Early Detection of Botnet Activities using Grammatical Evolution

Selim Yilmaz[0000−0002−9516−6892] and Sevil Sen[0000−0001−5814−9973]

WISE Lab., Department of Computer Engineering, Hacettepe University, Ankara, Turkey,
selimy@cs.hacettepe.edu.tr
ssen@cs.hacettepe.edu.tr

Abstract. There have been numerous studies proposed for detecting botnets in the literature. However, it is still a challenging issue as most of the proposed systems are unable to detect botnets in their early stage and they cannot perform satisfying performance on new forms of botnets. In this study, we propose an evolutionary computation-based approach that relies on grammatical evolution to generate a botnet detection algorithm automatically. The performance of the proposed flow-based detection system reveals that it detects botnets accurately in their very early stage and performs better than most of the existing methods.

Keywords: Botnet · flow-based detection · evolutionary computation · grammatical evolution

1 Introduction

Botnet is a number of compromised devices called as *bots or zombies* which are controlled through a special Command and Control (C&C) channel by an intruder node known as *botmaster*. By taking advantage of computational resources of bots, botnets are used for performing several distributed illegal activities such as spamming, distributed denial-of-service attacks. The increasing size of botnets have now reached to an unprecedented level such that more than 80% of the Internet traffic is propagating through botnets [1]. Distributing such bots are cheap, however they could have drastic effects on the economy. It is reported that malware distribution has caused a damage of 13.2 billion to 67.2 billion USD to the global market within only two years in the past [1]. Hence, developing robust and effective detection systems towards the different forms of botnets has become a must.

In this study, we proposed an evolutionary computation-based botnet detection system for early detection of botnets. Grammatical Evolution (GE) is employed due to its capability in generating interpretable computer programs for security experts. As inputs to GE, the TCP/UDP packet flows have been preferred to the packet payloads due to allowing us to inspect encrypted or obfuscated network traffic. Since most of the network traffic today is encrypted, monitoring information in the packet headers for a possible suspicious activity has become the only way. Furthermore, since the packet headers correspond to only a small fraction of the whole network traffic, it reduces the computational overhead. In addition, working on the smaller time windows rather than all individual flows enables our system to detect bot activity before the attack phase. To

the best of the authors' knowledge, this is the first study that explores the use of grammatical evolution for detecting botnets at the early stage and that uses the most recent dataset in the literature for identifying new botnet types.

The rest of the paper is organized as follows: Section 2 summarizes the related approaches in the literature. Section 3 explains grammatical evolution algorithm in detail. The proposed framework is outlined in Section 4. The experimental setup and results are provided in Section 5. Section 6 concludes the paper.

2 Related Work

Machine learning-based techniques have already been proposed to detect different types of botnets (i.e., IRC, HTTP, and P2P) within the recent years. However, most of these approaches employ deep packet inspection, which cannot analyze encrypted traffic. Given that the majority of today's network traffic is encrypted, researchers have moved their focus on developing flow-based detection systems as complementary to such systems recently. In this section, such flow-based detection approaches are reviewed.

Huseynov et al. [2] proposed a method based on the similarity of communication patterns between botmasters and bots. It relies on a semi supervised method where only 10% of all traffic was labeled. Firstly, similar activities are clustered by using ant colony optimization and then, clusters are identified based on their similarities with the labeled data. Narang et al. [3] applied different feature selection methods for identification of botnet activities. The authors concluded that built time considerably increases when the full feature set is applied.

Kirubavathi et al. [4] proposed a method for detecting HTTP-based botnets. Therefore, the inputs to the proposed neural network consist of only TCP related features. They extended their study in [5] to study other types of communication schemes (IRC & P2P) and to investigate other classification methods (i.e., J48, Naïve Bayes, SVM). In another study based on neural networks, Nogueira et al. [6] proposed a framework called botnet security system (BoNeSSy) where historical profiling provided by each application were used to train the network.

A behavioral-based P2P botnet identification system was proposed by Saad et al [7]. They regarded network traffic as data stream where bots tend to behave differently over time. The authors also studied the ability of five different learning algorithms for online botnet detection. Although the performance of these algorithms was promising, they were not enough to satisfy all the requirements of an online detection system. Another behavioral-based botnet detection method by Wang et al. [8] identifies malicious domain names and IP addresses by using fuzzy pattern recognition technique.

Livadas et al. [9] proposed an IRC-based botnet detection system of two stages. The first stage filters out non-chat flows. In the second stage, real chat flows are distinguished from botnet flows using J48, Naïve Bayes, and Bayesian network classifiers. Fedynyshyn et al. [10] proposed a host-based approach to detect botnet traffic by also employing random forest and J48 classification algorithms.

3 Grammatical Evolution

Grammatical evolution is an evolutionary computation technique inspired by the biological process of generating a protein from the genetic material of an organism. It first generates a certain number of individuals which represent candidate solutions for the targeted problem and then, examines the 'fitness' of each individual and finally creates new individuals by applying genetic operators (such as crossover, mutation, and, etc.). These are the steps of a single evolution step. The best solution is yielded by GE at the end of the evolution process.

GE is capable of generating a solution (program) in any language. The grammar of this language is expressed in a notation named Backus-Naur Form (BNF). BNF grammars consists of `terminals`, which are the end-items in the language and `nonterminals`, which can be expanded by the terminals or nonterminals. A grammar is represented by a four-tuple {`T,N,P,S`}. `T` represents the terminal set, `N` represents the nonterminal set, `P` is production rule set comprising of a number of grammar rules, and `S` is a start symbol indicating the entry-point of the grammar. To generate a program, GE makes use of a 8-bit binary string genomes (called *codons*) that are assigned to every individual and mapped to a sequence of integer values. The mapping process in GE results in a higher genetic diversity in the population than the other evolutionary-based approaches, which is the main advantage of GE [11].

To elaborate the process of automatic program generation, one-step evolution of GE on a symbolic regression problem is explained. Let the problem to be examined be $f(X) = X^4 + X^3 + X^2 + X$, and the BNF notation for the grammar be:

```
(1)        <expr>::= <expr><op><expr>      (0)
                   | (<expr><op><expr>)    (1)
                   | <pre-op>(<expr>)      (2)
                   | (<var>)               (3)
(2)          <op>::= +                     (0)
                   | -                     (1)
                   | /                     (2)
                   | *                     (3)
(3)      <pre-op>::= Sin                   (0)
                   | Cos                   (1)
                   | Exp                   (2)
                   | Log                   (3)
(4)         <var>::= X                     (0)
                   | 1.0                   (1)
```

The symbols enclosed by brackets (<>) are the nonterminals and others are the terminals, starting symbol (`S`) is `<expr>`. Suppose the genome of an individual is:

220	35	84	42	251	15	47	66

The first codon value (220 in this example) is used to expand the first nonterminal item in the language, which is initially indicated by `S`. Every encountered nonterminal in the BNF is replaced with the following rule:

$$rule = (codon\ integer\ value)\ MOD\ (\#\ of\ rules\ of\ nonterminal)$$

The first nonterminal here is <expr> and the codon value is 220 which selects rule 0 (220 *MOD* 4 = 0) and <expr> is replaced by <expr><op><expr>. The left-most nonterminal item is still <expr> but the codon value is now 35 which results in a selection rule of 3 (35 *MOD* 4 = 3). Now the language has become (<var>)<op><expr>. This process continues until all nonterminal items are replaced to terminals. In the case where the last codon value is reached but there still remains nonterminals in the language, the codon pointer switches back to the first codon value that is 220 –this process is called *wrapping*. The final language generated according to the genome and the grammar is '(X) / (1.0)'. Please refer to [12] for the detailed description of the algorithm.

4 The Proposed Method

The proposed approach comprises of three consecutive phases. The first phase involves obtaining traffic flows in different time windows from a real-world dataset (in *.pcap* format) in order to determine the optimal window size for obtaining high detection accuracy in botnet detection. Moreover, a number of distinctive features of network flows are extracted for botnet detection. In the second phase, grammatical evolution is employed to evolve a detecting algorithm automatically by using these features. The final phase evaluates the performance of all evolved detection algorithms. The general framework of our approach has been demonstrated in Figure 1. Each phase is explained in detail in the following two subsections.

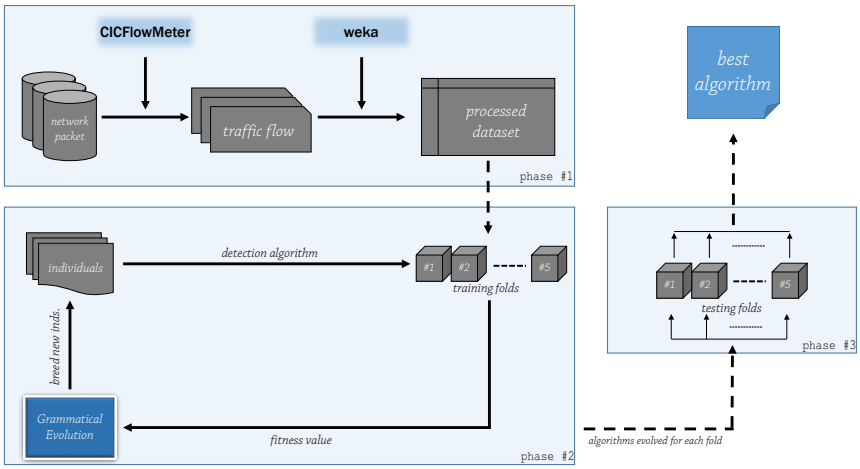

Fig. 1. General framework of the proposed botnet detection system

4.1 Flow generation and feature extraction

A flow is a collection of packets having the same five-tuple that is source IP, destination IP, source port, destination port, and protocol. These features can be extracted directly

from the TCP/UDP headers. However an extra computation is required to extract additional statistical features from these packets such as flow duration. CICFlowMeter [13], a Java-based bidirectional flow generator and analyzer tool integrating jNetPcap [14] library, has been used to generate traffic flows and to extract all traffic-related features from these flows. This tool supports 76 traffic-related statistical features calculated separately for the forward and backward directions. The identifying attributes like 'flow ID' have been discarded from the feature set as they might not be good indicators especially when the training and testing data come from different network domains. The tool terminates a flow depending on the flow timeout or the FIN flag. In this study, the flow timeout value is set to 30 mins. It labels a flow *bot* or *benign* considering the IPv4 addresses of the sender or the receiver. The more information about the feature set can be found in [13]. In addition to these features, in this study, bidirectional initial packet sizes are also extracted from the flow as they are known to be very effective in identifying botnet activity.

Individual flows are split into different time windows. In order to obtain an appropriate window-length, we have conducted a pre-experiment, in which every individual flow has been windowed from 120 s to 360 s in the multiples of 60 s. The results showing the effect of the window's size on botnet detection are discussed in Section 5.2. In the training, the optimal window size of this pre-experiment is used. Moreover, as the feature values are in a high scale, all features' values are mapped to a range from 0 to 1.

4.2 Evolution of botnet detection algorithm

In this study, GE has been explored for the evolution of an algorithm which detects botnets by inspecting the most discriminative features of the flow. Every individual in GE represents a candidate detection algorithm and is evaluated depending on its detection ability which has been measured by the following 'fitness' equation:

$$fitness = TP - (\omega \times FP) \tag{1}$$

where TP and FP are the true positive and false positive, respectively, and ω (=3, empirically found) is a constant factor which increases the magnitude of FP. The idea behind why ω constant has been introduced into the equation is to avoid over-stimulation to the botnet patterns as the dataset is highly imbalanced (with extracted 2,390,624 benign and 79,428 botnet flows). We have employed 5 fold cross-validation and thus the whole dataset is divided into 5 folds such that 4 of them have been used for training and the remainder has been used for the testing of the evolved algorithms. Due to the stochastic nature of evolutionary-based algorithms, we have run GE 10 times per a fold (50 runs in total). Java-based evolutionary computation toolkit (ECJ) [15] has been used for the GE implementation. All the GE parameters used during the training have been provided in Table 1. The parameters not listed here are the default parameters of ECJ.

The BNF grammar for the problem is provided in Figure 2. Traffic-related features generated through CICFlowMeter and random values (`rnd`) generated from a uniform distribution between 0 and 1 are employed as operands. A number of arithmetic and relational operations as well as mathematical functions are defined as operators. The

Table 1. GE Parameters

Parameters	Value
Functions	+, -, *, /, sin, cos, log, ln, sqrt, abs, exp, ceil, floor, max, min, pow, mod, <, ≤, >, ≥ ,== ,! =, and, or
Terminals	rnd(0,1), features in [13]
Population Size	50 (number of elite individuals = 5)
Crossover probability	0.9
Mutation probability	0.1
Selection strategy	Tournament selection (Tournament size: 7)
Generations	1000

detection performance of all detection algorithms has been evaluated on the folds spared for testing, then the best algorithm has been determined, which corresponds to the final phase of our methodology. The details regarding this phase is outlined in Section 5.2.

```
(1)   <algorithm>  ::= if(<cond>) { alert(); }
(2)   <cond>       ::= <cond><set-op><cond> | <expr><rel-op><expr>
(3)   <expr>       ::= <expr><op><expr> | (<expr><op><expr>)
                     | <pre-op>(<expr>) | <var>
(4)   <op>         ::= + | - | / | *
(5)   <pre-op>     ::= sin | cos | log | ln | sqrt | abs | exp | ceil
                     | floor | max | min | pow | modulus
(6)   <rel-op>     ::= < | ≤ | > | ≥ | == | !=
(7)   <set-op>     ::= and | or
(8)   <var>        ::= rnd | feature set in [13].
```

Fig. 2. BNF grammar of the botnet detection problem.

5 Experiments

5.1 Datasets

We have incorporated two datasets. The first dataset is the well-known Information Security and Object Technology (ISOT) dataset. It contains traffics belonging to the P2P Storm and Waledac botnets captured from 2007 to 2009. Benign traffics, on the other hand, were recorded from the everyday traffics such as HTTP web browsing, gaming packets, torrent packets like Azureus. The second dataset is CICIDS2017. It was obtained from the traffic generated between July 3 to July 7 in 2017, where different attacks were implemented for each day. We have used only botnet (Ares) and benign packets of this dataset captured on July 7. Please refer to [7] and [16] for the detailed description of the ISOT and CICIDS2017 datasets, respectively.

5.2 Results

As discussed earlier, we have conducted a pre-experiment in order to investigate the effect of the window size on botnet detection. GE has been run 5 times under the same settings given in Table 1 but with 500 generations. The results show that the algorithm detects bot activity with a higher performance as the window size increases up to 240 s (see Fig. 3). Hence, the optimal window size is set to be 240 s for the evolution of botnet detection algorithm in this study.

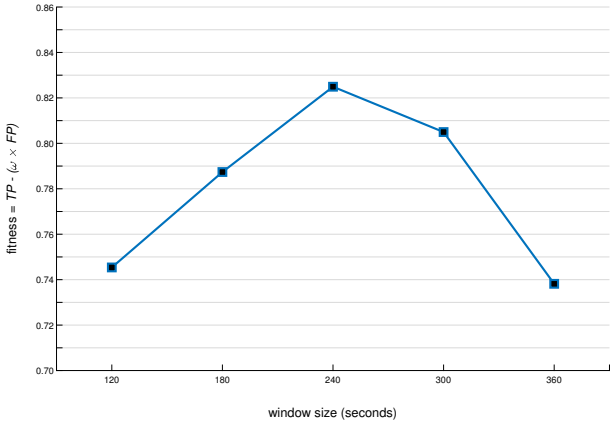

Fig. 3. Effect of the window size on botnet detection

The best, worst, and the mean statistical findings of the best detection algorithm generated from every testing fold have been provided in Table 2. As it is shown in the table, the best algorithm has shown very similar performances on different test settings. The best performance suggests that the evolved algorithm has been able to detect bot traffic with an accuracy of 92.92%. The higher rate of precision value shows that the algorithm has successfully detected bot activities with rare *false alarms*, which is known to be crucial for such an imbalanced traffic.

The evolved algorithm that gives the best accuracy is given in Algorithm 1. GE is known with its ability in generating interpretable programs. Although the best evolved program is readable, it is not easily interpretable since we have employed a large number of functions and features (see Figure 2). In addition to that, we have not limited the length of the generated program. We can conclude from the algorithm that, out of the 78 features, only 31 features have been enough to distinguish the bot traffic from the benign traffic. With 12 source originated and 11 destination originated features used in the detection algorithm, it can be deduced that forward and backward related features have an equal effect on the botnet detection.

In addition to the detection accuracy, the detection phase is another important evaluation metric for the botnet detection systems. In this study, we have also analyzed

Table 2. Statistical performance analysis of the best evolved algorithm

	Performances (%)		
	Best	**Mean**	**Worst**
TP	88.77	88.10	87.17
FP	2.54	2.80	2.98
Precision	97.19	96.93	96.75
Accuracy	92.92	92.65	92.17
F1 Score	92.60	92.30	91.76

```
if ((Flow_Byts_s + min((Subflow_Bwd_Pkts) , (pow(((Init_Fwd_Win_Byts / (Subflow_Fwd_Byts -
Fwd_Pkts_s))) , (0.4421))))) < (Fwd_Pkt_Len_Mean mod max(((max(((Fwd_Header_Len +
floor(ln(pow((pow(((sqrt((pow((Fwd_Header_Len) , (Flow_IAT_Mean)) mod 0.8803)) x
((Down_Up_Ratio + min((Init_Fwd_Win_Byts) , (Fwd_1st_Payload))) / Bwd_Pkt_Len_Mean))) ,
((((Fwd_Blk_Rate_Avg - ((Bwd_Byts_b_Avg - (Pkt_Size_Avg - Bwd_1st_Payload)) mod
Bwd_IAT_Std)) x (pow((pow((Bwd_Pkts_s) , (Init_Fwd_Win_Byts))) , (Flow_IAT_Std)) mod
URG_Flag_Cnt)) + (Init_Bwd_Win_Byts + Subflow_Bwd_Byts))))) , (pow((pow(((Bwd_IAT_Max mod
(max((Idle_Std) , (Fwd_Pkt_Len_Max)) mod Subflow_Bwd_Pkts))) , (0.3514))) , (0.8459)))))))
, ((0.7302 x Bwd_URG_Flags))) + (sin((Bwd_Pkt_Len_Max mod Fwd_Pkt_Len_Std)) x
Subflow_Bwd_Byts))) , (min((pow((min((Fwd_IAT_Min) , ((Pkt_Len_Mean mod Fwd_1st_Payload))))
, (Bwd_Pkt_Len_Max))) , (min((Fwd_PSH_Flags) , (Fwd_Seg_Size_Avg)))))))) {alert();}
```

Algorithm 1: The best evolved algorithm by GE

the detection time of the evolved best algorithm. For this purpose, we have considered 59,933 botnet flows that are correctly classified by our proposed detection algorithm. These individual flows have different numbers of time windows with a maximum of 17 windows. The analysis shows that the algorithm have missed to detect only 69 flows at the first window. In other words, 99.88% of all individual flows are detected at the first window.

In order to perform a better performance evaluation of the proposed algorithm, we have compared its accuracy on three datasets separately to the state-of-the-art systems developed for botnet detection. However it is not easy to conduct a fair comparison due to the different network environments, bot binaries, and etc. [17]. The comparative results (see Table 3) show that the proposed method performs best on the ISOT dataset. In addition, the proposed algorithm has performed better than most of the existing methods.

6 Conclusion

In this study, an evolutionary computation-based botnet detection system is proposed to address the aforementioned issues raised by existing techniques. GE is employed to generate a detection algorithm in a readable form to distinguish botnet traffic from normal traffic. P2P-based botnets, which are the latest and the most challenging type of botnets currently available, are focused in this study. To the best of our knowledge, it is the first study that explores the use of GE for detecting botnet flows. The results show that the proposed method has achieved a very high detection accuracy and performed better than most of the detection methods. In addition, it has accurately detected 99.88% of botnet traffics in their first window which proves the early detection capability of

Table 3. Comparison results

Study	Dataset	C&C structures	Accuracy (%)
Huseynov et al. [2]	ISOT	P2P	72.15
Nogueira et al. [6]	Generated traces	IRC, P2P, HTTP	87.56
Saad et al. [7]	ISOT	P2P	89.00
Huseynov et al. [2]	ISOT	P2P	89.85
Livadas et al. [9]	*Dartmouth* trace [18]	IRC	92.00
Narang et al. [3]	ISOT & Generated trace	P2P	92.55
Fedynyshyn et al. [10]	Generated traces	IRC, P2P, HTTP	92.90
Present study	ISOT & CICIDS2017	P2P	92.92
Wang et al. [8]	Generated traces	IRC, P2P, HTTP	95.00
Present study	CICIDS2017	P2P	95.09
Present study	ISOT	P2P	96.24
Kirubavathi and Anitha [4]	Generated traces	HTTP	99.02
Kirubavathi and Anitha [5]	ISOT & Others*	IRC, P2P, HTTP	99.14

* They merged 11 datasets including various types of bots like Skynet, Rbot and etc.

our approach. Therefore, we can easily conclude that grammatical evolution is very promising and applicable for botnet detection.

References

1. Karim, A., Salleh, R.B., Shiraz, M., Shah, S.A.A., Awan, I., Anuar, N.B.: Botnet detection techniques: review, future trends, and issues. Journal of Zhejiang University SCIENCE C 15(11), 943–983 (Nov 2014)
2. Huseynov, K., Kim, K., Yoo, P.D.: Semi-supervised botnet detection using ant colony clustering. In: Proc. Symp. Cryptogr. Inf. Secur.(SCIS). pp. 1–7 (2014)
3. Narang, P., Reddy, J.M., Hota, C.: Feature selection for detection of peer-to-peer botnet traffic. In: Proceedings of the 6th ACM India Computing Convention. pp. 16:1–16:9. Compute '13, ACM, New York, NY, USA (2013)
4. Kirubavathi Venkatesh, G., Anitha Nadarajan, R.: Http botnet detection using adaptive learning rate multilayer feed-forward neural network. In: Askoxylakis, I., Pohls, H.C., Posegga, J. (eds.) Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems. pp. 38–48. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
5. Kirubavathi, G., Anitha, R.: Botnet detection via mining of traffic flow characteristics. Computers & Electrical Engineering 50, 91 – 101 (2016)
6. Nogueira, A., Salvador, P., Blessa, F.: A botnet detection system based on neural networks. In: 2010 Fifth International Conference on Digital Telecommunications. pp. 57–62 (June 2010)
7. Saad, S., Traore, I., Ghorbani, A., Sayed, B., Zhao, D., Lu, W., Felix, J., Hakimian, P.: Detecting p2p botnets through network behavior analysis and machine learning. In: 2011 Ninth Annual International Conference on Privacy, Security and Trust. pp. 174–180 (July 2011)
8. Wang, K., Huang, C.Y., Lin, S.J., Lin, Y.D.: A fuzzy pattern-based filtering algorithm for botnet detection. Computer Networks 55(15), 3275 – 3286 (2011)
9. Livadas, C., Walsh, R., Lapsley, D., Strayer, W.T.: Using machine learning techniques to identify botnet traffic. In: Proceedings. 2006 31st IEEE Conference on Local Computer Networks. pp. 967–974 (Nov 2006)

10. Fedynyshyn, G., Chuah, M.C., Tan, G.: Detection and classification of different botnet c&c channels. In: Calero, J.M.A., Yang, L.T., Mármol, F.G., García Villalba, L.J., Li, A.X., Wang, Y. (eds.) Autonomic and Trusted Computing. pp. 228–242. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
11. O'Neill, M., Ryan, C.: Grammatical evolution. IEEE Transactions on Evolutionary Computation 5(4), 349–358 (Aug 2001)
12. Ryan, C., Collins, J., Neill, M.O.: Grammatical evolution: Evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) Genetic Programming. pp. 83–96. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)
13. CICFlowMeter: Network Traffic Flow Analyzer. http://netflowmeter.ca/netflowmeter.html, accessed: 2018-10-25
14. jnetpcap. http://jnetpcap.com, accessed: 2018-07-01
15. ECJ: A java-based evolutionary computation research system. https://www.cs.gmu.edu/ eclab/projects/ecj/ (2017)
16. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy. SCITEPRESS - Science and Technology Publications (2018)
17. Lu, W., Rammidi, G., Ghorbani, A.A.: Clustering botnet communication traffic based on n-gram feature selection. Computer Communications 34(3), 502 – 514 (2011), special Issue of Computer Communications on Information and Future Communication Security
18. Henderson, T., Kotz, D., Abyzov, I.: The changing usage of a mature campus-wide wireless network. In: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking. pp. 187–201. MobiCom '04, ACM, New York, NY, USA (2004)