UNSUPERVISED DETECTION OF MALWARE IN PERSISTENT WEB TRAFFIC

Jan Kohout[⋆†] *Tomáš Pevný*[⋆†]

[⋆]Cisco Systems
[†]Faculty of Electrical Engineering, Czech Technical University in Prague

ABSTRACT

Persistent network communication can be found in many instances of malware. In this paper, we analyse the possibility of leveraging low variability of persistent malware communication for its detection. We propose a new method for capturing statistical fingerprints of connections and employ outlier detection to identify the malicious ones. Emphasis is put on using minimal information possible to make our method very lightweight and easy to deploy. Anomaly detection is commonly used in network security, yet to our best knowledge, there are not many works focusing on the persistent communication itself, without making further assumptions about its purpose.

Index Terms**—** malware, outlier detection, persistent communication

1. INTRODUCTION

Bots differ from other types of malware by having a command and control (C&C) channel through which the botmaster controls the bot. The C&C channel can be implemented by using different network paradigms (p2p networks, central or fluxing servers, etc.) and protocols (custom, HTTP, plain TCP or UDP, etc.), but there has to be a communication path between C&C server and bots. The channel is maintained through the life of the bot, and once it is lost, the control over the bot is lost, too. This definition implies that the channel needs to be persistent in the sense that the bot receives the commands repeatedly in time. However, bots are not the only type of malware which produces persistent communication. Malware can repeatedly check connection to the Internet, perform click fraud, or download advertisements all of which can manifest as a persistent connection.

On the other hand, user's legitimate activities produce persistent connections as well. Repeated visits of a news portal, e-mail account, social network or application checks for updates being examples. All of these make the communication within a typical corporate network very heterogeneous and therefore provides good conditions for malware to hide its activities.

Many companies block traffic other than HTTP to mitigate unwanted activity, but malware has caught up and uses HTTP as well which stimulates the need for a technique that separates malicious HTTP traffic from benign. The goal of this paper is to utilize outlier detection for identification of infected computers within a network. The proposed method relies on the fact that statistical fingerprint of malicious persistent connection is recognizably different from other, legitimate connections. The method is very lightweight, as it does not inspect traffic content and it uses only information about transferred bytes and timings of individual requests. This allows to effectively scan communication in large corporate networks.

Although outlier detection methods in network security have been already widely studied, this work differs from prior art by (i) focusing on persistent communication only, without further assumptions about its purpose and (ii) by proposing a novel representation which well describes recurring communication and well captures its important characteristics.

The method is evaluated on data from web proxy logs (HTTP traffic), but as we do not use any text information from the HTTP headers, this method can be easily used, for example, on transport layer data.

2. RELATED WORK

The closest works can be found in the area of machine learning for detection of C&C channels, where the assumption about similarity of bots communication within one botnet is frequently made [1]. BotMiner [2] is a very complex system designed for detecting C&C by correlating communication patterns of network hosts. For successful detection it needs multiple hosts infected with the same botnet inside the network. BotSniffer [3] clusters the communication by using features derived from the content of packets and as previous works it needs more hosts to be infected. Ref. [4] based their detector on the assumption that access patterns of human client will have higher variability than malware client. As shown below, we made the similar observation in malware's persistent connections.

The definition of persistent connection used in this work is based on [5]. Ref. [6] uses it to train a supervised classifier to identify botnet's persistent communication. The similar approach on a flow level is used in [7].

Our representation of network communication bears some similarity to Ref. [8], which uses sizes and inter-arrival times of packets from the IP level flows for application protocol fingerprinting and classification, but our representation and objects of modelling (persistent connections vs. flows) are different.

3. THE ALGORITHM

The algorithm's description is broken into two parts: first, fingerprints of persistent connections are defined; second, the choice of outlier detection algorithm is justified.

Before diving into the actual description, we define the terms used hereafter. *Web request* is one HTTP request message [9] for a particular resource. *Connection* is a set of web requests that share the same local and remote endpoints. In this work, the endpoints are identified by local user's username (local endpoint) and target second level domain (remote endpoint). *Persistent connection* is a connection in which requests occur repeatedly over a certain time period, e.g., one day. This definition is slightly fuzzy allowing various mechanisms selecting persistent connections. Below we use modification [10] of the original approach [5] for better memory efficiency.

Furthermore, we use a limit K_u to filter out persistent connections to remote endpoints (domains) used by more than K_u local users. The reason is that these domains are expected to be popular services and therefore not malicious (e.g., Google). For such services we recommend to build a special model to identify outliers within them, which can be an indication of service misuse. Nevertheless, in our database of malware samples we did not have such sample and therefore could not verify this.

3.1. Fingerprints of persistent connections

A fingerprint characterizing statistical properties of a persistent connection is extracted from all of its web requests from a certain period of time. Our experiments used one day period. Fingerprints use only following four quantities from each web request:

1. **bytes sent** r_{up} from the client to the server,
2. **bytes received** r_{down} by the client from the server,
3. **duration**: r_{td} (in milliseconds) of handling the request,
4. **inter-arrival time** r_{ti} (in seconds) elapsed between start of the current and previous request.

Thus, for purpose of this paper, the request r can be reduced to a 4-tuple $r = (r_{\mathrm{up}}, r_{\mathrm{down}}, r_{\mathrm{td}}, r_{\mathrm{ti}})$ and the terms tuple and web request can be used interchangeably.

The fingerprint is a joint histogram of the four quantities estimated from all tuples from a given persistent connection.

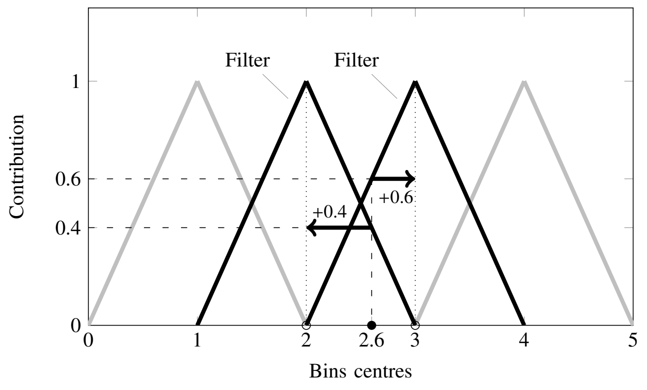

Fig. 1. Example of updating a one-dimensional *soft* histogram with value 2.6. It contributes with 0.4 to the bin centred in 2 and with 0.6 to the bin centred in 3. Filters that influence the contribution are highlighted.

The advantage of the joint histogram is that it captures dependencies between the quantities. Its disadvantage is that in order to capture the dependencies accurately, the number of bins can be high. In our implementation it was $11^4 = 14641$. We argue that the dimension of the fingerprint is not a problem because as shown below, the histogram will be typically very sparse, especially for persistent connections made by malware. Moreover, the field of text document analysis [11] works with high dimensional yet sparse data as well. In order to narrow the range of modelled values and equalize the variances on low and high values, all values in tuples are transformed to logarithmic scale before calculating the joint histogram.

The common approach to build a histogram, which is here called *hard* histogram, is to use each sample to update only the one bin into which the given sample falls. For example, i^{th} bin with bounds $[b_i, b_{i+1})$ is updated by 1 irrespectively if the sample is close to b_i or b_{i+1}. This strict quantization makes values of histogram bins sensitive to small variations in the data.

To decrease this sensitivity, we borrow an idea from an array of overlapping filter banks frequently used in digital signal processing [12]. The update procedure in so-called *soft* histogram divides the contribution of a sample to two neighbouring bins proportionally to their distance. The situation for one-dimensional case is illustrated in Figure 1. When the soft histogram is updated by a sample u, two nearest bins with centres in $\lfloor u \rfloor$ and $\lfloor u \rfloor + 1$ are updated by $1 - (u - \lfloor u \rfloor)$ and $u - \lfloor u \rfloor$, respectively.

To capture joint frequencies of multiple quantities, the *soft* histogram is naturally extended to multiple dimensions. Without loss of generalization it is assumed that m-dimensional *soft* histogram (in this work, we use $m = 4$) has bins centred at integer lattice points $[b_1, ..., b_m] \in \{0, ..., n\}^m$, where n acts as an upper bound on values to be inserted to histogram.

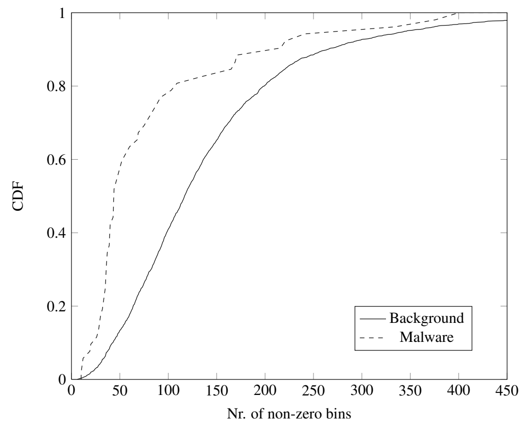

Fig. 2. Cumulative distribution functions (CDF) of number of non-zero bins in *soft* histograms of background (i.e., assumed to be mostly benign) and malicious persistent connections.

Based on the values that are commonly observed in web requests we used $n = 10$. Updating *soft* histogram with a tuple $(u_1, u_2, \ldots, u_m)$ means first calculating indices l_i and contributions v_i to "left" bins as

$$l_i = \lfloor u_i \rfloor, \quad v_i = 1 - (u_i - l_i),$$

and then updating all bins centred in vertices

$$\{(l_1 + i_1, \ldots, l_m + i_m) | (i_1, \ldots, i_m) \in \{0, 1\}^m\}$$

with values $\prod_{j=1}^{m} v_j^{1-i_j} (1 - v_j)^{i_j}$.

The motivation behind this representation is the observation that malicious persistent connections tend to have lower variability than that of the regular user. This is illustrated in Figure 2 which shows the cumulative distribution of non-zero bins in histograms of malicious and background persistent connections. By background connections we mean all connections revealed in traffic from three different companies in which we assume that the vast majority of traffic is benign (for details about the data used for this measurement see Section 4.1). The curve for malware is steeper which means that the number of non-zero bins in histograms is lower and the connections are more regular.

The fingerprint construction is finalized by mapping the *soft* histogram to a $(n + 1)^m$-dimensional column vector.

3.2. Outlier detection

Fingerprints of persistent connections are numerical vectors of fixed length on which most of outlier detection algorithms can be readily applied [13]. Since malware's fingerprints can form small clusters, we promote to use OutRank [14] algorithm, because it should be robust against cases when outliers form small clusters. The formation of small clusters can be caused, for example, by multiple infections of the same malware family with similar fingerprints or by a single malware instance maintaining several persistent connections with the same purpose (to improve robustness). The experimental comparison of OutRank and k-nearest neighbours based detector [15], which is an asymptotically consistent density level estimator, supports the choice.

The choice of distance measure between two fingerprints was inspired by measuring similarity of vectorized text documents, because in analogy to them, fingerprints are sparse. We use cosine distance defined as

$$d(x_1, x_2) = 1 - \frac{x_1^T \cdot x_2}{||x_1|| \cdot ||x_2||},$$

where x_1, x_2 are fingerprints of two persistent connections. Note that since all items of the fingerprint are non-negative, the cosine distance is always between zero and one.

Finally, the term frequency - inverse document frequency (TF-IDF) weighting is applied on the fingerprints, which puts more emphasis on less frequent non-zero items. It multiplies each item b_i in all fingerprints by $\log\left(\frac{N+1}{N_{b_i}+1}\right)$, where N is the total number of fingerprints and N_{b_i} is the number of fingerprints with non-zero value of the item b_i. TF-IDF weighting is very common in fields utilizing sparse representations, such as text documents analysis [16] and computer vision [17].

4. EXPERIMENTAL EVALUATION

4.1. Experiments setup

The ability of *soft* histograms representation to separate malicious and benign traffic by means of outlier detection was evaluated in the following way: We compared (i) the OutRank and k-NN algorithms and (ii) fingerprints based on *soft* and *hard* histograms. Combination of two detection algorithms with two versions of histograms lead to four different detectors.

The evaluation was performed on web proxy logs from one day of traffic in three distinct companies (referred as A, B and C) acquired during 2014. Based on the total volume of traffic we set the limit $K_u = 10$. The total number of persistent connections used for evaluation was 1732 in company A, 681 in company B, and finally 836 in company C.

Because reliable manual labelling of all 3249 persistent connections in the data sets is difficult and subjective to human judgement, we treated all these persistent connections as legitimate. Malicious web requests produced by malware were obtained from 14 different malware binaries executed in a controlled environment of malware laboratory. These binaries included variants of malware labelled by AV engines as ZeroAccess, Kelihos, ZBot, Asprox, Win32.Injector, Wapomi, Somoto, SS.Worm-generic and Downloader.UFN. From them we isolated 50 persistent connections. Infection of a user by a malware sample was simulated by adding all

Data set	*hard* histogram		*soft* histogram	
	k-NN	OutRank	k-NN	OutRank
A	0.855	0.892	0.862	**0.942**
B	0.889	0.896	0.912	**0.935**
C	0.871	0.867	0.880	**0.933**

Table 1. Average AUC values (computed over the 14 malware samples) for all combinations of hard and *soft* histograms, and OutRank and k-NN outlier detection algorithms. Higher is better with one being maximum, best results on each company are bold-faced.

traffic from the malware sample's persistent connections to the background data. Because none of the domains utilized in malicious persistent connections were visited by any user in the background data, there was no risk of collision.

The accuracy of detection was measured by the area under the ROC (receiver operating characteristic) curve (AUC), which is a common measure in cases when detection threshold cannot be determined beforehand. AUC was calculated for every combination of background data set (out of 3) and each malware sample (out of 14), which lead to 3×14 evaluations of every detector.

4.2. Experimental results

Average AUCs over all 14 malware samples for combinations of outlier detection algorithm and histogram (fingerprint) types are presented in Table 1. We can see that in all cases detectors employing fingerprints based on *soft* histograms performed better irrespective to used outlier detection algorithm. Similarly, all detectors with OutRank outlier detection algorithm were always better than those with the k-NN. This means that the proposed fingerprints based on *soft* histograms with OutRank algorithm performed the best on our data with average AUC being 0.936.

To get further insight into the detector we analysed 10 persistent connections with the highest outlier scores assigned by the best detector from each data set. Deeper analysis of these most anomalous connections revealed 3 persistent connections in data set A, 1 connection in data set B and 2 connections in data set C that were related to malicious activity. The most frequently observed type of legitimate but anomalous connections were those trying to reach a web server which was either unavailable (returning response code 50X) or the requested resource could not be accessed.

In Section 3.2, we have argued that OutRank algorithm should be more robust with respect to small clusters of outliers which we can expect to be present in real data. Figure 3 shows average AUC of OutRank and k-NN algorithms with fingerprints based on *soft* histograms for varying number of malware samples inserted into the background data sets. The average is calculated over all three background data sets and random selections of malware. We can observe that the AUC of the k-NN detector drops more rapidly than that of OutRank as the number of infected users increases. This means that in our settings the OutRank is indeed more robust against multiple infections.

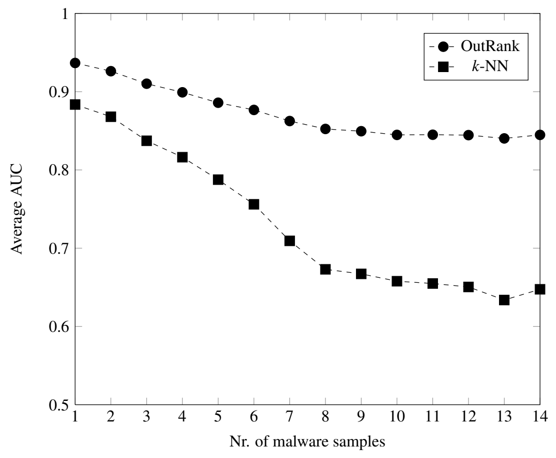

Fig. 3. Average values of AUC for OutRank and k-NN outlier detection algorithms with fingerprints based on *soft* histograms for varying number of malware samples inserted into the background data sets. The average is calculated over all three background data sets and random selections of malware.

5. CONCLUSION

This paper proposed a statistical description of a distribution of sizes, timings and durations of requests in persistent web connections. The description is based on a joint histogram of modelled quantities by so-called *soft* histogram, which is smoothed version of the traditional histogram with crisp bin bounds. We demonstrated that malicious persistent connections are more regular than those of legitimate users. Therefore, they manifest themselves as outliers.

The advantage of the proposed description was demonstrated by creating detectors utilizing off-the-shelf outlier detection methods. The detectors were evaluated by identifying malicious persistent connections inserted into the traffic from 3 different companies. On average we achieved the best accuracy of AUC 0.936.

The proposed method has two key advantages: it is lightweight and it does not inspect the content, thus it can be used for encrypted traffic. It is well suited as a filter before more sophisticated yet computationally more expensive methods.

6. ACKNOWLEDGEMENT

The work of T. Pevný was also supported by the Grant Agency of Czech Republic under the project P103/12/P514.

7. REFERENCES

[1] H. R. Zeidanloo and S. Rouhani, *Botnet Detection by Monitoring Common Network Behaviors: Botnet Detection by Monitoring Similar Communication Patterns*, LAP Lambert Academic Publishing, 2012.

[2] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th Conference on Security Symposium*, 2008.

[3] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, 2008.

[4] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: Detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*, 2012.

[5] F. Giroire, J. Chandrashekar, N. Taft, E. Schooler, and D. Papagiannaki, "Exploiting temporal persistence to detect covert botnet channels," in *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, 2009.

[6] G. Fedynyshyn, M. C. Chuah, and G. Tan, "Detection and classification of different botnet C&C channels," in *Proceedings of the 8th International Conference on Autonomic and Trusted Computing*, 2011.

[7] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, 2013.

[8] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *SIGCOMM Comput. Commun. Rev.*, 2007.

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," 1999, [Online].

[10] J. Jusko, M. Rehák, and T. Pevný, "A memory efficient privacy preserving representation of connection graphs," in *Proceedings of the 1st International Workshop on Agents and CyberSecurity*, 2014.

[11] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, 1975.

[12] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications*, Prentice-Hall, Inc., 1996.

[13] C. C. Aggarwal, *Outlier Analysis*, Springer New York, 2013.

[14] H. D. K. Moonesinghe and P.-N. Tan, "OutRank: a graph-based outlier detection framework using random walk," *International Journal on Artificial Intelligence Tools*, 2008.

[15] K. Sricharan and A. O. Hero III, "Efficient anomaly detection using bipartite k-NN graphs," in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, 2011.

[16] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[17] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-hash and tf-idf weighting.," in *BMVC*, 2008.