A study guide to “An Empirical comparison of botnet detection methods”

Thank you, Dr. David Eargle, for your editing and input

This document is a study guide for “An Empirical comparison of botnet detection methods” (Garcia et al 2014). A website for the project is available (https://www.stratosphereips.org/datasets-ctu13), and python source code for analyses they ran (https://sourceforge.net/projects/botnetdetectorscomparer/).

Garcia et al. (2014) discusses some concepts that may be unfamiliar to non-domain experts. For example, it would be difficult to extract meaningful insights without first basic understanding of the following:

- Networking, understanding of how devices talk to each other is mandatory to move forward. Having experience with packet capture programs or the structure of other traffic diagnostics will help.
- Botnets, how they can operate, and the difference between benign (normal) and malicious (botnet) traffic
- Machine learning as well as model evaluation metrics. One of the papers main contributions is an adaptation to classical model evaluation that will be difficult to understand without grasp of the fundamentals.

This document is structured as follows: First, a section with important preliminary concepts is included. Following this, each section-header in this document refers to a specific section of Garcia et al. (2014), with subsection headers in this document being either direct quotes, or broad questions that a reader may have while reading a particular section of Garcia et al. (2014).Preliminary information
The material in this section covers background information on botnets, packet captures (pcaps), and netflows.

What is a botnet?

A botnet is a network of compromised internet connected systems infected and under control by a malicious actor. Botnets can be used to launch DDoS attacks, rake personal information, or steal computer time to mine cryptocurrencies or spread more malware.

Botnets are typically controlled in a Command and control structure, with a botmaster pushing commands and information out to the net from a central server or network of server. They can also run on a peer 2 peer structure making it harder to locate the origin of the botnet since there is no obvious locatable c&c.

A second definition of a botnet from searchsecurity (https://searchsecurity.techtarget.com/definition/botnet):

“The term botnet is derived from the words robot and network. A bot in this case is a device infected by malicious code, which then becomes part of a network, or net, of infected devices controlled by a single attacker or attack group. A bot is sometimes called a zombie, and a botnet is sometimes referred to as a zombie army. Both names (bot and zombie) imply the mindless automatic propagation of something malicious (malware) by agents that are possessed in some way (by the threat actor).

The botnet malware typically looks for vulnerable devices across the internet, rather than targeting specific individuals, companies or industries. The objective for creating a botnet is to infect as many connected devices as possible and to use the computing power and resources of those devices for automated tasks that generally remain hidden to the users of the devices."

What is a Netflow? And why are they using them?

Why use netflows instead of packet captures (pcaps)? Netflows relate more as summary information, while pcaps are nitty gritty full-detail dumps. Netflows can take up less space as they contain less information I.E Netflows specify total packets and bytes while pcaps specify what was in the packets, shows it, and the information transmitted. And since netflows have way less data overhead things like routers and switches can create, and store, netflows for later inspection.

Why do Netflows have less data overhead?

A netflow may say a conversation between two IP's consisted of 128 bytes of information, and this netflow itself is only 8 bytes in size. While the entire pcap of this conversation would be greater than 128 bytes.

A typical output of a NetFlow command line tool (`nfdump` in this case) when printing the stored flows may look as follows:

```
Date flow start          Duration Proto   Src IP Addr:Port      Dst IP Addr:Port     Packets    Bytes Flows
2010-09-01 00:00:00.459     0.000 UDP     127.0.0.1:24920   ->  192.168.0.1:22126        1       46     1
2010-09-01 00:00:00.363     0.000 UDP     192.168.0.1:22126 ->  127.0.0.1:24920          1       80     1
```

https://en.wikipedia.org/wiki/NetFlow

This screenshot is an example of raw netflow statistics. Notice the difference between what you would see in a pcap tool like Wireshark and the Netflow format. Netflow enabled devices can keep track of and store the above information.

At this high level it is still possible or easier to identify foreign IP's causing unusual load on the system (talking to the botnet or being used by the C&C for illicit activity) Or to identify known IP's on the network that have been infected and are sending out unusual communication.

Section 1

Overview and frame of reference

Garcia et al. (2014) discusses ideas about why locating botnets and stopping them is difficult. This difficulty is substantiated by poor academic and scientific practice in the malware analysis field that translates over into the working world. While reading, identify main concepts of botnet detection and the botnet detection industry that would affect a company or business and their ability to safely protect their assets (a security mindset).

Garcia et al. (2014) is useful because

The paper makes what it claims to be the first real attempt at comparing botnet detection methods and evaluating them with the usual machine learning evaluations, FP, FN, but also adds *time* as a metric of evaluation which is very important. The earlier you can detect and deter a botnet the more you can

minimize spread and damage. The authors also produced a dataset of multiple botnet attack scenarios and published the netflows for use by other researchers and practitioners. A current glaring problem in studying botnet attacks is most researchers' experiments are conducted on proprietary or otherwise private datasets. Meaning other experiments cannot be conducted on the datasets thus halting any comparison of methods. Among other things, its contributions aim to remedy comparison difficulties by providing the botnet-oriented comparison metrics, their community dataset to evaluate on, and a methodology for future experiments to follow.

Section 2: Comparison of Methods

"The comparison of detection methods is usually considered a difficult task" (Section 2, page 2)

Why are detection methods hard to compare? Well the datasets that malware detection methods are used on are often private (Next section) Malware research is also often detached from typical academic experimental rigor (next section) Malware analysts and researchers poorly document steps taken in their experiment that make reproducing their results difficult as documented by multiple papers Aviv, Rosssow, Tavallae etc. The combination of dataset privacy and prior research being hard to compare doubles down on making things difficult and doesn't give good reason to progress the ability to compare.

Section 2.1

"Datasets tend to be private" (Section 2.1 page 2)

Why are datasets used in malware analysis and experimentation typically private? Datasets can be owned by companies who paid time and money to collect them, and only wish for them to be used for certain reasons or won't release them for competitive concerns (proprietary). In Malware analysis sometimes the issue is privacy of the network entities in the actual dataset as they can be quite detailed. Possessing a dataset of an infected network could reveal to a non-academic, possibly malicious entity, vital information that could put the network at risk for different malware attacks or botnet recruitment. Or just details of a network in general, or academic selfishness.

"Regarding the creation of datasets for malware-related research, Rossow et al. (2012) presented a good paper about the prudent practices for designing malware experiments" (Section 2.1 page 2)

Rossow et al. describes a prudent paper as prudent "if they are correct, realistic, transparent, and do not harm others." (Rossow Section 1, 1) And brings up that experimental transparency is lacking throughout the malware research academic community. Which makes comparison and research validation very difficult. Malware experiments as referred to here are actually academic/scientific research projects. Rossow calls out the malware research community as does many other writers saying the steps taken when conducting an experiment need to be verbatim to make reproduction of the experiment easy. Rossow et al define new vital malware experimentation goals consisting of transparency, realism, correctness, and safety (Rossow Section 1, 2) In terms of correctness, all of their recommendations outline issues in dataset correctness that need to be addressed. Transparency, nomenclature of malware is difficult, but it is important to list names of identical malware so people can apply research to appropriate malware. Malware is also constantly changing so it's important to list *dates* when samples were collected and executed. The recommendations continue - Rossow wants the Malware-

scientific community to simply be much more detailed in their approaches to more closely align network experimentation with other experimental sciences

Section 2.2 Datasets Available

"IRC-Based Botnet attack" (Section 2.2 page 2)

– IRC stands for internet relay chat. An IRC Botnet is a botnet which communicates through a chatroom. I.E Your malware infiltrates a PC, downloads an IRC client and begins sending you or your C&C communication information. Of note, IRC Botnets have fallen out of fashion as nobody *typically* communicates using IRC clients anymore. They use instant messaging or other identifiable messaging systems. IRC Packets are so out of fashion that they themselves raise red flags as internet traffic; Sometimes to the point of some firewalls blocking them automatically.

"…but the bot used for the attack was developed by the authors and therefore it may not represent a real botnet behavior." (Section 2.2 page 2)

Here is an example of the type of data that is readily available to a malware analysis experiment. A dataset available with authorization. However, it was created using a privately engineered botnet attack rather than a real-world attack or even a pseudo-simulation using an attack that is already seen in the public net. If an experiment was to be done on this, its results would not prove beneficial to botnet comparison or evaluation.

"They were published as CSV text files, where each line is a one-minute aggregation of the number of attempted connections of one IP address. Unfortunately, the aggregation method may not be suitable for comparisons with other proposals." (Section 2.2 page 3)

– Why would the aggregation method matter here? Well the aggregation method of smashing netflows into one-minute clips per IP removes lots of information that is useful. And if you wanted a good comparison using these datasets you would need to change how your malware analysis model reads information to work with this data. Later you can see how this sort of publication differs from the authors dataset. The authors dataset allows for much greater use with their Netflows, while it'd be even better with the pcaps those were withheld for privacy concerns (they let their machines become infected and then did *not* filter their traffic to the outside web, allowing for private IP's and internet conversations to be captured, by withholding the pcaps they shelter a lot of private information)

"Unfortunately, there is only one infected machine for each type of botnet…" (Section 2.2 Page 3)

– This sort of layout of a testing dataset, having only one infected machine per botnet, can be entirely unrepresentative of how a real botnet would function. And it could produce skewed results when used with botnet detection methods unalike the one the dataset was intended for. A good example of a botnet that could function using many more than one infected device would be the Mirai botnet which functioned primarily using Internet of things (IOT) enabled devices. Your household may only have a few personal computing devices, but how many IOT devices do you own? Mirai is just one example of a botnet structure that could easily surpass one infected device per network making evaluation metrics derived from a testing dataset of one machine per botnet less than adequate.

Section 3 The CAMNEP detection method

"CAMNEP processes NetFlow data provided by routers or other network equipment to identify anomalous traffic by means of several collaborative anomaly detection algorithms." (Section 3, page 3)

What is important here is that CAMNEP takes in *other* anomaly labelers – anomaly labelers are not normal classification models, they are unstructured learning models that identify what is "normal" and labels other datapoints that are distant from "normal" as anomalous – and *blends them* to create a new algorithm. This is what is known as a ***blender model*** or ***model ensembling***. If you wish to learn more about model blending feel free but understanding more than this is unnecessary to understand the main contributions of the paper.

"This unpredictability, together with the additional robustness achieved by the use of multiple algorithms, makes the evasion attempt a much more difficult task than just avoiding a single intrusion detection method." (Section 3, page 7)

CAMNEP's model blending allows for extremely unpredictable classification processes which in turn make for good evasion deterrents. To learn more about CAMNEP, its blending processes, and agent-based modeling, you can read the original paper here https://ieeexplore.ieee.org/document/4983378

Section 4 the BCLUS detection method

"The BClus method is a behavioral-based botnet detection approach. It creates models of known botnet behavior and uses them to detect similar traffic on the network. It is not an anomaly detection method."(Section 4 page 7)

"Anomaly detection" is unsupervised learning. BCLUS is not that, BCLUS clusters traffic (NetFlows) from each individual IP and then classifies each cluster as malicious or not. This results in labels existing for each individual netflow as well.

Section 4.1 Separate the netflows into time windows

"The main reason to separate the NetFlows in time windows is the huge amount of data that had to be processed. Some of our botnet scenarios produced up to 395,000 packets per minute."

Breaking netflows into time windows allowed the huge amount of data collected to be processed, with some scenarios consisting of 395,000 packets per minute, that data had to be shrunken down to some margin. Having time windows also allows for time to be a measurable attribute now, which is extremely important in botnet detection. A network security monitor will want to know when they were targeted or infected and having time separation in your detection method helps that.

Section 4.2 Aggregate the netflows by source IP address

"Inside each time window, the NetFlows are aggregated during one aggregation window. The width of the aggregation window should be less than the width of the time window, which was of two minutes." (section 4.2, page 107)

This is where the Netflows are finally aggregated by source IP and creates the IP feature vector. And that each unique IP can have multiple aggregations, one source IP can have aggregated instances at different points in time.

Section 4.3 cluster the aggregated netflows

"After generating the clusters, the task of the BClus method is to find which of them belong to botnets." (section 4.3 page 8)

Each aggregation from the previous section, one unique source IP with an amount of Netflows determined by the time and aggregation windows, is now clustered as the BClus method attempts to differentiate between benign and botnet traffic. Here a feature set is created for each of these clusters which is then transitioned into the next section which classifies each of these clusters.

Section 6 Creation of the dataset

"In order to compare the methods, a good dataset is needed." (Page 10)

This is one of their main contributions, and if all else fails it's a good one. As explained in previous sections there is a complete lack of good, reliable, "made-for-evaluation" data in the community. The dataset was created with analysis in mind. Including ground-truth labels assigned form the get-go, multiple scenarios, and attacks that communicate outside of the network (non-simulated).

Section 6.2.1

"The distribution of labels on each experiment is shown in Table 4." (Page 12)

Table 4 – Distribution of labels for each scenario in the dataset. This table is good to internalize for the upcoming portion. Notice how certain scenarios maintain an extreme bias towards background traffic but a few manage to have substantial botnet traffic. This will be important to remember in model evaluation.

Section 6.2

"First, the argus tool was used to convert each pcap file into a bidirectional Argus binary storage file. The exact configuration of argus is published with each scenario. Second, the ra Argus client tool was used to convert each Argus binary storage file into a NetFlow file."

The NetFlow file format is the cornerstone of a lot of the paper's analysis. To reiterate, NetFlows were used because of their minimal overhead in comparison to raw pcap's while still containing usable features. For each botnet scenario an entire copy of the networks traffic was made using misc. pcap method; Then it is transformed using argus software. A network administrator may find this a great skill to know. Knowing how to go from pcaps to NetFlows would allow use of the same algorithms here in this paper.

Section 7 comparison methodology

Section 7.1 Comparison Methodology

The language here isn't too bad. They want people to constantly work from the same dataset and append their methods results to the dataset to allow easy comparison *instead* of researchers trying to run *other* researchers' methods against the dataset. This way people figure out their own problems before contributing.

"The main advantage of this approach is that the details of the methods remain private" (Section 7.1 page 13) – Possibly people selling botnet detection methods They do not specify their dataset by name, so this comment can still apply.

"To implement this methodology we created and published a new tool called *Botnet detectors Comparer"* (Section 7.1 page 13)

This tool provides a final evaluation of your method against the dataset they created and creates a confusion matrix using the evaluation. It also computes the new error metrics they discuss in the next section, where they modify the classical ML model evaluation metrics (TP, FP, etc) to incorporate time. An important modification of evaluation when response time is essential to security.

Section 7.2 New Error Metric

"The second step was to migrate from a NetFlow-based detection to an IP-based detection. The classical error values (TP, FP, TN, FN) were redefined as follows:"

Here their new evaluation metric assigns evaluation-labels at the IP-address level…

Further Reading

To better understand the academic landscape of malware analysis and machine learning experimentation I recommend continuing with a few papers Garcia et al. (2014) cited.

- https://www.usenix.org/legacy/event/cset11/tech/final_files/Aviv.pdf
- https://ieeexplore.ieee.org/document/5464348
- https://oaklandsok.github.io/papers/rossow2012.pdf