Distributed malware detection

Péter Kenyeres[1], Tamás Mészáros[2], Attila Szentgyörgyi[1], Gábor Fehér[1]

Budapest University of Technology and Economics Department of Telecommunications and Media Informatics[1]
Budapest University of Technology and Economics Mathematical Institute[2]

Abstract. Widespread usage of broadband Internet connections has allowed the birth of a new threat against service providers and subscribers as well. Botnets are vast networks of compromised hosts under the control of single masters who possess the ability to launch crippling denial of service attacks, send vast quantities of unsolicited e-mail messages and infect thousands of vulnerable systems with privacy-violating spyware and other forms of malicious software. Our goal is to propose a distributed architecture and introduce novel algorithms for malicious (potential botnet) activity recognition based on network traffic statistics generated by NetFlow. Scalability and robustness were the main principles during the design of the architecture. In this paper, we demonstrate that we are able to reduce the number of NetFlow records significantly with an aggregation scheme. Furthermore, we are able to detect botnet participant computers (zombies) with the help of aggregated samples originating from other networks, while the algorithms provide utmost anonymity to participants.
Keywords: distributed botnet detection, netflow

1 Introduction

In the last decade the global Internet threats transformed considerably from the previous plain attacks executed individually to those distributed attacks that are capable of disabling whole infrastructures. This new kind of threat - indirectly or directly - seeps into the everyday life of millions of people and it does not spare the business world either. In most cases botnets are responsible for these attacks.

Actually, malicious botnets are multitude of infected computers that are remotely controlled by master host via one or more controller hosts. The master host itself is a computer that is used by the owner of the botnet to send commands to controllers. In most cases, these controllers are infected hosts as well and take a part in the network's coordination: relaying the instructions to executive hosts (bots).

Botnets are used for various malicious purposes such as: distributed denial-of-service (DDoS) attacks, sending spam, phising or trojan e-mails, serving phising sites, distributing pirated media, stealing personal information, performing click fraud, etc. Besides, they also have aggressive exploit activity as they rope in new vulnerable systems to increase size of the network.

However, it is relatively easy to detect attacks [10] [17] [18] [19] the elimination or paralysis of botnets raise more serious challenges.

In this paper we introduce a novel security architecture which is capable to work globally, scalable, efficient and can be anonymous. The architecture relies on a peer-to-peer (P2P) distributed hash table (DHT) to satisfy the scalability and the global availability requirements. Because of the high volume of network traffic NetFlow [20] is used to reduce the storage space required for traffic logs. Data anonymization is a key issue in the system, because joined peers do not have the intention of revealing its traffic properties. With the proposed architecture network administrators will be able to detect new threads and efficiently can react to the infections.

The remainder of the paper is organized as follows. In Section 2 the related work is presented. In Section 3 the system model is introduced including the system architecture and the different type of nodes participating in it. In Section 4 the components of our system and realization of the design priorities are presented. In Section 5 efficiency of algorithms are evaluated. Finally, some open questions are discussed and the results are summarized in Section 6.

2 Related Work

There has been a rich literature on NetFlow-based data collection and its applications. Network monitoring and anomaly detection are the focuses of researches. The passive monitoring system introduced by Sprint [12] was installed within the Sprint IP backbone network. Data was collected from different monitoring points and it was transmitted into a central repository for further analysis. The Gigascope approach [11] is similar to the Sprint's passive monitoring system, because Gigascope is using a central repository for further analysis as well. The passive monitoring infrastructure CoMo [13] allows users to query network data gathered from multiple administrative domains. Numbers of distributed monitoring nodes are available in the network to serve the user's needs. LOBSTER [14] is another passive network traffic monitoring infrastructure including forty sensors in twelve countries in three continents. With LOBSTER more than 600,000 sophisticated cyber attacks were captured during a one-year period.

Liu Bin et. al. [15] proposed a flow analysis and monitoring system based on NetFlow as well. The system was designed in client-server model for enterprise networks. The system relies on different modules, such as data collection module receiving and analyzing NetFlow and display module serving the web users. In addition, the IDS module included into the system runs with low computational complexity and high detection accuracy.

Our proposed architecture offers more than the aforementioned ones, because it does not only captures the network traffic, detects and indicates new threats, but processes the incoming anonymized data and redistributes the results to the nodes of the network. With these capabilities the system is able to react fast and efficient way to new infections.

3 System Architecture

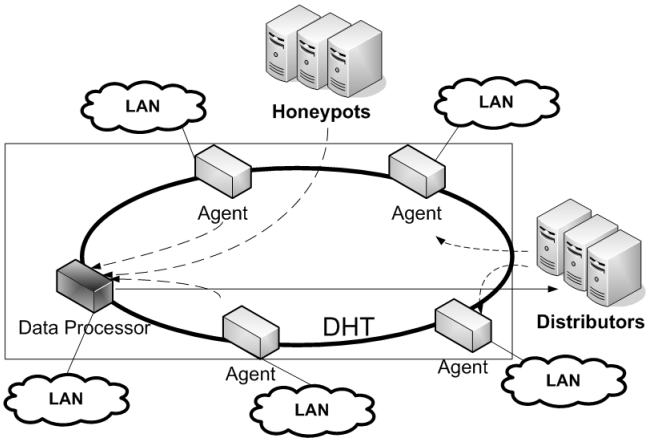

Fig. 1. System architecture with agents, honeypots data processors and distributors

The proposed architecture is depicted in Figure 1. and consists of four different types of nodes: agents, honeypots, data processors and distributors. The first three are connected via a P2P network designed for distributed data search and transfer considering scalability issues. This property is required to attach as many nodes to the system as possible to reach a globally available and distributed malware detection system. The roles of the components in the system are the following:

- Agents: Computers in enterprises and home users are connected to the Internet via a local gateway. This gateway separates their own LAN from the Internet. In enterprise networks or just in a SOHO environment gateways are capable to dump network traffic. Agents can be placed next to gateways, because it is a requirement for agents to collect and process traffic information of their LAN. Agents are connected to the P2P network and can communicate with other agents. Nevertheless, agents have another important role in the network: they monitor the traffic and try to detect malicious activities coming from its own subnet, such as DoS attack or spam. When an agent has detected a new threat, it should search the current designated data processor using the P2P network and has to send the anonymized and compressed traffic data of the suspicious node to the flow processor.
- Honeypots: These entities mark the suspicious traffic. When a new threat was detected, honeypots should create traffic traces of the malware, mark their command and control (C&C) channel and have to send the marked and anonymized trace to the current data processor.

– Data processor: It is a designated node of the P2P network. At the same time only one data processor presents in the network, but data processor is changing at times. It collects the reports of malicious activities and the corresponding anonymized and compressed flows from agents and the anonymized and marked flows from honeypots. Its task is to create clusters from the data, evaluate the results and if malicious activities are detected it will have to send the network traces to a distributor.
– Distributors: Distributors are responsible for collecting anomalous traces and sharing them with the agents. Distributors are independent from the P2P network. They accept requests from the data processors and store them to be available for the agents as a regular update.

4 System Components

4.1 Overlay Network

Robustness is a requirement for the architecture to eliminate DoS attacks. Resources (for instance: zombie networks) controlled by the attacker can be divided into two sets. One set of the zombies disable the detection system by distributed denial of service attack (DDoS) while the other set of infected computers commit the originally planed attack. Several papers [6] [7] [8] describe methods to defend a computer against flood attacks. Our transfer solution applies a protection against DoS as well, by using the Secret Overlay Service (SOS) [9], because this method guarantees the further high of design priorities, such as flexibility, scalability and fairness in task distribution.

Basically, SOS is a large distributed firewall which has two essential parts. First of all, targets are protected by sophisticated filters against 'unauthorized' traffic. On the another hand, a multilevel hierarchy endeavors to hide the access of target nodes from the attacker. This hierarchy with filtering mechanisms makes this scheme robust against the DDoS attacks, because each component is easily replicated within the architecture. If the attacker blocks a node A its duties will be reassigned by the DHT and node B will take charge. Hence, the attacker has to block node B also, while his attack against node A can not to be stopped. Otherwise, that one who is released rejoins to the system. The resistance of a SOS network against DoS attacks considerably depends on the number of nodes that participate in the overlay.

Task Distribution Method For the analysis of the collected attack samples a responsible node is appointed in the DHT. The current data processor is determined by locally stored and maintained seed value. This seed value is denoted by S and it serves as input to the hash function of the DHT. The value received in this manner always selects one node from the DHT. The chosen node transmits the packets towards to the *data processor* by SOS routing mechanism.

The next value of S has to be locally computable to reduce operational message overhead. The designation of the data processor by changing value of S may happen according to the following methods fundamentally:

– Time interval: in this case, S is actually a numerator increased after a certain interval. Typically, this interval means a couple of hours. For more details see also the results in Section 5.
– Victim network address: at this time the IP address (S) of the attacked network is used as input of the hash function.
– Attack types: certain attack types are predefined (e.g.: scanning, DDoS, spam, etc.). All these types are associated to unique values. Finally, these values are going to serve as S.

4.2 Flow Aggregation

Our method is based on NetFlow [20] logs. The most important fields in a NetFlow record are the source and destination IP addresses, the source and destination port numbers and the transport protocol, since these define a session. If four of them (including the IP addresses and the protocol) are the same in two NetFlow records, namely the same IP addresses are communicating with the same transport protocol, and at least one of them on the same port, we can assume that the two records belong to the same session and it is unnecessary to treat them separately. In the first step flow regrouping can be done by putting flow records with the similar connection parameters to the same group to represent each group by a single flow.

Note that if it is true for two flows A and B that $srcIP_A = destIP_B$, $destIP_A = srcIP_B$, $srcPort_A = destPort_B$ OR $destPort_A = srcPort_B$, then these flows can be aggregated, because these flows represents the different direction of the same connection. The up and down direction can be chosen arbitrary.

First, outlier filtering is applied for both directions to exclude anomalies and irrelevant data. This filtering due to all dimensions respectively is done by computing the m mean and the ς variance of the values. If the variance is relatively high (if $\frac{\varsigma}{m}$ is grater than a fixed ε_0), then the flow with the most outlying value will be discarded. This step is iterated until there will be no more outliers. The last step is the aggregation of the remaining flows in each group to obtain a representant. The values for packets, octets and active time will be added up, the earliest start time and the latest end time will be selected, and 5 more values will be computed: number of flows aggregated, mean packet size, mean active time, duration (the time elapsed between the earliest start time and the latest end time), up/down+down/up (up stands for the sum of octets in the flows with direction up and down). This 5-tuple will represent a group. The IP addresses, port numbers and the transport protocol are omitted to get a kind of anonymity.

4.3 Flow Processing and Sample Generation

The incoming aggregated NetFlow logs have to be classified to obtain flow samples belonging to the botnet traffic we want to detect. Logs are sent by agents which detected an attack. If this agent is a honeypot, the traffic logs will contain botnet traffic related flows (C&C channel communication and attack). These

flows are trusted in the sense that these are originated from a trusted entity and can be used as a sample of the botnet traffic. For this reason these flows are referred as labelled flows. The flows captured by honeypots that do not belong to the C&C channel can be labelled differently or simply omitted. The classifier should handle the huge dataset size and the consequences of multiple botnet activities. To address these challenges a semi-supervised learning technique is applied, which is a modified version of the general method discussed in [1]. Clustering [2] is applied to partition the data set that consists of labelled and unlabelled flows. Several previous works [3] [4] [5] demonstrated that clustering of Internet traffic using flow statistics has the ability to group together flows according to the same traffic. In this paper we applied the K-Means algorithm [2], since it is simple, easy to implement, offers fast computation, demonstrated good results in previous works e.g. [3] and converges in a few number of iterations. After clustering the supervised learning is applied to label the clusters using the labelled flows. Unlabelled flows are used to improve the precision of the classifier.

It is not our purpose to identify all of the clusters, our aim is just to select those, which belong to botnet communication. Now we will discuss the details of the classification method.

Cluster Identification and Sample Generation The output of K-Means is a set of vectors, which are the centers of the clusters. If a vector x is given, it is assigned to the cluster with the nearest center. Next step is the identification of the botnet traffic related cluster(s). We use a probabilistic method, similar to the one described in [1]. Let p_i be the probability of the event that the ith cluster, C_i, $i = 1, 2, \ldots, K$ is the cluster belonging to the botnet communication. These p_i probabilities are estimated with the maximum likelihood estimate $\frac{n_i}{n}$, where n_i is the number of labelled vectors in the ith cluster and n is the total number of labelled vectors. According to these estimated probabilities the cluster with the highest probability is considered to belong to the botnet communication. For the sample we consider the cluster center and calculate an ε threshold value. This value has the largest radius such that the sphere around the center of this cluster with such radius is disjoint from all of the spheres around the other cluster centers with the same radius. So the sample will be the (C, ε) pair.

Multiple Sources If the NetFlow log contains only one botnet trace, it can be assumed that the same cluster will contain the most of the labelled flows with high probability. If it is not the case, then several clusters according to each source with relatively high estimated probability will be available. In that case all clusters over a p_0 (a priori chosen) probability can be selected, and a sample can be constructed for all of them. Each of them will belong to a different botnet.

4.4 Sample Redistribution

Sample redistribution can be performed via distributors in the way that agents connect periodically to make regular updates or distributors can push new sam-

ples to the agents. Distributors could share the samples via HTTP protocol using e.g. a web service or any standardized way. However, sample distribution is an important part of the system, there are standardized ways to perform this operation. This is the reason why it is not the key issue of the paper. In addition, we note that further investigation is needed to find the best solution for the problem.

4.5 C&C Channel Recognition

After agents have downloaded the samples they can apply the C&C channel recognition procedure. First of all, all agents have to aggregate their flows to present a similar data structure to the aggregated sample. It not just decreases the size of the data set, but offers relatively fast search and preserves anonymity as well.

To select all botnet related flows from the agent's flow set the clustering method discussed in Section 4.3. can be applied. Let x be a vector from the agents aggregated flow set. Then the following steps are required:

1. Calculate the distances of the feature vector x from the centers in the samples: $d_1 = d(x, C_1), \ldots, d_r = d(x, C_r)$
2. Select an index i, if there exists such that $d_i < \varepsilon_i$ (Note that if such an index exists, then it will be unique.)

We can assume that this vector belongs to the corresponding cluster and so to the botnet communication. It's because if this vector is added to the training data set, then after the next iteration of K-Means the vector will be an element of this cluster.

5 Experimental Evaluation

We have implemented the aggregation algorithm in native C (a code of 4000 lines in total). We tested it on NetFlow logs collected from the network of the Department of Telecommunications and Media Informatics at the Budapest University of Technology and Economics (BME). The logs each 10 minutes long were collected in the time period 17-19. April 2008. The total size of the data was 5.059 GB, about 100 million flows.

First, the aggregation was made for each log separately. For all the 387 of 10-minute logs the compression ratio of the algorithm was between 0.3 and 0.35. So we can say that it reduced the size of the data set by it's 2/3. The average single-threaded running time for one 10-minute log was about 10 seconds. Figure 2 shows the size of the original and the aggregated data sets in the time period of one day.

Next, we did the aggregation for longer time intervals. Clearly, it improved the compression ratio, because in a longer time interval more flows were grouped together. Figure 3. a. shows, how this ratio is improved by increasing the length

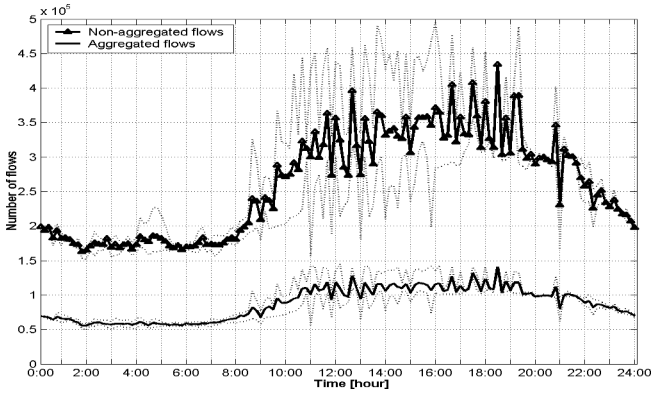

Fig. 2. Sizes of the original and the aggregated data sets over one day

of the time interval observed. In contrary to the compression ratio, when the length of the observed time interval was increased, the running time increased as well. Figure 3. b. shows this phenomenon.

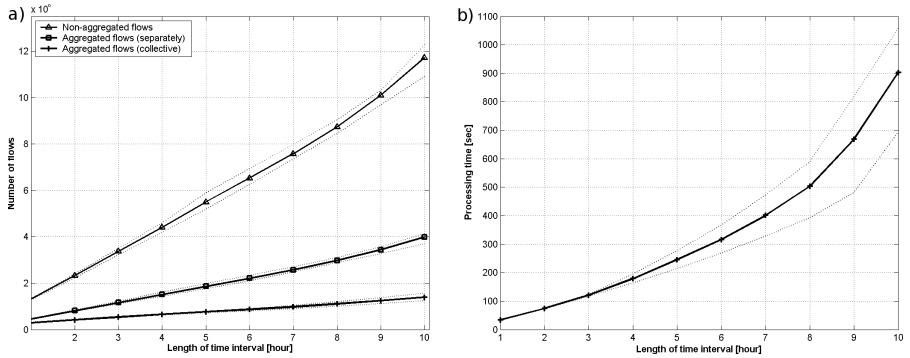

Fig. 3. The compression ratio over different time intervals

According to the results we can state that the compression ratio will be growing if longer time interval is selected. When the compression ratio increases, the processing time increases as well. Beyond a certain point this kind of delay can not be tolerated any longer and at this point a threshold can be stated. This threshold takes place, where the growing of the processing time turns from linear to exponential. In addition, it depends on the resources of the running environment. Our results was generated by a desktop computer with a Pentium Core2 3.0 GHz processor and 2 GB of RAM and the threshold was at approximately 7 hours. However, recent study of the Storm botnet [16] evinced that bots are short-lived: it takes them just over 4 minutes after boot-up until receiving a

control message, most of them remain in operation only for a little under 4 hours. Thus, approximately seven-hour aggregation interval is sufficient for the detection.

Further, we implemented a test version of the algorithm for the C&C channel recognition in Matlab. The data set came from a laboratory simulation. We simulated two virtual LAN networks, installed with Windows XP operating systems, and we infected with a botnet client. The two subnets were connected to a gateway, that was in connection with an IRC server, a botnet controller and a victim as well. Besides the legal traffic generated by the computers of the subnets, such as FTP, HTTP and e-mail, we simulated an attack against the victim directed by the botnet controller. The sample for the C&C channel was generated from the NetFlow log of the first network. With this sample we could achieve a 100% result in the second network that is we have found all flows belonging to the C&C channel communication. (The verification was due to a port analysis, the botnets C&C channel communication was done on port 6667)

However, this result is quite promising, more testing on more realistic data sets are required.

6 Conclusion and Future Work

In this paper, we have shown an architecture for distributed malware detection. After the base system we presented solution proposals on all emerging piece of the problem. In addition, we proposed two algorithms: one for the reduction of the huge amount of network statistical data and another for the detection with the help of samples which was generated from the attacks. We demonstrated the strength of the algorithms: i) the aggregation method reduced the NetFlow entries to one third in practice ii) the detection algorithm was able to find botnet clients using the aggregated samples. We note that these samples provide anonymity in that sense they do not contain any kind of valid IP information. Consequently, each and every user can be sure that their network traffic is not revealed totally. Hereby, spying usage of the system is not possible. As a result, there is no need to establish mutual and unconditional trust among all participants. This property of the architecture can facilitate to make extensive use of the system.

Our future work includes adding a more sophisticated algorithm for the botnet C&C channel recognition and improving further the speed of the algorithms. Although, gathering of NetFlow data from real networks, where the traffics were reliably identified, is a quite complicated task, it is necessary to validate the results of the detection and prevention.

References

1. Erman, J., Mahanti, A., Arlitt, M., Cohen, I., Williamson, C.: Offline/Realtime Traffic Classification Using Semi-Supervised Learning, Performance Evaluation. Vol. 64. No. 9-12, pp. 1194-1213. (2007)

2. Duda, R. O., Hart, P. E., Stork, D. G.: Pattern Classification. Wiley, second edition (2001)
3. Erman, J., Arlitt, M., Mahanti, A.: Traffic Classification using Clustering Algorithms. In: SIGCOMM'06 MineNet Workshop, Pisa, Italy (2006)
4. McGregor, A., Hall, M., Lorier, P., Brunskill, J.: Flow Clustering Using Machine Learning Techniques. In PAM 2004, Antibes Juan-les-Pins, France (2004)
5. Zander, S., Nguyen, T., Armitage, G.: Automated Traffic Classification and Application Identification using Machine Learning. In: LCN'05, Sydney, Australia (2005)
6. Yau, D.K.Y., Lui, J.C.S., Liang, F., Yam, Y.: Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles, In: Networking IEEE/ACM Transactions on, vol.13, pp. 29-42. (2005)
7. Peng, T., Leckie, C., Ramamohanarao, K.: Protection from distributed denial of service attacks using history-based IP filtering, Communications, In: ICC '03. IEEE International Conference on, vol.1, pp. 482-486. (2003)
8. Kargl, F., Maier, J., Weber, M.: Protecting web servers from distributed denial of service attacks, International World Wide Web Conference, pp. 514-524. ACM, Hong Kong (2001)
9. Keromytis, A. D., Misra, V., Rubenstein, D.: SOS: Secure Overlay Services, In: ACM SIGCOMM, pp. 61-72., Pittsburgh, USA (2002)
10. Karasaridis, A., Rexroad, B., Hoeflin, D.: Wide-scale botnet detection and characterization, In: HotBots'07, 1st conference on 1st Workshop on Hot Topics in Understanding Botnets, pp. 7-7. USENIX Association, Cambridge, USA (2007)
11. Cranor, C., Johnson, T., Spataschek, O., Shkapenyuk, V.: Gigascope: a stream database for network applications, In: ACM SIGMOD International Conference on Management of data, pp. 647-651. (2003)
12. Fraleigh, C., Diot, C., Lyles, B., Moon, S., Owezarski, P., Papagiannaki, D., Tobagi, F.: Design and Deployment of a Passive Monitoring Infrastructure, In: S. Palazzo (ed.) IWDC 2001, LNCS, vol. 2170, pp. 556-575. Springer, Heidelberg (2001)
13. Iannaccone, G., Diot, C., McAuley, D., Moore, A., Pratt, I., Rizzo, L.: The CoMo White Paper, Technical Report IRC-TR-04-17, Intel Research (2004)
14. Antoniades, D., Polychronakis, M., Papadogiannakis, A., Trimintzios, P., Ubik, S., Smotlacha, V., Oslebo, A., Markatos, E. P.: LOBSTER: A European Platform for Passive Network Traffic Monitoring, In: 4th International Conference on TRIDENTCOM, pp. 1-10. Innsbruck, Austria (2008)
15. Bin, L., Lin, C., Jian, Q., Jianping, H., Ungsunan, P. D.: A NetFlow based flow analysis and monitoring system in enterprise networks, Computer Networks vol. 52 issue 5, pp. 1074-1092 (2008)
16. Kreibich, C., Kanich, C., Levchenko, K., Enright, B., Voelker, G. M., Paxson, V., Savage, S.: On the Spam Campaign Trail, In: 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats (2008)
17. Sekar, V., Duffield, N., Spatscheck, O., Van Der Merwe, J., Zhang, H.: LADS: Large-scale Automated DDoS detection System, In: USENIX ATC, pp. 171-184. (2006)
18. Gu, G., Zhang, J., Lee., W.: BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic, In: 15th Annual Network and Distributed System Security Symposium (NDSS) (2008)
19. Choi, H., Lee, H., Lee, H, Kim H.: Botnet Detection by Monitoring Group Activities in DNS Traffic, In: 7th IEEE International Conference on Computer and Information Technology (CIT), pp. 715-720., Aizu-Wakamatsu, Japan (2007)
20. Cisco Systems NetFlow Services Export Version 9, RFC 3954, http://www.ietf.org/rfc/rfc3954.txt