DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis

Leyla Bilge
Symantec Research Labs
leylya_yumer@symantec.com

Davide Balzarotti
Eurecom
balzarotti@eurecom.fr

William Robertson
Northeastern University
wkr@ccs.neu.edu

Engin Kirda
Northeastern University
ek@ccs.neu.edu

Christopher Kruegel
UC Santa Barbara
chris@cs.ucsb.edu

ABSTRACT

Botnets continue to be a significant problem on the Internet. Accordingly, a great deal of research has focused on methods for detecting and mitigating the effects of botnets. Two of the primary factors preventing the development of effective large-scale, wide-area botnet detection systems are seemingly contradictory. On the one hand, technical and administrative restrictions result in a general unavailability of raw network data that would facilitate botnet detection on a large scale. On the other hand, were this data available, real-time processing at that scale would be a formidable challenge. In contrast to raw network data, NetFlow data is widely available. However, NetFlow data imposes several challenges for performing accurate botnet detection.

In this paper, we present DISCLOSURE, a large-scale, wide-area botnet detection system that incorporates a combination of novel techniques to overcome the challenges imposed by the use of NetFlow data. In particular, we identify several groups of features that allow DISCLOSURE to reliably distinguish C&C channels from benign traffic using NetFlow records (i.e., flow sizes, client access patterns, and temporal behavior). To reduce DISCLOSURE's false positive rate, we incorporate a number of external reputation scores into our system's detection procedure. Finally, we provide an extensive evaluation of DISCLOSURE over two large, real-world networks. Our evaluation demonstrates that DISCLOSURE is able to perform real-time detection of botnet C&C channels over datasets on the order of billions of flows per day.

1. INTRODUCTION

Malware continues to run rampant across the Internet, and among the myriad forms that modern malware can assume, botnets represent one of the gravest threats to Internet security. Through the large-scale compromise of vulnerable end hosts, botmasters can both violate the confidentiality of sensitive user information—for instance, banking or social network authentication credentials—as well as leverage groups of bots as an underground computational platform for performing other illicit activities.

Accordingly, a great deal of research has focused on methods for detecting and mitigating the deleterious effects of botnets. Research to date has largely followed one of two major approaches: (i) vertical correlation, or detecting command and control (C&C) channels used by botmasters to communicate with each infected machine [17,19,21,29]; and (ii) horizontal correlation, where botnet detection is based upon patterns of crowd behavior exhibited by collections of bots in response to botmaster commands [6,16,18,32, 35]. Once bots or, ideally, C&C servers have been identified, a number of actions can be performed, ranging from removal of infected endpoints from the network, to filtering C&C channels at edge routers, to orchestrated take-downs of the C&C servers themselves.

Unfortunately, while previous botnet detection approaches are effective under certain circumstances, none of these approaches scales beyond a single administrative domain while retaining useful detection accuracy. This limitation restricts the application of automated botnet detection systems to those entities that are informed or motivated enough to deploy them. Thus, we have the current state of botnet mitigation, where small pockets of the Internet are fairly well protected against infection while the majority of endpoints remain vulnerable.

This situation is not ideal. Botnets are an Internet-wide problem that spans individual administrative domains and, therefore, a problem that requires an Internet-scale solution. In particular, botnets can continue to wreak havoc upon the Internet despite the deployment of localized detection systems by focusing on propagation through less well-protected populations.

Two of the primary factors preventing the development of effective large-scale, wide-area botnet detection systems are seemingly contradictory. On the one hand, technical and administrative restrictions result in a general unavailability of raw network data that would facilitate botnet detection on a large scale. On the other hand, were this data available, real-time processing at that scale would be a formidable challenge. While the ideal data source for large-scale botnet detection does not currently exist, there is, however, an alternative data source that is widely available today: NetFlow data [10].

NetFlow data is often captured by large ISPs using a distributed set of collectors for auditing and performance monitoring across backbone networks. While it is otherwise extremely attractive, NetFlow data imposes several challenges for performing accurate botnet detection. First, and perhaps most critically, NetFlow records do not include packet payloads; rather, flow records are limited to aggregate metadata concerning a network flow such as the flow duration and number of bytes transferred. Second, NetFlow records are half-duplex; that is, they only record one direction of a network connection. Third, NetFlow data is often collected by sampling the monitored network, often at rates several orders of magnitude or more removed from real traffic.

Each of these characteristics of NetFlow data complicates the development of an effective botnet detector over this domain. The detector must be able to distinguish between benign and malicious network traffic *without* access to network payloads, which is the component of network data that carries direct evidence of malicious behavior. The detector must also be able to recognize weak

ACSAC '12 Dec. 3-7, 2012, Orlando, Florida USA

signals indicating the presence of a botnet due to the combined effects of half-duplex capture and aggressive sampling.

In this paper, we present DISCLOSURE, a large-scale, wide-area botnet detection system that incorporates a combination of novel techniques to overcome the challenges imposed by the use of NetFlow data. In particular, we identify several groups of features that allow DISCLOSURE to reliably distinguish C&C channels from benign traffic using NetFlow records: (i) flow sizes, (ii) client access patterns, and (iii) temporal behavior. We demonstrate that these features are not only effective in detecting current C&C channels, but that these features are relatively robust against expected countermeasures future botnets might deploy against our system. Furthermore, these features are oblivious to the specific structure of known botnet C&C protocols.

While the aforementioned features are sufficient to capture core characteristics of generic C&C traffic, they also generate false positives in isolation. To reduce DISCLOSURE's false positive rate, we incorporate a number of external reputation scores into our system's detection procedure. These additional signals function as a filter that reduces DISCLOSURE's false positive rate to a level where the system can feasibly be deployed on large-scale networks.

We provide an extensive evaluation of DISCLOSURE over two real-world networks: a university network spanning a small country where no NetFlow sampling occurred, and a Tier 1 ISP where NetFlow data was sampled at a rate of one out of every ten thousand flows. Our evaluation demonstrates that DISCLOSURE is able to perform real-time detection of botnet C&C channels over data sets on the order of billions of flows per day. In particular, we show that DISCLOSURE is able to recognize approximately 65% of known botnet C&C servers in both settings while producing 1% false positives. Furthermore, we demonstrate DISCLOSURE's ability to detect previously unknown C&C servers by manually verifying 20 and 91 true positive alerts from the university and ISP networks, respectively.

Finally, we report on our operational experience in deploying and testing DISCLOSURE on real large-scale networks, highlighting the most critical areas for tuning the performance of the detection system. The contributions of the paper is as follows:

- We present DISCLOSURE, a large-scale, wide-area botnet detection system that reliably detects botnet C&C channels in readily-available NetFlow data using a set of robust statistical features. To our knowledge, DISCLOSURE is the only NetFlow-based system that does not assume *a priori* knowledge of particular C&C protocols.
- We incorporate several external reputation systems into DISCLOSURE's detection procedure to further refine the accuracy of the system.
- We evaluate DISCLOSURE over two real-world networks, and demonstrate its ability to detect both known and unknown botnet C&C servers at scales not previously achieved.
- We report on our operational experience with DISCLOSURE, and highlight important tuning considerations for deployment and reproducibility.

2. SYSTEM OVERVIEW

DISCLOSURE is a botnet detection system designed to identify C&C servers by employing NetFlow analysis. Figure 1 shows an overview of the system architecture. The upper half of the figure describes the detection model generation process, where a supervised machine learning algorithm is used to train models on a subset of NetFlows targeting known (i.e., labeled) benign and C&C servers.

The flows in this labeled data set are first processed by the *feature extraction module*. This module reduces the flows to a number of distinct features: flow size-based features, client access pattern-based features, and temporal features, which are described in detail in Section 3. The features extracted from the training set are then forwarded to DISCLOSURE's *learning module*, which is responsible for building detection models. The learning module can be tuned with several thresholds to obtain an optimal balance between detection and false positive rates.

The bottom half of the graph represents the detection phase, where the models that have been previously generated are applied to unlabeled NetFlows in order to distinguish benign traffic from C&C communication. Since the aim of DISCLOSURE is not to identify bot-infected machines but to detect C&C servers, the first task of the detection phase is to filter those NetFlows that cannot be attributed to a server; this process is explained in Section 5.4. Then, the flows are forwarded to the *feature extraction module*. Finally, the resulting feature vectors are processed by the *detection module* to produce the final list of suspected C&C servers.

Note that the results of DISCLOSURE can be further processed by a *false positive reduction* filter. The goal of this additional module is to correlate the results of DISCLOSURE with the information obtained from other security feeds in order to reduce the probability of misclassification. For example, in Section 4, we present a novel technique that associates a reputation score to the autonomous systems to which the C&C servers belong.

3. FEATURE SELECTION AND CLASSIFICATION

In this section, we present the features extracted by DISCLOSURE from NetFlow data in order to detect botnet C&C channels at scale, and discuss why these features are suitable for discriminating between C&C channels and benign traffic. We then describe the particular machine learning techniques we use to build detection models over these features.

3.1 NetFlow Attributes

NetFlow is a network protocol proposed and implemented by Cisco Systems [10] for summarizing network traffic as a collection of network flows. Network elements such as routers and switches capture these NetFlows and forward them to NetFlow collectors. A network flow is defined to be a unidirectional sequence of packets that share specific network properties (e.g., IP source/destination addresses, and TCP or UDP source/destination ports). Each flow has a number of associated attributes, or summary statistics that characterize various general aspects of its behavior. In this paper, the NetFlow attributes we analyzed for extracting features to identify C&C servers are: the source IP address, the destination IP address, the TCP source port, the TCP destination port, the start and finish timestamps, and the number of bytes and packets transferred.

Since DISCLOSURE is primarily focused on identifying botnet C&C channels, it is imperative that the system can reliably distinguish servers from clients. Therefore, as an intermediate preprocessing stage, NetFlow data is analyzed by the server classifier that labels each observed IP address according to whether it provides one or more network services. In particular, since multiple services can be made available for each IP address, we represent each server as a 2-tuple of IP address and port, $s_i = \langle a, p \rangle$, where $a \in A$ is an IP address, A is the set of all IP addresses, $p \in P$ is a TCP port, and P is the set of all ports.

A common, and legitimate, criticism of early attempts to perform machine learning-based detection over NetFlow data is that the features that were selected were often not *robust*. Hence, the resulting detection systems would often overfit models to the specific behavior of malware represented in the training set—for instance, the particular server port used by a given malware sample. Such features, however, do not generalize to classes of malware such as botnets. For example, using our example of learning a model on server ports, it is clear that the use of a particular server port is not an intrinsic property of botnet behavior. Therefore, the design of DISCLOSURE's feature extractor module emphasizes the selection of those NetFlow attributes that best capture invariants in botnet behavior without resorting to specialization to a particular C&C protocol.

3.2 Disclosure Feature Extraction

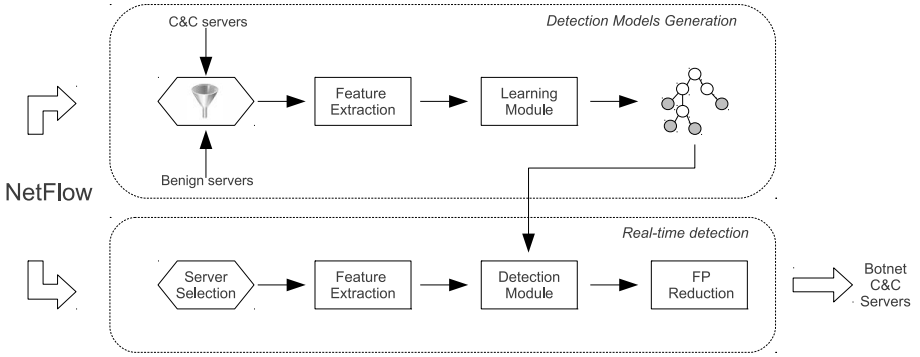


Figure 1: The system architecture of Disclosure. In the training phase (upper half), labeled training samples are used to build detection models. In the detection phase (lower half), the detection models are used to classify IP addresses as benign or associated with C&C communications.

3.2.1 Flow Size-Based Features

The first class of features extracted from NetFlow data are based on *flow sizes*, which simply indicate the total number of bytes transferred in one direction between two endpoints for a particular flow. Our premise for analyzing flow sizes in NetFlow data is that the flow size distributions for C&C servers are significantly and *necessarily* different from flow size distributions for benign servers.

We attribute this difference to several factors. First, the main role of the botnet C&C channel is to establish a connection between the bots and the C&C server. This channel should be both reliable as well as relatively innocuous in appearance. Thus, flows carrying botnet commands or information harvested from infected clients are preferred to be as short as possible in order to minimize their observable impact on the network. Considering that network monitoring tools are widely used and that a botnet's local network impact usually scales linearly with the number of bot infections, tuning for stealth is an important goal. Moreover, due to the limited number of commands in typical C&C protocols, flow sizes tend not to fluctuate significantly. On the other hand, flow sizes generated during accesses to a benign server usually assume a wide range of values.

The preliminary analysis we performed on known sets of benign servers and C&C servers supports our premise. Hence, we designed a set of methods to extract features to detect the behavioral difference between C&C servers and benign servers with respect to flow size.

DISCLOSURE extracts flow size-based features by first grouping all flows according to the server s_i that they originate from or are destined to. Let $s_i \in S$ be a server, and $c_j \in C$ be a client. Then, flow sizes are grouped by time intervals $j = 0, 1, 2, \ldots$, where $F_{i,j}$ denotes a series of flow sizes for flows from endpoint i to j, where endpoints can be drawn from C or S. Once this set has been derived, the following feature sets are extracted.

Statistical features. This group of features characterizes the regularity of flow size behavior over time for both benign and C&C servers. In particular, we extract the mean $\mu^{F_{i,j}}$ and standard deviation $\sigma^{F_{i,j}}$ separately for both incoming and outgoing flows of each server.

Autocorrelation features. Autocorrelation is widely used for cross-correlating a signal with itself in the signal processing domain [7], and is useful for identifying repeating patterns in time series data. A series of flow sizes $F_{i,j}$ can be converted to a time series by ordering sizes by time. Since the autocorrelation function also requires a time series that is sampled periodically as input, we segment the time series by fixed intervals and take the mean over each interval; empirical testing suggested that a period of 300 seconds is appropriate. Once a periodically sampled time series $\hat{F}_{i,j}$ has been derived from $F_{i,j}$, the series is processed by the autocorrelation function, and features are extracted from the output. Here, we use a discrete autocorrelation coefficient $R_{\hat{F}_{i,j}\hat{F}_{i,j}}$ with lag j normalized by the variance σ^2, where

$$R_{\hat{F}_{i,j}\hat{F}_{i,j}} = \frac{\sum_{i=j}^{n} x_i \overline{x}_{i-j}}{\sigma^2}.$$

The autocorrelation function outputs the correlation results for each period in the time series. This output is further processed by taking the mean and standard deviation over these values to derive the final autocorrelation features.

Unique flow sizes. In addition to the statistical features described above, DISCLOSURE also includes features that count the number of unique flow sizes observed, and performs statistical measurements of occurrence density for each of them during the analysis time. Specifically, an array is constructed in which the elements are the number of occurrences of a specific flow size. Afterward, statistical features are computed over this flow size incidence array to measure its regularity.

3.2.2 Client Access Patterns-Based Features

One typical property of botnets is that the bots frequently establish a connection with the C&C server. These connections tend to be ephemeral, as longer-lived connections might draw undue attention to a bot's presence.

Our basis for selecting features to extract in order to distinguish malicious client access patterns from benign ones is that all of the clients of a C&C server (i.e., bots) should exhibit similar access patterns, whereas the clients of a benign server should not. Since all bots share the same, or nearly identical, malicious program, they tend to access C&C servers similarly unless specifically programmed otherwise. On the other hand, clients of benign services tend to exhibit much more varied patterns due to the vagaries of human action. DISCLOSURE extracts two sets of features to characterize client access patterns typical of C&C servers and those typical of benign servers.

Regular access patterns. For each server s_i and client c_j, DISCLOSURE prepares a time series $T_{i,j}$ of flows observed during the analysis period. Then, a sequence of flow inter-arrival times $I_{i,j}$ is derived from the time series by taking the difference between consecutive connections; that is,

$$I_{i,j} = \bigcup_{k=1}^{n} t_{i,j,k} - t_{i,j,k-1},$$

where $t_{i,j,k}$ is the k^{th} element of $T_{i,j}$. Then, statistical features are computed over each inter-arrival sequence, including the minimum, maximum, median, and standard deviation. Finally, we derive the final features for each server s_i by generating statistical features across the set of clients that accessed s_i. This allows DISCLOSURE

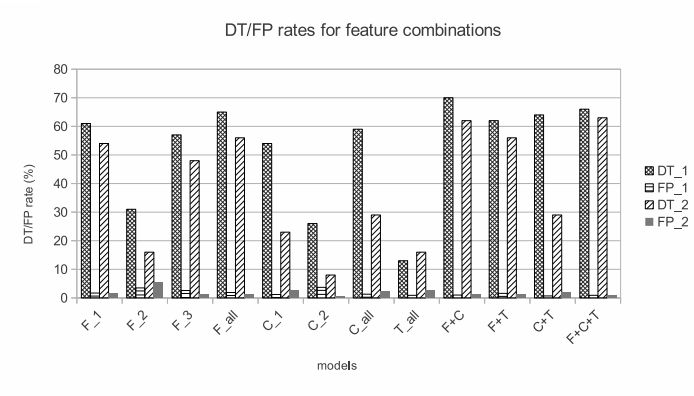


Figure 2: Detection rates (DT) and false positive (FP) rates for different feature combinations. We note that the DT:FP ratio is most favorable when all features are used in the detection procedure.

to not only find regular patterns in clients, but to determine whether the set of clients accessing a server behave similarly.

Unmatched flow density. When a bot is unable to communicate with a legitimate C&C server, it detaches from the rest of the botnet and becomes a zombie. This might happen because the C&C server was shutdown, or its IP address has been blacklisted. Since the zombie cannot distinguish between these situations and transient network errors, it continues querying the server. This can result in a significant number of flows to a server that do not have a matching flow in the opposite direction. It is also possible that a benign server is unreachable for a period of time. However, the behavior of a benign server's clients is significantly different than the behavior of bots that lose access to their C&C servers. This is because when a benign user is aware that a server is offline, it typically does not insist on continuing to query the server indefinitely. Therefore, DISCLOSURE extracts statistics regarding the number of unmatched incoming and outgoing flows to detect this behavior. Specifically, let $U_{i,j}$ be the number of unmatched flows for server s_i in time interval t_j, where

$$U_{i,j} = \sum_{j \in C} \text{abs}\left(|F_{i,j}| - |F_{j,i}|\right).$$

Then, DISCLOSURE derives the mean and standard deviation over a time series of $U_{i,j}$ as a statistical feature.

3.2.3 Temporal Features

Connections to a benign server are subject to diurnal fluctuations representative of the server's user population. On the other hand, connections to C&C servers are dictated by the botmaster, and require no user intervention. As previously mentioned, the majority of botnets configure their bots to contact the C&C server periodically and with relatively short intervals. Therefore, bot-infected machines connect to C&C servers during periods of the day that benign clients do not. For example, many benign servers receive a high volume of traffic during the day, and very little—or nothing—during the night.

To capitalize on this observation, DISCLOSURE extracts a set of temporal features that characterize the variability of client flow volume as a function of time, such that the system can discriminate between uniform client flow distributions indicative of C&C servers and benign traffic that follows well-known diurnal patterns. Specifically, DISCLOSURE segments a time series of client and flow volume by hour-long intervals per server s_i, and calculates statistical features over these.

3.3 Building the Detection Models

To build detection models for identifying C&C servers, we experimented with a number of machine learning algorithms, including the J48 decision tree classifier [26], support vector machines [12], and random forest algorithms [23]. Random forest classifiers, known to be one of the most accurate learning algorithms, combine multiple classification methods to achieve more predictive results. In particular, the random forest classifier builds a number of decision trees, where each node in a tree encodes a decision using one or more features that partition the input data. The leaves of each decision tree correspond to the set of possible labels (i.e., {`benign`, `malicious`}), and the output of all of the trees are then ensembled such that the average behavior among all trees is produced as the final decision. In our testing, the best ratio between detection rates (DT) and false positive rates (FP) were produced by the random forest classifier. Furthermore, the classifier is efficient enough to perform online detection in our application. Consequently, DISCLOSURE uses the random forest classifier to build its detection models.

We evaluated our detection models against NetFlow data collected from two networks: a university network (N_1) that does not apply sampling, and a large Tier 1 network (N_2) that samples one out of 10,000 flows. Figure 2 shows the detection rates (DT_1 for N_1 and DT_2 for N_2) and false positive rates (FP_1 for N_1 and FP_2 for N_2) for individual features sets, and all possible combinations among different feature sets. The feature sets we evaluated are the set of statistical features extracted from (i) the flow size (F_1); (ii) the flow size-based features extracted from the output of the autocorrelation function (F_2); (iii) unique flow sizes for each server (F_3); (iv) the combination of all flow size-based features (F_{all}); (v) the features for characterizing client access patterns (C_1); (vi) unmatched flow density (C_2); (vii) the combination of all client access pattern-based features (C_{all}); (viii) temporal features (T_{all}); (ix) the combination of client access pattern and flow size-based features ($F+C$); (x) the combination of flow size and temporal features ($F+T$); (xi) the combination of client access pattern and temporal features ($C+T$); and, finally, (xii) the combination of all feature sets ($F+C+T$).

Figure 2 indicates that individual feature sets are not as effective as combinations of multiple feature sets. Furthermore, increased levels of feature aggregation results in better detection rates with less false positives. Finally, we note that the most promising results were achieved on both data sets by using all possible feature sets as input to the classification process. Hence, DISCLOSURE uses detection models that include all features sets ($F+C+T$) to detect botnet C&C channels.

4. FALSE POSITIVE REDUCTION

NetFlow data, by its nature, provides limited information about the real activities that are carried out in a network. As a consequence, a botnet detection system based only on the analysis of NetFlow data could produce results that are likely to contain some false positives (FP).

As we explain in Section 5, DISCLOSURE can be tuned to decrease the overall FP rate to 0.5% or below. However, given the volume of NetFlow data that must be processed every day in large networks, even a misclassification rate less than a fraction of a percent can result in an unacceptably large number of false alarms. Note that some existing malicious activity detection systems have shown to be useful for specific classes of malware or attacks. Clearly, it would be beneficial to correlate the detection results of our system with the results of some previously built systems. Therefore, in our architecture, we include a component that has the aim of correlating the results that DISCLOSURE produces with the public feeds of other malware analysis or detection platforms. The main insight here is that by integrating different data sources, it is possible to further reduce DISCLOSURE's FP rate to a manageable level.

We have built a reputation-based component for FP reduction that uses three public services that provide reports about a wide range of malicious activities on the Internet. The first service we

make use of is FIRE [3,31]. FIRE is a system that identifies organizations and ISPs that have been observed to engage in malicious activities. FIRE's website reports detailed information about many autonomous systems (AS), including a maliciousness score, relative rankings among other ASes, as well as the number of C&C servers, exploit servers, and spam and phishing servers the AS has been hosting over time. In our implementation, we separate each type of information into two time series: one representing the current year, and one containing previous historical data. Afterward, we compute statistical features for each time series. For instance, for the time series built from the number of C&C servers observed before 2011, we compute the minimum, mean, and maximum values. After we repeat this step for each time series, we compute a final score by aggregating all the values together by assigning a weight of 0.8 to the value for the current year, and 0.2 to the previous years.

The second public service we use in our FP reduction component is EXPOSURE [2,5]. EXPOSURE is a system that uses passive DNS analysis methods to detect malicious domains. EXPOSURE currently analyzes data obtained from a large number of recursive DNS servers, and reports its findings on daily basis. For each domain, it provides the associated IP address list and the ASes in which they are located. Leveraging this information, we count the number of malicious domains detected in each AS and build a reputation score according to the density of maliciousness for each AS reported by EXPOSURE.

The last source of information we use for FP reduction is Google Safe Browsing [4], a service that reports maliciousness information about a large number of web sites. This tool can also be used to query specific AS numbers to obtain the percentage of web sites in that AS that host malicious services.

For each IP address that DISCLOSURE labels as a potential botnet C&C server, the FP reduction component fetches the associated AS number and corresponding reputation scores from FIRE, EXPOSURE, and Google Safe Browsing. Each of these individual reputation scores are then aggregated using a weighted linear combination. That is, given the reputation scores r_1, r_2, r_3 and corresponding weights w_1, w_2, w_3 for FIRE, EXPOSURE, and Google Safe Browsing such that $\sum_i w_i = 1$, the final reputation score R is calculated as

$$R = \sum_{i=1}^{3} w_i r_i,$$

where $0 \leq R \leq 1$. If R is below a tunable threshold we denote as RepThresh, this indicates that a particular server is located in a network that is historically *not* associated with malicious activities, and the corresponding alert is discarded as a FP.

We are aware of the fact that the FP reduction component can introduce an opportunity for the attacker to evade our system. For example, she could place her C&C server in a network with a high reputation score. However, note that this increases the burden on the attacker, and forces her to move away from more vulnerable targets located in ASes with lower reputation scores towards potentially better-protected networks. Therefore, we believe that, on a large scale, this is a favorable result.

5. EVALUATION

In this section, we present the design and results of several experiments we conducted to evaluate DISCLOSURE's detection accuracy, false positive (FP) rate, and performance. We also present deployment considerations, and conclude with a discussion of resilience to evasion.

The accuracy of DISCLOSURE's classification procedure greatly depends upon the environment in which the input NetFlows have been collected. For example, NetFlow collectors placed in a small company network versus those placed in a large ISP will likely observe significantly different volumes of traffic. To bound the storage requirements at each collector, sampling rates might be configured to match the particular traffic volume specific to each site.

Network	C&C Servers	Benign Servers
University Network (N_1)	892	1489
Tier 1 ISP (N_2)	2000	1742

Table 2: IP addresses in our labeled data set derived from data observed in N_1 and N_2.

To measure how DISCLOSURE responds to varying levels of sampling, we evaluated our system in two distinct environments: a medium-size network connecting multiple universities with no sampling, and a Tier 1 ISP network configured with a sampling rate of 1:10,000.

5.1 NetFlow Data Sets

Our NetFlow data sets were drawn from two separate environments: a university network located in Europe, and an ISP network located in the USA and Japan. Hereinafter, we refer to the university network as N_1 and to the Tier 1 ISP network as N_2.

Table 1 shows summary statistics for the two data sets. The N_1 data set was collected for a period of 18 days between the 7th and the 25th of September 2011. The NetFlow data of N_1 is not sampled and, therefore, all network flows present in the monitored network are represented in the data. The sensor in N_1 produced an average of 1.2 billion network flows per day. During this period, we collected 22 billion flows between 28 million unique IP addresses.

In contrast, we collected NetFlow data observed at N_2 for a period of 40 days between the 1st of June 2011 and the 10th of July 2011. The sensors in N_2 were configured to sample flows at a rate of 1:10,000. The data was harvested by 68 sensors, each of which was responsible for monitoring and forwarding NetFlow traffic collected from specific autonomous systems (ASs). The sensors collected approximately 400 million network flows per day between 50 million unique IP addresses.

5.2 Ground-Truth Data Sets

The accuracy of the classification models generated by a machine learning algorithm greatly depends on the quality of the training set [33]. In our case, to train the features used by DISCLOSURE, we required a ground-truth list containing both known C&C servers and known benign servers.

The malicious server data set consisted of 4295 IP addresses associated with real C&C servers observed in the wild during approximately three weeks preceding our experiments. The list of botnet C&C servers was provided to us by a company that specializes in threats intelligence. This list is used by the company as the core of their reputation-based detection engine. Hence, the aim is to be both complete with regard to current threats and keep FPs down. The reputation-based engine is deployed on a few hundred sites, and operational experience shows that FPs are minimal and coverage is at least comparable with deployed anti-virus tools (i.e., the engine captures threats that AV software installed on these sites misses).

We constructed our benign server training set from ranking information provided by Alexa [1]. In this case, we assume that the top 1,000 popular web sites reported by Alexa are not involved in malicious activities and, in particular, are not responsible for hosting botnet C&C servers. Alexa reports the top popular web sites grouped by geographical regions as well. In order to obtain a comprehensive list of benign servers, we combined the "Alexa Top 1000 Global Sites" with the most visited websites in the regions where N_1 and N_2 are located.

Once the benign domains lists were compiled, we resolved each DNS name on both lists to obtain the corresponding list of IP addresses. Note that we executed the DNS queries for each list from the same network geographical locations of the corresponding network (Europe for N_1, and US for N_2). Hence, the number of IP addresses collected for each network is different. This process resulted in 2,958 unique IP addresses for N_1, and 3,047 IP addresses for N_2.

Table 2 shows the number of benign and malicious servers in

Network	Sampling	Flows per day	Unique IP Addresses
Inter-University Network (N_1)	1:1	1.2 billion	28 million
Tier 1 ISP (N_2)	1:10.000	400 million	50 million

Table 1: Summary statistics for each of the two NetFlow data sets for N_1 and N_2.

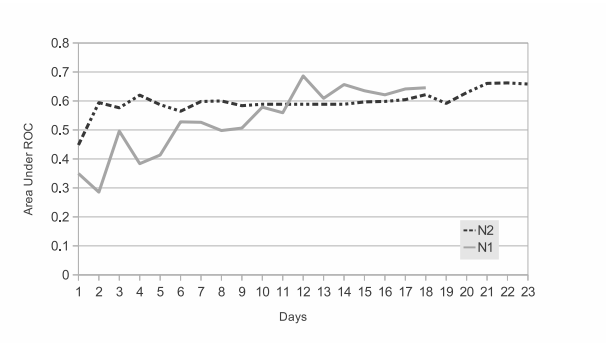


Figure 3: Area under ROC curves with different training set lengths for N_1 and N_2.

our labeled data set that were observed in the traffic of N_1 and N_2 respectively.

5.3 Labeled Data Set Detection and False Positive Rates

In the initial experiment, we evaluated DISCLOSURE's ability to recognize known botnet C&C servers from the ground truth constructed in the previous section. DISCLOSURE's detection rate (DR) and FP rates were measured by generating ROC curves for each data set under two configurations each that controlled the level of input data filtering performed prior to detection.

DISCLOSURE requires a minimum number of observed flows to a particular server in order to provide accurate results. This minimum is a threshold we denote by MinFlows, and can be set by a security administrator according to the volume of traffic at a particular site and any sampling that may be applied. We evaluated two values for MinFlows for each data set: 20 and 50. For each experiment, we excluded any servers that did not have at least one port that received more than MinFlows flows. We then evaluated the accuracy of DISCLOSURE's detection models by performing a 10-fold cross-validation.

We also considered varying the size of the training set as an additional tunable parameter. Figure 3 shows a summary of DISCLOSURE's accuracy, measured by computing the area under the ROC curve for different training windows. The curve for N_2 is almost constant. In comparison, the curve for N_1 steadily increases over the first 15 days before plateauing. This is due to the fact that the number of known C&C servers observed in the university network is low (see Table 2). Therefore, more time is required to collect enough data to properly train the models. For this reason, we decided to train DISCLOSURE with all the available data.

Figure 4 shows the individual ROC curves obtained by varying the classification threshold ClassThresh, i.e., the boundary separating benign scores from malicious scores, of DISCLOSURE's detection module. Consequently, each point in the ROC curves represents a possible setup configuration of the system. Security administrators can thus precisely tune DISCLOSURE to achieve a reasonable trade-off between FPs and false negatives based on the traffic characteristics of the network. Each graph also contains a short synopsis of possible working points. For example, configuring the system for a very high DR is usually too costly in terms of FPs. On the other end of the scale, it is often possible to achieve a 0% FP rate if we accept the fact that only one out of three C&C servers will be detected.

Point on the ROC curve	Servers Flagged as C&C (N_1)	Servers Flagged as C&C (N_2)
1.0% FP	12,383	4,937
0.5% FP	7,856	3,166
0.3% FP	6,295	1,958
0.0% FP	132	960

Table 3: Servers flagged as malicious by Disclosure for each of the networks N_1 and N_2 (without incorporating reputation scores).

Despite the differences between the two data sets, the results are similar. For instance, with MinFlows set to 20 flows and ClassThresh tuned to produce a 1% false positive rate, the system detects 64.3% of the C&C servers in the university network and 66.9% in the ISP network. This similarity emerges from the composition of all features, where the individual contribution of each feature is quite different in the two environments. For instance, most of the features are better suited to the unsampled data set, where traffic patterns are clearly preserved. However, some of the features—for instance, the *unmatched flow density*—provide the best results when applied to large networks, even in presence of a high sampling rate. The mixture of these two classes of features makes DISCLOSURE less sensitive to variability in the NetFlow collection environment and, therefore, more robust.

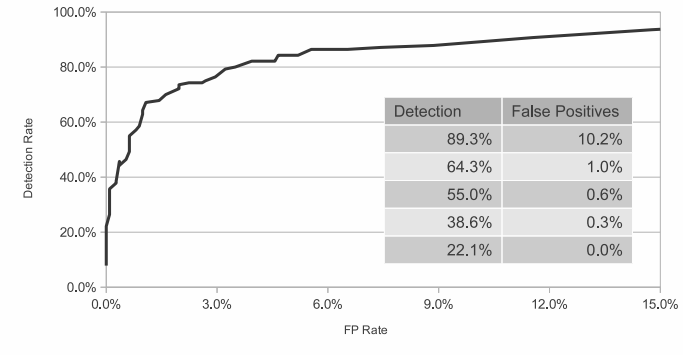
Another important difference between the two experiments is the fact that in the small network (N_1), DISCLOSURE provides better results for a higher value of the minimum flow threshold, while in the large network (N_2) it performs better with a lower threshold. This phenomenon is due to the fact that in the second case, the sensors are only collecting 1 flow out of every 10,000. Therefore, a high value for MinFlows would filter out all small-to-medium size botnets, leaving only a few large ones for the analysis. As a result, the features are now trained on a very few C&C samples and, therefore, tend to produce inaccurate models. This is an important issue to keep in mind when configuring the system. In general, if MinFlows is set too low, the features are exposed to samples that do not show sufficient regularity because an insufficient number of flows are observed in the traffic. If, on the other hand, MinFlows is set too high, the majority of the botnets are discarded, and the features are trained on too few samples. In both extremes, the result is a set of poorly trained models.

Finally, we manually verified the features of the benign servers that DISCLOSURE wrongly classified as being botnet C&C servers. In several cases, the network or the server were probably malfunctioning, and the clients (in most of the cases less than 10) were repeatedly trying to send the same data over and over again at regular intervals, and receiving no answer back from the server. This behavior, even though not malicious *per se*, is indeed quite similar to that exhibited by bot-infected machines.

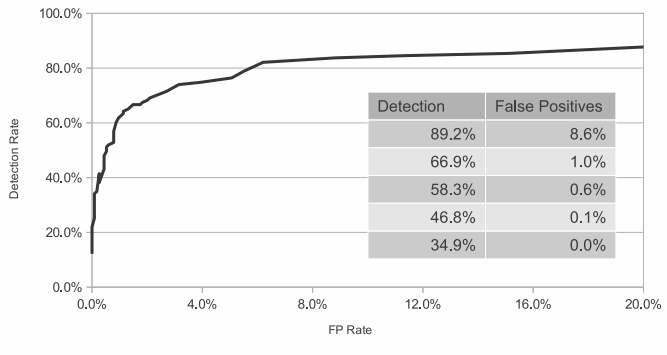
5.4 Real-Time Detection

In the previous section, we presented the results obtained with labeled data sets containing known benign and botnet C&C servers. In order to apply DISCLOSURE to the remaining unlabeled data, we needed to perform three separate operations.

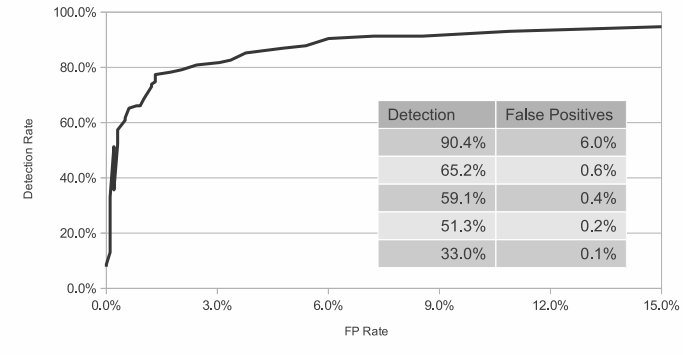
First, since DISCLOSURE is meant to discover C&C servers and not infected machines, we need to restrict the analysis to the servers only. In order to separate them from the clients, we apply the following heuristic: an IP address belongs to a server if the number of flows directed towards its top two ports (i.e., the two that receives the most connections) account for at least 90% of the flows



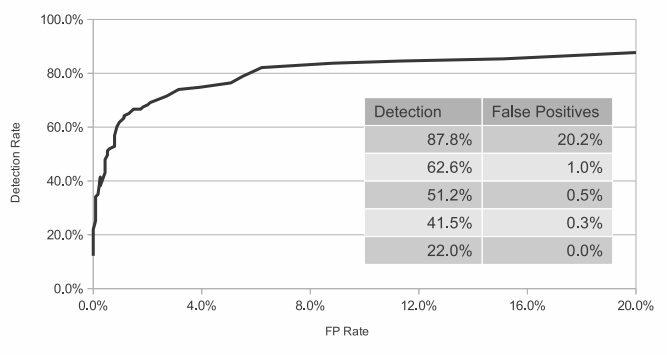
(a) N_1 with MinFlows $= 20$.



(b) N_2 with MinFlows $= 20$.



(c) N_1 with MinFlows $= 50$.



(d) N_2 with MinFlows $= 50$.

Figure 4: Classification accuracy for each data set (N_1 and N_2) with MinFlows $\in \{20, 50\}$.

Point on the ROC curve	C&C Servers after the RF (N_1)	C&C Servers after the RF (N_2)
1.0% FP	1779	1516
0.5% FP	1448	688
0.3% FP	1236	271
0.0% FP	20	91

Table 4: Servers flagged as malicious by Disclosure for each of the networks N_1 and N_2 (incorporating reputation scores). Here, RF refers to "reputation filter."

towards that address. From the count, we removed the ports used less than 3 times to filter out the noise generated by the fact that servers may also have outgoing connections. By adopting this technique, we identified 82,580 servers in N_1 and 530,011 servers in N_2.

The second step consisted of setting the value of the MinFlows threshold. According to the results obtained in the labeled data set, we decided to perform the rest of the experiments with the threshold set to 50 flows for N_1 and to 20 for N_2. After applying the threshold, we were left with 53,426 servers in N_1 and 48,713 in N_2.

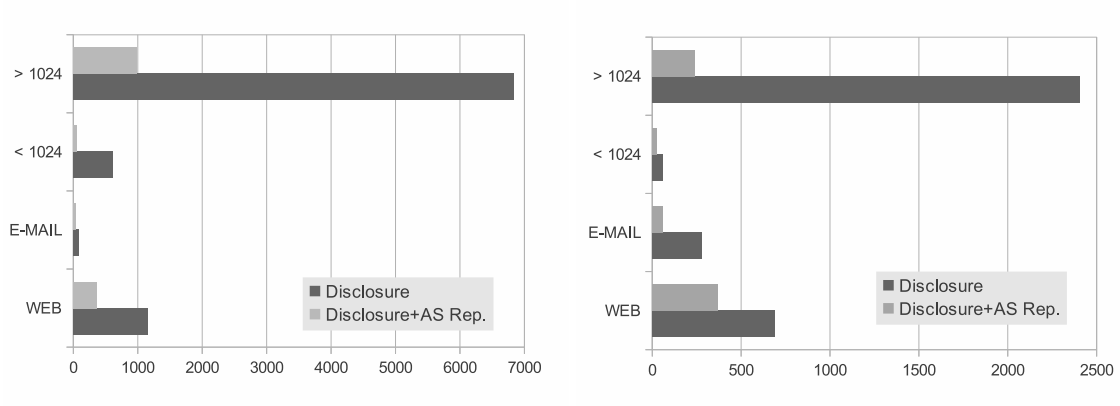
Finally, we needed to select the operational point on the ROC curve ClassThresh (i.e., the trade-off between DR and FP rates). Table 3 shows the number of servers detected in the two networks obtained with four different configurations of the system.

Despite the fact that the various configurations were chosen to minimize the number of FPs generated by the system, the number IP addresses suspected of being C&C servers is still relatively high. Therefore, to further reduce the probability of misclassification, we combined the results of Disclosure with a reputation score based on the information provided by EXPOSURE [2, 5], FIRE [3, 31], and Google Safe Browsing [4]. As explained in Section 4, this approach has the effect of narrowing down the results to the servers that have a higher probability of being malicious.

The way in which the reputation score is computed can be tuned according to the desired results and the number of daily alerts that the security administrator can tolerate. The more aggressive the filtering, the smaller the set of IP addresses flagged as C&C servers. In our experiments, we increased the strength of the FP reduction until we reduced the amount of alerts to a level that can be manually verified. The results are reported in Table 4.

Figure 5 shows the ports distributions of the C&C servers detected by Disclosure in the 0.5% false positive setting for N_1 and N_2. The graphs report the two most frequently used protocols: HTTP-related (ports 80, 443, 8080, 8000) and SMTP/IMAP (ports 25, 143, and 993). The remaining ports are grouped in two categories: the reserved ports (0-1023), and the registered and ephemeral ports (1024-65535). This classification is based only on the port number and not on identification of the true protocol. For instance, a botmaster can run a C&C server on port 25 to avoid firewalls, but that does not mean that he will adopt the SMTP protocol as well. It is interesting to note that the majority of the services identified by Disclosure run on ports higher than 1024. However, the distribution changes significantly after the FP reduction is applied. In fact, the reputation system filters out around half of the HTTP services, but cuts between 70 and 90% of the services running on high port numbers.

Finally, we manually investigated the C&C servers detected by Disclosure to gain some insight into the accuracy of the detection models and the reasons for misclassification. To this end, we chose the most conservative configuration: Disclosure configured for 0% FP + Reputation filter. With this setup, during one week



(a) N_1 port distribution.

(b) N_2 port distribution.

Figure 5: Port distributions of the C&C servers detected by Disclosure for both N_1 and N_2, with and without AS reputation scores.

of operation, DISCLOSURE reported 91 previously unknown C&C servers on the ISP network, and 20 on the university network.

We first manually queried popular search engines for each of the 111 entries. In 36 cases (32.4%), we found evidence of malware that was related to those IP addresses.[1] The fact that one third of our reports were confirmed by other sources is a strong support of the ability of DISCLOSURE to successfully detect C&C servers. Out of the remaining servers, 30 were associated with HTTP-related ports. After a manual investigation, seven of them seemed to be legitimate web sites—even though it is unusual that a small real estate company or a personal page in the Philippines would receive the large number of connections we observed in our traffic. Four pages were default placeholders obtained with a fresh installation of a web server; the number of NetFlow entries and varying flow sizes is suspicious, although this could be attributed to the web server not having a default virtual host configured. Four servers returned errors related to either unauthorized access or bad requests. Three of the HTTPS servers did not use the SSL/TLS protocol but some other form of binary protocol. The remaining servers were unaccessible at the time we checked them, which was approximately three weeks after the data was collected. Of the non-HTTP services, only four were still running at the time the checks were performed. Three of these appeared legitimate, but the remaining service was a web server located in Russia listening to a non-standard port. Finally, interestingly, eight servers were located in the Amazon cloud network, which is rapidly increasing in popularity for hosting ephemeral malicious services.

5.5 Performance Evaluation

As described in Section 2, the detection phase consists of two modules: feature extraction and detection. The detection module is highly efficient, requiring only several minutes to process an entire day's worth of data. Hence, detection performance is constrained by the analysis of input NetFlow data to extract the requisite features for analysis.

However, since the extraction of each feature is an independent process, the feature extraction procedure is an example of an embarrassingly parallel problem that can be easily distributed on multiple machines should the need arise. Nevertheless, even with the large amount of input data for our evaluation networks, we have not found it necessary to parallelize feature extraction. The current prototype implementation of DISCLOSURE consists of a number of Perl and Python scripts, all running on the same server: a 16 core Intel(R) Xeon(R) CPU E5630 @ 2.53 GHz with 24 GB of ram.

In the course of our experiments, we run all individual feature extraction modules in series in 10 hours 53 minutes for 24 hours of data. Therefore, DISCLOSURE is able to perform at approximately 2x real-time.

5.6 Deployment Considerations

To deploy DISCLOSURE to a real network, the administrator should configure three main settings: the minimum flows threshold MinFlows, the classification threshold ClassThresh, and the FP reduction threshold RepThresh. This setup can be accomplished by performing the following steps:

1. Choose the MinFlows threshold.
   This value should be selected according to the NetFlow sampling rate for the monitored network and the amount of available training data. If the threshold is set too high, the system will not have enough C&C samples to properly train. But, if it is set too low, the system will train on poor data, and produce inaccurate models.
2. Choose an operational point on the ROC curve for ClassThresh.
   This value should be selected according to the traffic volume of the network and the misclassification rate that can be tolerated. On one extreme, the system will be able to detect most of the C&C servers, but it will also generate too many FPs. On the other end of the scale, the system will miss many C&C servers, but the results will be much more precise.
3. (OPTIONAL) Apply and tune the FP reduction module using RepThresh.
   To reduce the number of alerts in large networks, DISCLOSURE can be coupled with other detection or verification techniques. In this paper, we propose the use of an AS reputation-based score to filter the servers hosted in benign networks. The weights for the constituent reputation systems can be modified to have a more aggressive or a more lightweight filtering contribution, and the overall reputation score filtering strength can be adjusted by setting RepThresh.

5.7 Evasion Resilience

The detection approach presented in this paper is predicated on the assumption that existing botnets often exhibit a regular, detectable pattern in their communication with the C&C server. However, we have not discussed how strong this requirement is and how difficult it might be for an attacker to perturb this regularity to avoid detection.

To answer this question, we designed two botnet families (hereinafter B_1 and B_2) that attempt to evade our system by inserting a random delay between consecutive connections and a random

[1]This evidence included reports from ThreatExpert, various sandbox malware analysis tools, MaliciousUrl.com, or the offensive IP database.

length padding in each flow. In our implementation, we employed two different randomization functions. The first randomization function produces uniformly distributed values on a fixed range. This is intended to model a botnet in which the programmer uses a random number generator to select a value from a fixed range. The second family adopts a more sophisticated approach and generates random numbers from a Poisson distribution. In this case, we model a more complex scenario in which the botmaster tries to mimic the flow inter-arrival times of benign services, which are known to be well-approximated as a Poisson process [22].

In our experiment, we generated 300 C&C servers for both B_1 and B_2. First, we randomly specified the size of each botnet and the duration of its activity. Afterward, we created synthetic NetFlow data for each server, using one of the aforementioned randomization functions to generate random flow sizes and intervals between consecutive flows.

Each botnet was created according to the following parameters:

Botnet lifetime	1 - 33 days
Number of bots	1,000 - 100,000
Flow sizes	4 - 3076 bytes
Delay between flows	1 min - 1 hr

The only significant difference between the two botnet families is that for B_1, the delay between consecutive flows between each bot and the the C&C server was a uniformly-distributed random value between 1 minute and 1 hour. For B_2, the delay was, instead, drawn from a Poisson distribution whose *mean* was randomly chosen in the 1 minute to 1 hour range. We decided to set 1 hour as an upper bound since, in order to maintain a reasonable flexibility and control over the botnet, a botmaster must be able to to send commands to the infected machines with a delay that is no longer than an hour or two.

Finally, we added the synthetically-generated NetFlows to our labeled data set and re-ran the classification evaluation using a 10-fold cross-validation. In both cases, DISCLOSURE was able to detect all the experimental C&C servers belonging to B_1 and B_2. In addition, the addition of these synthetic botnets to the training set had the side effect of actually increasing the overall detection rate. In other words, some of the real botnets that were not detected by DISCLOSURE in our normal experiments were detected after we added the synthetic data. This implies that our detection models were not properly trained to detect this kind of variability in the C&C channel behavior. However, by adding many new samples with a randomized behavior to supplement the training set, DISCLOSURE was able to subsequently detect real botnets that present similar access patterns.

6. RELATED WORK

In the last couple of years, much work has been done to investigate the topologies of botnets, understand how they operate, and create novel approaches to detect them. In this section, we analyze and discuss the state-of-the-art in network-based botnet detection, as well as the previous work on NetFlow-based anomaly detection.

6.1 Network-based Botnet Detection

Botnet-related research can be divided to two groups: work that focuses on the measuring botnets [11, 15, 20, 27] and work that focuses on detecting them [6, 16–19, 21, 29, 32].

A number of botnet detection systems perform horizontal correlation. While initial detection proposals relied on some protocol-specific knowledge about the C&C channel [19, 21], subsequent techniques overcame this shortcoming [17, 29]. The main limitation of systems that perform horizontal correlation is that they need to observe multiple bots of the same botnet to spot behavioral similarities. This is significant because as botnets decrease in size [11], it becomes more difficult to protect small networks, and a botmaster can deliberately split infected machines within the same network range into different botnets.

A second line of research explored vertical correlation to be able detect individual bot-infected machines. A number of systems focus on specific protocols such as IRC [6, 16, 32]. More advanced systems in this category provide generic solutions. For example, BotHunter [18] correlates the output of three IDS sensors and a scan detection engine to identify different phases in the lifecyle of bots. Wurzinger et al. [35] automatically generates detection models to identify single bot infected machines without any a priori knowledge on the C&C protocol. The problem with the approach, however, is that the system is only able to detect known instances of botnets.

6.2 Anomaly Detection Through NetFlow Analysis

To date, there has been a considerable amount of research on anomaly detection using NetFlow analysis. While some of the works proposed anomaly detection methods to detect specific kinds of malware such as worms [34] or spamming botnets [28], others tried to propose more general approaches to distinguish malicious traffic from benign traffic [8, 13, 30].

Wagner et al. [34] present an entropy-based approach to identify fast worms in real-time network traffic. Dewaele et al. [13] extract sub-traces from randomly chosen traffic traces, model them using Gamma laws, and identify the anomalous traces by tuning the deviations in the parameters of the models. Brauckhoff et al. [8] present a histogram-based anomaly detector that identifies anomalous flows by combining various information extracted from multiple histogram-based anomaly detectors. Sperotto et al. [30] analyzed the time series constructed from both flow and packet sizes, and tested them to find whether they were sufficient for detecting general intrusions.

Another line of research focuses on the analysis of the impact of sampling methods applied on NetFlow traffic. Mai et al. [25] analyze a set of sampling techniques experimented with two classes of anomalies. The results show that all types of sampling techniques introduce a significant bias on anomaly detection. Another work [9] as well studied the impact of packet sampling on anomaly detection metrics. Their analysis concludes that entropy-based anomaly detection systems are more resilient to packet sampling because the sampling still preserves the distributional structure.

6.3 Botnet Detection with NetFlow Analysis

Only a few papers exist that propose to use NetFlow analysis to specifically detect botnets. For example, Livadas et al. [24] propose a system that identifies the C&C traffic of IRC-based botnets by using machine learning-based classification methods.

Francois et al. [14] present instead a NetFlow-based method that uses the PageRank algorithm to detect peer-to-peer botnets. In their experiments, the authors created synthetic bot traces that simulate the NetFlow behavior of three P2P botnet families.

Both works succeeded in the identification of a specific type of botnet traffic, IRC in the first case and peer-to-peer in the second. DISCLOSURE, on the other hand, can successfully detect C&C servers without any prior knowledge about the internals of the C&C protocol. Moreover, our experiments shows how DISCLOSURE can be used to perform real-time detection on large datasets.

7. CONCLUSIONS

Botnets continue to be a significant problem on the Internet. Accordingly, a great deal of research has focused on methods for detecting and mitigating the effects of botnets. While the ideal data source for large-scale botnet detection does not currently exist, there is, however, an alternative data source that is widely available today: NetFlow data [10]. Though it is attractive due to its ubiquity, NetFlow data imposes several challenges for performing accurate botnet detection. In particular, packet payloads are not included, and the collected data might be sampled.

In this paper, we present DISCLOSURE, a large-scale, wide-area botnet detection system that incorporates a combination of novel techniques to overcome the challenges imposed by the use of NetFlow data. In particular, we identify several groups of features

that allow DISCLOSURE to reliably distinguish C&C channels from benign traffic using NetFlow records: (i) flow sizes, (ii) client access patterns, and (iii) temporal behavior. Our experiments demonstrate that these features are not only effective in detecting current C&C channels, but that these features are relatively robust against expected countermeasures future botnets might deploy against our system. Furthermore, our technique is oblivious to the specific structure of known botnet C&C protocols.

We provide an extensive evaluation of DISCLOSURE over two real-world networks: a university network spanning a small country where no NetFlow sampling occurred, and a Tier 1 ISP where NetFlow data was sampled at a rate of one out of every ten thousand flows. Our evaluation demonstrates that DISCLOSURE is able to perform real-time detection of botnet C&C channels over data sets on the order of billions of flows per day.

8. ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 257007, the National Science Foundation (NSF) under grant CNS-1116777, and Secure Business Austria. Engin Kirda also thanks Sy and Laurie Sternberg for their generous support.

9. REFERENCES

[1] Alexa Web Information Company. `http://www.alexa.com/topsites/`, 2009.

[2] EXPOSURE: Exposing Malicious Domains. `http://exposure.iseclab.org/`, 2011.

[3] FIRE: FInding RoguE Networks. `http://www.maliciousnetworks.org/`, 2011.

[4] Google Safe Browsing. `http://www.google.com/safebrowsing/diagnostic?site=AS:as_number`, 2011.

[5] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *18th Annual Network and Distributed System Security Symposium (NDSS'11)*, 2011.

[6] J. Binkley and S. Singh. An Algorithm for Anomaly-based Botnet Detection. In *Usenix Steps to Reduce Unwanted Traffic on the Internet (SRUTI)*, 2006.

[7] G. E. P. Box, G. M. Jenkins, and G. Reinsel. Time Series Analysis: Forecasting and Control. In *3rd eddition Upper Saddle River, NJ: PrenticeâĂŞHall*, 1994.

[8] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K.Salamatian. Anomaly extraction in backbone networks using association rules. In *ACM Internet Measurement Conference (IMC'09)*, 2009.

[9] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina. Impact of packet sampling on anomaly detection metrics. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, 2006.

[10] B. Claise. Cisco systems netflow services export version 9, 2004.

[11] E. Cooke, F. Jahanian, and D. McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. In *1st Workshop on Steps to Reducing Unwanted Traffic on the Internet*, pages 39–44, 2005.

[12] N. Cristianini and J. Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. In *Cambridge University Press*, 2000.

[13] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho. Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures. In *Proceedings of the 2007 workshop on Large scale attack defense (LSAD'07)*, 2007.

[14] J. Francois, S. Wang, R. State, and T. Engel. Bottrack: Tracking botnets using netflow and pagerank. In *IFIP Networking 2011*, 2011.

[15] F. Freiling, T. Holz, and G. Wicherski. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In *10th European Symposium On Research In Computer Security*, 2005.

[16] J. Goebel and T. Holz. Rishi: Identify bot contaminated hosts by IRC nickname evaluation. In *Workshop on Hot Topics in Understanding Botnets*, 2007.

[17] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In *Usenix Security Symposium*, 2008.

[18] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In *16th Usenix Security Symposium*, 2007.

[19] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In *15th Annual Network and Distributed System Security Symposium (NDSS)*, 2008.

[20] J. John, A. Moshchuk, S. Gribble, and A. Krishnamurthy. Studying Spamming Botnets Using Botlab. In *6th Usenix Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[21] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale Botnet Detection and Characterization. In *Usenix Workshop on Hot Topics in Understanding Botnets*, 2007.

[22] D. E. Knuth. Seminumerical algorithms. In *The Art of Computer Programming, Volume 2, Addison Wesley*, 1969.

[23] A. Liaw and M. Wiener. Classification and regression by randomforest. In *R News*, volume 2/3, page 18, 2002.

[24] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer. Using machine learning techniques to identify botnet traffic. In *the 2nd IEEE LCN Workshop on Network Security (WoNS'2006)*, 2006.

[25] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang. Is sampled data sufficient for anomaly detection? In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, 2006.

[26] J. Quinlan. C4.5: Programs for machine learning. In *Morgan Kaufmann Publishers*, 1993.

[27] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *Internet Measurement Conference (IMC)*, 2006.

[28] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *SIGCOMM Comput. Commun.*, 2006.

[29] M. Reiter and T. Yen. Traffic aggregation for malware detection. In *DIMVA*, 2008.

[30] A. Sperotto, R. Sadre, and A. Pras. Anomaly characterization in flow-based traffic time series. In *Proceedings of the 8th IEEE international workshop on IP Operations and Management*, IPOM '08, pages 15–27, 2008.

[31] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda. Fire: Finding rogue networks. In *2009 Annual Computer Security Applications Conference (ACSAC'09)*, 2009.

[32] W. Strayer, R. Walsh, C. Livadas, and D. Lapsley. Detecting Botnets with Tight Command and Control. In *31st IEEE Conference on Local Computer Networks (LCN)*, 2006.

[33] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 2009.

[34] A. Wagner and B. Plattner. Entropy based worm and anomaly detection in fast ip networks. In *SIG SIDAR Graduierten-Workshop uber Reaktive Sicherheit (SPRING'06)*, 2006.

[35] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda. Automatically generating models for botnet detection. In *ESORICS 2009 : 14th European Symposium on Research in Computer Security*, 2009.