

Using Session-Keystroke Mutual Information to Detect Self-Propagating Malicious Codes

Syed Ali Khayam

NUST Institute of Information Technology
National University of Sciences & Technology (NUST)
Rawalpindi, Pakistan
khayam@niit.edu.pk

Hayder Radha*

Department of Electrical & Computer Engineering
Michigan State University
East Lansing, MI 48824, USA
radha@egr.msu.edu

Abstract—In this paper, we propose an endpoint-based joint network-host anomaly detection technique to detect self-propagating malicious codes. Our proposed technique is based on the observation that on any endpoint there exists very high correlation between benign network sessions and the keystrokes that trigger these sessions. Specifically, users generally use a few keystrokes to trigger most of the benign network sessions. On the other hand, malicious sessions originating from a compromised endpoint will not have the session-keystroke correlation. We leverage this observation in a novel information-theoretic framework that characterizes the session-keystroke correlation in terms of their mutual information. Changes in session-keystroke mutual information are used to detect malicious codes in an automated and real-time fashion. To evaluate the proposed anomaly detector, we use actual traffic and keystroke data collected on benign and infected endpoints. We show that the proposed anomaly detector provides almost 100% detection with negligible false-alarm rates and significantly surpasses the accuracy of existing techniques.

I. INTRODUCTION

A recent and dramatic increase in malicious network traffic has compelled security researchers to reexamine classical intrusion detection with emphasis on self-propagating malicious codes. To combat these rapidly evolving malware, we need defense mechanisms which can detect anomalous activity for novel attacks with few (if any) assumptions about the attack technique. The challenge of such anomaly detection systems is the characterization of benign behavior. To that end, traditional anomaly detectors are either: (a) host-based systems that detect anomalies by monitoring a host’s OS behavior (such as tracking OS audit logs, processes, command-lines or keystrokes) [1]–[6], or (b) network-based systems that detect anomalies by observing unusual traffic patterns [1], [7]–[9].

In this paper, we propose an endpoint-based¹ joint network-host anomaly detector which exploits the observation that when a user is actively using his/her computer most of the benign traffic is triggered by a small subset of keystrokes and mouse clicks. Based on this observation, we propose to

correlate the last input from the keyboard or mouse hardware buffer with every new network session². To leverage the session-keystroke correlation for anomaly detection, we use the information-theoretic mutual information measure [11].

To obtain benign profiles of end-users, we have spent 12 months collecting traffic and keystroke statistics of a diverse set of endpoints in home, office, and university settings. An endpoint’s traffic-keystroke profile contains information about session timestamps and the keystrokes preceding sessions. For malicious activity, we use actual self-propagating malicious codes with varying propagation rates and scanning techniques.

We jointly and marginally model the keystroke and session information as random variables. We then compute the mutual information [11] of these random variables on a window-by-window basis. We observe that the mutual information is consistently high in the time windows containing benign data. However, once malicious traffic with a marginal keystroke distribution is inserted into the benign profile, there is a significant drop in the mutual information. This drop is due to the fact that many keys that are generally used very frequently by the users are never used to initiate legitimate network activity. For a user active on his/her endpoint, the malicious network sessions that are not initiated by the user are logged with unlikely keystrokes, thereby causing a decrease in the session-keystroke mutual information.

To use the proposed technique for automated anomaly detection, we use a small subset of the benign profiles to generate the joint and marginal distributions of keystrokes and network sessions. Based on the statistics of these distributions, we ascertain a mutual information threshold below which an alarm is raised. For all our experimental evaluations, we observe almost 100% detection accuracy and negligible false-alarm rates. We compare the performance of our proposed malware detector with two state-of-the-art anomaly detectors, namely the maximum-entropy detector [7] and the rate-limiting detector [9], and show that the proposed detector provides consistently and substantially better performance than the existing techniques.

*This work was supported in part by NSF Award CNS-0430436, NSF Award CCF-0515253, MEDC Grant GR-296, and an unrestricted gift from Microsoft Research.

¹Throughout this paper, we focus on malware detection at *endpoints* with human users, where “an endpoint is an individual computer system or device that acts as a network client and serves as a workstation or personal computing device.” [10].

²We define a *network session* as a bidirectional communication between two IP addresses. Thus communications between the same IP addresses on different ports are considered part of the same network session.

II. RELATED WORK

Some existing host-based anomaly detectors model benign user behavior using commands given by a user in a textual OS environment [3]–[6]. Due to the high market penetration of graphical operating systems, we argue that a practical anomaly detector should model graphical rather than textual behavioral features of end-users.

A recent anomaly detector called BINDER [2] correlates keystrokes with OS processes and raises an alarm whenever a process initiated without end-user input accesses the network. The main issue with BINDER is that it rigidly flags processes that are created without user input as malicious. This inherent rigidity adversely impacts BINDER’s ability to detect slightly novel attacks. For instance, BINDER would not be able to detect memory-resident malware because its detector is invoked only when a new process is created. (There have been many well-known worms that were memory-resident; two most famous examples are CodeRed II and Witty.) Moreover, a trojan-horse with a propagation module will not be detected by BINDER as the trojan’s process is inadvertently created by the user.

The only endpoint- and network-based malware detection technique that the authors are aware of is rate limiting [9]. A rate-limiter places outgoing network sessions of an endpoint in a queue, from where sessions are released at a constant rate. The size of the queue is monitored to detect malicious activity.

There are two recent studies that have proposed information-theoretic measures for network-based anomaly detection. The first technique proposed by Lakhina *et al.* [8] detects anomalies at a border router using entropies of the distributions of source ports, destination ports, and origin/destination pairs. The second study by Gu *et al.* [7] classifies packets into 2,348 distinct classes based on their destination ports and protocol information. Benign probability of each class is learned by maximum-entropy estimation. Anomalies are detected by computing Kullback-Leibler divergence [11] between the learned/benign probabilities and the port frequencies observed in the current time-window. An anomaly is detected if the divergence of a class exceeds a threshold value more than d times in a given window.

III. DATA COLLECTION

In this section, we briefly explain the two main datasets collected for this study. The first dataset comprises benign session-keystroke profiles collected from several hosts with regular human users. In addition to the joint session-keystroke data, on some hosts we have also collected data of all the keystrokes. This all-keystrokes data is used to define marginal probability distributions later in the paper. The second main dataset comprises real malicious traffic.

A. Joint Session-Keystroke Data

We have investing up to 12 months in collecting joint session-keystroke data on a diverse set of 14 endpoints. Users of these endpoints included home users, research students, and

technical/administrative staff with Windows 2000/XP laptop and desktop computers. Most endpoints were single-user machines, but some endpoints were shared among multiple users. The endpoints used in this study were running different types of applications, including peer-to-peer file sharing software, online multimedia applications, network games, SQL/SAS clients etc.

Data were collected by a multi-threaded windows application called `argus` which logged session-level information. Here a *session* represents a bidirectional communication between two IP addresses. Communication between the same IP address on different ports is considered part of the same network session. This data granularity reduces the complexity, while providing complete information about sessions originating from or terminating at an endpoint. Each session is logged using the information contained in the *first* packet of the session. A session expires if it does not send/receive a packet for more than τ seconds. In the collected data, τ is set to 10 minutes.

For each logged session, `argus` also logs the last keystroke or mouse click that was pressed before the first packet of the session. We generically refer to keyboard and mouse inputs as keystrokes or keys in this paper. The last keystroke is associated with a session only if the key was pressed no more than γ seconds before the session. If there was no key pressed in the last γ seconds before a session then a void keystroke value of zero is inserted. In the collected traces, γ is set to 10 seconds. Throughout this work, we only focus on sessions with non-zero keys. We assume that the last pressed key has initiated the associated session, that is, an inherent correlation relationship is assumed between the last key and the consequent session. Clearly, this correlation will not be present when a malicious code is trying to propagate from an oblivious end-user’s computer, and hence perturbations in the session-keystroke correlation can be leveraged at that point to detect the malicious code.

Each entry of the log file has the following 3 fields that are of interest in this work: `<session direction, timestamp, keystroke>`, whose explanation is given below:

- `session direction`: one byte flag indicating outgoing unicast, incoming unicast, outgoing broadcast, or incoming broadcast packets;
- `timestamp`: millisecond-resolution time of session initiation;
- `keystroke`: The virtual key code, as defined by Microsoft MSDN library [12], of the last (keyboard or mouse) key that was pressed before the session. The MSDN virtual key codes assign integer values to keystrokes and mouse clicks. All standard keystrokes and mouse clicks are mapped to integers between 1 and 254.

Throughout this paper, we refer to this jointly collected session and keystroke data as *session-key* or *key-session* data. Moreover, keystrokes observed in this joint profile are referred to as the *session initiation keys*.

The endpoints used for data collection in this study are very diverse because they: (i) operated in different environments (home, office, university, or both home and work in some cases); (ii) generated different traffic loads (total number of sessions varied from 11,996 for endpoint 14 to 444,345 for endpoint 4); and (iii) had different session rates (varying from 0.19 sessions per second (sps) for endpoint 6 to 5.28 sps for endpoint 4.) We also observed that on all hosts more than 90% of the sessions are initiated using 10 keys. This is a preliminary indication that the correlation of the session-key data is consistent across endpoints, and therefore can be leveraged to detect malicious activity.

The joint session-key data described above can be used to develop a joint session-key probability distribution. In addition to the correlated/joint data, the information-theoretic framework proposed later in this paper also requires marginal distributions of sessions and keystrokes. Since marginal session information is already provided by the session-key data, the only missing component here is the distribution of all the keystrokes that are pressed on a host.

B. All-Keystrokes Data

To develop a marginal distribution of keystrokes, we had to log frequencies of all the keys that are pressed on a host. Due to strict privacy constraints imposed by the university, and due in part to user reservations, we were unable to collect such data on all the participating hosts. The authors installed a custom-developed keylogger on their own computers (endpoints 5 and 13) and collected keystroke data for more than a month. Each entry of the keylogger contains 2 fields, `<timestamp, keystroke>`, in the same format as described above. This dataset is referred to as the *all-keys* data in this paper. For the remaining endpoints, an average of the all-keys data of endpoints 5 and 13 is used as the *marginal keystroke distribution*, which is simply a normalized histogram of the frequency of usage of the keystrokes.

In addition to benign data, we have also collected malware data generated by real malicious codes. The following section explains collection of the malicious traffic data.

C. Malware Traffic Data

We installed original and unpatched releases of Windows 2000 and Windows XP on a computer using Microsoft Virtual PC 2004 [13]. The advantage of using virtual machines (VMs) was that once a virtual host was infected, we could reinstall it by overriding just a few key files. We assigned static IP addresses to both virtual machines and connected them to the Internet. These hosts were then compromised by the following malware: `Zotob.G`, `Forbot-FU`, `Sdbot-AFR`, and `Dloader-NY`. (Further details of the malware used in this paper can be found at [14]–[16].) We also requested network administrators and research collaborators in our university to share malware binaries and source codes with us. This way we acquired `SoBig.E@mm` and the C source code of `MyDoom.A@mm`, which are mass-mailing worms. Finally, we

downloaded binaries or source codes of the following malware from the Internet: `Blaster`, `Rbot-AQJ`, and `RBOT.CCC`.

The malicious codes used in this paper are quite diverse with varying attack ports, transport protocols and scan rates; we observed the highest scan rate of 46.84 scans per second (sps) with `Dloader-NY`, while `MyDoom-A` and `Rbot-AQJ` have very low scan rates of 0.14 and 0.68 sps, respectively. We show later that for existing network-based anomaly detectors the low-rate `MyDoom-A` and `Rbot-AQJ` are more difficult to detect than high-rate malware.

D. Offline Evaluation Methodology

A vulnerable VM was infected with each of the 9 malicious codes. We then used `argus` to log malicious traffic traces from the VM in the same format as the benign session-key data. While this provided us complete information about the malicious sessions, we did not have information about the keystrokes that a user will be pressing when a malicious code is trying to propagate after compromising his/her machine. The only way to realistically generate such data is to infect participating endpoints with malicious codes without informing the user of that endpoint. Clearly, such a procedure is not feasible. Therefore, for each malicious session we generate an associated keystroke using the marginal keystroke distribution generated from the all-keys data.

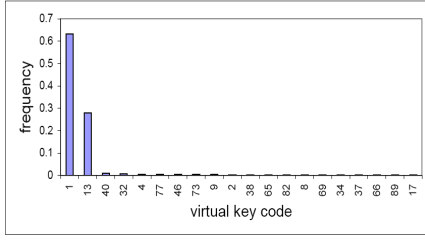
We insert T minutes of malicious traffic data of each malicious code in the benign session-key profile of each endpoint at a random time instance. Specifically, for a given endpoint's benign session-key profile, we first generate a random infection time t_I (with millisecond accuracy) between the endpoint's first and last session times. Given n malicious sessions starting at times t_1, \dots, t_n , where $t_n \leq T$, we create a special *infected* profile of each host with these sessions appearing at times $t_I + t_1, \dots, t_I + t_n$. Thus in most cases once a malware's traffic is completely inserted into a benign profile, the resultant profile contains interleaved benign and malicious sessions starting at t_I and ending at $t_I + t_n$. For all malware used in this study, we use $T = 15$ minutes.

IV. MUTUAL INFORMATION BASED CHANGE DETECTION

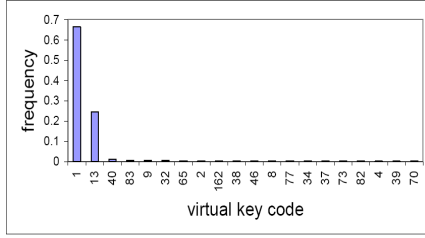
Like prior endpoint-based studies, we focus solely on outgoing unicast traffic since incoming packets can be easily blocked using firewalls. Also, we only focus on the scenario when the end-user is actively using his/her computer, although he/she may not be accessing the Internet. This is achieved by only processing sessions with non-zero keystroke values; recall that a zero keystroke value implies that no key was pressed right before the session. Detection when a user is inactive cannot employ keystroke data, thereby requiring purely network-based approaches.

A. Correlation in the Session-Key Data

Fig. 1 shows the normalized frequencies of the 20 most-used session initiation keys for two endpoints. In both cases, more than 85% of the times network sessions are initiated by the `left mouse click` or the `Enter` key. (Similar results



(a) endpoint 5



(b) endpoint 13

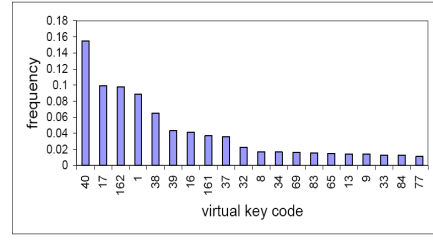
Fig. 1. Normalized histograms of 20 most-used session initiation keystrokes. Histograms are generated from the session-key data. Virtual keys codes 1 and 13 correspond to the left mouse click and the Enter key, respectively [12].

are observed for the remaining endpoints.) Fig. 2 shows the normalized histograms of all the keystrokes that are pressed on a host. Note that the all-keys distribution looks quite different from the session-key distribution of Fig. 1. For one thing, the all-keys distribution of Fig. 2 is much more spread out (i.e., has higher variance) than the session-key distribution of Fig. 1. Also, contrary to the session-key-based keystroke histogram, less than 50% sessions are initiated by the two most-commonly used keys. Lastly, left mouse click or Enter are not in the two most-commonly used keys in either Fig. 2(a) or (b). These results can be summarized as follows: (i) users frequently employ only a few session initiation keys to trigger network sessions, thus there is strong correlation between these few session initiation keys and network sessions; (ii) frequencies of session initiation keys are very consistent across different users, consequently making this a common benign feature that can be leveraged to detect abnormal behavior; (iii) frequencies of keys that are generally used on a host are quite different from frequencies of session initiation keys.

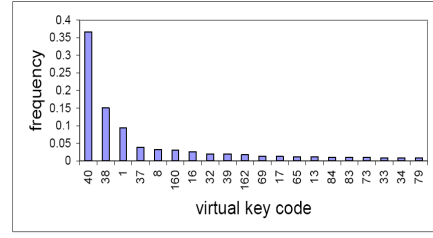
Based on the above discussion, we deduce that session-key correlation is a feature that is common across users and can be used for malware detection. To formally leverage this observation, the following section uses a mutual information based framework.

B. Session-Keystroke Mutual Information

Mutual information [11] is an information-theoretic measure of the similarity between two probability distributions. Consider two random variables X and Y with marginal distributions $p(x)$ and $p(y)$, and a joint distribution $p(x, y)$. The mutual information of these random variables is defined



(a) endpoint 5



(b) endpoint 13

Fig. 2. Normalized histograms of 20 most-used keystrokes. Histograms are generated from the all-keys data. Virtual keys codes 40, 38 and 17 correspond to the down arrow key, the up arrow key and the control key, respectively [12].

as

$$I(X; Y) = \sum_x \sum_y p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)}. \quad (1)$$

Mutual information is a non-negative measure of the similarity between X and Y , with $I(X; Y) = 0$ when X and Y are independent. In general, $I(X; Y)$ increases with respect to the correlation between X and Y .

To leverage mutual information in the present context, we define X as a binary random variable which characterizes the probability of whether or not a session was initiated in the last time window. That is, $X \in \{0 \Rightarrow \text{no session in time window}, 1 \Rightarrow \text{one or more sessions in time window}\}$. Moreover, we define Y as a random variable characterizing the probability of a keystroke, i.e., $Y \in K_n$, where $K_n = \{1, 2, \dots, 254\}$, due to MSDN's keystroke definition [12]. Specifically, the marginal $p(Y)$ distribution is simply the normalized all-keys histogram, such as the ones shown in Fig. 2. Then the session-keystroke mutual information can be written as:

$$I(X; Y) = \sum_{y \in K_n} \left[p(x=0, y) \log_2 \frac{p(x=0, y)}{p(x=0)p(y)} + p(x=1, y) \log_2 \frac{p(x=1, y)}{p(x=1)p(y)} \right]$$

We derive the marginal $p(X)$ distribution using the first 500 entries of each endpoint's benign session-key profile. More specifically, $p(X)$ is computed by counting the total number of windows n with one or more sessions between the 1-st session and the 500-th session. We also count the total number of windows N (with and without sessions) in that time frame. Then, $p(X=1) = N/n$ and $p(X=0) = 1 - p(X=1)$. The joint distribution $p(x=1, y=j)$ then simply corresponds to the joint probability that a network session was initiated using keystroke j .

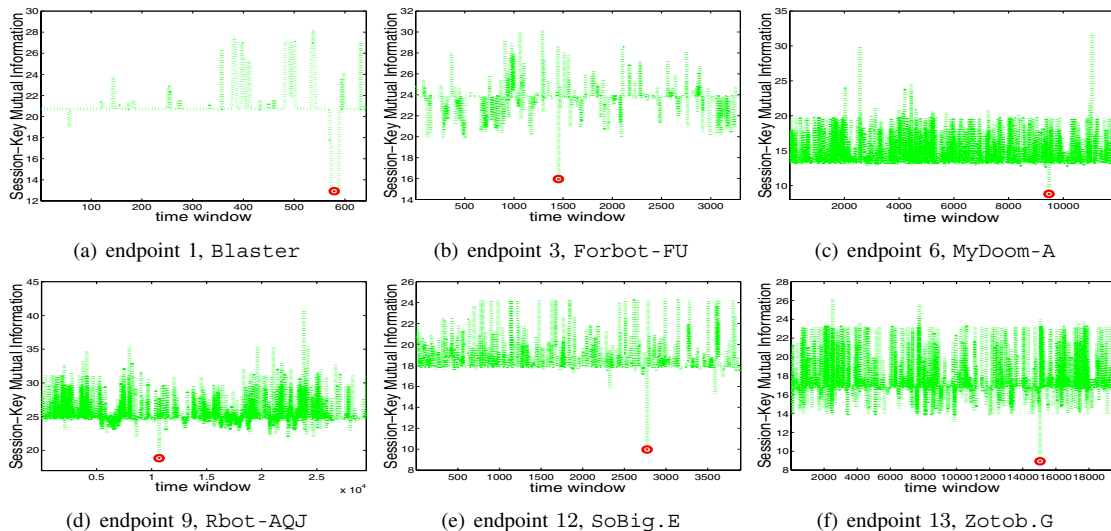


Fig. 3. Mutual of the session and keystroke random variables at infected endpoints. Infection start times are marked with a circle. Infections last approximately 15 minutes. Each non-overlapping time-window is 60 seconds.

As mentioned earlier, the keystroke information is not logged when there are no network sessions in a window. That is, we do not have the distribution $p(x=0, y)$. Hence we cannot use the mutual information expression of (B.56) in its present form. To resolve this problem, we employ a *partial mutual information* measure $I(X=1; Y)$, which only uses the $p(x=1, y)$ probability distribution. Since the partial mutual information employs only one outcome of the random variable X , it can be written as

$$I(X=1; Y) = \sum_y p(x=1, y) \log_2 \frac{p(x=1, y)}{p(x=1)p(y)}$$

Note that due to the binary nature of the session random variable X , the partial mutual information $I(X=1; Y)$ is in fact the self-information of $p(x=1, y)$ normalized by $p(x=1)$. For brevity, we continue to refer to this measure as mutual information.

The above characterization describes the correlation between network sessions and keystrokes in a simple and intuitive manner. Based on previous results, we know that for legitimate activity X and Y are highly correlated. Therefore, their mutual information should be high. Once a self-propagating malicious code starts initiating sessions, the keystrokes will be drawn from the marginal $p(X)$ distribution and therefore the correlation between X and Y should drop.

We compute mutual information on a window-by-window basis. The results reported in this paper use a window size of 60 seconds. (Qualitatively similar results were obtained for other window sizes.) In each window with one or more sessions, we compute the joint distribution $p(x, y)$. The joint distribution $p(x, y)$ is used in equation (1), where $p(X)$ and $p(Y)$ are generated from the training all-keys and session-key data, respectively. (500 sessions from the session-key data are used to generate an estimate of the marginal probability $p(X=1)$.)

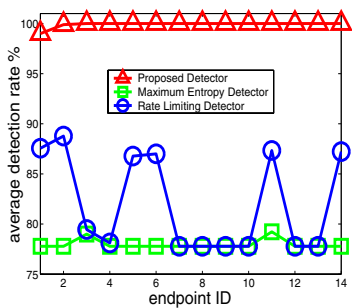
When we used mutual information for detection of randomly inserted infections, we observed a number of noisy spikes due to variations in benign user behavior. We use a median filter to remove the spikes that arise due to inherent changes in legitimate user behavior. Henceforth, all mutual information results reported use an order-7 median filter.

The mutual information of different endpoints randomly infected with a single infection of a malicious code are outlined in Fig. 3. It can be observed in Fig. 3 that session-keystroke mutual information clearly highlights anomalous behavior in all cases. The drop in session-key mutual information is revealed for both high- and low-rate malware, and for endpoints with high and low session rates. Contemporary network-based anomaly detectors implicitly or explicitly use this rate for detection. Therefore, and as will be shown later, detection and false alarm rates of such detectors are dependent on the scanning rate of the malicious code. Our technique jointly considers sessions and keystrokes and is therefore not entirely dependent on the session rate.

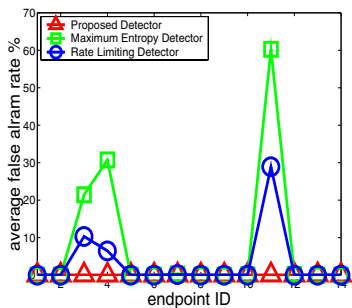
C. Performance Evaluation

To leverage the median-filtered session-key mutual information in a real-time and automated fashion, we train the detector using the first 10 values of session-key mutual information computed from the benign session-key profile of each endpoint e . We find the sample mean μ_e and sample variance σ_e^2 of the 10 mutual information values of endpoint e . An alarm is raised when when the filtered mutual information observed in a window is less than $\eta = \lceil \mu_e + \sigma_e \rceil$.

We inserted 100 non-overlapping infections of each malicious code in every endpoint's benign profile. As discussed earlier, each infection was approximately $T = 15$ minutes. For the 100 infections of a particular malicious code on an endpoint, the percentage detection rate for that malicious code is computed by simply counting the number of infections that



(a) detection rate



(b) false-alarm rate

Fig. 4. Comparison of detection and false-alarm rates of the proposed Mutual Information based malware detector with maximum-entropy [7] and rate-limiting [9] detectors. Each point is averaged over 9 malicious codes with 100 random infections per malicious code per endpoint.

are detected by the malware detector. The false alarm rate is computed by taking the ratio of the total number of false alarms with the total evaluated time-windows (i.e., windows with one or more sessions).

The average detection and false alarm rates for each endpoint are shown in Fig. 4. It is clear from 4(a) that the detection rate of the proposed technique is 100% for all endpoints and all malware, except endpoint 1 which has an average detection rate of 99.66%. Also, note in Fig. 4(b) that the proposed mutual information based detector has negligible false alarm rates at all endpoints. The highest false alarm rate we observed was approximately 0.43%, with many endpoints having almost zero false alarms. It can also be seen in Fig. 4 that the proposed detector clearly outperforms the maximum-entropy detector [7] and the rate limiting detector [9] by providing much higher detection rates and much lower false alarm rates than these existing and state-of-the-art techniques. This high accuracy of the proposed detector is a consequence of combining network- and OS-level information in a joint information-theoretic framework.

V. ATTACKS AND COUNTERMEASURES

In this section, we discuss two attacks that can circumvent the proposed malware detector and possible countermeasures to mitigate these attacks.

A. Mimicry Attack

In a mimicry attack [18], a malicious code tries to hide its traffic inside benign traffic to avoid detection. A mimicry

attack can be launched against the detector proposed in this paper by a malware which always initiates its scanning sessions after a certain predefined time has elapsed since the last keystroke. Such a malicious session will not be evaluated by the proposed detector. To mitigate this attack, the time threshold for logging the session initiation keystroke can be made adaptive.

B. Attack by Modifying Hardware Buffers

A malicious code can insert its own keystroke entries if the OS allows users to directly modify hardware buffers containing keystroke information. To defeat this attack, a trusted computing platform [19] or a virtual machine [13] must be employed.

VI. CONCLUSION

We used network sessions, keystroke and malware traffic on actual endpoints to show that the joint distributions of sessions and keystrokes change when an endpoint is compromised by a malicious code. We proposed an anomaly detector that employed mutual information to detect the changes in session-key mutual information. The proposed detector provided very high detection rates and extremely low false-alarm rates.

REFERENCES

- [1] L. Me and C. Michel, "Intrusion Detection: A Bibliography," Tech. Rep. SSIR-2001-01, Sep 2001.
- [2] W. Cui, R. H. Katz and W-T. Tan, "BINDER: An Extrusion-based Break-In Detector for Personal Computers," *Usenix Security Symposium*, Apr 2005.
- [3] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State Transition Analysis: A Rule-based Intrusion Detection Approach," *IEEE Trans. on Software Engineering*, (21)3, pp. 181-199, Mar 1995.
- [4] S. Jha, K. Tan, and R.A. Maxion, "Markov Chains, Classifiers, and Intrusion Detection," *IEEE CSFW*, Jun 2001.
- [5] N. Ye, "A Markov Chain Model of Temporal Behavior for Anomaly Detection," *IEEE Workshop on Information Assurance and Security*, Jun 2000.
- [6] W. DuMouchel, "Computer Intrusion Detection Based on Bayes Factors for Comparing Command Transition Probabilities," Tech. Rep. 91, National Institute of Statistical Sciences, 1999.
- [7] Y. Gu, A. McCullum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," *ACM/Usenix IMC*, Oct. 2005.
- [8] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *ACM SIGCOMM*, Aug. 2005.
- [9] M. M. Williamson, "Throttling viruses: Restricting propagation to defeat malicious mobile code," *ACSAC*, Dec. 2002.
- [10] Endpoint Security Homepage, <http://www.endpointsecurity.org/>.
- [11] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley-Interscience, 1991.
- [12] MSDN Library, <http://msdn.microsoft.com>.
- [13] Microsoft Virtual PC 2004, <http://www.microsoft.com/Windows/virtualpc>.
- [14] Sophos Virus Info, <http://www.sophos.com/virusinfo/>.
- [15] Symantec Security Response, <http://securityresponse.symantec.com/avcenter>.
- [16] TrendMicro Virus Encyclopedia, <http://au.trendmicro-europe.com/smb/vinfo>.
- [17] Trusted Computing Alliance, <https://www.trustedcomputinggroup.org>.
- [18] D. Wagner and P. Soto, "Mimicry Attacks on Host-Based Intrusion Detection Systems," *ACM CCS*, Nov. 2002.
- [19] Trusted Computing Alliance, <https://www.trustedcomputinggroup.org>.