Using Machine Learning Techniques to Identify Botnet Traffic

Carl Livadas Robert Walsh David Lapsley W. Timothy Strayer
Internetwork Research Department
BBN Technologies
{clivadas,rwalsh,dlapsley,strayer}@bbn.com

Abstract

To date, techniques to counter cyber-attacks have predominantly been reactive; they focus on monitoring network traffic, detecting anomalies and cyber-attack traffic patterns, and, a posteriori, combating the cyber-attacks and mitigating their effects. Contrary to such approaches, we advocate proactively detecting and identifying botnets prior to their being used as part of a cyber-attack [12]. In this paper, we present our work on using machine learning-based classification techniques to identify the command and control (C2) traffic of IRC-based botnets — compromised hosts that are collectively commanded using Internet Relay Chat (IRC). We split this task into two stages: (I) distinguishing between IRC and non-IRC traffic, and (II) distinguishing between botnet and real IRC traffic.

For Stage I, we compare the performance of J48, naive Bayes, and Bayesian network classifiers, identify the features that achieve good overall classification accuracy, and determine the classification sensitivity to the training set size. While sensitive to the training data and the attributes used to characterize communication flows, machine learning-based classifiers show promise in identifying IRC traffic. Using classification in Stage II is trickier, since accurately labeling IRC traffic as botnet and non-botnet is challenging. We are currently exploring labeling flows as suspicious and non-suspicious based on telltales of hosts being compromised.

1. Introduction

One of the most vexing cyber-security threats today is the use of very large, coordinated groups of hosts for brute-force attacks. These large groups of hosts are assembled by turning vulnerable hosts into so-called *zombies*, or *bots*, after which they can be controlled from afar. A collection of bots, when controlled by a single command and control (C2) infrastructure, form what is called a *botnet*. Botnets obfuscate the attacking host by providing a level of indirection and separating the assembly of the botnet and its use for attack by an arbitrary amount of time. Botnets, often involving thousands of hosts, are increasingly being used to launch highly-effective cyber-attacks.

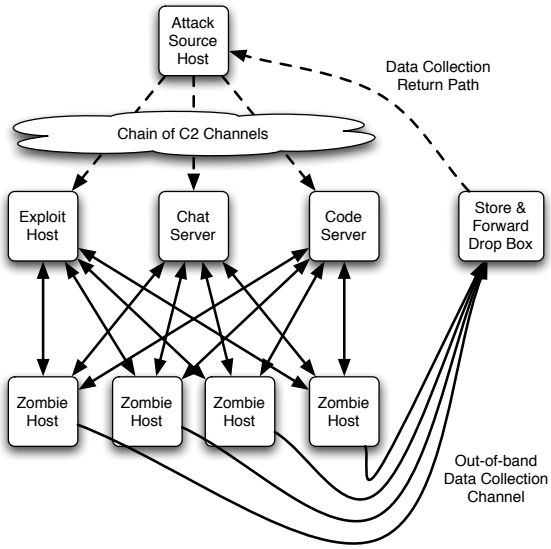

Figure 1. Chat-Based Botnet Architecture

Figure 1 depicts a high-level block diagram of a botnet that uses a chat service as the C2 channel, the *de facto* C2 channel of choice of recent botnets [4,5,7,8,13]. The exploit host initially compromises the zombies using well-known host vulnerabilities. Subsequently, it instructs the zombies to subscribe to a particular chat session. Finally, this chat session is used to command the zombies into performing coordinated tasks, such as fetching an executable from a designated *code server* and running it at a particular point in time.

Despite the development of various reactive techniques to detect and trace a slew of cyber-attacks, detecting and tracing the origin of botnet-based attacks is in its infancy. Reactive techniques involve detecting the attack while it is in progress and promptly tracing it to its true origin. Since

the setup of a botnet and its use to launch a cyber-attack may be separated by arbitrarily long intervals of time, reactive traceback techniques are not directly applicable.

Advocating a proactive approach to botnet-based attack attribution, we propose detecting and identifying botnets prior to their being used as part of a cyber-attack [12]. This task can be broken down into: (1) detecting flows that likely comprise botnet C2 traffic, (2) correlating these flows to identify groups of flows that pertain to the same botnets, (3) identifying the C2 hosts, which are one logical step closer to the hosts instigating the attacks.

In this paper, we present our preliminary work on using machine learning-based classification techniques to carry out the first sub-task. We use two-stage approach. In Stage I, we classify communication flows (*i.e.*, TCP connections) observed as either chat or non-chat flows. Here, the underlying assumption is that the C2 flows of chat-based botnets indeed resemble real chat flows. We argue that this is the case for two reasons: (1) botnet C2 channels have, to date, been observed to exchange text-based commands that are of comparable length to that of real chat messages, and (2) in order for botnet flows not to alarm chat server administrators, they must roughly resemble real chat connections. Thus, we expect that a classifier that is trained to differentiate between chat and non-chat flows will classify botnet chat flows as chat flows.

Using classification in Stage II is trickier. We are currently investigating whether machine learning-based classifiers can be used to distinguish between botnet and real IRC flows. Here, the underlying assumption is that botnet IRC flows are, in some subtle respects, different than real IRC flows. The challenge here is to accurately label IRC flows as either botnet or real IRC flows. Because the traces available to us are anonymized and their payloads have been discarded, labeling IRC traffic as either botnet or real IRC traffic is challenging. Nevertheless, we are exploring the possibility of labeling flows as either suspicious or non-suspicious using telltales of hosts being compromised.

One of the challenges of using machine learning techniques is the need for *ground truth*, an accurately labeled data set that can be used for purposes of training (and testing). Obtaining ground truth is particularly challenging. For Stage I — that of discerning chat from non-chat flows — obtaining ground truth entails being able to label flows as either chat or non-chat. In the advent of payload encryption and the use of random ports for communication, very little information can be leveraged to carry out this labeling. We focus on botnets that use Internet Relay Chat (IRC) as the C2 channel. Since most IRC servers use the default IRC port and IRC traffic is unencrypted, the default IRC port and the packet payload (if available) can be used in identifying IRC traffic.

For Stage II — that of discerning botnet from chat flows — obtaining ground truth is more elusive. General telltales of botnet flows are, to our knowledge, not publicly available. We are currently exploring two approaches of labeling IRC flows are either botnet or real IRC flows. First, we build and use a botnet testbed based on our own safe re-implementation of the Kaiten botnet [6, 13]. The IRC flows from our testbed can thus be used both for training and testing purposes. Second, we use telltales of hosts being compromised to label flows as suspicious and non-suspicious. This work still in progress.

2. Background

Several techniques have been developed to automatically identify and, often, classify communication streams [1, 9–11]. Dewes *et al*. [1] propose a scheme for identifying chat traffic. Their approach relies on a combination of discriminating criteria, including service port number, packet size distribution, and packet content. Sen *et al*. [11] use a signature-based scheme to discern traffic produced by several well-known P2P applications. Their approach relies on identifying particular characteristics in the syntax of packet payloads exchanged as part of the operation of the particular P2P applications, and then using these characteristics to discern the traffic of each application. The recent trend toward using non-standard ports and encryption may reduce the effectiveness or, even, prevent the use of these techniques.

Others [9, 10] have proposed using statistical techniques to characterize and classify traffic streams. Roughan *et al*. [10] use traffic classification to identify the class of service (CoS) of traffic streams and, thus, enable the on-the-fly provision of distinct levels of quality of service (QoS). The authors attempt to classify traffic streams into four major traffic classes: interactive, bulk data transfer, streaming, and transactional. Moreover, a multitude of traffic statistics can be used to classify flows and these statistics may pertain to either packets, flows, connections, intra-flow, intra-connection, or multi-flow characteristics. Roughan *et al*. investigate the effectiveness of using average packet size, RMS packet size, and average flow duration to discriminate among flows. Given these characteristics, simple classification schemes produced very accurate traffic flow classification.

In a similar approach, Moore and Zuev [9] apply variants of the naive Bayesian classification scheme to classify flows into 10 distinct application groups. They 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 characteristics, this search is also effective in pruning the number of traffic flow characteristics used for classification. 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.

3. Description of Data

3.1. Real-Life Traces

We use a set of network traffic traces collected from Dartmouth's wireless campus network [3]. Wireless traffic sniffers were used to collect complete TCP/IP headers of all packets transmitted over a period of four months (11/1/2003–2/28/2004) from 18 locations around campus, including academic, library, residential, and social buildings. The traces are anonymized (in terms of the source and destination IP addresses) and contain no payload information.

These traces are valuable in many respects. First, the number of hosts monitored and the collection duration makes the traces good candidates for including a wide variety of traffic. Second, the variety in the placement of the wireless sniffers ensure the collection of a variety of activities, *e.g.*, activities carried out either in the privacy of one's dorm, or in the anonymity of a library. Thus, even if the traffic one is looking for is rare, there is a good chance that the traces will contain enough of this traffic to perform statistical analysis and characterization.

Given the proliferation of packet payload encryption, building a botnet identification system that relies upon parsing payload data is not wise. However, had it been available, payload data could be used to label flows more accurately and to evaluate the accuracy of other labeling schemes.

3.2. Testbed Traces

We set up a botnet testbed modeled on the chat-based botnet architecture of Figure 1. As depicted in Figure 2, our setup involved an IRC server, a code server, 13 zombies running a safe reimplementation of the Kaiten botnet code [6, 13], an attacker, and a victim host.

We used this testbed to obtain actual traces of the communications between the various botnet entities. Our experiments entailed using the IRC server to instruct the zombies to download attack code from the code server and to subsequently launch a coordinated UDP attack on the victim host. The traces involved IRC traffic between the bots and the IRC server, http traffic between the zombies and the code server (for downloading the attack code), and the UDP traffic involved in the coordinated UDP-attack on the victim host. The setup and the launch of the attack were successively repeated in order to increase the amount of trace data collected.

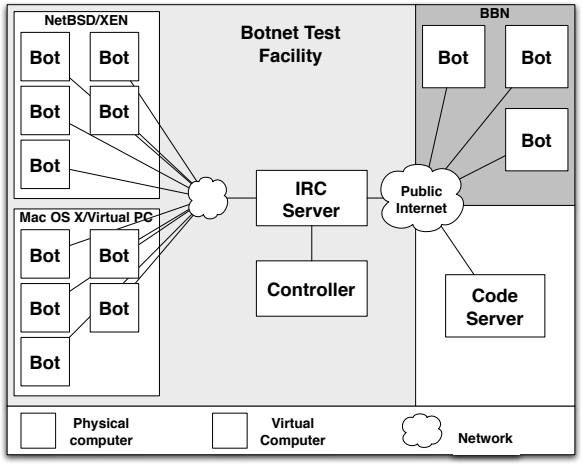

Figure 2. Botnet Testbed Architecture

4. Trace Pre-filtering

4.1. From Packets to Flows

Since IRC-based botnets use TCP, we retain TCP packets and discard all others (UDP, ICMP, etc.). We characterize flows using attributes based on TCP and IP packet headers. These can be interpreted even if the encapsulated payload is encrypted. The headers contain detailed information, some of which is only relevant to the networking stack. Other information may be too application- and OS-specific to be used consistently across flows.

Table 1 summarizes the flow characteristics that we collected for each of the flows in the traffic traces we used in our work. These include the cumulative application payload size, the IP protocol type (TCP), the IP source and destination addresses, the source and destination ports, and TCP flags. Moreover, we record flow start and end times, packet counts, byte counts, statistics for variance, client/server role for the connection (as indicated by the initial three-way-handshake of TCP), and a histogram of application payload sizes (this adds a finer grain breakdown of the distribution of packet sizes of each flow). For experimental purposes, we also recorded the packet counts associated with TCP push and maximum window size.

4.2. Heuristic Flow Filtering

We use a set of heuristics crafted to discard flows that are unlikely to be botnet flows. These heuristics are very effective in reducing the total number of flows considered in the subsequent classification stages of our work.

We eliminate all port-scanning activity from the data set — flows containing only TCP Syn or TCP Rst indicate that communication was never established. These provide no information about chat or C2 flows.

Table 1. Traffic Flow Characteristics

start/end	Flow start/end times
IP-proto	IP protocol of flow
TCP flags	Summary of TCP SYN/FIN/ACK flags
pkts	Total pkts exchanged in flow
Bytes	Total Bytes exchanged in flow
pushed pkts	Total packets pushed in flow
duration	Flow duration
maxwin	Maximum initial congestion window
role	Whether client or server initiated flow
Bpp	Average Bytes-per-packet for flow
bps	Average bits-per-second for flow
pps	Average packets-per-second for flow
PctPktsPushed	Percentage of packets pushed in flow
PctBppHistBin0–7	Percent of packets in one of eight packet size bins; these variables collectively form a histogram of packet size for flow
varIAT	Variance of packet inter-arrival time for flow
varBpp	Variance of Bytes-per-packet for flow

Peer-to-peer file sharing is a significant load on the Internet and may take place on chat ports by co-incidence (since the chat port is not reserved) or by intent (to avoid identification and filtering). We dropped "high bandwidth" flows, thus eliminating software updates and rich web page transfers. This elimination is more significant for the non-chat subset of flows and serves to focus subsequent machine learning modeling techniques on the more important area of overlap between either IRC and non-IR, or botnet and real IRC flows.

Finally, we eliminated short-lived flows — flows of only a few packets or a few seconds. These do not correspond to bots that are standing by "at the ready."

4.3. Filtered Traces

In our work presented in this paper, we use two sets of flows: the set obtained after filtering the flows from a particular residential building from the Dartmouth traces and that obtained after filtering the flows collected from a particular botnet experiment in our botnet testbed facility. We henceforth refer to these flow sets as the *Dartmouth* flows (or trace) and the *Testbed* flows (or trace), respectively. The *Dartmouth* trace involves 227784 flows, 7343 of which were IRC flows (as dictated by the use of the default 6667 IRC port). The *Testbed* trace involves 74 flows, 38 of which were IRC flows.

5. Results

In this section, we present our work on using machine learning-based classifiers to identify IRC-based botnet C2 flows. We first classify flows into IRC and non-IRC flows; then, among the flows identified as IRC flows, we distinguish between authentic IRC and botnet flows.

All the results in this paper were obtained using the WEKA ML toolbox [14]. We use the flow characteristics in the lower part of Table 1 as the initial set of flow features/attributes. We do not use the characteristics in the upper part of the table for classification purposes — they either are inconsequential in classifying flows, or correspond to accumulated quantities, which are indirectly captured by the corresponding rates or percentages and the flow duration.

We use the false negative rate (FNR) and the false positive rate (FPR) to evaluate the performance of the classifiers considered. The relative importance of each of these metrics depends on the ultimate use of the classification results. A low FNR guarantees that only a small fraction of the IRC/botnet flows will be discarded during our botnet identification process. A low FPR guarantees that the set of flows identified as IRC/botnet will not be infested by non-IRC/botnet flows. In the first stage of our approach — that of identifying IRC traffic — we expect traffic traces to involve a large number of non-IRC/botnet flows. A low FPR would thus be beneficial in cutting down the number of flows that need to be examined during the second stage — that of discerning botnet from chat flows. In Stage I, we evaluate the classifiers considered in terms of the FNR in identifying our botnet testbed IRC flows. We consider our botnet testbed IRC flows to be representative examples of actual botnet IRC flows. Thus, their accurate identification as IRC traffic is crucial in allowing their consideration in Stage II.

5.1. Stage I: Identifying Chat Traffic

We explore the effectiveness of machine learning-based classification in identifying chat traffic in three dimensions: the classification scheme, the subset of characteristics/features used to describe the flows, and the size of the training set size.

In this stage, we use the *Dartmouth* trace to evaluate the performance of classifying flows as IRC and non-IRC and use the *Testbed* trace to determine whether the classifiers accurately identify the IRC flows of the *Testbed* trace as IRC flows; that is, the *Dartmouth* trace is used for both training and testing while the *Testbed* trace is only used for testing. Unless otherwise specified, when using the *Dartmouth* trace we used half the flows for training and the rest for testing the resulting classifiers.

5.1.1. Varying the Classification Scheme

We first compared the performance of three classification schemes, namely J48, naive Bayes, and Bayesian networks. J48 is the WEKA [14] implementation of C4.5 decision trees [2]. In this model, the classification is performed using a decision tree in which each internal node corresponds to a test on one or more attributes and each leaf corresponds

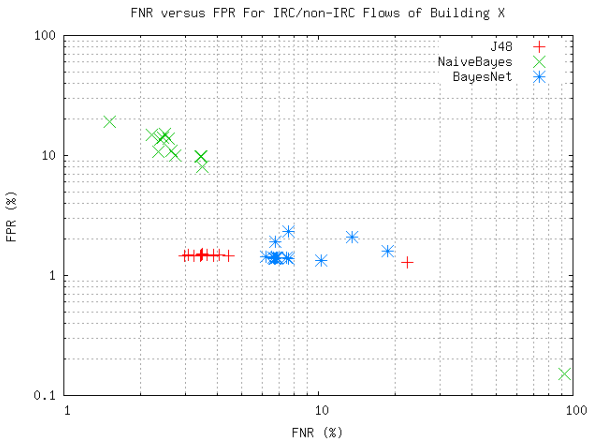

Figure 3. FNR and FPR of J48, Naive Bayes, and Bayesian Net Classification Schemes for IRC/non-IRC Flows of the *Dartmouth* Trace

to a decision outcome. Naive Bayes classifiers presume that the features describing a particular sample are independent. Thus, the maximum *a posteriori* probability that a sample belongs to the class C is equal to the product of the prior probability for C and each of the conditional feature probabilities for the given sample. The Bayesian networks technique uses a directed acyclic graph to capture the dependence among sample features. Classification of samples is carried out based on this graphical representation of the conditional probability distributions of the sample features.

Figure 3 depicts the FNR vs. FPR scatter plot for several runs of J48, naive Bayes, and Bayesian networks for the labeled *Dartmouth* trace. Each data point corresponds to a different subset of the initial flow attribute set. Figure 3 reveals clustering in the performance of each of three classification techniques. Naive Bayes seems to have low FNR, but higher FPR. The Bayesian networks technique seems to have low FPR, but higher FNR. J48 seems to strike a balance between FNR and FPR.

Only the naive Bayes classifiers were successful in achieving low FNR in the case of our botnet testbed IRC flows — notably, one of our naive Bayes classifiers accurately classified 35 out of the 38 botnet testbed IRC flows, thus achieving an FNR of 7.89%. In contrast, the J48 and the Bayesian networks classifiers, possibly tuned too tightly to the training set, performed very poorly. Since the naive Bayes classifier is the only one that showed potential in accurately classifying our botnet testbed IRC flows, it would be preferable to the J48 and Bayesian network classifiers.

5.1.2. Varying Flow Characterization Attribute Sets

The classification accuracy of the three classification schemes that we investigated in the previous section depends heavily on the set of attributes used to characterize flows. We investigated which of the attribute sets provide the most flow differentiation benefit. We considered three different approaches. First, we use the decision trees obtained from several runs of J48 to identify the attributes with the most differentiating power. Then, we explore the attribute selection and exploration functionality of WEKA [14]. Finally, we explore the attribute sets according to our intuitive understanding of how IRC flows should differ from other types of flows; for instance, we expect the average packet size of IRC flows to be smaller than that of other types of traffic, such as HTTP.

Table 2. Attributes Used at Different Levels of J48 Decision Tree

Attribute	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
duration						
maxWindow						
role					√	
Bpp	√	√			√	
bps						√
pps						
PctPktsPushed			√			√
PctBppHistBin0					√	√
PctBppHistBin1				√		√
PctBppHistBin2						
PctBppHistBin3						
PctBppHistBin4						
PctBppHistBin5						
PctBppHistBin6			√			
PctBppHistBin7						
varIAT						√
varBpp				√		

Our first approach to evaluating the differentiation power of particular attributes involves examining the decision trees produced by J48. By construction, attributes that appear higher up in the decision tree have more discriminatory power. Thus, by examining a particular decision tree we can identify the attributes of more discriminatory power.

Table 2 depicts the levels of the decision tree in which each of the flow attributes appears for an example use of J48 in classifying the *Dartmouth* flows as either IRC or non-IRC flows. A simple examination of the table reveals the flow attributes that have the most discriminatory power in this case, namely the Bytes-per-packet (Bpp), the % of packets that are pushed (PctPktsPushed), and the variance in the Bytes-per-packet (varBpp).

After training a number of J48 classifiers on a variety of attribute sets and training sets, we observed a great variation in the resulting J48 trees. This inconsistency in identifying the most discriminatory attributes indicates that this approach may not, on average, be accurate. Nevertheless, it may be used as a crude technique for identifying which attributes are more important than others. Indeed,

Table 3. Search-Based Optimal Attribute Sets

Attribute	Baseline Attribute Set			Optimal Attribute Set		
	J48	NB	BN	J48	NB	BN
duration	√	√	√	√	√	√
maxWindow	√	√	√	√	√	√
role	√	√	√	√	√	√
Bpp	√	√	√	√	√	√
bps	√	√	√	√	√	√
pps	√	√	√	√	√	√
PctPktsPushed	√	√	√	√	√	√
PctBppHistBin0	√	√	√	√	√	√
PctBppHistBin1	√	√	√	√	√	
PctBppHistBin2	√	√	√		√	√
PctBppHistBin3	√	√	√	√		
PctBppHistBin4	√	√	√		√	
PctBppHistBin5	√	√	√		√	
PctBppHistBin6	√	√	√			
PctBppHistBin7	√	√	√			
varIAT	√	√	√			
varBpp	√	√	√			
FNR (%)	3.07	2.65	6.66	3.23	3.23	6.74
FPR (%)	1.48	11.00	1.40	1.47	8.05	1.39
Testbed Trace FNR (%)	100	68.42	100	100	100	100

our examination of several J48 trees revealed that the Bytes-per-packet (Bpp) and the variance in the Bytes-per-packet (varBpp) often appear high up in the J48 decision trees. Thus, they are of particular discriminatory value.

WEKA provides several search techniques for exploring the attribute space. We used exhaustive searches starting from the initial attribute set search and using the classification performance of the respective classification schemes as the search performance metric. Table 3 depicts the attribute sets obtained by running such exhaustive searches for the J48, naive Bayes, and Bayesian network classifiers. Table 3 includes the FPR and FNR achieved by the baseline and optimal attribute sets for each of these classification techniques.

In the case of the J48 and the Bayesian network classifiers, the performance difference between the initial and optimal attribute sets is negligible. In the case of naive Bayes, the optimal attribute set trades off FNR and FPR. Moreover, the optimal attribute set performs worse in identifying the testbed botnet traces.

Lastly, we explored selecting attributes based on how we expect IRC traffic to differ from other types of traffic. For instance, because of the nature of chat, we expect packets to involve the transmission of small sentences involving a small number of characters. Thus, we expect chat traffic to involve small packets compared to other services, such as long ftp transfers, that predominantly use MTU-size packets. Another example is the variance of the number of bytes per packet. Since in the case of chat, packets correspond to chat exchanges, we expect the variance in the packet sizes to be large. In contrast, we expect the variance of the packet size for other applications, such as bulk transfers, to be quite small.

Table 4. Intuition-Based Attribute Sets

Attribute	Baseline Attribute Set	Intuition-Based Attribute Sets		
duration	√			√
maxWindow	√			
role	√	√	√	√
Bpp	√	√	√	√
bps	√	√	√	√
pps	√	√	√	√
PctPktsPushed	√	√	√	√
PctBppHistBin0–7	√	√	√	√
varIAT	√		√	√
varBpp	√		√	√
FNR (%)	2.65	2.46	2.21	2.49
FPR (%)	11.00	14.17	14.73	15.04
Testbed Trace FNR (%)	68.42	47.37	18.42	7.89

We identified the flow duration as a useful attribute. Apart from differentiating flows according to how long they persist, the duration and the average rates of the various flow characteristics indirectly capture the absolute flow characteristics that we excluded from our initial attribute set; for instance, the duration and the average Bps capture the total number of bytes transmitted during the given flow.

Table 4 presents the performance of several of the intuition-based attribute sets that we investigated. Once again we present the FPR and FNR, for each of the attribute sets evaluated. Due to space limitations, we only present the results for the Naive Bayes classification scheme.

Table 4 reveals a low variability in the FNR and FPR rates achieved with respect to the attribute set for the flows of the *Dartmouth* trace. However, the attribute set does affect the FNR achieved in the case of the testbed botnet flows; the FNR dropped from the initial 68.42% to 7.89%. Since a low FNR for our botnet testbed IRC flows is critical, these results demonstrate that a careful selection of the flow attributes used is crucial.

The far right column of Table 4 indicates that solely excluding the maxWindow flow attribute results in a classifier that retains a higher percentage of the *Testbed* IRC flows. It follows that the maxWindow used for IRC flows in the *Dartmouth* trace (which is used for training the classifiers) is different than that in the *Testbed* trace. Indeed, our *Testbed* IRC flows are among unix-based machines, while we conjecture that the IRC flows in the *Dartmouth* trace involve predominantly Windows machines. This example demonstrates how important it is to train on flows that are indeed prototypical of what one is looking to identify.

5.1.3. Varying the Training Set Size

We evaluated the sensitivity of the naive Bayes classification scheme on the size of the training set. The experiments in this section were performed as follows. We started off

with all the *Dartmouth* flows, set aside 10% of the flows for testing, and used the remaining 90% for generating training sets of different sizes. These training sets were random subsets of the flows set aside for training. Moreover, they involved 1–9% and 10-100% of the number of flows set aside for training, resulting in 19 random training sets. We repeated this process 10 times, generating 10 different testing sets and 10 different sets of training sets. We used the attribute set appearing on the far right of Table 4; this attribute set was the one achieving the lowest FNR for our testbed botnet flows.

Figure 4 is a scatter plot of the FNR and FPR as a function of the training set size. The mean FNR remains quite constant throughout the range of training set sizes that we used. Moving from the left of the graph to the right, there are three distinct regions of the graph. For the smallest training set, the FNR is a bit higher. For the middle range, the variance of the FNR grows but there are some runs that perform distinctly better. In these runs, the training sets represent the respective testing sets well. For large training sets, the FNR has values of higher mean but lower variance.

The observed effect of the training set size on the FPR is slightly different. The mean FPR increases slightly, while the variance decreases with the increase with the training set size. This indicates that in the case of FPR, smaller training set sizes can be used and, indeed, they result in lower FPRs.

These FNR and FPR statistics indicate that the benefit to using large training sets is low. In the case of our experiments, training sets on the order of 10K flows were adequate (and often preferable) for both the *Dartmouth* and the *Testbed* flows.

5.2. Stage II: Identifying Botnet Traffic

We are currently exploring two approaches of labeling IRC flows as either botnet or real IRC flows. First, we are using our botnet testbed to collect both real and botnet IRC flows and label them based on our knowledge of the experimental setup. Second, we are exploring the possibility of labeling flows as either suspicious or non-suspicious using telltales of hosts being compromised. For example, a host that is initiating port scans may be deemed compromised and its IRC flows may be labeled as suspicious. We are also looking into obtaining real-life traces that contain packet payloads and using these to positively identify botnet IRC through textual payload analysis. Once identifying a sufficiently accurate labeling scheme, we'll investigate whether machine learning-based classifiers can be used to distinguish between botnet and real IRC flows.

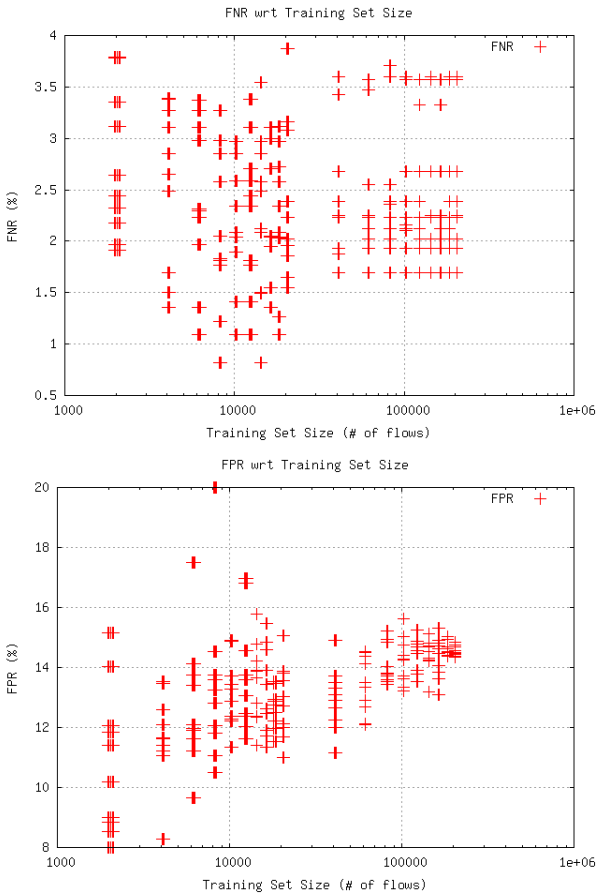

Figure 4. FNR and FPR of Naive Bayes classification with respect to training set size for the *Dartmouth* trace.

6. Conclusions

In this paper, we used machine learning techniques to identify C2 traffic of IRC-based botnets. We split this task into two stages: (I) distinguishing between IRC and non-IRC traffic, and (II) distinguishing between botnet and real IRC traffic. In Stage I, a naive Bayes classifier performed best, achieving both low FNR (2.49%) and low FPR (15.04%) for the *Dartmouth* flows and low FNR (7.89%) for the *Testbed* flows. While some J48 and Bayesian network classifiers performed better for the *Dartmouth* flows, they classified the *Testbed* IRC flows poorly. For the feature sets and the traces we used, we observed that training sets of 10K flows were sufficient and that the benefit of using larger sets was minimal. For Stage II, we are currently running more testbed experiments and collecting testbed botnet and real IRC flows to be used for training. We are also evaluating the use of telltales of hosts being compromised to label flows as suspicious and non-suspicious and to distinguish between botnet and real IRC flows.

Acknowledgments

We thank David Kotz for the Dartmouth traces and Mark Allman for his insightful comments on paper drafts.

References

[1] C. Dewes, A. Wichmann, and A. 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] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., 2 edition, 2001.

[3] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 187–201. ACM Press, September 2004.

[4] T. Holz. A Short Visit to the Bot Zoo. *IEEE Security & Privacy*, 3(3):76–79, May 2005.

[5] E. Levy. The Making of a Spam Zombie Army. *IEEE Security & Privacy*, 1(4):58–59, July 2003.

[6] McAfee. 2006/3/6; DDoS-Kaiten `http://vil.nai.com/vil/content/v_99371.htm`.

[7] B. McCarty. Automated Identity Theft. *IEEE Security & Privacy*, 1(5):89–92, Sept. 2003.

[8] B. McCarty. Botnets: Big and Bigger. *IEEE Security & Privacy*, 1(4):87–90, July 2003.

[9] A. W. Moore and D. 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.

[10] M. Roughan, S. Sen, O. Spatscheck, and N. 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.

[11] S. Sen, O. Spatscheck, and D. 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.

[12] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley. Detecting Botnets with Tight Command and Control. To Appear in 31st IEEE Conference on Local Computer Networks (LCN'06), 2006.

[13] The Honeynet Project. *Know Your Enemy : Learning about Security Threats*. Addison-Wesley Professional; 2 edition (May 17, 2004), Mar. 2004.

[14] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (2nd Edition)*. Morgan Kaufmann, San Francisco, CA, 2005.