Journal of Naval Science and Engineering
2013, Vol.9, No.2, pp. 1-11

A REAL-TIME SYSTEM FOR ABUSIVE NETWORK TRAFFIC DETECTION

Georgios KAKAVELAKIS, Lt.Cdr.

Hellenic Navy General Staff, Hellenic Republic
gekakavelakis@gmail.com,gkaki@yahoo.com

Abstract

Abusive network traffic—to include unsolicited e-mail, malware propagation, and denial-of-service attacks—remains a constant problem in the Internet. Despite extensive research in, and subsequent deployment of, abusive-traffic-detection infrastructure, none of the available techniques addresses the problem effectively or completely. The fundamental failing of existing methods is that spammers and attack perpetrators rapidly adapt to and circumvent new mitigation techniques. Within this framework, we develop a real-time, online system that integrates transport layer characteristics into the existing SpamAssasin tool for detecting unsolicited commercial e-mail (*spam).*

Keywords: *Network traffic, spam, machine learning, system analysis, auto-learning*

1. INTRODUCTION

Electronic mail (e-mail) is one of the most popular applications of the Internet, enabling users to easily communicate by exchanging electronic messages at no upfront cost, quickly, reliably and easily. Unfortunately, this popular communication media has been exploited [1]. Many methods have been proposed to address the increasing problem of spam, such as content filtering, rule-based and learning-based systems. Spammers, however, adapt accordingly and find countermeasures, such as fake IP [2] addresses or compromised hosts, also known as botnets, to evade blacklisting. Traffic-characterization studies [3, 4, 5] try to address these issues by examining network characteristics associated with spam behavior at the IP and TCP [6] level. These techniques are promising

since it is more difficult for spammers to thwart such characteristics by manipulating the IP or TCP layer.

2. RELATED WORK

Traffic-characterization methods are a recent novel approach to differentiating sources of abusive traffic. Several prior works are directly relevant to our research. These methods try to identify spam by leveraging the network or transport-layer properties. Whereas spammers have the ability to alter the content of a message or spoof an IP address or sender domain, they have much less power to forge network (e.g., IP) or transport-level (e.g., TCP) properties.

Ramachandran et al. [7] examine the spamming behavior at the network layer (IP layer) focused on network-level properties such as: IP address space, autonomous systems (AS), BGP route announcements. Another approach on traffic characterization was proposed by Schatzmann et al. [8] focused on the network-level characteristics from the perspective of an AS or ISP. Their idea is based on the assumption that a large number of e-mail servers perform some level of pre-filtering (e.g., blacklisting). Hao et al. [9], however, showed that AS alone as a feature may cause a large rate of false positives. Further studies [4, 10] have shown that a spammer can evade this technique by advertising routes from a forged AS number [9].

In a spirit similar to Ramachandran et al., Beverly and Sollins [3] explored transport-layer characteristics in order to determine whether spam e-mail presents different behavior from legitimate e-mail. Their idea is based on the premise that spammers have to send large volumes of e-mail to be effective, which suggests that the network links involved would experience contention and congestion. Therefore, transport-layer properties such as number of lost segments and the roundtrip time (RTT) would have different metrics in such a

contentious environment, allowing discrimination between spam and legitimate behavior. Moreover, Ouyang et al. [5] conducted a large-scale empirical analysis of transport-layer characteristics on 600K+ messages, based on the work of Beverly and Sollins. Performance-wise, they showed that transport-layer features are stable over time and can classify spam with 85–92% accuracy.

3. SYSTEM ARCITECTURE

An overview of our real-time system is shown in (Figure 1). It comprises four main components: SpamAssassin, SpamFlow Analysis Engine, SpamFlow Plug-in, and the SpamFlow Classification Engine. We refer to SpamFlow Analysis Engine, SpamFlow Plug-in, and SpamFlow Classification Engine as *spamflow*, *plugin*, and *classifier*, respectively. Furthermore, we have a separate process running in promiscuous mode, which captures every packet of the SMTP [11] session using libcap and stores it to disk.

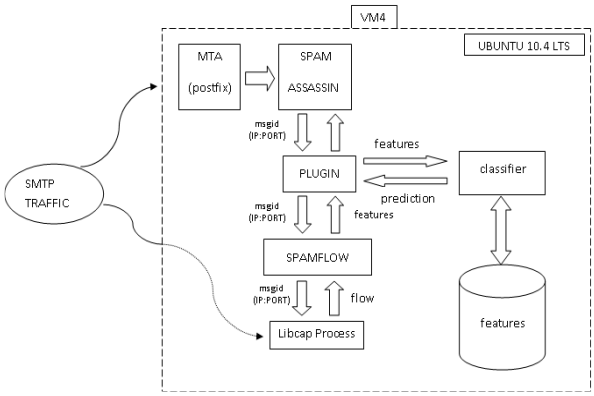

Figure 1. System Architecture

3.1. Spamassassin

SpamAssassin [12] is an open-source, rule-based, content filter. Each rule is assigned a score using a genetic algorithm. All scores are then aggregated to produce an overall score for each message. The classification

process involves comparing the overall score with a user-defined threshold and if the score is above the threshold, then the message is classified as spam; otherwise, as legitimate. Moreover, using a modular architecture, SpamAssassin can be extended to include other filtering techniques, such as real-time blackhole lists (RBLs), whitelists, collaborative filtering, learning-based techniques (e.g., naïve Bayes), and others.

3.2. Spamflow

Spamflow [3] serves as our network analyzer. It promiscuously, using libpcap [13], listens on the network interface, builds corresponding flows, and extracts TCP features for each flow. We modified *spamflow* for our purposes to extract TCP features for a given message identified by the (host IP address, host port number) tuple. Finally, we modified our mail server to add to the header of each e-mail the (IP address, TCP port number) identification tuple of the remote mail-transport agent (MTA) [14] sending the mail.

3.3. Spamflow Plugin

Spamflow cannot operate as a standalone application for real-time traffic analysis therefore, we developed, using Perl [15], a module that integrates spamflow into SpamAssassin and allows it to operate in a real-time fashion. It performs two main tasks that are related to spamflow and classifier. The first task is to provide spamflow with the 2-tuple identifier of the current message under inspection and receive in return the features that correspond to the given message identifier. Once plugin obtains the features, the second task involves classification: passing the features, via XML-RPC [16] , over to the classifier and retrieving the corresponding classification.

3.4. Spamflow Classification Engine

As mentioned above, we have implemented *classifier* using Python [17] and the Orange [18] machine-learning package. Our *classifier* implementation comprises three machine-learning algorithms: naïve Bayes [19,20], decision trees (C4.5) [21], and support-vector machines (SVM) [22]. We selected three algorithms because we wanted to examine if the classification performance of our system is a function of the classification method, and these algorithms are known to provide good performance.

4. RESULTS

This section describes the results from our system evaluation in both a virtual test bed and a live-test environment. A virtual test bed provides insights about the behavior of a system and allows for more controllable conditions, allowing us to reach more reliable and reproducible results. Live testing, on the other hand, is important because it reveals how the system interacts with possibly unknown features of the external environment [23].

4.1. Test Bed Evaluation

An overview of our virtual environment is shown in (Figure 2). It consists of three building blocks: the client side, server side, and network emulator. The client side generates, through an e-mail replayer, the required SMTP [12] traffic, which is then received, analyzed, and classified on the server side.

The replayer reads from the TREC public spam corpus [24] containing 92,187 messages, of which 52,788 are spam and 39,399 are legitimate. For each message the replayer sets the type of service (tos) field in the IP header of each

message to some value, depending on its class. Thus spam and legitimate messages have different tos values, which allows us to redirect them through different paths in our network emulator, and finally transmits the message, after having established an SMTP session with our mail server.

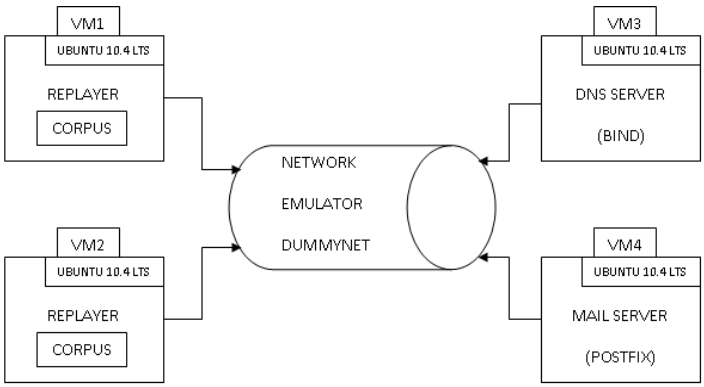

Figure 2. Virtual System Architecture

The role of the network emulator is to simulate congestion, in the form of longer delay, delay variance, retransmissions, etc., that large volumes of spam traffic will cause on the link. In our case, our goal is to reproduce the TCP characteristics that spam TCP traffic exhibits, such as TCP timeouts, retransmissions, resets, and highly variable roundtrip time (RTT) estimates [3]. For our evaluation, we selected Dummynet [25], a publicly available tool that allows packets to pass through virtual network links to introduce delay, loss, bandwidth constraints, queuing constraints, etc.

Table 1 shows how the three classifiers performed with respect to training times and classification throughput. Examining the results, we observe that naïve Bayes provides the higher throughput among the three classifiers, and this conforms to the fact that its decision rule is much simpler than the other two, whereas C4.5 has the lowest training time. SVM, on the other hand,

achieves the lowest throughput and the largest training time, due to the more complex decision model.

	Training Times (msec) across samples				Throughput (msgs/sec)
	10	100	1000	10000	
Bayes	0.884	15.016	105.453	104.843	1300
C4.5	0.151	0.964	16.017	29.785	1100
SVM	0.721	12.691	224.250	260.018	700

Table 1**.** System Performance

The significant takeaway from these measurements is that, taking into account the relative independence of our system from the classification method, we can select the classification model that fit our needs. For example, the low training time of C4.5 makes it a good candidate when we need to retrain often and want to minimize idle times.

4.2. Live Testing

We deployed our system in a live environment for a selected domain from January 25, 2011 to March 2, 2011, where we collected a trace of 5,926 e-mail messages. Ground truth was first established via SpamAssassin. We then manually examined all the legitimate messages and relabeled those that were false negatives. Even though the volume of traffic captured is small and represents a small portion of the Internet traffic, the results with respect to accuracy, precision, and recall were strong.

Auto-learning is the incremental process of building the classification model based on exemplar e-mail messages that achieve certain threshold values. In our case, we use the flow features of e-mail messages otherwise classified via orthogonal methods as having very high or very low scores. More specifically, we explicitly retrain each classification model each time we

observe a message with a particularly high score from the other SpamAssassin categories (rule- and Bayesian-word based) that meets our threshold criteria; i.e., having a score above or below our threshold. After retraining is complete, we evaluate our models on subsequent messages until we observe one or more messages with scores above or below our thresholds, at which point we stop and retrain the models.

We set up two thresholds: one for spam messages and one for legitimate based on the spam and ham score distributions, which proved effective as it allowed the classifiers to be trained on total of 2,685/5,510 (48.7%) spam and 267/416 (64.2%) ham messages. Figure 3 shows the classification performance of the three classifiers as a function of cumulative training samples received.

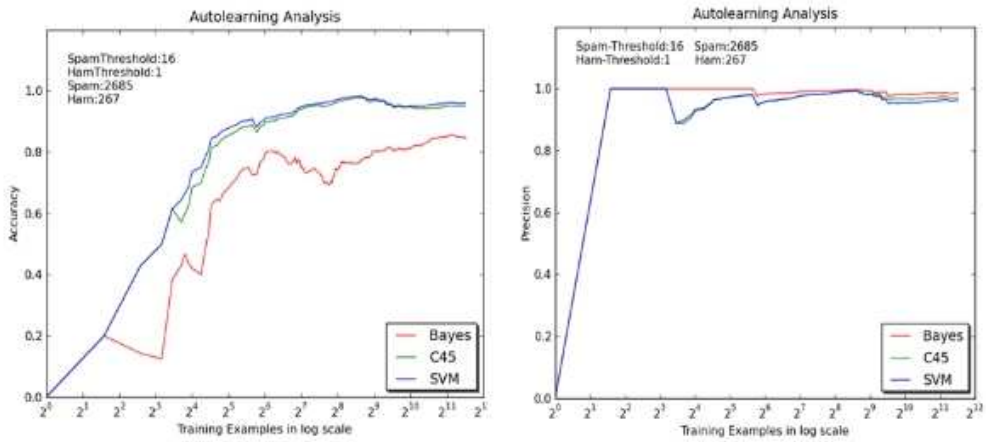

Figure 3. Autolearning Analysis

We observe a gradual improvement in the performance in all metrics, with C4.5 and SVM achieving constant high rates above 95% in accuracy, precision, and recall with as few as 1,024 (2^{10}) training examples. While accuracy is interesting, it is essential that a spam classifier have a very low false

positive rate. For this reason we also show precision results for each classifier. In other words, of those emails identified as spam, what fraction truly are. With as few as 64 (2^6) training examples, all reached constant high precision rates above 95%.

To better understand the sensitivity of our auto-learning results to the imposed thresholds, we experiment with a spam threshold three deviations above mean. By increasing the spam threshold, the SpamFlow auto-learning uses fewer spam-training examples. However, we expect to have higher confidence in their true disposition of spam with the higher threshold. With the spam score threshold raised to 40, we trained the classifiers with only 30 spam training examples many fewer than the number of ham examples. The results are interesting with the precision rate remaining above 97% across all classifiers with as few as six training flows.

5. CONCLUSIONS

In this research, we implemented the necessary infrastructure to perform real-time, on-line transport-layer classification of email messages. We detail the system architecture to integrate network transport features with SpamAssassin, an MTA, and a classification engine. Our testing reveals that the system can handle realistic traffic loads. Using our techniques, we achieve accuracy, precision, and recall performance greater than 95 percent after receiving only $\approx 2^{10}$ messages during live, real-world production testing.

We note, however, that our live-testing corpus is small. Our intent in this work was to demonstrate the practical feasibility of using transport network traffic features. In future work, we plan to investigate SpamFlow's performance in large, production systems against much larger volumes of traffic. Our hope is to enable the practical deployment of transport-layer based abusive traffic detection and mitigation techniques to system administrators.

REFERENCES

[1] Messaging Anti-Abuse working Group (MAAWG), "Email metrics program: The network's operator perspective," Tech. Rep. 13, November. 2010. Available: http://www.maawg.org/sites/maawg/files/news/MAAWG_2010-Q1Q2_Metrics_Report_13.pdf.

[2] J. Postel. (1981, September). Internet protocol. Internet RFC 791 Available: http://www.faqs.org/rfcs/rfc791.html.

[3] R. Beverly and K. Sollins, "Exploiting transport-level characteristics of spam," in *CEAS 2008 - Fifth Conference on Email and Anti-Spam*.

[4] X. Zhao, D. Pei, L. Wang, D. Massey and A. Mankin, "An analysis of BGP multiple origin AS (MOAS) conflicts." in *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement (IMW)*, 2001.

[5] T. Ouyang, S. Ray, M. Allman and M. Rabinovich, "A Large-Scale Empirical Analysis of Email Spam Detection through Transport-level Characteristics," *Technical Report 10-001, International Computer Science Institute.*, January 2010.

[6] Postel. (1981, September). Transmission control protocol. *Internet RFC 793* Available: http://www.ietf.org/rfc/rfc793.txt.

[7] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers." in *Proceedings of ACM SIGCOMM*.

[8] D. Schatzmann, M. Burkhart and T. Spyropoulos, "Inferring spammers in the network core," in *Passive and Active Conference*, Seoul,South Corea, 2009, pp. 229–238.

[9] S. Hao, N. A. Syed, N. Feamster, A. G. Gray and S. Krasser, "Detecting spammers with SNARE: Spatio-temporal network-level automatic reputation engine." in *In Proceedings of the 18th Conference on USENIX Security Symposium.*

[10] J. Karlin, S. Forest and J. Rexford, "Autonomous security for autonomous systems." *Computer Networks*, vol. 52, pp. 2908–2923, 2008.

[11] J. Klensin. (2001, April). Simple mail transfer protocol. *Internet RFC 2821 (Standards Track)* Available: http://www.ietf.org/rfc/rfc2821.txt.

[12] J. Mason. Filtering spam with SpamAssassin. Presented at HEANet Annual Conference. Available: http://wiki.apache.org/spamassassin/PresentationsAndPapers.

[13] V. Jacobson, C. Leres and S. McCanne, "Packet Capture Library (pcap)," vol. 1.0.0, October 27, 2010.

[14] D. Crocker, "Mail transfer agent," in *Internet RFC 5598-Internet Email Architectur*e pp. 31.

[15] W. Larry. Perl. Available: http://perldoc.perl.org/.

[16] D. Winer. (1998, April). XML-RPC specification. Available: http://www.xmlrpc.com/spec.

[17] G. Van Rossum. Python. Available: http://www.python.org/

[18] Laboratory of Artificial Intelligence, Faculty of Computer and Information Science, University of Ljubljana, Slovenia, "Orange: A Component Based Machine Learning Library for Python," vol. 2.0, 2010.

[19] R. O. Duda and P. E. Hart, "Bayes decision theory," in *Pattern Classification and Scene Analysis*Anonymous John Wiley & Sons, 1973, pp. 10.

[20] T. M. Mitchell, *Machine Learning.* McGraw-Hill, 1997.

[21] J. R. Quinlan, "C4.5: Programs for Machine Learning," 1993.

[22] V. N. Vapnik, "Estimation of Dependencies Based on Empirical Data," 1992.

[23] M. Carbone and L. Rizzo, "Dummynet Revisited," *ACM SIGCOMM Computer Communication Review*, vol. 40, pp. 12–20, April 2010.

[24] G. Cormack and T. Lynam. 2005 TREC public spam Corpus.

[25] L. Rizzo, "Dummynet: A Simple Approach to the Evaluation of Network Protocols," *ACM Computer Communication Review*, vol. 27, pp. 31–41, 1997.