEE International Conference on*, 1085–1090. IEEE.

Yang, S.; Korayem, M.; AlJadda, K.; Grainger, T.; and Natarajan, S. 2017. Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach. *Knowledge-Based Systems* 136:37–45.

Ye, N. 2000. A markov chain model of temporal behavior for anomaly detection. In *In Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, 171–174.

Zhang, J.; Zulkernine, M.; and Haque, A. 2008. Random-forests-based network intrusion detection systems. *Trans. Sys. Man Cyber Part C* 38(5):649–659.