HAL
archives-ouvertes.fr

Cleaning Your House First: Shifting the Paradigm on How to Secure Networks

Jérôme François, Giovane César Moreira Moura, Aiko Pras

To cite this version:

Jérôme François, Giovane César Moreira Moura, Aiko Pras. Cleaning Your House First: Shifting the Paradigm on How to Secure Networks. 5th Autonomous Infrastructure, Management and Security (AIMS), Jun 2011, Nancy, France. pp.1-12, 10.1007/978-3-642-21484-4_1 . inria-00613607

HAL Id: inria-00613607

https://hal.inria.fr/inria-00613607

Submitted on 5 Aug 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

Cleaning Your House First: Shifting the Paradigm on How to Secure Networks

Jérôme François[1], Giovane C. M. Moura[2], and Aiko Pras[2]

[1] University of Luxembourg - Interdisciplinary Centre for Security, Reliability and Trust
jerome.francois@uni.lu
[2] Centre for Telematics and Information Technology (CTIT)
Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)
Design and Analysis of Communications Systems (DACS)
Enschede, The Netherlands
{g.c.m.moura, a.pras} @utwente.nl

Abstract. The standard paradigm when securing networks is to filter *ingress traffic* to the domain to be protected. Even though many tools and techniques have been developed and employed over the recent years for this purpose, we are still far from having secure networks. In this work, we propose a paradigm shift on the way we secure networks, by investigating whether it would not be efficient to filter *egress traffic* as well. The main benefit of this approach is the possibility to mitigate malicious activities before they reach the Internet. To evaluate our proposal, we have developed a prototype and conducted experiments using NetFlow data from the University of Twente.

1 Introduction

When it comes to protecting their networks, Internet Service Providers (ISPs) and businesses employ a large set of specialized tools aiming at mitigate attacks targeting their networks. Examples of such tools include network firewalls, Network Intrusion Detection Systems (NIDS), antivirus, web proxies, and mail filters. Stills, we are far from having secure networks. To better illustrate this, take as example one of the largest security threats on the Internet nowadays: botnets [1]. By definition, a *botnet* is a network of compromised hosts (also known as *bots*/*zombies*) controlled by a botmaster, via a Command and Control (C&C) channel. They are used for different purposes, such as phishing, malware propagation, distributed denial of service (DDoS), and spamming. It is estimated 85% of the more than 100 billion daily spam messages are sent by bots [2].

Behind the current security problems, there might be a subtle defense approach decision: ISPs and businesses usually focus on protecting their own network from the outside world, filtering mostly *ingress traffic*. However, much less attention is usually given to *egress traffic*, meaning that malicious traffic find little or no barrier to leave the originating domain. One example is spam – most companies filter heavily incoming mail, but usually they do not much when their own users spam other domains [3]. Due to that, by the time a security event is detected, it has already taken its own share from routers, network links and computers that it had to go through to reach its final target, imposing direct and indirect costs.

This left us wondering if it would not be the case of *changing the paradigm* on how we secure our networks, filtering egress traffic as well. Thus, this leads us to the following research question: *what can be achieved by filtering egress traffic from a particular domain?* – that is, why not "clean your house before looking for dirty at other's houses?"

Some research works suggest that is worth doing. Van Eeten *et al.*, for example, have shown that 10 ISPs account for 30% of unique IP addresses sending spam worldwide [4]. According to that, by filtering outgoing mail from only 10 ISPs, we could reduce almost one third of all spam. In another work, de Vries *et al.* [3] have shown that egress mail traffic can be easily filtered with higher detection rates.

In order to filter *egress traffic*, many sources of data can be employed, such as mail server logs, network traces, DNS blacklists. In this work we propose the use of flow records [5]. The main advantage is scalability, since flow records provide summarized information about the network traffic, thus coping much better with cfurrent high speed multi-gigabit lines. Besides that, by using flows records, the communication patterns in a network can be evaluated instead of having to process the content of each packet [6]. Finally, flows records are application independent, so they can potentially be used to detect and block any type of malicious activity in the network.

To analyze flow records, in this work we employ cluster analysis, since it is a unsupervised learning technique that does not require *a priori* knowledge about malicious communication patterns (in contrast to signature-based NIDS). The assumption is that we can detect different types of malicious traffic using flow records and cluster analysis. To prove technical feasibility of our proposal, we have developed a prototype and conducted an evaluation on network flows obtained from the University of Twente.

The rest of this paper is organized as follows: Section 2 provides background information and introduces the architecture proposed for detecting intra-domain malicious hosts. Next, Section 3 details the clustering algorithms employed. After that, Section 4 covers the experiments and the results obtained. Next, Section 5 presents the related work. Finally, Section 6 concludes the paper and proposes future work.

2 Intra-domain Malicious Hosts Detection Architecture

Figure 1 shows the proposed architecture for detecting intra-domain malicious hosts. NetFlow-enabled routers export flow records to a collector. This information is stored on a database and fed into the Anomaly Detection Engines (step 1 – in between parentheses – in the same figure) which are responsible for analyzing the input data. A broad range of attacks can be analyzed such as DDoS or port scan but we focus on spamming since botnets are well used for spamming [7].

Hence, we aim to find bots by looking at group of hosts having similar communication patterns with some of them involved in spamming activity. In this way, detecting spammers helps to figure out entirely botnet related traffic (C&C as well as other malicious activities). To do that, we first obtain a list email senders (step 2 in Figure 1) and exclude legitimate mail servers from our domain (step 3). Next, hosts sending many more emails than others are considered as potential spammers (step 4).

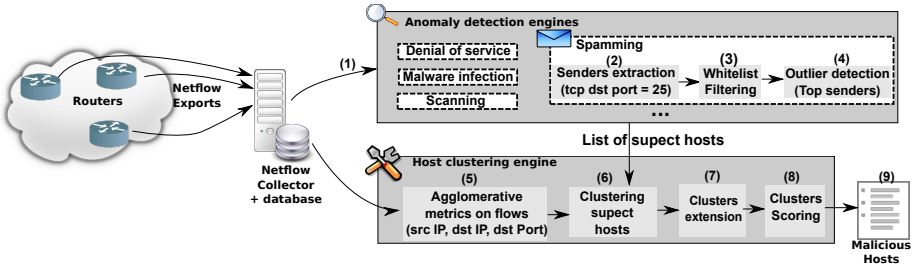


Fig. 1: Intra-domain malicious hosts detection architecture

After obtaining a list of spamming hosts, we compute the following aggregated metrics (obtained from Botminer [8]) for all flows: the average number of individual flows per hour (*fph*), the average number of packets per flow (*ppf*) and the average number of bytes per packet (*bpp*).

In the end, for each flow we have the following tuple: <source IP address, destination IP address, destination UDP/TCP port, fph, ppf, bpp>. These metrics (step 5) allow to easily compare flow records from different hosts in order to find shared communication patterns (step 6). For example, if messages sent by a bot master reach two different hosts in our network, it is expected that they have similar properties, such as similar number of packets and bytes. In the same way, bots from a peer-to-peer botnet should exhibit similar communications to route the messages and to maintain the coherency of routing tables when nodes join or leave the network.

Once these agglomerative metrics are computed, the next step is to compare such metrics for flows related to hosts exhibiting abnormal activities in order to cluster them (step 6). It is important to note that it is applied on flows and not on hosts. It means that it also helps to distinguish the malicious traffic pattern from the benign ones for a single host.

The major advantage is the reduction the number of flows to be analyzed. Then, we extend the clusters to other flows (step 7) by comparing them with flows related to suspect hosts, reducing the overall complexity of the algorithm. Finally, a score is computed (step 8) for each cluster based on the similarity of flows within it and the number of hosts it contains which are tagged as suspect. Hosts as source of Netflow included in highly scored clusters are declared as malicious (step 9).

3 Detection Algorithms

3.1 Anomaly Detection Engine – Top Email Senders

On step 2 in Figure 1, we have to find a list of spamming hosts. After listing all hosts that have connections to mail servers *outside* our domain (machines have outgoing TCP connections on port 25), we remove legitimate IP addresses of legitimate mail servers from UT (step 3). Finally, we compute two metrics for each remaining host:

– n_i: the number of mail flows records per the host i,

– b_i: the total volume of email data sent per host i (in bytes).

The idea behind combining these metrics is that we can detect both hosts contacting many different mail servers and hosts sending too much mail data (specially related with spam campaigns that include attachments, such as PDF files). A more complex approach to detect spam using flow records was proposed by Sperotto *et al.* [9]. However, in our case we employ a faster and simpler approach because the output is the list of potential spammers for which no decision have to be taken and so can include benign hosts which will be discarded afterwards.

Therefore, a host is considered as a spamming one if the number of emails and bytes sent is higher than the observed average plus a margin expressed as a multiple of the standard deviation. Considering all hosts, the average number of emails sent by an individual host is avg_n and the corresponding standard deviation is std_n. In the same way, avg_b and std_b refer to the number of bytes. i is a spamming host if:

$$(n_i > avg_n + \sigma std_n) \; or \; (b_i > avg_b + \gamma std_b) \tag{1}$$

In this paper, γ and σ are set to 3 based on preliminary experiments. The corresponding hosts form the set S which is constructed in a linear time (iteration over all email senders).

3.2 Email Senders Clustering

Before starting the clustering algorithm, we obtain the following tuple for each single flow: <source IP address, destination IP address, destination UDP/TCP port, number of bytes, number of packets>. After that, we divide this set into two subsets: F_s, a subset that contains all the flows related to the spamming hosts and F_a, that contains all the remaining flows from the other machines. Then, we compute for each flow $f \in \{F_s \cup F_a\}$ the metrics introduced in section 2 (fph_f, ppf_f, and bpp_f).

In order to reduce the computational complexity, the first clustering process focuses on the suspect IP addresses (potential spammers) and creates clusters containing aggregated flow information from F_s. Hence, the goal is to find similar communication patterns involving multiple suspect hosts. Without any prior knowledge, unsupervised clustering is required. Besides, there is no assumption about the shape of clusters (following a certain distribution) and that is why nearest neighbor clustering [10] is fitted in our case. Medoids based methods are excluded because they entail computational overhead.

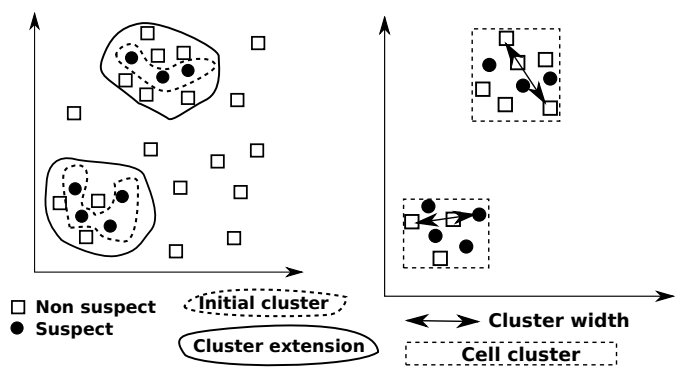
Nearest neighbor clustering assumes that two data points belong to the same cluster if the distance, $dist(d_1, d_2)$, between them is lower than the threshold θ. Regarding our context, each data point represents a tuple f as a vector $[fph_f, ppf_f, bpp_f]$. After normalizing the values, we applied the Euclidean distance on the vectors since this is commonly used in various contexts with multidimensional data.

The algorithm iterates over all f_i of F_s and compute $dist(f_i, f_j)$ for all $f_j \in F_s \; and \; f_j \neq f_i$. The pairs of points which the resulting distance is lower than θ are aggregated into one cluster. If the aggregated points were prior assigned to another clusters, all points belonging to them are also aggregated (merging). The result is a set of clusters C.

Like many unsupervised algorithm, computing the distance between each pair of data points is needed which implies a quadratic complexity. Thus, this clustering process is only applied to a limited subset of points which were previously selected and form the set F_s.

3.3 Extending Clusters

The assignation list represents for each flow, the cluster where it is affected. Assuming K clusters, $C = \{c_1, \ldots, c_K\}$, the assignation list is $A = \{a_1, \ldots, a_{|F_s|}\}$ such that $a_i = c_j$ if the flow $f_i \in F_s$ is assigned to the cluster c_j. Each remaining non suspect point is assigned to the closest cluster except if this distance is too high that lead to consider the host as a benign one. They are represented by points outside of clusters in Figure 2a which shows a toy example in two dimensions. There are a first set of initial clusters constructed in the previous step and then the clusters are extended.



(a) Cluster construction (b) Cluster scoring

Fig. 2: Clustering algorithms on a toy example

The distance between a new point to assign and a cluster is the minimal distance between this point and any point of the cluster. The assignation list denoted $A' = \{a'_1, \ldots, a'_{|F_a|}\}$ is defined as follows:

$$a'_i = \begin{cases} a_k \ if \ f_k = \underset{f_j \in F_s}{\arg\min} dist(f_i, f_j) \ and \ dist(f_i, f_k) < \theta' \\ unassigned \ else \end{cases} \tag{2}$$

From a computational point of view, the distance between each non suspect data point and suspect data points have to be computed. It is equivalent to $|F_s| \times |F_a|$ iterations. Considering the quadratic complexity of the constructions of clusters in the previous step, the total number of iterations is $|F_s| \times (|F_s| + |F_a|)$ whereas naive clustering of all flows would have led to $(|F_s| + |F_a|)^2$ iterations.

3.4 Scoring

Once clusters are created, the goal is to identify those containing hosts with higher probability of being malicious. Since our approach relies on the malicious activities and the similar communication patterns, a score is assigned to each cluster based on these criteria. In brief, the first component of score named $score_anomaly_i$ represents the proportion of hosts related to malicious activities in the cluster c_i:

$$score_anomaly_i = \frac{|\{f_j \in F_s, a_j = c_i|\}}{|\{f_j \in F_s, a_j = c_i|\} + |\{f_k \in F_s, a_k = c_i|\}} \tag{3}$$

The other component of the global score considers also the similarity among the flow information containing in the cluster. The lower the distances between the points of a cluster, the higher the score is. This can be regarded as the width of the cluster (maximal distance between two internal points as shown in figure 2b). The computation of the width can be long since clusters should contain hundreds or thousands of points and computing all pair-wise distance is quadratic. Therefore, we propose a simple method inspired from grid clustering techniques [11] where each cluster is represented as a squared cell like in the toy example in figure 2b. Only one iteration per point is needed to compute the coordinate of the cell since the goal is to find the maximal and minimal value for each dimension (two in the toy example and three in our context). Assuming the cluster c_i, the similarity score is defined:

$$score_sim_i = dist(FMin_i, FMax_i) \tag{4}$$

where $FMin_i$ and $FMax_i$ are fictive points containing minimal and maximal values for fph_f, ppf_f and bpp_f subject to f assigned to c_i. For example, the first feature of $FMin$ is:

$$\min_{\{f_j \in F_s, a_j = c_i\} \cup \{f_j \in F_a, a'_j = c_i\}} fph_{f_j}$$

Since, the iterations have only to cover each point one time, the complexity is $O(n)$ where n is the number of points in a cluster. Traditional methods have to compute pair-wise distances for extracting then the minimal, maximal or the average one. Unlike our method, the complexity is $O(n^2)$.

Finally, the global score of the cluster c_i is the usual mean of both scores:

$$S_i = \frac{score_anomaly_i + score_sim_i}{2} \tag{5}$$

If the score is higher than the threshold ψ, all source IP addresses related to the cluster are considered as malicious. It includes spamming hosts as well as other ones thanks to the cluster extension process.

4 Experimental Evaluation

In this section we describe the evaluation conducted to prove technical feasibility of our proposal. As describe in Figure 1, the first step is to obtain NetFlow data from external data sources. For this experiment, we have obtained two NetFlow datasets from the University of Twente (a /16 network):

- **Dataset A**: 1 hour of flow records (April 10th, 2010, from 5:00 PM to 6:00 PM CEST) – a total of more than 12 million records;
- **Dataset B**: 2 hours of flow records (April 10th, 2010, from 3:00 PM to 5:00 PM CEST) – more than 24 million records.

Dataset A was used in the anomaly detection engine in Figure 1, while the dataset B was used in the host clustering engine. Next we present the detection results and the validation.

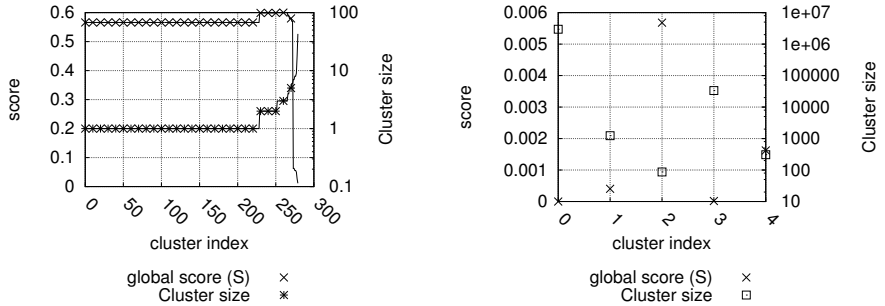
4.1 Malicious Host Detection

After obtaining the aforementioned datasets, we analyze the first one (A) to find hosts that have contacted more mail servers outside of our domain (steps 2 and 3 in Figure 1). Then, we removed legitimated mail servers from this list, and two hosts have been automatically selected to be fed into the host clustering engine. The first host was tagged as suspect since it has contacted 45 distinct mail servers outside our domain (some of them more than one time) within one hour, in a total of 250 flow records. Thus, we can assume that at least 250 accounts were target. The other host tagged as suspect has contacted 12 distinct mail servers in a total of 12 flow records.

Next, we have computed the metrics defined in Section 2 for the dataset B (step 5). In the end, we had 5,424,333 entries in the metrics table with the following format: IP source/destination, destination port, fph, ppf, and bpp. It is important to emphasize that we have computed these metrics for not only mail flow records, but all flow records. By doing that, our algorithm is suitable detecting common communication patterns and not only spam. So even, it helps to distinguish potential C&C flows from other ones even if there are few hosts.

In step 6 and 7, two level clustering is applied on the metrics obtained in the previous step, regard the parameter θ. We assume that $\theta = \theta'$ (the similarity within a cluster has to be same when comparing spamming hosts or any other hosts). Therefore, when θ increases, more points are grouped within each cluster and so the total number of clusters decreases. This is shown in Figures 3a and 3b where each cluster is represented by two points (the score and the size equivalent to the number of distinct source IP addresses among all flows of the cluster). The x-axis represents only an arbitrary cluster index but also indicates the total number of clusters which is the maximal index. More complex method, based on cluster characteristics, might be used to fine-tune the parameters similar to [12].

In Figure 3a, there are two main groups of clusters. Firstly, many clusters have a very low scores. They represents normal group of IP addresses which the underling applications exhibit different patterns. Secondly, there are many clusters with high score (> 0.5). In fact, the corresponding sizes are very low (one or two IP addresses). Therefore, the high score is only due to the bias introduced by the anomaly score (*score_anomaly*) equal 1. Indeed, such clusters may easily contain 100% of flows related to potential spammers since they contain only one or two IP addresses. That is why these clusters are discarded for further analysis. However, there is a third group of outlier scores below 0.1 between these extrema.



(a) Cluster construction with $\theta = 1e-12$ (b) Cluster construction with $\theta = 1e-06$

Fig. 3: Cluster construction

In order to figure out them easily, θ is increased in Figure 3b to merge clusters with few IP addresses. This case is an extreme one showing very few clusters and highlighting only one score greatly higher than others (the second index on x-axis). Therefore, this case was chosen in the end of the clustering process. The selected cluster contains 100 different flow tuples from 52 distinct IP addresses under University of Twente domain, which represent the list of malicious hosts obtained in the step 7.

4.2 Complexity

The core construction of initial clusters is quadratic (section 3.2) due to the calculation of all pairwise distances. To avoid this drawback, some samples (potential spammers) are selected in a linear time (33k iterations) regarding the number of SMTP flows (section 3.1). Then, the two selected hosts coverages $143k$ records. It leads to $143k^2$ calculations for the initial clustering. Then, $143k \times 24M$ operations are necessary for the cluster extension which represent around 0.6% if no prior selection and clustering were performed ($24M^2$ iterations to compute all pair-wise distance). Finally, the scoring process has to deal with about 3,000,000 points in the worst case (the biggest cluster). The complexity of our scoring method is linear whereas traditional approaches are quadratic (section 3.4). So, the number of iterations is also divided by 3M.

Even our approach has similarities with other ones [8], we have really focus on reducing the running time by optimizing the algorithms. This is particularly important when monitoring large networks.

4.3 Validation and Egress Traffic filtering

When evaluating the performance of techniques for intrusion detection, researches usually rely on labeled datasets, which contains meta-information about the attacks observed. Usually such kind of datasets are available in pcap format, *i.e.*, complete network traces. Since our technique is based on flow records instead, we could not benefit from these datasets. Even though Sperotto *et al.* [13] have provided the first labeled

dataset for flow-based intrusion detection, this could not be used in our research, since in this work we evaluate egress traffic instead. More than that, their dataset is based on a single host – which is not suitable for our clustering technique. Due to that, we have to check manually the flow records associated to the 52 malicious hosts obtained as a result of our detection technique to try to find whether malicious behavior was observed or not. Some interesting findings were obtained:

- One desktop PC was found having 7151 SMTP flows to 245 different mail servers located in many different countries for a 24 hours period. Since flows contain only a summary of a connection, we cannot tell how many messages are sent per flow. Assuming, in this case, that only one message was sent per flow, we have a total of almost 5 mail messages send per second, which is very unusual for a desktop. Figure 4 shows the number of SMTP flows to each mail server. Moreover, the same machine have contacted two different IRC servers, in a total of 1193 flows. Such behavior is typical for a machine belonging to a spamming IRC botnet.
- One Windows desktop found running a non-authorized service on UDP and TCP port 56168. After checking with the Security Administrator at UT, it was found that this machine was the desktop of a professor that was unaware of it. He was promptly notified. In this two hour period, this machine has been contacted by 72579 different IP addresses on the aforementioned port, transmitting more than 66MB of data. Also, a hidden web server was found on this machine, which was contacted by 353 different hosts. We have extended the analysis for this machine and found out that 330,925 different IP addresses reached it on April 10th 2010 – a very suspect behavior for a desktop. We suspect this machine may be working as a botmaster (remotely controlled) or as a coordination point for botnets like those used by Storm [14].
- Another computer on the wireless network – which, by definition, should not run any services – was found running a suspect service on port 23352, for TCP and UDP (mostly UDP). In this 2 hours period, the machine was contacted by 96.609 different hosts from various countries. Since this machine is mobile, we could not reach it by the time of the analysis. For this period, 19GB of data was transfered to these different hosts.
- One machine from the student network was also running a suspect service on UDP and TCP port 32861, which was contacted by 77,434 different hosts in two hours. 67 MB of data were transfered in this case.
- Another host was found running a suspect service on TCP and UDP port 39563, contacted by 79824 distinct IP addresses from various locations. 66 MB of data were transfered in this two hours.

Even though the validation process was manually and not extensively executed, this results shows that our approach was able to detect the involved hosts based on very small list of potential suspect addresses – 2 spamming hosts detected before applying clustering. By blocking such malicious hosts in our domain ("cleaning our houses first"), we can avoid their malicious activities to reach the Internet. Some estimates can be calculated from blocking malicious hosts: assuming that every flow of the spamming bot we found represents a single spam, by blocking only this host, we could avoid 7151

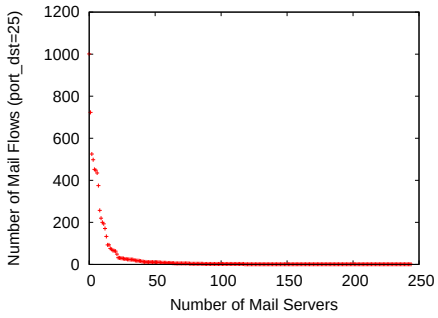


Fig. 4: Number of flows from one suspect host to different mail servers

spam messages to reach the Internet. If the botnet this host is part has a size of 100k bots, by dismantling it (looking at the IRC traffic) we could potentially avoid 715 millions spam messages to reach the Internet in a single day. That represents 878.2 GB of data by extrapolating the monitored metrics. The same reasoning can be extended to other machines that our previous analysis has figured out and as well to other source of network attacks, such as DDoS. The more egress filtering is employed by ISPs and businesses, the more malicious traffic can be blocked from the Internet.

5 Related Work

Van Eeten *et al.* [4] have evaluated a dataset of 63 billion spam messages obtained between 2005 and 2008. By analyzing the IP addresses of the sources, they observed that 10 ISPs account for 30% of unique IP addresses sending spam worldwide, 50 ISPs for half of all sources. Even though this study was performed on a not up-to-date data set, it suggests the benefits that can be achieved by filtering egress traffic of few ISPs.

In another work, de Vries *et al.* [3] have shown that egress mail traffic can be easily filtered using lightweight techniques. However, in this work the authors rely on the message's content when filtering the traffic. A survey on flow-based intrusion detection, on the hand, was presented by Sperotto *et. al* [6]. Differently from this work, the authors focus on detecting malicious host on the Internet, while in this work we target intra-domain malicious hosts.

Deploying a honeypot to be infected by a bot software is usually a direct and convenient way to study a botnet but it may not be efficient [15]. Tracking infected hosts can also be done by monitoring DNS requests of the machines [16] especially for an IRC botnet. A lot of techniques detect a botnet relying on the malicious activities such as scanning or denial of service attacks [17]. In our approach, we leverage the same knowledge but then we improve the botnet detection by detecting common communication patterns of the C&C channel. P2P botnets are usually detected by active techniques such as in [18]. Graph algorithms may be also employed to infer interesting properties of bots relationships [19].

In this work, we have conducted a study case on spamming hosts. We based our work on two works by Gu *et. al* [8] [20]. In our work, we correlate malicious activities

with C&C detected communication patterns. The main difference between BotMiner [8] and our approach is that we apply clustering only to a small subset of Netflows resulting in clusters which are extended afterwards to all Netflows. It leads to a huge improvement of the complexity and so the running time.

6 Conclusions and Future Work

In this paper we propose a new paradigm to be employed by ISPs and businesses for protecting their own networks. Instead of solely filter ingress traffic, in this work we investigate the benefits that can be achieved by filtering *egress* traffic as well (cleaning our own house first). The motivation is that if such policy is widely adopted, the overall amount of malicious traffic on the Internet could be significantly reduced. That would ultimately lead to saving considerable amounts of money and computer/network resources.

Therefore, in this paper we investigated the following research question: *what can be achieved by filtering egress traffic from a particular domain?* To answer this question, we have refined clustering techniques to analyze flow records from the University of Twente. As our results have shown, we were able to detect many suspect hosts. By detecting and blocking one of these hosts, for example, we could have been able to avoid that 7151 spam messages could reach the Internet in first place. The same reasoning applies to the other suspect hosts. More than that, our results have shown that such filtering could help to detect and dismantle botnet operations *outside* the monitored domain. The benefits of egress filtering would only increase as more ISPs and businesses adopt it as a common practice.

As future work, we intend to combine surpervised detection methods (since they are more efficient to detect known attacks) with clustering analysis. The idea is to combine both methods for improving the detection accuracy in various scenarios. Finally, we plan to obtain an economic model in order to estimate how much can be saved by filtering egress traffic.

Acknowledgments The authors would like to thank Radu State, Anna Sperotto, Marc Berenschot, Ramin Sadre, and Olivier Festor for their valuable comments, suggestions, and contributions for this work.

References

1. Arbor networks. Worldwide infrastructure security report (2009 report). Technical report, 2010.
2. John P. John, Alexander Moshchuk, Steven D. Gribble, and Arvind Krishnamurthy. Studying spamming botnets using Botlab. In *NSDI'09: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, pages 291–306, Berkeley, CA, USA, 2009. USENIX Association.
3. W. W. de Vries, G. C. Moreira Moura, and A. Pras. Fighting spam on the sender side: A lightweight approach. In F. A. {Aagesen} and S. J. {Knapskog}, editors, *Proceeings of 16th EUNICE/IFIP WG 6.6 Workshop (EUNICE 2010), Trondheim, Norway*, volume 6164/2010 of *Lecture Notes in Computer Science*, pages 188–197, Berlin, 2010. Springer Verlag.

4. Michel van Eeten, Johannes M. Bauerb, Hadi Asgharia, Shirin Tabatabaiea, and Dave Randc. The Role of Internet Service Providers in Botnet Mitigation: An Empirical Analysis Based on Spam Data. In *WEIS 2010: Ninth Workshop on the Economics of Information Security*, 2010.
5. Cisco Systems. Cisco IOS NetFlow, August. 2010.
6. A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. An Overview of IP Flow-Based Intrusion Detection. *Communications Surveys Tutorials, IEEE*, 12(3):343 –356, 2010.
7. Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. Spamming botnets: signatures and characteristics. *SIGCOMM Comput. Commun. Rev.*, 38(4):171–182, 2008.
8. Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *USENIX Security Symposium (SS)*, pages 139–154, San Jose, CA, July 2008.
9. Anna Sperotto, Gert Vliek, Ramin Sadre, and Aiko Pras. Detecting spam at the network level. In *EUNICE '09: Proceedings of the 15th Open European Summer School and IFIP TC6.6 Workshop on The Internet of the Future*. Springer-Verlag, 2009.
10. William H. Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1):7–24, December 1984.
11. Erich Schikuta. Grid-clustering: a fast hierarchical clustering method for very large data sets. In *In Proceedings 15th Int. Conf. on Pattern Recognition*, pages 101–105, 1996.
12. Jérôme François, Humberto Abdelnur, Radu State, and Olivier Festor. Automated Behavioral Fingerprinting. In *12th International Symposium on Recent Advances in Intrusion Detection (RAID)*, St Malo France, 2009.
13. Anna Sperotto, Ramin Sadre, Frank van Vliet, and Aiko Pras. A Labeled Data Set for Flow-Based Intrusion Detection. In Giorgio Nunzi, Caterina Scoglio, and Xing Li, editors, *IP Operations and Management*, volume 5843 of *Lecture Notes in Computer Science*, pages 39–50. Springer Berlin / Heidelberg, 2009.
14. P. Porras, H. Sadi, and V. Yegneswaran. A Multi-perspective Analysis of the Storm (Peacomm) Worm.
15. Ping Wang, Lei Wu, Ryan Cunningham, and Cliff C. Zou. Honeypot detection in advanced botnet attacks. *Int. J. Inf. Comput. Secur.*, 4(1):30–51, 2010.
16. Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *ACM SIGCOMM conference on Internet measurement (IMC)*, pages 41–52, 2006.
17. Anestis Karasaridis, Brian Rexroad, and David Hoeflin. Wide-scale botnet detection and characterization. In *First Workshop on Hot Topics in Understanding Botnets (HotBots)*. USENIX, 2007.
18. Thorsten Holz, Moritz Steiner, Frederic Dahl, Ernst Biersack, and Felix Freiling. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–9, Berkeley, CA, USA, 2008. USENIX Association.
19. Jérôme François, Wang Shaonan, Radu State, and Thomas Engel. BotTrack: Tracking Botnets using NetFlow and PageRank. In *To appear in IFIP/TC6 NETWORKING 2011*, Valencia, Spain, May 2011.
20. Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. Bothunter: detecting malware infection through ids-driven dialog correlation. In *USENIX Security Symposium (SS)*, August 2007.