

Using Machine Learning Techniques to Identify Botnet Traffic

Carl Livadas, Bob Walsh, David Lapsley, Tim Strayer
InterNetwork Research Department
BBN Technologies

Abstract—In this paper, we use machine learning techniques to identify the command and control 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 IRC traffic 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. A naive Bayes classifier performs best, achieving both low false negative (2.49%) and false positive (15.04%) rates for real-life IRC/non-IRC flows and low false negative (7.89%) rates for our botnet testbed IRC flows. While some J48 and Bayesian network classifiers perform better for real-life IRC/non-IRC flows, they classify our botnet testbed IRC flows poorly. For the feature sets and the traces we considered, we observed that training sets of 10K flows are sufficient and that the benefit of using larger sets is minimal.

Using classification in Stage II is trickier. Because the traces available to us are anonymized and stripped of their payload, labeling IRC traffic as botnet and non-botnet is challenging. We propose and explore an alternative approach for this labeling. We use telltales of hosts being compromised to label IRC traffic as suspicious and non-suspicious. We then evaluate the performance of naive Bayes classifiers on the thus labeled traffic. Although effective for the real-life suspicious/non-suspicious flows, the resulting classifiers performed poorly in classifying our botnet testbed IRC flows. We conclude that more accurate labeling, either by using our botnet testbed traffic, or by using more accurate botnet telltales, is crucial for this stage.

I. INTRODUCTION

One of the most vexing cyber-security threats today is the use of very large, coordinated groups of hosts for brute-force attacks, intrusions, and generating unsolicited emails. These large groups of hosts are assembled by turning vulnerable hosts into so-called *zombies*, after which they can be controlled from afar. A collection of zombies, also called *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. Their sheer power, however, is what has fueled their proliferation; botnets often involve thousands of hosts that can be collectively commanded to launch highly effective coordinated cyber-attacks.

Figure 1 depicts a high-level block diagram of a botnet. This is a crude depiction of the entities setting up and indirectly controlling a set of zombies using a chat service, 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.

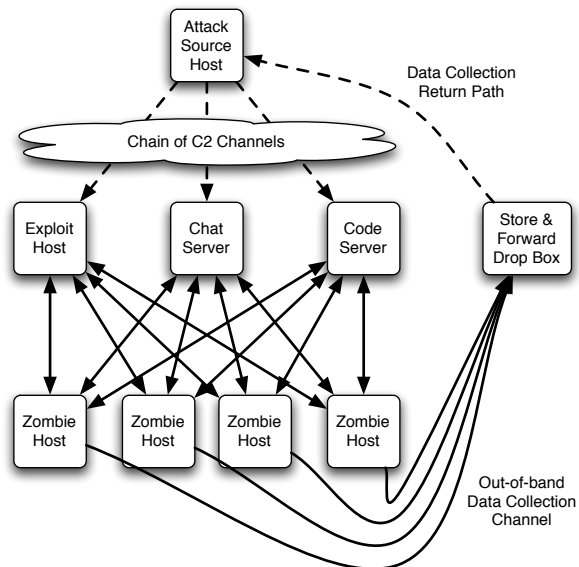


Fig. 1. Chat-Based Zombie Network Architecture

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

We have recently begun investigating techniques to detect and identify botnets prior to their being used as part of a cyber-attack. 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 host, which is presumably one logical step closer to the attack host.

In this paper, we present our preliminary work on using machine learning 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 among chat and non-chat flows will classify botnet chat flows as chat flows.

In Stage II, we classify chat flows as either botnet chat flows or real chat flows. Here, the underlying assumption is that botnet chat flows are, in some subtle respects, different than real chat flows. We thus investigate whether these subtle differences can be modeled by machine learning techniques.

Our two stage approach is preferable to a one stage classification step. The first stage is used to drastically reduce the number of flows that are candidates of being botnet flows. This allows the classifiers in the second stage to train on a smaller number of flow types, *i.e.*, botnet chat and real chat flows, and achieve higher classification accuracy.

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 tackle this problem using two approaches. 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 demonstrate how other telltales may be used for labeling flows for Stage II by presenting a trial example. Although our example may admittedly not be adequate, we use it to demonstrate an alternative approach to obtaining the elusive ground truth for Stage II. Given a set of IRC flows, we identify these flows that involve hosts that show signs of being compromised. In particular, we use evidence of the SubSeven trojan [10] as an indication of a compromised host. Although labeling of IRC flows into botnet and authentic IRC based on this indication of compromise may not be accurate, it splits the IRC flows into two subsets. If classifiers can indeed be used to discern among these two subsets of flows, then machine learning techniques should be capable of discerning among botnet and authentic IRC flows, were ground truth readily available.

II. BACKGROUND

Several techniques have been developed to automatically identify and, often, classify communication streams [1, 9, 11, 12]. Dewes *et al.* [1] propose a scheme for identifying chat traffic. Their approach relies on a combination of discriminative criteria, including service port number, packet size distribution, and packet content. Sen *et al.* [12] 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, 11] have proposed using statistical techniques to characterize and classify traffic streams. Roughan *et al.* [11] 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 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.

III. DESCRIPTION OF DATA

A. 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.

B. Testbed Traces

We set up a botnet testbed modeled on the chat-based botnet architecture of Figure 1. 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.

IV. DATA MODELING

A. Reduction of Packets to Flows

Since IRC-based botnets use TCP, we retain TCP packets and discard all others (UDP, ICMP, etc.). We characterize

TABLE I
TRAFFIC FLOW CHARACTERISTICS EVALUATED

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

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-dependent for us to rely upon. So, we only preserve application payload length, IP protocol type (TCP), IP source address, IP destination address, source port, destination port, and TCP flags. We also 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 data lengths (this adds a finer grain breakdown of the distribution of packet sizes within particular flows). For experimental purposes, we also included packet counts associated with TCP push and maximum window size.

Table I summarizes the flow characteristics that we collected for each of the flows in the Dartmouth traces. The characteristics in the top of the table were not used for classification purposes — they either involve characteristics that seemed inconsequential in classifying flows, or are accumulated quantities, which are indirectly captured by the corresponding rates or percentages and the flow duration.

B. Reduction of Flows for Machine Learning

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.

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 chat and non-chat.

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.”

V. RESULTS

In this section, we present our work on using machine learning-based classifiers to identify IRC-based botnet flows. As noted earlier, we first classify flows into IRC and non-IRC flows; then, among the IRC flows identified, we distinguish between authentic IRC and botnet flows. For each of these tasks, we use the flow data obtained from the Dartmouth traces after applying the flow reduction techniques of Section IV-B. We also use the flow characteristics in the lower part of Table I as the initial set of flow features/attributes. All the results in this paper were obtained using the WEKA ML toolbox [14].

Throughout our work, 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. Given the large number of non-IRC/botnet flows (in comparison to the number of IRC/botnet flows), a low FPR in the first stage of our work — that of identifying IRC traffic — is more effective in cutting down the number of flows that need to be examined during the second stage of our botnet flow identification. We also evaluate the classifiers considered in terms of the FNR in identifying our botnet testbed IRC flows. In the case of Stage I, this will guarantee the examination of these flows in Stage II.

A. 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 section, we use two sets of traces; those collected from a particular residential building on Dartmouth’s campus, henceforth referred to as Building X, and those obtained from preliminary experiments on our botnet test facility. After applying the flow reduction heuristics described in Section IV-B, the Building X trace resulted in 227784 flows, 7343 of which were IRC flows (as dictated by the use of the default 6667 IRC port). As of this writing, our testbed botnet traces resulted in 74 flows, 38 of which were IRC flows. Moreover, unless otherwise specified, we used half of the Building X flows for training and other half for testing the classifiers considered. Our botnet testbed traces were only used for testing.

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 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 2 contains an FNR vs. FPR scatter plot for several runs of J48, naive Bayes, and Bayesian networks for the labeled Building X trace. Each data point corresponds to a different subset of the initial flow attribute set. Figure 2 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.

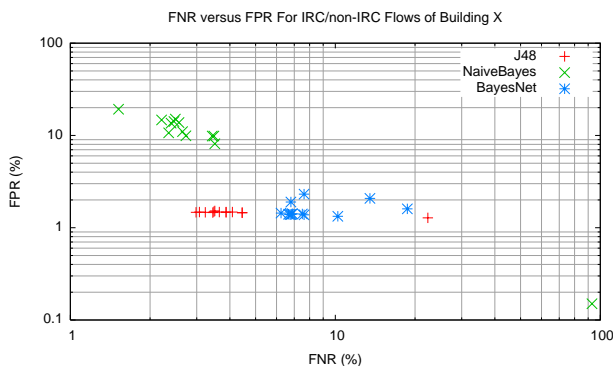


Fig. 2. FNR and FPR of J48, Naive Bayes, and Bayesian Net Classification Schemes for IRC/non-IRC Flows of Building X

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.

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.

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

TABLE II
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	✓	✓	✓	✓	✓	✓
varLAT	✓	✓	✓	✓	✓	✓
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	Botnet	100	68.42	100	100	100
Trace FNR (%)						

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.

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, our examination of several J48 trees revealed that the Bytes-per-packet (Bpp), the % of packets that are pushed (PctPktsPushed), and the variance in the Bytes-per-packet (varBpp) appear often high up and, thus, 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 II depicts the attribute sets obtained by running such exhaustive searches for the J48, naive Bayes, and Bayesian network classifiers. Table II 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 inconsequential. In the case of naive Bayes, there seems the optimal attribute set seems to trade-off the FNR and the FPR. Moreover, the optimal attribute set seems to perform 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.

TABLE III
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 Botnet Trace	68.42	47.37	18.42	7.89
FNR (%)				

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.

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 III 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 III reveals a low variability in the FNR and FPR rates achieved with respect to the attribute set for the Building X flows. 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.

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 the set of all Building X flows, set aside 10% of the flows for testing, and used the remaining 90% for generating training sets of different sizes. These training sets corresponded to random subsets of the flows set aside for training, whose size corresponded to 1-9% and 10-100%; that is, 19 different training set sizes. In order to obtain results with some sort of statistical significance, 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 III; this attribute set was the one achieving the lowest FNR for our testbed botnet flows.

Figure 3 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 seem to represent the respective testing sets well. For large training sets, the FNR seems to stabilize to values of higher mean but lower variance.

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

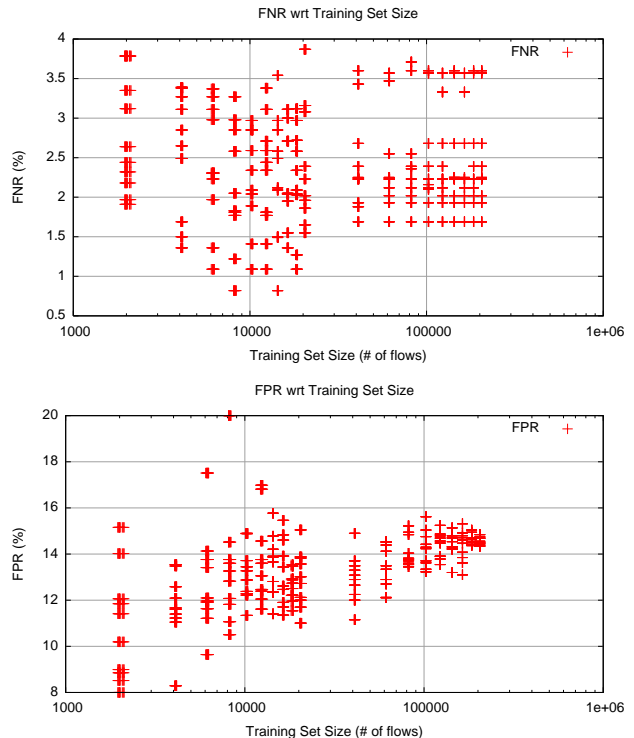


Fig. 3. FNR and FPR of Naive Bayes classification with respect to training set size for the Building X traces.

In the case of our botnet testbed IRC flows, small training sets perform poorly (FNRs of 50-60%). Medium and large training sets have comparable average FNRs. However, the minimum achieved FNRs for medium training sets (FNRs of 0-10%) are lower than those achieved by larger training sets. This comes, of course, at the cost of FNR variance.

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 Building X flows and for our botnet testbed IRC flows.

B. Stage II; Identifying Botnet Traffic

In this section, we describe our preliminary results on the classification of IRC flows into botnet IRC and real IRC flows. Since we do not have enough botnet testbed IRC flows for training, we used a more indirect approach to labeling. We labeled flows that involve compromised hosts as botnet flows and all others as real chat flows. Arguably, the fact that a host may be compromised in some respect does not directly imply that all its chat flows are botnet chat flows. However, these telltales split the flows into suspicious and non-suspicious chat

flows, which may be sufficient to identify a small subset of IRC flows that are botnet flows with higher probability.

Since the Dartmouth traces are anonymized and contain no packet payloads, accurate telltales of compromise are hard to come by. Nevertheless, we present a simple example of the sort of telltales that may indeed prove effective. We label IRC flows as either suspicious or non-suspicious depending on whether any of the two communicating hosts have shown indications of running the SubSeven trojan [10]; that is, using the default SubSeven port (27374).

Figure 4 presents the FNR vs. FPR plots of the performance of J48, naive Bayes, and Bayesian network classifiers on IRC flows from Building Y of the Dartmouth traces. We label the IRC flows of Building Y as suspicious and non-suspicious according to our SubSeven compromise criterion. Then, we use 90% of the flows for training and the remaining flows for testing. We repeat this experiment 10 times. We present the performance of classifiers trained on two different attribute sets: our initial attribute set and the attribute set obtained by running an exhaustive search using the Bayesian network classifier in WEKA. Figure 4 indicates that our Bayesian network classifiers achieve lower FNR and FPR than the J48 and naive Bayes classifiers. The Bayesian network classifier for the optimal attribute set achieves FNR between 10-20% and FPRs between 30-40%. Most of the resulting classifiers performed poorly in accurately classifying our botnet testbed IRC flows.

These results indicate that machine learning techniques are capable of capturing subtle differences in flows, but that more accurate labeling is required to achieve low FNR for our botnet testbed IRC flows. Although possibly crude, this simple example is used to demonstrate our proposed approach. One of our major thrusts going forward is to achieve much more accurate botnet IRC flow labeling. We are running more testbed experiments to obtain a larger pool of testbed botnet IRC flows. We are also looking into obtaining traces that contain packet payloads, which can be examined for more accurate botnet telltales such as textual C2 instructions.

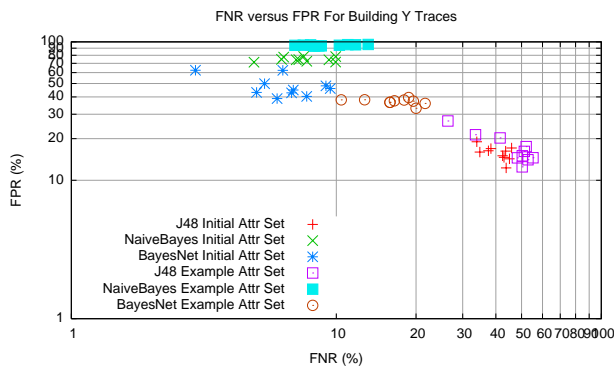


Fig. 4. FNR and FPR of J48, Naive Bayes, and Bayesian Network Classification Schemes for Building Y.

VI. CONCLUSIONS

We used machine learning techniques to identify the command and control 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;

it achieved both low FNR (2.49%) and low FPR (15.04%) for the Building X flows and low FNR (7.89%) for our botnet testbed IRC flows. While some J48 and Bayesian network classifiers performed better for the Building X flows, they classified our botnet testbed IRC flows poorly. For the feature sets and the traces we considered, we observed that training sets of 10K flows were sufficient and that the benefit of using larger sets was minimal. In Stage II, due to the absence of accurate labeling, we explored using telltales of the SubSeven trojan to label flows as suspicious and non-suspicious. With this labeling, Bayesian network classifiers achieved the best trade-off between FNR (10-20%) and FPR (30-40%). None of the resulting classifiers, that were viable in terms of FNR and FPR for the Building Y flows, accurately classified our botnet testbed IRC flows. This indicates that our labeling criterion may not be representative of botnet traffic and that more accurate labeling, either through more extensive botnet testbed traffic, or by using more accurate botnet telltales, is crucial for this stage of botnet traffic identification.

ACKNOWLEDGMENTS

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

REFERENCES

- [1] DEWES, C., WICHMANN, A., AND FELDMANN, A. An analysis of internet chat systems. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement* (New York, NY, USA, 2003), ACM Press, pp. 51–64.
- [2] DUDA, R. O., HART, P. E., AND STORK, D. G. *Pattern Classification*, 2 ed. John Wiley & Sons, Inc., 2001.
- [3] HENDERSON, T., KOTZ, D., AND ABYZOV, I. The changing usage of a mature campus-wide wireless network. In *Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking (MobiCom)* (September 2004), ACM Press, pp. 187–201.
- [4] HOLZ, T. A Short Visit to the Bot Zoo. *IEEE Security & Privacy* 3, 3 (May 2005), 76–79.
- [5] LEVY, E. The Making of a Spam Zombie Army. *IEEE Security & Privacy* 1, 4 (July 2003), 58–59.
- [6] MCAFEE. 2006/3/6; DDoS-Kaiten - http://vil.nai.com/vil/content/v_99371.htm.
- [7] MCCARTY, B. Automated Identity Theft. *IEEE Security & Privacy* 1, 5 (Sept. 2003), 89–92.
- [8] MCCARTY, B. Botnets: Big and Bigger. *IEEE Security & Privacy* 1, 4 (July 2003), 87–90.
- [9] MOORE, A. W., AND ZUEV, D. 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* (New York, NY, USA, 2005), ACM Press, pp. 50–60.
- [10] PEIKARI, C., AND CHUVAKIN, A. *Security Warrior*. O'Reilly Media, Inc., 2004.
- [11] ROUGHAN, M., SEN, S., SPATSHECK, O., AND DUFFIELD, N. 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* (New York, NY, USA, 2004), ACM Press, pp. 135–148.
- [12] SEN, S., SPATSHECK, O., AND WANG, D. Accurate, scalable in-network identification of p2p traffic using application signatures. In *WWW '04: Proceedings of the 13th international conference on World Wide Web* (New York, NY, USA, 2004), ACM Press, pp. 512–521.
- [13] THE HONEYNET PROJECT. *Know Your Enemy: Learning about Security Threats*. Addison-Wesley Professional; 2 edition (May 17, 2004), Mar. 2004.
- [14] WITTEN, I. H., AND FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques (2nd Edition)*. Morgan Kaufmann, San Francisco, CA, 2005.