A look at data feature extraction and classification techniques for predicting botnet assaults

Bradley Walls
Management Information Systems
University of Arizona
Tucson, AZ
blwalls@email.arizona.edu

Abstract— **Botnets as a cyber security risk continue to evolve and transform in response to security measures deigned to stop them. Consequently, cyber security professionals must continue to develop new counter measures to defeat these threats. In this paper we build upon a prior body of work that uses machine learning algorithms to detect botnets within raw network traffic. Our approach is to look at network flow between two end points and to extract 23 distinguishing features from their packet exchange. We then train machine learning algorithms to classify the network flow as either malicious or benign. The dataset we use is the standard ISOT Dataset compiled by University of Victory. The machine learning algorithms implemented are a neural network, a decision trees, a SVMs, and an ensemble method known a RUSBoost. Training each of the machine learning algorithms with the 23 features using 5-fold cross validation we achieved an average of 99% in recall, precision, and accuracy across all methods.**

Keywords—botnet; cyber-security; PCAP; netflow; machine learning; fealture extraction

I. INTRODUCTION (*HEADING 1*)

The term "bot" (a.k.a. web bot, network bot, software bot) originates from the word robot. What this implies is that a bot, like a robot, is a piece of software capable of responding to commands and carrying out autonomous operations. When used for data gathering and beneficial applications such as web spidering, bots are welcome and useful tools. However, when used to distribute malicious software or attack computers, bots pose a significant threat to computer security. Criminals have evolved to use large number of bots in a collaborative fashion, known as botnets, to perform distributed denial-of-service (DDOS) attacks and distribute malware at unprecedented rates. While much research has been done in an attempt to thwart botnet attacks, the growing sophistication of hackers requires a reciprocal response in research to defend against such attacks.

In recent years, advanced network monitoring software has facilitated the collection of large datasets of network traffic and events. Forensic analysis of these datasets provides an opportunity to discover identifying characteristics of bots and botnets. This paper proposes two key research objectives in regards to this data:

- Examine datasets for features that may act as powerful discriminants in detecting new bot and botnet threat events.
- Gain a deeper understanding of the data and relevant algorithms for selecting feature sets and matching classification algorithms to fully exploit the data available for detecting bot and botnet threats.

II. MOTIVATION

On September 17, 2014, in a Statement before the House Homeland Security Committee, James Comey, Director FBI [1], described the state of Cyber Threats with the U.S. In his statement, Mr. Comey elucidated the substantial level of effort the FBI is engaging in to combat these threats and to stop "… the world's most dangerous botnets." Mr. Comey continued by highlighting some successes, "...Over the past several years, the FBI's efforts to combat these significant cyber threats have caused the disruption and dismantlement of numerous botnets, including Butterfly Bot, Rove Digital, Coreflood, ZeroAccess, and GameOver Zeus, resulting in numerous arrests, extraditions, and convictions." The implications of this address to the House Homeland Security Committee is that botnets, regardless of recent advances in detection and defense remain, a significant threat to domestic security.

Chris Rodrigues, a Senior Industry Analyst (Information & Network Sectuity) reiterates this sentiment in a recent SPIE article [2] as he advocates that security professionals remain diligent in their pursuit of new protections against the continually changing tactics utilized by disreputable individuals. Mr. Rodrigues indicates that the most recent trends have hackers and hacker groups utilizing complex botnets for covert operations to defeat security measures. To compound the problem work by Mullaney [3] and Lu [4] indicate that botnets are migrating to the Internet of Things (IoT) and mobile devices. The results are undeniable, with daily disruptions and data theft from large corporations such as Target, Home Depot, the US Postal Service, and many others. It is obvious, new techniques for defense against bots and botnets must be explored.

III. BACKGROUND

Often in the literature botnets are characterized as simply a coordinated group of bots spread across many computers controlled by an individual to accomplish harmful actions. While this may be accurate from a high level point of view, the reality is that botnets are instantiated in a variety of very different ways and understanding the differences is important to detecting, defending against, and defeating them. In this section of the paper we look at several dimensions of botnets and discuss how we will focus our research. The dimensions that we discuss below are:

- Good – vs – Bad
- Terminology
- Lifecycle
- Spreading Bots
- Protocols
- Topologies
- Command and Control (C&C)
- Commercial / Open Source Detection Software

A. Good –vs- Bad

Initially bots were small programs run on the Internet to perform an automated, repetitive function, such as searching for information on web sites across the Internet. The use of bots is appealing because they can be easily configured to do simple tasks at speeds much faster than humans. They will run 24 hours a day and through replication can be used to harness the power of distributed processing. One relatively well known and beneficial botnet is SETI@home [5]. SETI@home is a voluntary distributed computing project started by the Space Science Laboratory at the University of California Berkeley. For this project volunteers purposefully install a bot on their own personal computer (PC). The bot, when activated, makes contact with SETI servers and downloads some astronomical data. During times when a SETI-enabled PC is not being used the SETI bot processes the data and returns the results to the SETI servers without intervention of the PC's owner.

It did not take long for the criminal element on the Internet to realize how bots and botnets could be used to enhance their criminal endeavors. In addition to the DDOS attacks and malware distribution functions mentioned above, botnets are also know for other mischievous and/or harmful actions such as click fraud, phishing, and spam spreading. Within the context of this paper we consider bots from the perspective that they are used for nefarious purposes.

B. Terminology

As the cyber security landscape has evolved so has the terminology that is used to describe the components in this domain. The following list is comprised of a few key terms often used within the botnet arena.

- Bot / Web bot / Network Bot / Software Bot: The program (application) which is run that allows and attacker to gain control over an affected computer.
- Botnet: A network of bot infected computers that have been compromised.
- Bot Herder: The individual(s) responsible for the creating the botnet and who will eventually control and command the botnet.
- Zombie or Drone: An individual computer that has been infected by a bot and is part of a botnet.
- Botmaster: This is the bot server which commands and controls a botnet. The botmaster is operated by the Bot Herder.

C. Lifecycle:

To gain a cursory understanding of how botnets work it is useful to know the major stages in creating and operating a botnet. These key stages are illustrated in fig. 1 below:

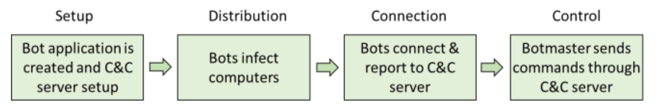

Setup
Distribution
Connection
Control
Bot application is created and C&C server setup
Bots infect computers
Bots connect & report to C&C server
Botmaster sends commands through C&C server

Figure 1: Key stages of the botnet lifecycle.

- In the Setup Stage the bot is created. During bot conception certain decisions are made regarding motivation, design, and implementation. Infection vectors are programmed, payloads are planned, and C&C details are configured.
- During the distribution stage bots propagate according to configuration parameters. Bots scan for vulnerabilities to take advantage of infection vectors.
- Once an infection is successful the bot activates an internal script that connects back to the C&C server (or peer), reports its status, and waits for commands.
- After the desired number of bots have reported in and joined the botnet, the botmaster can control the bots and command them to action. The botmaster can even update the bots, providing them with new capabilities and even a new C&C framework.

D. Spreading Bots

Bots are spread in a similar fashion to any type of malware. The goal is to get individuals to inadvertently install and activate the bot on their personal computer systems. This is typically done by exploiting a computer system vulnerability, exploiting weak security policies, or social engineering. For example an unaware computer user is tricked into visiting a faked website. The website scans for vulnerabilities and exploits them or tricks the user into downloading malware. Negash & Che [6], in their paper on modern botnets, provide additional details and examples of to the botnet infection phase.

E. Protocols

One common method of classifying botnets is by the type of networking protocol they use for communication. The three primary application layer networking protocols used for botnet communication are: Internet Relay Chat (IRC) Protocol, Peer-to-Peer (P2P) Protocol, and Hypertext Transfer Protocol (HTTP). These are discussed in detail below.

1) IRC

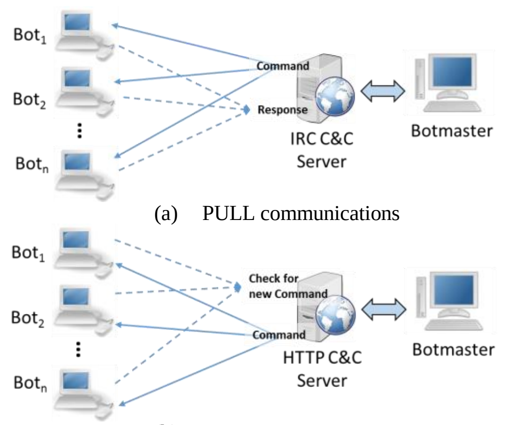
IRC is an open-standard electronic chat protocol that was designed for chat programs but was adapted to be used for the first generation of botnets. The original idea behind IRC was to provide a mechanism for users to hold group chat sessions in real-time across the Internet. A basic IRC system comprises two programs: a server and a client. The server run continuously on a computer whose address is well known. For botnets this acts as the C&C server. The client is the program that runs as an application on the computer(s) that will connect to the server. These clients when started, connect to the server and joins one or more "chat" channels. Many clients can connect to one server. In the case of a botnet, each instance of the client program is a bot. When IRC bots are first started they will connect to a preprogrammed server address and port number and then proceed through a standard set of exchanges. Researchers have used knowledge of this standard and predictable communication exchange to tackle the problem of IRC botnets. Since these type of botnets were the original botnets, much research has been done in this area, and advanced detection techniques have been developed with detection rates typically in the 99% to 100% range and few to no false alarms. Early work by Lividas, et al. [7], provides an example of successfully utilizing machine learning (ML) to detect IRC botnets. Although IRC botnets are the most simple and easy to detect, continuing code modifications by botnet designers have motivated researchers to continue their pursuit of new and updated detection methods. Awadi and Belaton [8] recently published new work using a spatial-temporal analysis of C&C traffic and behavior to make the detection of IRC botnets more robust.

2) P2P

Following a historical timeline, subsequent to IRC botnets came the evolution of P2P botnets. P2P networks share connections and resources directly with other "peer" computers on a network rather that going through a managing server. The P2P network protocol is designed to "overlay" or specify a logical network on top of another physical network. This means that nodes of the P2P network connect and communicate to other nodes in the P2P network through virtual or logical links independent of the underlying physical links. While there are many details to this kind of network, when it comes to botnets utilizing P2P the key feature is that each node keeps a routing table of connected nodes. The implication is that if a node can't accomplish a task or pass a message, it can search for one that can. This one feature was a boon for botnet designers as it got around the main vulnerability of IRC botnets, one central C&C server. As cyber security professionals developed tools to find and take down the C&C server, thereby rendering the botnet useless, the P2P botnets have gotten around this by distributing C&C to all nodes in the network. Initial work by researchers was done to gain a better understanding of these botnets. Examples of some of this research include Grizzard, et al. [9], who presented a case study that took a detailed look into the operation of the Trojan.Peacomm P2P botnet and Holz, et al. [10], who did a deep dive into the function of the Stormnet P2P botnet. As a result of the new complexity of P2P botnets, different ideas for detection of botnet began to emerge. One idea was to look for anomalies in network traffic flows. This approach has gained of momentum and is showing promising results in detection botnets of all flavors. Narang, et al. [11], successfully applied this approach specifically to P2P botnets.

3) HTTP

Simply put, HTTP is the foundation for communication on the World Wide Web (WWW), so it is only natural that botnets would eventually adopt this network communications protocol. The move away from P2P is thought to have occurred for a couple of reasons. First, implementation of a P2P communication protocol is complex and difficult to manage. Second, botmaster commands are propagated through the network and distributed by other bots. Consequently, the delivery and response status of instructions is not easily monitored by the botmaster. This has led botnet developers back to a model using a single C&C server, but with the advantages of using HTTP. The primary advantage gained from using HTTP is that HTTP Botnet traffic activity is merged (effectively hidden) among all the other traffic of the Internet. IRC and P2P botnets had their own protocol structures that could be parsed out of the majority of all other Internet traffic, but this is not the case for HTTP botnets. Even though HTTP botnets use a central C&C server there is a significant difference from the IRC botnet use of a central C&C server that provides a significant advantage: IRC botnets use a PUSH C&C communication strategy where HTTP botnets use a PULL C&C communication strategy. Fig. 2 below illustrates the differences:



(a) PULL communications

(b) PUSH communications

Figure 2: C&C Communication Strategies.

In the IRC botnet PUSH C&C communication strategy IRC bots connect to the preprogrammed IRC C&C server and corresponding channel and remain connected waiting for the botmaster to issue commands. The HTTP botnet PULL C&C communication strategy differs in that the bots are programmed to periodically visit preprogrammed C&C web servers to check for new commands and updates. In 2012 a security industry report [12] summarized nine of the most dangerous botnets for that year - six of them were HTTP based. Given the seriousness of the HTTP botnet threat researcher have rallied to come up with innovate strategies to detect them. Eslahi, et al. [13], exploits the knowledge of the HTTP botnet connection process and the preprogrammed check-in nature of these bots to derive a periodicity classification model for detecting HTTP botnets. Even with work like Eslahi's, due to the recent arrival of HTTP botnets on the botnet scene, research in this area is minimal by comparison to the number of studies on IRC and P2P botnets.

F. Topologies

Another popular way of differentiating botnets is by the architectural layout of their network connections. However, in most cases the protocol determines and/or limits the topology possibility for the botnet. The three primary networking topologies used for botnet connections are: Centralized, Decentralized, and Hybrid.

1) Centralized

The centralized topology is the configuration used by IRC botnets and most HTTP botnets.

- Advantages are the simplicity of programing, managing, and commanding.
- A disadvantage is its single point of failure. If the C&C Server is removed the entire botnet is rendered ineffective.

2) Decentralized

The decentralized topology is the configuration used by P2P botnets.

- An advantage is that there is no single point of failure, multiple bots can be removed and the botnet can still function.
- Some disadvantage are the complexity of programming, managing, and commanding.

3) Hybrid

Various attempts by botnet architects have been attempted at combining both centralized and decentralized topologies to gain the most from the advantages of each, while minimizing the impact of their disadvantages. The paper by Wang, et al. [26], discusses one such hybrid architecture. This topology creates a type of tiered network where the first tier is a limited sized P2P network of "servant" bots and the second tier is composed of client bots. In the top tier the P2P bots are responsible for distributing C&C orders, and the limited sized reduces complexity and eases management concerns. The second tier has each client bot report to two servant bots, which eliminates the single point of failure.

IV. Prior Research

Bots and botnets have been a significant threat and a major source of problems for security professionals over the years. As such, they have caused significant security concerns, and much research has been conducted on these software robots. However, the nature of bots is dynamic and the way botnets are used is ever changing. In light of the evolving nature of this type of software construct, continuing efforts on various battle fronts are underway to devise strategies and methodologies for defeating this malware. In the prior section we presented a few examples of research that looked at a specific category of botnet. In this section we highlight some additional efforts using detection approaches that can be applied to two or more types of botnets.

One recent paper by Khattak, et al. [14], does a thorough job of capturing the contemporary understanding of botnets. The paper looks into botnet behaviors, how they are used, and command and control considerations. A survey of current botnet detection mechanisms is provided and an assessment of existing defensive techniques is discussed. The key take away from this paper is that the authors suggest new benefits may be achieved through the coupling of detectable botnet features with matching predictive analysis mechanisms.

Another paper by Chakchai So-In, et al. [15], puts forth the idea of taking a more active approach to network security and investigates six traditional classification models for network intrusion detection. The described data mining and classification models are applied to the KDD CUP 1999 dataset which contains up to 41 attributes of network access behaviors and a variety of known networking threats. The dataset was derived through an intrusion detection simulation of the U.S. Air Force local area networks.

Other research has produced standard frameworks and software for botnet detection. Bothunter [16] is one such software package for botnet intrusion detection that is freely distributed (www.bothunter.net). Bothunter works by correlating network traffic flows to infection sequence models. Follow on work by Guofei et al. [17] developed a framework, titled Botminer, which proposes a technique for botnet detection that is protocol independent. Botminer perform detection by clustering traffic on the basis of malicious activity and communication patterns. Lastly in this sequence of standard frameworks is additional work by Gu et al. [18] that evaluates C&C activity. Their framework, BotSniffer, is based on the observation that pre-programmed activities of C&C bots within the same botnet will likely demonstrate spatial-temporal correlation and similarity, thus providing a detectable flag.

The last two papers discussed in this section present the currently popular idea of using data mining techniques applied to network traffic flows. The prevailing idea behind these techniques is to look at a complete network flow and analyze the statistics of the flow either individually or over a specified periods of time. A network flow is the group of packets transmitted over a network from a source location to a destination during a single connection session or call. The statistics and features of these flows are then evaluated using data mining and machine learning techniques to develop classification parameters for identifying botnets. In the approach by Zhao, et al. [19], 13 attributes are captured from network flows to train a decision tree classifier. Their paper justifies the selection of a decision tree methodology by presenting prior work that compares detection results using a decision tree to results from neural networks, support vector machines, Gaussian and nearest neighbor classifiers, and Naïve Bayes algorithms. Kirubavathi & Anitha [20] use a set of four attributes (some aggregate) captured from network flows to train three classifiers: a boosted decision tree, a Naïve Bayesian classifier, and a support vector machine. A unique contribution of their paper is the examination of flow statistics using varying sized sliding time windows to bound numerical calculation. Techniques from both papers show true positive rates at >90% and false positive rates around ≈ 5%.

An interesting component of these and other papers publish over the past year or two is the inclusion of one or two common botnet datasets as part of the performance evaluation. Prior to this, most botnet papers used a dataset captured in their own sandbox environment or a proprietary dataset from a security company. However, since the creation of the ISOT Dataset [21], more and more research seem to be utilizing this data as a comparative baseline for evaluating how their algorithms are performing.

V. Approach

For this research we will be looking exclusively at TCP flows to detect botnets. There are several motivations:

- TCP is the foundation for P2P and HTTP-based botnets.

- The number of studies focusing on the detection of HTTP-based botnets is relatively low (compared to the number of those on IRC-based and P2P botnets)
- In a May 2016 security industry report [22] the ten top malware threats were summarized. Seven of these ten malware threats utilized HTTP alone or in a hybrid combination with another communication mechanism.
- The newest cyber security danger is the mobile botnet. According to the security reports [23], over half use the HTTP protocol the remaining are either SMS based or a combination of SMS-HTTP.

As a starting point to develop our detection approach we will adopt the current research trend of using the ISOT dataset as the baseline standard for performance evaluation. The ISOT Dataset is described in detail at the University of Victory webpage [21] and is a cleverly crafted experimental dataset combining several publicly available malicious and non-malicious datasets. From a birds-eye perspective the ISOT dataset is provided as 12GByte data file with 161 million packets. To handle such a large set of data we first used a product call CapLoader by Netresec (http://www.netresec.com/) to create flows out of our packets and to filter out non-TCP traffic. We then saved the flow data of interest into multiple *.pcap files of a manageable size. Next, we created a custom Java program using the jNetPcap library to step through the flows and calculate flow statistics. These statistics were subsequently saved into a CSV file. For the final stage of our study we used machine learning tools within the MATLAB® environment to train, test, and validate several classification approaches using our feature set. Fig. 3 below depicts our approach and the following subsections discuss the details.

A. *ISOT Data & network packets*

As previously mentioned, the ISOT Dataset is described in detail at the University of Victory webpage [21] and is an experimental dataset combining publicly available malicious and non-malicious datasets. The malicious datasets contain network traffic from the Waledac, Storm and Zeus botnets. The non-malicious network traffic is a mixture of traffic from a massively multiplayer online game (MMOG), a popular bit torrent, and general traffic from a medium size enterprise network. The network data was intelligently combined and replayed using the TcpReplay tool in order to homogenize the network behavior. The resulting dataset contains 23 representative subnets with both malicious and non-malicious traffic. One critical key to this dataset is the labeling of malicious –vs- non-malicious traffic using recoded MAC address labels.

B. *Aggregation & Network Flows*

A network flow (a.k.a. packet flow) is a sequence (or group) of packets from one destination to another that describes a "call" between two end points. This grouping is a necessary component of our approach because we rely on the temporal features of an ongoing communication sequence to provide the rich feature set that allows for the discrimination between malicious and non-malicious traffic. To accomplish the task of data partitioning and identification of flow groups we used the CapLoader tool. Caploader allowed us to load in the large ISOT dataset, separate the TCP flows from the UDP flows, and save the TCP flows in smaller size PCAP files for easier downstream processing.

Mimicking the flow approach to detection, mentioned above by Zhao, et al. [19] and Kirubavathi & Anitha [20], our approach builds upon this prior research by increasing the number of flow features used for analysis. The complete set of features used within this project are listed in the next section. The motivation for using flows is that prior research examining single packets primarily relied on additional information from historical captures such as signature of malicious payloads or known IP addresses of malicious content providers. It is the desire to move away from having to have such *a priori* knowledge about threats and move to a more general behavioral approach to detection. To this end, as discussed in the next section, we do not use IP addresses, port numbers, or payload signatures to inform our detection approach.

C. *Processed Flow Statistics*

To calculate the comprehensive set of features used in this research, we created a Java program that leverages the open source jNetPcap library. jNetPcap is an ideal library to use for calculating customized features. jNetPcap provides a Java wrapper for the libpcap & winpcap libraries allowing for the capture and management of network traffic. While the maximum number of features in previous work was 13, we took the approach of looking at expanding the number a network traffic feature presented to the machine learning algorithm in order to let the algorithms determine the correlations among features and make use of the potentially subtle discriminating power of any one of the features. It will be the goal of future work to more closely examine the nature of each feature and understand the rationale of how the features are used to provide discriminating power for classification.

Our Java program (provided at http://www.azsecure-data.org/) was written to take in the Pcap files saved by the CapLoader tool and derive the 31 features for each flow. The data was saved to a CSV file for easy downstream processing.

D. *Machine Learning & Classificaion*

For this research we use a family of machine learning algorithms that fall under the category of supervised machine learning (ML) algorithms. The overarching goal of supervised machine learning is to build a model that makes predictions

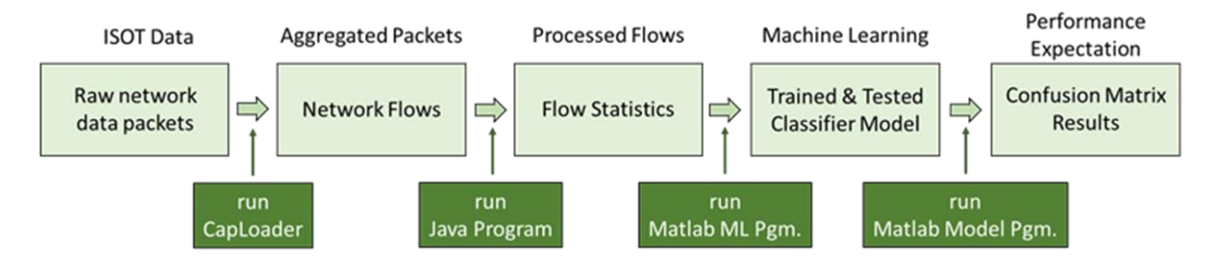


Figure 3: Data processing flow.

based on a coupling between observations and outcomes. Adaptive algorithms are used to identify patterns in data and learn from the observations. Supervised learning occurs in two steps, shown in fig. 4 below. In Step 1, a known set of input data and known responses to the data are used to train model. Next, in Step 2, the model is used to generate reasonable predictions in response to new data.

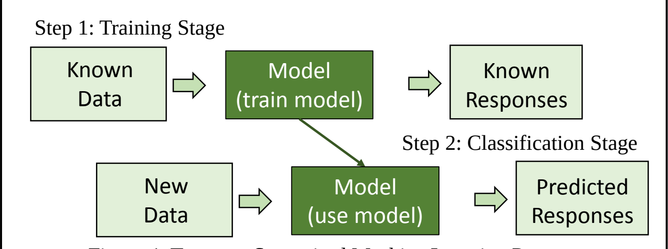


Figure 4: Two step Supervised Machine Learning Process.

Supervised ML is typically thought of as having two broad categories: regression and classification. In regression, continuous predictions for an observation are generated. Stock market predictions in response to some observed news report is one popular regression example. For classification the goal is about learning to assign objects to one of a set of classes given an observation. This second category of supervised ML algorithms is what we will be using for this project. Our goal is to assign a network flow to the category of either malicious traffic or non-malicious traffic given a set of features as our observation.

Neural networks, decision trees, SVMs, and ensemble methods are popular classification-focused supervised ML algorithms.

MATLAB® is a leading mathematical computing software that provides a convenient set of tools and libraries to support ML for our investigations. More specifically, MATLAB® supports each of the ML algorithms used in this research.

E. *Performance Analysis*

For performance analysis of our various algorithms we use the standard error matrix format also known as the confusion matrix. The confusion matrix gives a simple and compact way of visualizing performance of our machine learning algorithms.

One nice aspect of the confusion matrix is that it is easy to pick out Type I and Type II errors. The confusion matrix also makes it straight forward to calculate secondary performance metrics, such as sensitivity, precision, accuracy, and F1 score.

- The "False Positive" coordinate is equal to the Type I error. This is also known as a "false hit". Statistically speaking, Type I errors occur when the null hypothesis (H0) is true, but rejected.
- The "False Negative" coordinate is equal to the Type II error. This is also known as a direct "miss". Statistically speaking, Type II errors occur when the null hypothesis (H0) is false, but fails to be rejected.
- Sensitivity, also known as Recall, measures the proportion of positives that are correctly labeled as positive. It is calculated by:

$$Sensitivity\ (Recall) = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- Precision, also known as the Positive Predictive Value, measures the proportion of positive results that are positive. It is calculated by:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

- Accuracy is the degree of closeness of measurements of a quantity to that quantity's true value. It is calculated by:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

- F1 scores are a measure of a test's accuracy. It is the harmonic mean of Sensitivity and Precision. It is calculated by:

$$F1 = \frac{2 * True\ Positive}{(2 * True\ Positive) + False\ Positive + False\ Negative}$$

VI. Results

Prior to applying the network feature to the machine learning algorithm we performed data exploration. Our goal was to look for evidence that our features would have some measure of discriminating power.

Our initial data exploration included generating histograms of the mean values for each feature. For each chart we overlaid the plot for both malicious and non-malicious to see if we could identify any features that by themselves could provide some measure of discrimination. In fig. 5 below we show two examples of these plots - one that shows good discriminating power and one that is not so good.

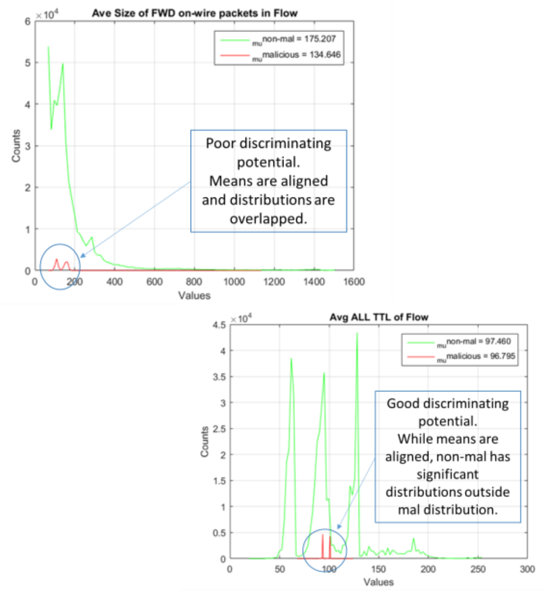


Figure 5: Examples of data exploration plots. Means of Average FWD On-Wire & All TTL Size.

After examining the individual feature means, we looked at some cross-feature distributions. Again, the motivation was to see if our features contained variables that would allow for discrimination between malicious and non-malicious network flows. In fig. 6 below we show three examples of these plots -

one that is not so good and two that show good discriminating power.

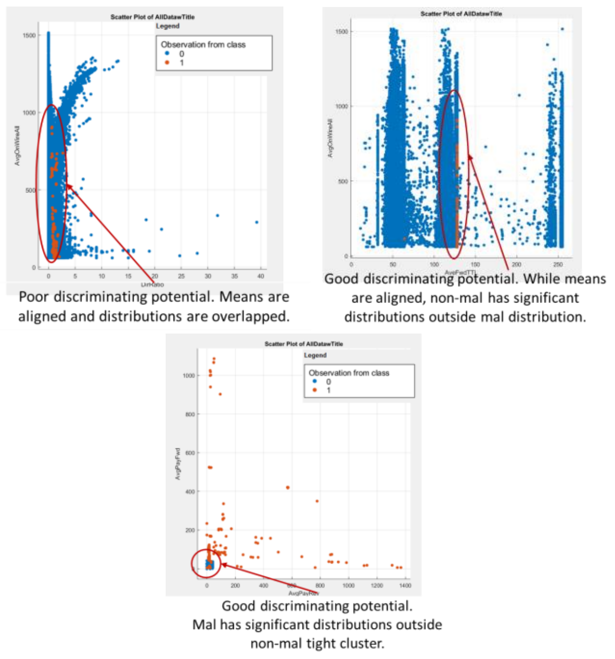
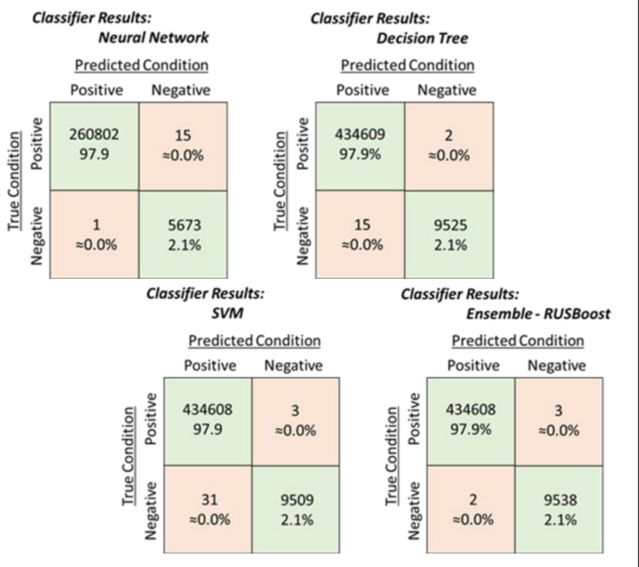


Figure 6: Examples of data exploration plots. Means of Average FWD On-Wire & All TTL Size.

Once our data exploration was complete and we had confidence that discrimination was possible, MATLAB® was used to implement the four popular machine learning algorithms: Neural Network, Decision Tree, SVM, and RUSBoot.

For the Neural Network we used a 70% hold out method to separate the data into training and testing data. In the case of the Decision Tree, SVM, and RUSBoot we used a 5-fold cross-validation for training and testing. The confusion matrices for each of our trained classifiers is provided in fig. 7.



Classifier Results: Neural Network

True Condition	Predicted Condition: Positive	Predicted Condition: Negative
Positive	260802 97.9	15 ≈0.0%
Negative	1 ≈0.0%	5673 2.1%

Classifier Results: Decision Tree

True Condition	Predicted Condition: Positive	Predicted Condition: Negative
Positive	434609 97.9%	2 ≈0.0%
Negative	15 ≈0.0%	9525 2.1%

Classifier Results: SVM

True Condition	Predicted Condition: Positive	Predicted Condition: Negative
Positive	434608 97.9	3 ≈0.0%
Negative	31 ≈0.0%	9509 2.1%

Classifier Results: Ensemble - RUSBoost

True Condition	Predicted Condition: Positive	Predicted Condition: Negative
Positive	434608 97.9%	3 ≈0.0%
Negative	2 ≈0.0%	9538 2.1%

Figure 7: Classification Confusion Matrix Results

The summary performance statistics are given in Table 1.

	Recall	Precision	Accuracy	F1 Score
Neural Network	0.999942	0.999996	0.99994	0.999969
Decision Tree	0.999995	0.999965	0.999962	0.99998
SVM	0.999993	0.999929	0.999923	0.999961
RUSBoost	0.999993	0.999995	0.999989	0.999994

Table 1: Summary performance statistics

The good True-Positive and True-Negative is indicative of a set of features that had good discriminating power. Comparing these results to those of our two primary reference studies that utilized flow features, we see improved outcomes through our use of additional features (Tables 2 & 3):

	Recall	Precision	Accuracy	F1 Score
SVM	0.925	0.912	0.920	0.918
Naïve Bayes	0.961	0.987	0.991	0.969
Boost_j48	0.979	0.958	0.959	0.968

Table 2: Results by Kirubavat & Anitha [20]

	Recall	Precision	Accuracy	F1 Score
Neural Network	0.53	0.336	0.545	0.412
Decision Tree	0.929	0.973	0.957	0.95
SVM	0.998	1.0	0.998	0.998
Naïve Bayes	0.991	0.984	0.984	0.987

Table 3: Results by Kalaivani & Vijaya [24]

While our improved results over the previous studies lends credibility to our feature selection and classification approaches, there are several important considerations to take into account when comparing results.

- Kirubavat & Anitha's paper includes 4 flow features, not including source and destination IP Address, as part of their feature set.
- Kirubavat & Anitha achieve an additional dimension to their captured data by using specified feature captured for a flow over a specified time window.
- Kirubavat & Anitha paper uses the ISOT data combined with other network traffic including additional botnet examples.
- Kalaivani & Vijaya's paper includes include 16 flow features, including source and destination IP Address, as part of their feature set.
- Kalaivani & Vijaya's paper uses the CTU-13 data combined this dataset is larger and more complex that the ISOT data with additional botnet examples.

VII. Limitation & Future Directions

One caveat to our investigation is that we examined each flow in its entirety. This means that statistical features for the flows are calculated over the entire flow. In the ISOT dataset some flows consist of a few quick packets and some flows are thousands of packets over minutes. We believe that this leads to two issues:

- Our approach as currently implemented is not suitable for real-time field deployment to prevent botnet attacks. Instead, the current research is better positioned as a forensics tool for offline analysis. However, with slight modification we could easily reposition this work.
- Analysis of botnet traffic using the ISOT dataset represents a first step towards validating our approach. However, the generalizability and true performance of our approach can only be validated by testing against a more expansive dataset that includes additional malware and non-malware network flows.

To move the state of botnet threat detection forward we see two natural extensions to this work in the areas of sample data and feature analysis.

Data Collection:

Datasets that meet the following criteria:

- It is real data
- It is recent data
- It contains a wide variety of network intrusions
- Malicious network intrusions can be identified after they have occurred

Feature Analysis:

For this work we took the brute force approach of simply collecting lots of features to create the vector inputs to the classification algorithms. Future work should include a more sophisticated examination of features, along with:

- Data mining to find patterns
- Analysis of features for understand of why they discriminate
- Assessment of feature set correlations
- Identification of additional classification algorithms
- Testing and evaluation of additional classifiers

VIII. Conclusion

In this research we achieved our overarching objective to examine network traffic datasets for features that may act as powerful discriminants in detecting new and existing botnet threat. In our approach we looked at 28 network flow features that when taken together as a whole and applied to a set of machine learning algorithms achieved an average of 99% in recall, precision, and accuracy. We saw consistent results across four different classification techniques indicating that the set of features we collected provides a robust mechanism for discrimination between malicious and non-malicious network flows for the dataset evaluated. We suggest future studies to include: a more analytic examination of the feature set, a more comprehensive dataset, and an expansion of investigated classification techniques.

The threat of botnets will continue and they will evolve. Even as of the writing of this report the market for botnets is expanding because now anyone, for a price, can rent a botnet [25]. The incentives for botnet creators is growing and as detection technologies advance so will the creativity and sophistication that is used to create this malware. Thus, we must continue to conduct research such as this to be vigilant in developing new way of detecting and defeating these threats.

References

[1] http://www.fbi.gov/news/testimony/worldwide-threats-to-the-homeland

[2] Rodrigues, Chris, "The Forgotten Barometer: Bot Detection as an Integral Security Technology," SPIE Stratecast Perspective & Insight for Executives (August 2014)

[3] Mullaney, C. (2012). Android.Bmaster: A million-dollar mobile botnet. Retrieved from http://www.symantec.com/connect/blogs/androidbmaster-million-dollar-mobile-botnet:Symantec

[4] Lu, Z., Wang, W., & Wang C. (2014). How can botnets cause storms? Understanding the evolution and impact of mobile botnets. INFOCOM, 2014 Proceedings IEEE (pp. 1501–1509), Toronto, ON.

[5] http://setiathome.ssl.berkeley.edu/

[6] Negash, N; Che, X. (2015) An Overview of Modern Botnets, Information Security Journal: A Global Perspective, 24:4-6, 127-132

[7] Livadas, Carl, et al. "Usilng machine learning technliques to identify botnet traffic." Proceedings. 2006 31st IEEE Conference on Local Computer Networks. IEEE, 2006.

[8] Awadi, A. H. R. A., & Belaton, B. (2015). Multi-phase IRC botnet and botnet behavior detection model. arXiv preprint arXiv:1501.03241.

[9] Grizzard, J. B., Sharma, V., Nunnery, C., Kang, B. B., & Dagon, D. (2007). Peer-to-Peer Botnets: Overview and Case Study. HotBots.

[10] Holz, T., Steiner, M., Dahl, F., Biersack, E., and Freiling, F. 2008. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET'08), Fabian Monrose (Ed.). USENIX Association, Berkeley, CA, USA, Article 9 , 9 pages.

[11] Narang, P., Reddy, J. M., & Hota, C. (2013, August). Feature selection for detection of peer-to-peer botnet traffic. In Proceedings of the 6th ACM India Computing Convention (p. 16). ACM.

[12] http://www.siliconindia.com/news/enterpriseit/9-Most-Dangerous-Botnets-of-2012-nid-135433-cid-7.html

[13] Eslahi, M., Rohmad, M. S., Nilsaz, H., Naseri, M. V., Tahir, N. M., & Hashim, H. (2015, April). Periodicity classification of HTTP traffic to detect HTTP Botnets. In Computer Applications & Industrial Electronics (ISCAIE), 2015 IEEE Symposium on (pp. 119-123). IEEE.

[14] Khattak, S. ; Ramay, N.R. ; Khan, K.R. ; Syed, A.A. ; Khayam, S.A. , "A Taxonomy of Botnet Behavior, Detection, and Defense,", Communications Surveys & Tutorials, IEEE Volume: 16 , Issue: 2 ,Publication Year: 2014 , Page(s): 898 – 924

[15] Chakchai So-In; Mongkonchai, N.; Aimtongkham, P.; Wijitsopon, K.;Rujirakul, K. ,"An evaluation of data mining classification models for network intrusion detection," Digital Information and Communication Technology and it's Applications (DICTAP), 2014 Fourth International Conference on, Publication Year: 2014 , Page(s): 90 – 94

[16] Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W., & Lee, W. (2007, August). Bothunter: Detecting malware infection through ids-driven dialog correlation. In Usenix Security (Vol. 7, pp. 1-16).

[17] Guofei, G., Roberto, P., Junjie, Z., Wenke, L.: BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Proceedings of the 17th Conference on Security Symposium. USENIX Association, San Jose (2008)

[18] Gu, G., Zhang, J., & Lee, W. (2008). BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. Proceedings of the 15th Annual Network and Distributed System Security Symposium.

[19] Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., & Garant, D. (2013). Botnet detection based on traffic behavior analysis and flow intervals. Computers & Security, 39, 2-16.

[20] Kirubavathi, G., & Anitha, R. (2016). Botnet detection via mining of traffic flow characteristics. Comp. & Electrical Engineering, 50, 91-101.

[21] www.uvic.ca/engineering/ece/isot/datasets/

[22] news.softpedia.com/news/top-10-malware-threats-may-2016-edition-505542.shtml

[23] www.virusbulletin.com/virusbulletin/2015/03/timeline-mobile-botnets/

[24] Kalaivani, P., and M. S. Vijaya. "Mining Based Detection of botnet traffic in Network Flow." IRACST. Vol. 6, No.1, Jan 2016

[25] www.zdnet.com/article/study-finds-the-average-price-for-renting-a-botnet/

[26] Wang, Ping, Sherri Sparks, and Cliff C. Zou. "An Advanced Hybrid Peer-to-Peer Botnet." IEEE Transactions on Dependable and Secure Computing 7.2 (2010): 113