Detecting Malicious Sessions Through Traffic Fingerprinting Using Hidden Markov Models

Sami Zhioua[1](✉), Adnene Ben Jabeur[2], Mahjoub Langar[3], and Wael Ilahi[3]

[1] King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
zhioua@kfupm.edu.sa
[2] École Polytechnique, La Marsa, Tunisia
adnenebj@gmail.com
[3] École Nationale des Ingénieurs de Tunis, Tunis, Tunisia
mahjoub.langar@enit.rnu.tn, waelilahi@gmail.com

Abstract. Almost any malware attack involves data communication between the infected host and the attacker host/server allowing the latter to remotely control the infected host. The remote control is achieved through opening different types of sessions such as remote desktop, webcam video streaming, file transfer, etc. In this paper, we present a traffic analysis based malware detection technique using Hidden Markov Model (HMM). The main contribution is that the proposed system does not only detect malware infections but also identifies with precision the type of malicious session opened by the attacker. The empirical analysis shows that the proposed detection system has a stable identification precision of 90 % and that it allows to identify between 40 % and 75 % of all malicious sessions in typical network traffic.

Keywords: Malware detection · Hidden Markov Model (HMM) · Malicious sessions · Traffic analysis

1 Introduction

Malware[1] is a significant threat and root cause for many security problems on the Internet, such as spam, distributed denial of service, data theft, or click fraud [1]. Malware attacks are getting more and more sophisticated. The recent campaign of malware-based attacks targeting the Middle East is a manifestation of this trend. Several organizations in the Middle East, in particular in the energy industry, reported infections with sophisticated malware in the few last years [2–5].

Most malware consist of (at least) two fundamental components: a client agent, who runs on infected hosts, and a control server application, widely known as Command and Control (C&C) server. Almost any malware-based attack involves a data communication between the infected host and the attacker. This includes sending control commands, stealing confidential files, opening remote control sessions (simple shell, remote desktop connection, keylogger session, webcam video communication session, etc.).

[1] Malware and Bot will be used interchangeably.

J. Tian et al. (Eds.): SecureComm 2014, Part I, LNICST 152, pp. 623–631, 2015.
DOI: 10.1007/978-3-319-23829-6_47

The proposed work falls into the network-based malware detection techniques. It deviates from most of existing work in the literature by not relying on the payload/body of traffic packets. The approach is based on general characteristics of packets such as size, direction, and delays between successive packets. The approach is inspired by a large body of work called traffic fingerprinting [6–11] used to attack anonymity protocols, in particular Tor [12]. A common type of traffic fingerprinting called website fingerprinting whose aim is to detect websites visited by a victim, showed recently very promising results (precision of 90 % [11]).

While existing work in the literature focuses only on identifying malware infections, the main contribution of this paper is to push the network-based malware detection technique further to recognize with precision the type of communication being carried out between the infected host and the C&C server (Remote desktop connection, camera session, keylogging session, etc.). Identifying the type of malicious session with precision has several applications, in particular in forensics investigations. To the best of our knowledge, this is the first work in the literature to tackle this problem.

2 Malicious Sessions

Attackers use different types of malware to infect home and business users having access to internet. The most common types of malware include trojans, spyware, and worms. Infected machines are typically part of a large network of owned machines called botnets. The attacker typically uses a Command and Control server (C&C) to remotely control the zombie machines. The remote control is typically done through a feature called Remote Administration Tool (RAT). A RAT provides the possibility to open several types of malicious sessions with an infected machine: initial connection, remote shell session, remote desktop session, keylogging session,webcam video streaming session, audio streaming session, chat session, upload/Download file session, and screenshot session.

3 Overview of the Detection System

The proposed malware detection system is based on network-level signatures. Figure 1 shows an overview of the detection system. The procedure starts by collecting a set of packet traces corresponding to each malware/malicious session. This can be achieved by using a host machine as a honeypot[2]. This machine is configured to attract malware attacks by automatically opening suspicious files, using unpatched versions of software, in particular, web browsers, visiting malicious websites, etc. Once an infection occurs, the next step is to keep observing/logging the network traffic so that to collect several instances of typical malware sessions (e.g. Download/Upload of files, screen snapshot transfer, remote shell, etc.). The set of packet trace instances/samples are then used to learn a network signature model for each type of session. In order to make the approach applicable even if the malware uses a form of network encryption, network signatures are only represented in terms of general characteristics of the packets, in particular, the

[2] For large scale systems, the honeypot machine can be replaced by a full honeyNet network.

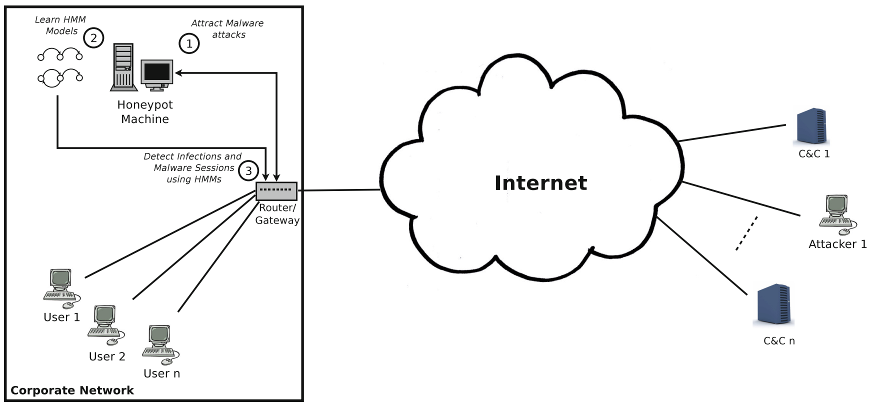

Fig. 1. Overview of the malware detection system using one honeypot and HMM models

direction of the packet (C&C to infected host or infected host to C&C), the size of the packet, and time delay between every successive packets, namely, the Inter Packet Time (IPT). For instance, Fig. 2 shows the packet trace corresponding to the initial handshake between a freshly infected host and the C&C of a famous Remote Administration Tool (RAT) malware called proRat [13]. The trace is composed of 8 packets all of which are simple TCP segments. Given several samples of such session, the next step is to generate a signature model that captures the pattern of the sequence of sizes and IPTs. The model used in this work is a Hidden Markov Model (HMM). The outcome of the learning phase is a database of HMM models, each model capturing the pattern of an observed malicious session. The database of HMMs can then be used to analyze the network traffic in order to detect new infections with the same malware and identify exactly the type of malicious sessions being opened. The detection system can be deployed according to different scenarios and at different locations in the network (e.g. Gateway, Router, Intrusion Detection System, Proxy, etc.).

4 HMM Based Signatures

Packet traces corresponding to malicious malware sessions can very well be represented using Hidden Markov Models (HMMs). HMM is a statistical Markov model especially known for its application in temporal pattern recognition. HMM has been mainly used for speech recognition [14] and bioinformatics [15].

Definition 1. A HMM is a tuple $(\mathcal{S}, T, O, Q, \pi)$ where

- $\mathcal{S}$ is a set of N states $\{s_1, s_2, \ldots, s_N\}$.
- $T : \mathcal{S} \rightarrow \Pi(\mathcal{S})$ is a state transition function which maps each state $\mathcal{S}$ to a probability distribution over $\mathcal{S}$. $T_{s_i \rightarrow s_j}$ denotes the probability of transition from s_i to s_j.
- O is a set of M observations $\{o_1, o_2, \ldots, o_M\}$.
- $Q : \mathcal{S} \rightarrow \Pi(O)$ is an observation function. Q_s^o denotes the probability of observing o while in state s.

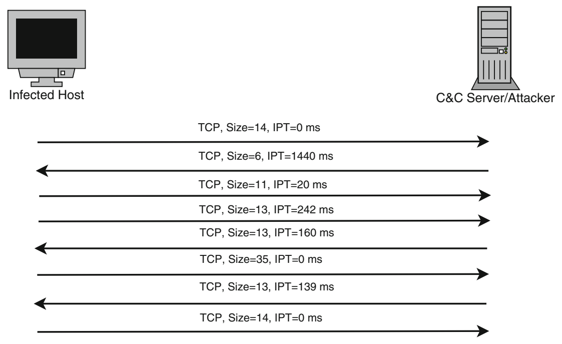

Fig. 2. Packets Trace of the initial handshake of proRAT malware attack

– π is the initial state distribution, where $\pi(s)$ denotes the probability of s being the initial state.

The sequence of exchanged packets between the infected host and the C&C server in every malicious session can be very well represented using an HMM. For example, the packet trace in Fig. 2 representing the initial handshake between the infected host and the C&C can be captured using an HMM with 8 states each corresponding to a packet in the trace. The observation in every state can be the IPT value. Since the space of possible IPT values (observations) is continuous, we represent malicious sessions with a Continuous HMM (CHMM) where every state defines a continuous probability distribution, in particular a Gaussian (Normal) distribution, over the space of observations. The HMM corresponding to the trace in Fig. 2 can be defined as follows:

Definition 2. The HMM corresponding to the packet trace in Fig. 2 is a tuple $(\mathcal{S}, T, \mathcal{O}, \mathcal{Q}, \pi)$ such that:

– $\mathcal{S} = \{1,2,3,4,5,6,7,8\}$
– $\mathcal{O} = [0,\infty)$
– $\mathcal{Q} : \mathcal{S} \to \mathcal{N}(\mu, \sigma^2)$
– $\pi = [1,0,0,0,0,0,0,0]$

$$T = \begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0 \end{pmatrix}$$

Figure 3 shows the graphical representation of the HMM of Definition 2.

One can note that the states of the HMM as defined in Definition 2 are not hidden since state 1 is always the first to be visited and the transition function is deterministic. Hence, one can argue that we could use a simpler Markov model where states are not hidden. The reasons to choose the HMM model are two fold. First, we need a model where different observations can be emitted from a single state. Second, HMMs come

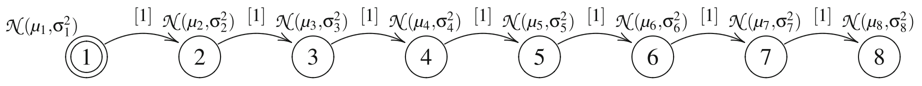
$\mathcal{N}(\mu_1,\sigma_1^2)$
[1]
$\mathcal{N}(\mu_2,\sigma_2^2)$
[1]
$\mathcal{N}(\mu_3,\sigma_3^2)$
[1]
$\mathcal{N}(\mu_4,\sigma_4^2)$
[1]
$\mathcal{N}(\mu_5,\sigma_5^2)$
[1]
$\mathcal{N}(\mu_6,\sigma_6^2)$
[1]
$\mathcal{N}(\mu_7,\sigma_7^2)$
[1]
$\mathcal{N}(\mu_8,\sigma_8^2)$
1
2
3
4
5
6
7
8

Fig. 3. Graphical representation of HMM of Definition 2

with a well established theory for learning the parameters and computing the probability of acceptance of a given observations sequence.

All the parameters of the HMM are known (Definition 2) except the parameters of the Gaussian distributions in every state, namely, the values of the mean (μ) and variance (σ^2). Typically, the observation functions of a HMM are learned based on a training set of observation sequences. Since the aim of the proposed HMMs is to model network signatures of malicious sessions, the training sequences should correspond to valid previously observed malicious sessions.

4.1 Learning the HMM Parameters

Learning HMM parameters based on a set of sequences is one of the basic problems with HMMs. In his seminal work [14], Rabiner shows how an HMM is trained given a single or multiple observation sequences. The main idea consists in starting from any HMM model then keep adjusting the parameters to maximize the probability to accept the observation sequences. Computing the probability of accepting an observation sequence by an HMM is another basic problem with HMMs.

Hence, what is required to learn the HMM parameters is a set of packet traces (samples) corresponding to each type of malicious malware sessions. These packet traces are given as input to the HMM learning algorithm which returns as output the HMM model. It is important to mention that very often, the HMM learning algorithm does not consider all samples in the training. An initial filtering step is carried out to rule out "noisy" samples. A noisy sample is a packet trace where the packet sizes do not match the packet sizes of the "majority" of the other samples. For instance, if most of the samples have the following sequence as packet sizes: $[14, -6, 11, 13, -13, 35, -13, 14]$ (the same as Fig. 2), and one sample of the same set has a sequence $[14, 11, 11, -6, 13, -13, -13]$. The latter is considered noisy and is not used for the training of the HMM model. Notice that negative values are used to distinguish between packets in different directions: positive value designates a packet going from the infected host to the C&C while a negative value designates a packet going in the opposite direction (C&C to infected host).

5 Implementation and Experimental Settings

Given a set of packet traces for each type of session, the HMM learning algorithm generates a set of HMMs. These HMM models are stored in a database. The detection of malware infections and the exact type of malicious sessions is achieved by scanning network traffic of all hosts in the private network and trying to identify packet traces accepted by some HMM models. This process is done in two steps: packet sizes matching and HMM acceptance. The detection system analyzes the traffic by maintaining a sliding window on the previously observed packets. The length of the window is equal to h_{max} representing the number of states of the longest HMM in the database. The aim of the HMM acceptance step is to make sure that the IPT values of the current packets in the sliding window exhibit a pattern very similar to the pattern modeled by any

HMM model in the HMM database. This is achieved looping over all HMMs in the database and computing the acceptance probability of the sequence of IPTs. A probability larger than a fixed threshold λ means that the last observed packets correspond to the malicious session associated with the current HMM.

6 Empirical Analysis

In order to assess the accuracy of our HMM based approach to identify malicious sessions, we used a set of commonly used malware RATs (Remote Administration Tools). Each RAT supports a set of sessions. Table 1 shows the list of RATs with the sessions considered in the analysis. An empty cell in the table indicates that the corresponding RAT-session combination was not considered in the analysis. The reason is that some combinations did not work when the infected host is running on a virtual machine[3].

Table 1. List of Remote Administration Tools (RATs) with the type of sessions considered in the empirical analysis.

RAT Name	Initial Infection	Remote Desktop	Chat	Keylogger	Upload File	Camera Streaming	Audio Streaming	Screenshot
Beast 2.07	✓	✓	✓	✓	✓			
Bifrost 1.2.1	✓	✓			✓	✓		✓
Blacknix 1.1	✓	✓		✓	✓	✓	✓	✓
jRAT 3.2.4	✓	✓	✓		✓		✓	✓
njRAT 0.7	✓	✓	✓	✓	✓	✓	✓	
Turkojan 4.0	✓	✓	✓		✓	✓	✓	
Dark Comet 5.3	✓		✓		✓	✓	✓	✓

For each combination malware/session corresponding to a checked cell in Table 1, 10 samples are collected. Each sample is a sequence of packets captured using *Tshark* sniffing tool.

The approach used to assess the precision of the proposed detection system is cross-validation [16]. Our experiment consists in applying a 5-fold cross-validation on the collected data.

Three well established measures in classification are used, namely, *precision*, *recall* and *F-measure*. *Precision* measures the fraction of packet traces identified correctly by the proposed system as malicious sessions of a certain type. *Recall* measures the fraction of the total set of malicious sessions in the traffic that are identified correctly by system.

$F1$ is a measure that combines both *precision* and *recall*.

The first experiment performed consists in applying 5-fold cross-validation on 10 samples of each malware/session combination. Only HMM models trained using 4 samples or more are considered. The log likelihood threshold for the HMM acceptance algorithm is fixed to -100. Figure 4 shows the results of the 5-fold cross-validation. Each of the first 5 histograms shows the three measure values for each fold. The last histogram is the average of the 5 folds. The average

[3] All the experiments were carried out using virtual machines both for the infected host and the attacker/C&C server.

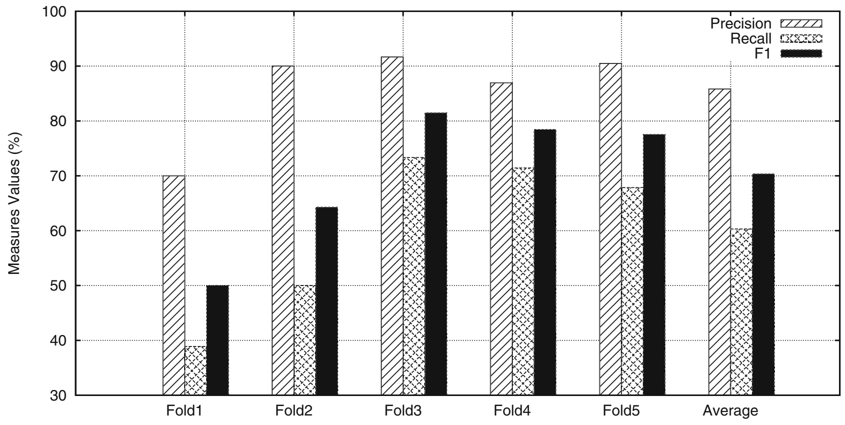

Fig. 4. Results of 5-fold cross-validation HMM based detection with 10 samples, a minimum number of training sequences of 4, and a log likelihood of -100.

precision is more than 85 %. This means that when the detection system identifies a particular session, there is 85 % chance the identification is correct. The average recall is around 60 %. This means that 60 % of all sessions in the testing phase have been correctly identified. The F1 measure is around 70 %.

The two remaining experiments performed aim to assess the efficiency of the system to detect particular malware sessions. Figure 5 shows the precision measures when 5-fold cross-validation is applied separately on each malware RAT. From the experiment' result, the detection system is very efficient in detecting Beast sessions (average recall more than 70 %) while it has hard time with Blacknix sessions (average recall value less than 30 %). Figure 6 shows the precision measures when cross-validation is applied separately on each session type. One can notice that the proposed system is relatively efficient to detect file transfer sessions (average recall of more than 60 %) but showing lower results for webcam video streaming sessions (average recall of 40 %).

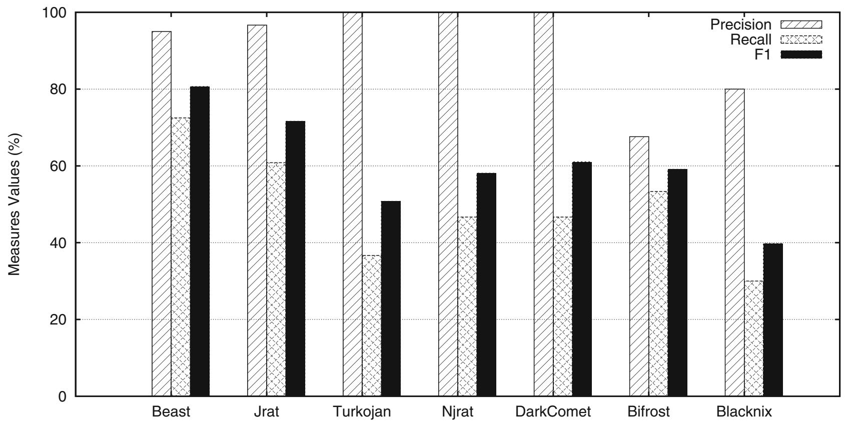

Fig. 5. Detection efficiency of each malware RAT.

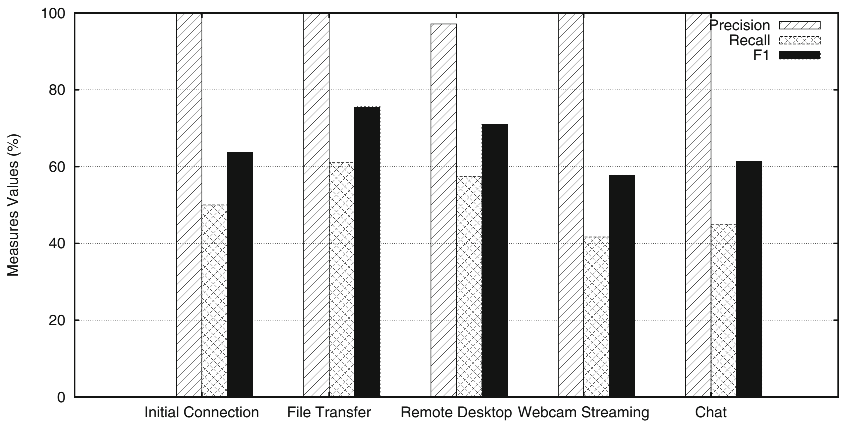

Fig. 6. Detection efficiency of each session type.

7 Conclusion

This paper presents a network-based malware detection system. Unlike typical network-based approaches found in the literature, the proposed system does not only detect malware infections but also identifies with precision the type of malicious session between the infected host and the attacker/C&C server. Signatures for malicious sessions are represented using HMMs. The empirical analysis shows that the proposed system has a high average precision (more than 85 % in almost all the experiment performed) and a good average recall (around 60 %). While the precision of the approach is stable (more than 85 %), the recall depends significantly on the quality of the HMM models which in turn depends on the number of packet trace samples used effectively in the training. For instance, training the HMM models using 7 packet trace samples yields a recall of 75 %.

Our plan for future work is to improve the filtering step in the HMM training to consider slightly noisy packet trace samples in the effective HMM training. This will further improve the efficiency of the detection system. At the implementation side, gathering the training samples needs to be further automated.

References

1. Siroski, M., Honig, A.: Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. No Starch Press, San Francisco (2012)
2. Falliere, N., Murchu, L., Chien, E.: W32.stuxnet dossier. Technical report, Symantec Security Response, February 2011
3. Gostev, A.: The flame: Questions and answers. Technical report, Kaspersky, May 2012
4. Bencsáth, B., Pék, G., Buttyán, L., Félegyházi, M.: Duqu: analysis, detection, and lessons learned. In: ACM European Workshop on System Security (EuroSec). ACM (2012)
5. Leyden, J.: Hack on Saudi Aramco hit 30,000 workstations, oil firm admits (2012). http://www.theregister.co.uk/2012/08/29/
6. Sun, Q., Simon, D.R., Wang, Y.M., Russell, W., Padmanabhan, V.N., Qiu, L.: Statistical identification of encrypted web browsing traffic. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, SP 2002, p. 19. IEEE Computer Society, Washington, DC (2002)

7. Liberatore, M., Levine, B.N.: Inferring the source of encrypted http connections. In: Proceedings of the 13th ACM conference on Computer and Communications Security, CCS 2006, pp. 255–263. ACM, New York (2006)
8. Herrmann, D., Wendolsky, R., Federrath, H.: Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naive-bayes classifier. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, CCSW 2009, pp. 31–42. ACM, New York (2009)
9. Panchenko, A., Niessen, L., Zinnen, A., Engel, T.: Website fingerprinting in onion routing based anonymization networks. In: Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, WPES 2011, pp. 103–114. ACM, New York (2011)
10. Cai, X., Zhang, X.C., Joshi, B., Johnson, R.: Touching from a distance: website fingerprinting attacks and defenses. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012, pp. 605–616. ACM, New York (2012)
11. Wang, T., Goldberg, I.: Improved website fingerprinting on tor. In: 12th ACM Workshop on Privacy in the Electronic Society, WPES 2013. ACM (2013)
12. Dingledine, R., Mathewson, N., Syverson, P.: Tor : the second-generation onion router. In: Proceedings of the 13th Usenix Security Symposium, August 2004
13. prorat trojan. http://en.wikipedia.org/wiki/ProRat
14. Rabiner, L.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)
15. Durbin, R., Eddy, S.: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge (1998)
16. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1137–1143 (1995)