Botnet Behavior Detection using Network Synchronism

Sebastián García, Alejandro Zunino, Marcelo Campo
ISISTAN Research Institute, Universidad Nacional del Centro. Also CONICET.
Campus Universitario, Tandil (B7001BBO), Buenos Aires, Argentina.
Tel.: +54 (2293) 439682. Fax.: +54 (2293) 439681
{eldraco,azunino}@gmail.com

Abstract. Botnets diversity and dynamism challenge detection and classification algorithms, which depend heavily on botnets protocol and can quickly become avoidable. A more general detection method, then, was needed. We propose an analysis of their most inherent characteristics, like synchronism and network load combined with a detailed analysis of error rates. Not relying in any specific botnet technology or protocol, our classification approach sought to detect synchronic behavioral patterns in network traffic flows and clustered them based on botnets characteristics. Different botnet and normal captures were taken and a time slice approach was used to successfully separate them. Results show that botnets and normal computers traffic can be accurately detected by our approach and thus enhance detection effectiveness.

Keywords: Botnet, detection, clustering, EM algorithm, security

1 Introduction

In the last decade botnets have evolved from being used as a personal activity platform to become a financially aimed structure controlled by malicious groups [1]. A botnet is a network of remotely controlled, compromised computers, used for malicious purposes. Every infected host in a botnet is called 'Bot' and the owner is called 'Botmaster'. From small DDoS (Distributed Denial of Service attacks) to world wide spam campaigns, botnets have become the technological backbone of a growing community of malicious activities. Technology to control malicious programs remotely first surfaced in late 1999 and since then their primary goal has been financial gain. Botnets resisted besiege security measures resting on their home based client attacks, circumventing security methods [2], encryption and anti-reverse engineering techniques. P2P [3] and Fast-flux [4] networks have also made botnets behavior analysis a challenging task.

Infected computers are believed to have a unique network behavior, but they also have a normal behavior at the same time. Being cleaned up and infected again later with a different type of malware makes the detection and isolation process very hard to accomplish in real environments. False positives are still the hardest problem. Botnet network behavior analysis has been proposed [6] as a probabilistic solution.

We aim at detecting botnet synchronization within different stages of their life cycle. Botnet phases such as scanning for new victims, connecting to the Command & Control server (from now on C&C), receiving orders, attacking with DDoS, sending

spam, updating, etc. have distinguishable and different synchronization patterns. C&C servers are the weakest link in botnets life cycle, as preventing bots from being controlled, negates Botmasters their utility. We propose a novel unsupervised anomaly detection approach, based on network behavioral synchronization patterns to detect botnets despite of their connection protocols.
To detect every type of botnet, we believed it is necessary to detect their most inherent characteristics. As analyzing connections within a time frame has been successfully proposed as a method to detect synchronism [7], we extended this idea of time frames to detect other types of synchronization as well. Synchronization patterns are the main tool to detect botnets in our work.
First we separated traffic flows into time slices of one second. Then, the amount of unique source IP addresses, destination IP addresses and destination ports in each slice were computed. Analyzing botnet synchronization from this point of view allowed us to detect relationships between time intervals and botnet activity. We proposed clustering algorithms [8] to group botnet time slices together.
This paper introduces a new botnet characteristic, the time slices separation of network flows with conglomeration of source IP address, destination IP address and destination ports.
Some advantages of this method include detection independent of encrypted flows or botnet protocol details, detection of both bandwidth intensive and stealthy botnets and detection within the first stages of infection. The preliminary experimental results reported, suggest that this detection can be accomplished successfully under certain conditions.
The rest of the paper is organized as follows: Section 2 describes previous work in the area; Section 3 shows details about the solution implemented; Section 4 explains corroboration procedures; Subsection 4.1 shows how network data has been obtained; in subsection 4.2 the data prepare steps are explained; Section 5 discusses experiments outcomes; Section 6 refers to future work we are planning and finally Conclusions are presented in Section 7.

2 Background

Several approaches have been proposed in recent years to detect botnets. Study of anomalous DNS usage [9], IRC protocol analysis [10] and P2P network features [5] among others [11] [12], but the variability of botnets and their rapid mutation have made researchers from different security domains consider the analysis of botnet behavior as a solid foundation for detection.
Particularly, network behavior detection has been studied from different perspectives: classifying traffic based on flow characteristics [13], temporal-frequent protocol analysis [14] and spatial-temporal correlation [6].
Behavior analysis based on protocol dependent features has proved to be successful only under certain conditions. There are three significant problems with this approach. First, most botnets cannot be detected by analyzing protocol dependent characteristics. Secondly, new unseen botnets are unlikely to be detected if a new protocol is

used. Finally, Botmasters can change their current malware code based on the protocol characteristics used to detect their botnet [15].
Botnet correlation was proposed [16] among other techniques [17], as all bots in a botnet normally act at the same moment. However, synchronization has not been deeply studied, and temporal correlation approaches were found to be evadable [18] under certain circumstances.
Our approach to temporal synchronization differentiates each stage of a botnet life cycle, determining where, when and how correlations are significant.

3 Proposed technique

The first step of our methodology consists in capturing network data from infected computers. This could be done properly using the tcpdump tool in any machine in the wired network. The second step includes information extraction from captures and flow separation. Tcptrace [19] tool was used to find out every TCP flow including its start time and network characteristics.
As we are seeking to detect synchronization, in the third step of our approach flows are divided in time slices. Each time slice contains information about every flow within a time interval. We compute then for every time slice, the amount of unique source IP addresses, unique destination IP addresses and unique destination ports. A time slice can then be represented by a four position array containing the slice Id, amount of unique source IP addresses seen in that time slice, amount of unique destination addresses seen and amount of unique destination ports seen. An example should look like this: {23, 1, 10, 1}.
A unique bot computer is expected to have very high flows rates within very short time periods during several of botnet phases [6]. Working with a one second time slice allows us to correctly monitor and identify bot behavior. The fourth step is the clustering of these slices using the EM algorithm [20]. Expectation-maximization is a method for finding maximum likelihood estimates of parameters in statistical models, and has been used successfully before in traffic flow analysis for characterizing communication connectivity patterns [21]. Wekas [22] implementation of this algorithm was used because of the independence assumption of the attributes in the model, making it suitable for our purposes.
Grouping network flows with EM enables us to assign each slice to a cluster with a certain probability because no amount of training data is sufficient to make a complete firm decision about cluster memberships.

4 Validation

Validation procedures included ensuring a clean experimental setting. Every computer was freshly installed, scanned with antivirus products and its traffic analyzed both with Snort NIDS and by hand. From this clean configuration, Virtual Machines were created to deal with normal computers in the experiments. Normal computers packet verification assured no scanning activities were conducted during the experiments.

This clean configuration was also used to create Virtual Machines for computers that were later infected. Confirmation of malware binaries using VirusTotal [23] and the EUREKA! automated Malware Binary Analysis Service [24] was conducted in conjunction with packet manual verification. The second verification step ensured that no flow was lost in the process by counting them within every result.
Validation in the third step was conducted through slice counting. For example, in a one hour capture, almost 3600 one second slices must be created.
The clustering step was verified by labeling data. Trustworthy botnet and normal experiments were performed, building an accurate cluster compare base. Once available, this information allowed interpretation of the resulting clusters and experiments improvement. Table 2 shows an example of this validation. Finally, an experimental validation was conducted over the assumptions about the proposed botnet behavior detection methodology. One of the main theoretical assumptions was that through the analysis of three parameters, detection of botnets and bots was possible. These three parameters are the amount of unique source IP addresses within a time slice (from now on sips), the amount of unique destination IP addresses within a time slice (from now on dips) and the amount of unique destination ports within a time slice (from now on dports). For example, founding 1 sips, 20 dips and 1 dips within a one second time frame could mean we have detected a port scanning activity, possibly looking for a vulnerability.
In the following subsections details about how data was accurately captured and processed are described.

4.1 Data Capture

The proposed algorithm was verified against a suite of labeled data flows. We considered a unique data flow as the group of packets exchanged between two hosts and two ports in a single connection. Raw pcap data of both normal network activity and botnet activity was captured. Normal traffic from University campus computers and DSL home connections included Web Browsing, Mail sending, Web sites with Ajax updates, edonkey like protocols, torrent protocols, operating system updates, and normal Web work using several Google tools. Almost 5037 unique normal network flows were collected. A total of 315.672 unique botnet flows were collected. A port scanning capture with 73830 flows was also achieved in order to analyze security tools behavior. Table 1 shows details about captures.

Table 1. Labeled network data captures details.

Name	Duration	Unique Flows
Botnet1	11h:12m:29s	37389
Botnet2	00h:30m:30s	5806
Botnet3	10h:11m:33s	34117
Botnet4	00h:01m:31s	23166
Botnet5	01h:00m:30s	89792
Botnet6	00h:46m:45s	58013
Botnet7	00h:18m:15s	46212
Botnet8	00h:20m:36s	21177

Normal1	00h:42m:20s	1416
Normal2	03h:28m:14s	1517
Normal3	04h:19m:36s	976
Normal4	23hs:37m:21s	1128
Scanner1	00hs:05m:27s	73830

Note: The Botnet1 capture corresponds to the 'Virut' malware in a single laboratory host, the Botnet2 capture corresponds to the 'Neeris' malware in a single laboratory host, the Botnet3 capture corresponds to 'eldorado' malware in a single laboratory host, captures Botnet4 to Botnet7 corresponds to a novel webbotnet infecting 10 hosts on a laboratory LAN. Botnet8 capture corresponds to a real LAN with four hosts infected. The Normal1 capture corresponds to several computers inside a University campus network, the Normal2 capture is a single computer inside a University campus network, the Normal3 capture is another single computer inside a University campus network and the Normal4 capture corresponds to an amule session in a home DSL computer. Finally Scanner1 capture corresponds to a complete nmap scanning of a LAN.

4.2 Data processing

Data processing was a fundamental part of the analysis as it determined which characteristics of data were taken into account. Processing steps include data verification, data labeling, data preprocessing and feature extraction among others.
From pcap captured data, the first phase of data processing consisted in flow extraction using the tcptrace tool.
The second phase was performed with a specially developed tool, called *tcptrace-reader.py* to process the tcptrace output file. As a result of this phase, a Weka compliant Arff (Attribute-Relation File Format) file was generated with proper information for every flow.
Phase three performed slice aggregation in a new Arff file using the *tcptrace-reader.py* tool for Weka processing. The fourth phase consisted of data labeling as we already addressed in Section 4. After clustering was done and slices were grouped together, a third Arff file was saved from Weka with clusters assignments.
Fifth phase used another specially developed tool called *analisis.sh* to analyze cluster assignments. This tool allowed us to validate clusters using slices labels.

5 Results and error analysis

Several experiments were conducted to analyze algorithm performance. Every experiment includes one botnet capture and one normal capture combined. This combination was done at tcptrace file level, concatenating both text files and modifying flows start times of one capture.
Error rate analysis was separated in two points of view. From the first, classical point of view, real positives were considered when bots flows were classified as botnet.

Following this reasoning, false positives were considered each time a normal computer flow was detected as botnet. This is the most simple and accepted point of view. In the second, more practical, point of view, error rate analysis was done taking into account the problems domain and scope. A real positive was considered now when a bot know IP address was classified as botnet at least once within a time frame. Following this reasoning, false positives were triggered only when a normal IP address was considered botnet at least once. This is sufficient for most practical detection scenarios. In the one hand, under this new conception, false positives shown in the experiments were truly one day false positives. In the other hand, no botnet IP address was missed in its corresponding one day time frame, meaning that at some point, a network administrator will find out **all** the botnets.
Note that since we are detecting botnet behavior and not normal behavior, we can not detect instances as normal, we can not decide when a cluster is representative of a normal behavior and thus we do not have False Negatives. As we did not define what normal means to us, we should have used the term *not-botnet* instead, but we still used the term normal for the sake of clarity.

We concluded that any cluster with more than 90% of botnets instances on it was in fact a good representative of botnet behavior. In Future research directions section we explain the next phase in this botnet cluster automatic identification.

First experiment, shown in Table 2, shows an almost balanced situation. During the 19 days, 4 hours and 41 minutes time frame, botnet data make up 99% of cluster 0 and 2, and make up more than 95% of cluster 3

Table 2. Detection percentages of 19519 Normal3 flows mixed up with 34042 Botnet3 flows

Cluster Number	Botnet Flows	Normal Flows	Botnet %	Normal %
0	14433	24	99.83%	0.16%
1	1517	18721	7.49%	92.50%
2	3539	14	99.60%	0.39%
3	14629	684	95.53%	4.46%

Table 3. Experiment one false positives detection rates

IP	Times as Real Normal	Times Detected as Botnet	Error %
10.1.1.1	344	180	52.32%
192.168.2.79	19099	542	2.83%

Note that IP address 10.1.1.1 had a very high false positive rate in near every experiment using Normal3 flows. This IP address was the web proxy of the network, and their flows were only RESET packets responses to web requests from normal computers. This behavior looked like a botnet. As a compromise solution, the network administrator can blacklist this IP address knowing it is the web proxy. From now on this IP address will be included in the tables but will not be included in the total error rate calculation.

The second experiment summarized in Table 4 shows an unbalanced analysis. The manual analysis concluded that botnet traffic in this cluster generated only one or two flows per slice directed to Windows shares, and so was easily confused with normal traffic.

Table 4. Detection percentages of 1517 Normal2 flows mixed up with 34118 Botnet3 flows

Cluster Number	Botnet Flows	Normal Flows	Botnet %	Normal %
0	2147	1358	61.25%	38.74%
1	3539	12	99.66%	0.33%
2	14469	81	99.44%	0.55%
3	13963	66	99.52%	0.47%

Table 5. Experiment two false positives detection rates

IP	Times as Real Normal	Times Detected as Botnet	Error %
10.1.1.1	2	1	50.00%
192.168.2.74	869	89	10.24%
192.168.2.76	264	31	11.74%

Experiment three, shown in Table 6, had an even more unbalanced situation.

Table 6. Detection percentages of 977 Normal3 flows mixed up with 34118 Botnet3 flows

Cluster Number	Botnet Flows	Normal Flows	Botnet %	Normal %
0	2147	916	70.09%	29.90%
1	13918	22	99.84%	0.15%
2	3539	14	99.60%	0.39%
3	14469	25	99.82%	0.17%
4	45	0	100%	0%

Table 7. Experiment three false positives detection rates

IP	Times as Real Normal	Times Detected as Botnet	Error %
192.168.2.79	954	60	6.28%

Analysis showed that every misdetection in this experiment was due to the normal trace being part of a large botnet trace in the same second.

An example of normal traffic along several hours with a peak of botnet traffic in the middle was made in experiment four and can be seen in Table 8. During botnet4 capture at least ten computers were infected and all of them used in a DDoS attack. It is worth noting that 192.168.1.9 IP address appeared both in normal and botnet captures and was a coincidence.

Table 8. Detection percentages of 988 Normal4 flows mixed up with 23305 Botnet4 flows

Cluster Number	Botnet Flows	Normal Flows	Botnet %	Normal %
0	0	32	0%	100%
1	0	642	0%	100%
2	0	150	0%	100%
3	1738	0	100%	0%
4	3	164	1.79%	98.20%
6	14261	0	100%	0%
7	2249	0	100%	0%
8	139	0	100%	0%
9	4915	0	100%	0%

Table 9. Experiment five false positives detection rates

IP	Times as Real Normal	Times Detected as Botnet	Error %
192.168.1.6	337	14	4.15%
192.168.1.9	629	123	19.55%

Experiment five, in Table 10, was a concatenation of both types of flows. The web-botnet of this experiment had more sips, dips and dports, and thus classification was better, but still they were not enough to avoid high error rates.

Table 10. Detection percentages of 1417 Normal1 flows mixed up with 46210 Botnet7 flows

Cluster Number	Botnet Flows	Normal Flows	Botnet %	Normal %
0	9282	549	94.41%	5.58%
1	1300	632	67.28%	32.71%
2	17412	0	100%	0%
3	381	222	63.18%	36.81%
4	17835	14	99.92%	0.07%

IP	Times as Real Normal	Times Detected as Botnet	Error %
10.1.1.1	231	115	49.78%
192.168.2.24	31	9	29.03%
192.168.2.56	650	277	42.61%
192.168.2.57	192	66	34.37%
192.168.2.59	60	26	43.44%
192.168.2.63	66	17	25.75%
192.168.2.66	4	2	50.00%
192.168.2.67	51	18	35.29%
192.168.2.69	48	18	37.50%
192.168.2.76	81	14	17.28%

Table 11. Experiment nine false positives detection rates

Experiment six, in Table 11, was the first experiment with a port scanning activity. This experiment aimed at trying to identify the differences between a real port scanners and a botnet. Note however that the reason of successfully separation has changed. The port scanner generated more flows per second that the botnet.

Table 11. Detection percentages of 73830 Scanner1 flows mixed up with 7060 Botnet8 flows

Cluster Number	Botnet Flows	Normal Flows	Botnet %	Normal %
0	2213	241	90.17%	9.82%
1	4847	1147	80.86%	19.13%
2	0	72442	0%	100%

Table 12. Experiment ten false positives detection rates

IP	Times as Real Normal	Times Detected as Botnet	Error %
192.168.2.76	73448	241	0 .32%

Experiment seven, in Table 13, was an analysis of two different types of flows. Normal traces had an average of almost 5 sips and dips, while botnet traces had an average of 10 sips and dips.

Table 13. Detection percentages of 1417 Normal1 flows mixed up with 32962 Botnet1 flows

Cluster Number	Botnet Flows	Normal Flows	Botnet %	Normal %
0	1159	536	68.37%	31.62%
1	13875	0	100%	0%
2	1389	881	61.18%	38.81%
3	13939	0	100%	0%
4	2600	0	100%	0%

Table 14. Experiment eleven false positives detection rates

IP	Times as Real Normal	Times Detected as Botnet	Error %
192.168.2.76	81	1	1.23%
192.168.2.69	48	1	2.08%
192.168.2.63	66	1	1.51%
192.168.2.59	60	1	1.66%
192.168.2.24	31	1	3.22%

6 Future research directions

Several improvements are planned for future research. First, UDP traffic must be taken into account because botnets uses this protocol.
We will analyze the total amount of transferred bytes within a single flow, in order to capture long lived botnet connections with their C&C server.

In the validation area, it would be very helpful to analyze if botnet and normal flows that started at the same second were successfully separated.
To overcome the DDoS problem, we are going to modify capture method in order to include enough data to detect these attacks.
As a final phase, we are working on analyzing cluster assignments to synthesize classification rules that later allows an automatic cluster classification.

7 Conclusion

This work applies the EM clustering algorithm to detect synchronization in bots and botnets behavior. This synchronization was studied as the relationship of IP addresses, ports and time frames only. Different combinations of these parameters mean different network behaviors, and a new approach is used to distinguish them. Behavioral techniques are used because current fingerprint matching algorithms or protocol dependent algorithms were not having good performances, and could not cope with every new variant of botnets.
Several botnets were captured and many experiments were conducted to successfully verify the method. Results show that, within certain life cycle phases and time frame, botnets can be differentiated from normal traffic accurately. Further study in the area is needed to find better false positive rates and an automatic cluster classification approach.

Acknowledgments

We acknowledge the financial support provided by ANPCyT through grants PAE-PICT 2007-02311 and PAE-PICT 2007-02312.

8 References

1. Wilson, C. (2007). Botnets, Cybercrime, and Cyberterrorism: Vulnerabilities and Policy Issues for Congress. Technical report, LIBRARY OF CONGRESS WASHINGTON DC CONGRESSIONAL RESEARCH SERVICE.
2. Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., Kemmerer, R., Kruegel, C., & Vigna, G. (2009). Your botnet is my botnet: analysis of a botnet takeover. In CCS '09: Proceedings of the 16th ACM conference on Computer and communications security, (pp. 635-647), New York, NY, USA. ACM.
3. Ha, D., Yan, G., Eidenbenz, S., & Ngo, H. (2009). On the effectiveness of structural detection and defense against p2p-based botnets. In Dependable Systems & Networks, 2009. DSN '09. IEEE/IFIP International Conference on, (pp. 297-306).
4. SSAC. (2008). Ssac advisory on fast flux hosting and dns. Technical report. ICANN Security and Stability Advisory Committee. From the ICANN Web site at: http://www.icann.org/en/committees/security/ssac-documents.htm.

5. Kang, J., Zhang, J.-Y., Li, Q., & Li, Z. (2009). Detecting new p2p botnet with multi-chart cusum. In Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC ’09. International Conference on, volume 1, (pp. 688-691).
6. Gu, G. (2008). Correlation-based Botnet Detection in Enterprise Networks. Published doctoral dissertation, Georgia Institute of Technology.
7. Gu, G., Zhang, J., & Lee, W. (2008). BotSniffer: Detecting Botnet Command and Control Channels in Network traffic. In Proc. 15th Network and Distributed System Security Symposium (NDSS), San Diego, CA.
8. Baeza-Yates, Ricardo A., & Riberiro-Neto, Berthier A. (1999). Modern Information Retrieval. ACM-Press / Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.
9. Villamarin-Salomon, R. & Brustoloni, J. (2008). Identifying botnets using anomaly detection techniques applied to dns traffic. In Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE, (pp. 476–481).
10. Mazzariello, C. (2008). Irc traffic analysis for botnet detection. Information Assurance and Security, 2008. ISIAS ’08. Fourth International Conference on, (pp. 318-323).
11. Shahrestani, A., Ramadass, S., & Feily, M. (2009). A survey of botnet and botnet detection. In Emerging Security Information, Systems and Technologies, 2009. SECURWARE ’09. Third International Conference on, (pp. 268-273).
12. Zhu, Z., Lu, G., Chen, Y., Fu, Z., Roberts, P., & Han, K. (2008). Botnet Research Survey. Computer Software and Applications, 2008. COMPSAC’08. 32nd Annual IEEE International, (pp. 967-972).
13. Strayer, W., Lapsely, D.,Walsh, R., & Livadas, C. (2007). Botnet detection based on network behavior. Advances in Information Security, Springer US, editor, Botnet Detection, volume 36, (pp. 1-24).
14. Lu, W., Tavallaee, M., Rammidi, G., & Ghorbani, A. A. (2009). Botcop: An online botnet traffic classifier. In Communication Networks and Services Research Conference, 2009. CNSR ’09. Seventh Annual, (pp. 70-77).
15. Stinson, E. & Mitchell, J. (2008). Towards Systematic Evaluation of the Evadability of Bot/Botnet Detection Methods. WOOT'08: Proceedings of the 2nd conference on USENIX Workshop on offensive technologies, USENIX Association, (pp. 1-9).
16. Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection. SS'08: Proceedings of the 17th conference on Security symposium. USENIX Association, (pp. 139-154).
17. Liu, L., Chen, S., Yan, G., & Zhang, Z. (2008). Execution-based bot-like malware detection. Lecture Notes in Computer Science, Information Security, volume 5222/2008, Springer Berlin / Heidelberg, (pp. 97-113).
18. Chen, Z., Chen, C., & Wang, Q. (2009). Delay-tolerant botnets. In Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on, (pp. 1-6).
19. Shawn, O. (2010), Retrieved March 30, 2010 from the Tcptrace Web site: http://www.tcptrace.org/.
20. Dempster, A.P., Laird, N.M., & Rubin, D.B. (1977). Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society. Series B (Methodological), 39(1) (pp. 1-38).
21. Chen, A., Li, L., & Cao, J. (2009). Tracking cardinality distributions in network traffic. INFOCOM 2009, IEEE, volume , (pp. 819-827).
22. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, H.I., (2009). The weka data mining software: An update. Paper presented at SIGKDD Explor. Newsl., 11(1) (pp. 10-18).
23. Hispasec Sistemas. (2010) Retrieved March 30, 2010, from the Virus Total Web site: http://www.virustotal.com/es/.

24. Yegneswaran, V., Saidi, H., Porras, P., Sharif, M., Mark, W., & President, V. (2008). Eureka: A Framework for Enabling Static Analysis on Malware. Technical report, Georgia Institute of Technology.