J Comput Virol (2011) 7:159–172
DOI 10.1007/s11416-010-0145-1

ORIGINAL PAPER

Joint network-host based malware detection using information-theoretic tools

Syed Ali Khayam · Ayesha Binte Ashfaq · Hayder Radha

Received: 21 July 2009 / Accepted: 12 July 2010 / Published online: 27 July 2010

Abstract In this paper, we propose two joint network-host based anomaly detection techniques that detect self-propagating malware in real-time by observing deviations from a behavioral model derived from a benign data profile. The proposed malware detection techniques employ perturbations in the distribution of keystrokes that are used to initiate network sessions. We show that the keystrokes' entropy increases and the session-keystroke mutual information decreases when an endpoint is compromised by a self-propagating malware. These two types of perturbations are used for real-time malware detection. The proposed malware detection techniques are further compared with three prominent anomaly detectors, namely the maximum entropy detector, the rate limiting detector and the credit-based threshold random walk detector. We show that the proposed detectors provide considerably higher accuracy with almost 100% detection rates and very low false alarm rates.

Parts of this work appeared in the Proceedings of IEEE International Conference on Communications (ICC) 2007 [1].

S. A. Khayam's work was supported in part by Pakistan National ICT R&D Fund and Higher Education Commission (HEC), Pakistan.
H. Radha's 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.

S. A. Khayam (✉) · A. B. Ashfaq
School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology (NUST), Islamabad, Pakistan
e-mail: ali.khayam@seecs.edu.pk

A. B. Ashfaq
e-mail: ayesha.ashfaq@seecs.edu.pk

H. Radha
Department of Electrical and Computer Engineering, Michigan State University (MSU), East Lansing, USA
e-mail: radha@egr.msu.edu

1 Introduction

A recent and dramatic increase in automated network intrusions has necessitated defense mechanisms that can curb the spread of self-propagating malicious software (malware[1]) in real-time. Effective containment of rapidly evolving worms and viruses requires real-time defense mechanisms that can detect *novel* (i.e., previously unknown) attacks. The challenge of anomaly detection systems is the characterization of benign behavior. Most of the contemporary anomaly detectors are either (a) network-based systems that detect anomalies by observing unusual network traffic patterns [2–26] or (b) host-based systems that detect anomalies by monitoring an endpoint's operating system (OS) behavior, for instance by tracking OS audit logs, processes, command-lines or keystrokes [27–31].

Recent statistics show that increasingly *network endpoints*, comprising client machines at homes and offices, are serving as extremely potent and viable launch pads and carriers for worm and virus infections [36,37]. Thus it is important that real-time defenses be deployed at the endpoints or at network points close to the endpoints. Recently, there has been some interest in network- and host-based malware detection at endpoints [27–31,19–22] or at servers close to the endpoints [23–25]. Most of these studies leverage some characteristics of past malware for endpoint-based

[1] Throughout this paper, the terms malware and self-propagating malware are used synonymously.

malware detection. While these malware characteristics hold true for some of the contemporary malware, their validity and efficacy are currently being questioned [32–44]. Consequently, there is a growing interest in identification of inherent features and models of benign/legitimate behavior [2], so that malicious activity can be detected using deviations from benign behavior, rather than relying on prior experiences of malicious activity.

In this paper, we propose novel features for malware detection at endpoints and show that the field of information theory provides very effective tools for accurate quantification of these features. To identify benign behavioral features of end-users, we have spent up to 12 months collecting traffic statistics of a diverse set of endpoints in home, office, and university settings. An endpoint's traffic profile contains information about session-level network activity, such as one-way hashed source and destination IP addresses, session direction (incoming or outgoing), source and destination ports, keystrokes, etc. For malicious activity, we use real and simulated worms. These worms vary in their propagation rates and scanning techniques. We evaluate the benign data profiles for behavioral features that get perturbed when the endpoint is compromised by a self-propagating malware. Based on the identified features, we propose two malware detection techniques.

The malware detection techniques proposed in this paper are joint network-host anomaly detectors which exploit 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 session. We use marginal keystroke data to show that the session initiation keys are not necessarily used as frequently by an end-user. To effectively exploit the session-keystroke correlation in real-time and automated fashion, we propose two information-theoretic measures, namely keystrokes' entropy and session-keystroke mutual information [45].

We compute the keystrokes' entropy and mutual information on a window-by-window basis. We observe that the entropy is consistently low and mutual information is somewhat 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 increase in the entropy and simultaneously there is a decrease in the mutual information. These entropy and mutual information perturbations are because of 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 who is active on his/her endpoint, the malicious network sessions that are not initiated by the user are logged with unlikely and diverse keystrokes thereby changing the keystrokes' distribution. To create an automated detection tool based on the keystroke distributions, 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 joint and marginal keystroke distributions, we develop entropy/mutual information thresholds above/below which an alarm is raised. For both entropy and mutual information based detectors, we observe almost 100% detection accuracy and negligible low false-alarm rates. Overall the mutual information detector has lower false alarm rates than the entropy detector. We compare the performance of our proposed malware detectors with three existing anomaly detectors, namely the maximum-entropy detector [14], the rate-limiting detector [19] and the credit-based threshold random walk detector (TRW-CB) [20]. We show that the proposed joint network-host based detection techniques provide consistently and substantially better performance than the techniques of [14,19] and [20].

The rest of this paper is structured as follows. Section 2 describes related work in this area. Section 3 details the benign endpoint profiles and malware collected for this study. Section 4 presents the proposed joint network-host based detection techniques which respectively employ entropy and mutual information of keystrokes and network sessions to detect malware. Section 5 provides a comparative performance evaluation of the proposed techniques with the ADSs of [14,19] and [20]. Section 6 identifies possible attacks on the proposed malware detectors and discusses defenses against these attacks. Section 7 summarizes key conclusions of this paper.

2 Related work

Most of the contemporary studies perform network-based anomaly detection at the enterprise network perimeter or the local network perimeter. Zou et al. [3] propose a malware warning center (MWC) and distributed ingress and egress sensors at a local network's perimeter. Similarly, Wu et al. [4] propose a network architecture and a distributed algorithm to detect multi-vector worms. Schechter et al. [20] used a combination of rate limiting and portscan detection on local network worm detector. Jung et al. [5] develop a network-level fast portscan detector that uses a threshold random walk (TRW) on typical access patterns to infer whether a host is malicious or benign. Weaver et al. [6] simplify the TRW algorithm to make it more amenable to hardware and software implementations. The simplified algorithm of [6] can accurately detect very low rate worms. Soule et al. [11] apply a Kalman filter to normal traffic and then use multiple anomaly detection techniques to detect abnormal behavior. Kim et al. [12] propose that gateway routers score each packet based on its legitimacy. Similarly, anomaly detectors that monitor

blocks of unused IP addresses are also becoming increasingly popular [15–18].

There has been some recent interest in detecting malware at servers near the endpoints. Whyte et al. [23] detect worms by monitoring (at the gateway router) connections that are not preceded by a DNS address resolution request. Gupta and Sekar [24] detect changes in traffic volume at a mail server to detect mass mailing worms. Xiong [25] trace attachment at mail servers to detect mass mailing worms. Barford et al. [9] use time-frequency signal analysis to develop a change detection algorithm. Krishnamurthy et al. [10] propose a sketch-based change detection algorithm. Lakhina et al. [7,8] propose a subspace method to detect and characterize network-wide volumetric traffic anomalies. The authors then extend their work in [13] and use entropy to detect anomalies. Another recent study by Gu et al. [14] uses maximum-entropy estimation to quantify a baseline distribution at a network gateway or router, which is in turn used to classify anomalous activity using the K-L divergence.

The most commonly used endpoint-based network-level malware detection technique is rate limiting. This technique proposed by Twycross and Williamson [19,21] limits the rate of an endpoint's network traffic to curb and detect malware propagation. Sellke et al. [22] extend rate limiting by proposing a branching worm propagation model and in turn using this model to develop a window-based rate limiting mechanism.

Wong et al. [39,40] show that rate limiting is not very effective on endpoints or local network perimeter, but can provide effective malware throttling if deployed on backbone routers. Panjwani et al. [44] evaluated whether portscans are precursor to malicious attacks. It was concluded in [44] that over 50% of attacks are not preceded by a portscan and, therefore, "port scans should not be considered as precursors to an attack." Moreover, Li et al. [41] show that statistical filtering-based defense mechanisms are effective when they are adapted in accordance with an attack. In [41] it is also shown that the performance of a statistical filter degrades significantly if the attacker is more adaptive than the filter.

In the host-based anomaly detection context, most of the existing detectors characterize benign user behavior by modeling commands given by a user in a textual OS environment [28–31]. Due to the high market penetration of graphical operating systems, it is important to model graphical behavioral features of end-users. A recent technique called BINDER [27] correlates keystrokes with OS processes and raises an alarm whenever a process is initiated without an end-user's input. There are important differences between BINDER and the detector proposed in this paper. First, BINDER is purely host-based and does not employ any network session information. Second, BINDER cannot detect memory-resident malicious codes 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`.) Since our technique uses both network and host information, it can detect memory-resident malware. Lastly, BINDER requires a whitelist of legitimate applications before deployment. The detector proposed in this paper can be deployed out-of-the-box after which all training is done online.

3 Data collection and simulation

In this section, we explain the two main datasets collected for this study. The first dataset comprises benign traffic and keystroke profiles collected from several hosts with regular human users. The second dataset comprises real malware traffic. Since university policy and user reservations prohibited us from infecting operational endpoints with malware, we first identify network- and host-based features perturbed by the introduction of malicious code into each system and then perform offline analysis by inserting malicious traffic at random instances in the endpoints' benign traffic profiles.

3.1 Benign traffic-keystroke profiles

Our first step towards the development of a network-based worm detector was to collect pertinent network traffic data. We started by investing up to 12 months in monitoring network profiles of a diverse set of 13 endpoints. Users of these endpoints included home users, research students, and technical/administrative staff with Windows 2000/XP laptop and desktop computers. The laptop endpoints were used by their users both at home and at work. Some endpoints, in particular home computers, 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. (More data details to follow.)

Data were collected by a multi-threaded windows application called *logger*, which runs as a background process storing network activity in a log file. The log file is periodically and securely uploaded to a secure copy (SCP) server. *logger* only logs session-level information, where a *session* corresponds to bidirectional communication between two IP addresses. Communication between the same IP address on different ports is considered part of the same network session. This session-level granularity reduces the complexity of the worm detector, 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.

Table 1 Statistics of benign profile collected for this study

Endpoint ID	Endpoint type	Total profile collection time (months)	Total sessions	TCP sessions (%)	UDP sessions (%)	Mean session rate (/sec)	Var in session rate (/sec)	Cumulative freq of ten most-used Src ports	Cumulative freq of ten most-used Dst ports	Cumulative freq of ten most-used session keys
1	Office	8	33,487	13.95	73.9	0.25	0.26	90.37	88.06	96.01
2	Office	10	21,066	50.45	42.29	0.22	0.43	47.8	87.53	92.32
3	Home	3	373,009	98.54	1.36	1.92	11.98	3.95	37.29	92.01
4	Home	2	444,345	57.37	41.91	5.28	25.93	5.86	10.82	94.86
5	Home/Univ	3	27,873	74.71	24.03	0.44	2.0	15.91	99.27	95.25
6	Univ	9	60,979	24.36	70.89	0.19	0.35	54.95	94.0	95.49
7	Univ	11	171,601	45.46	53.53	0.28	0.6	40.7	96.75	95.56
8	Univ	13	41,809	19.7	76.58	0.52	0.71	66.1	96.44	96.13
9	Univ	13	235,133	47.96	50.22	0.41	0.81	44.1	94.84	95.48
10	Univ	13	152,048	12.8	82.33	0.21	0.37	75.19	95.11	95.27
11	Univ	13	207,187	44.63	47.8	0.31	0.96	38.85	95.2	95.14
12	Home/Univ	13	100,702	65.94	32.24	0.33	0.73	24.78	95.0	95.13
13	Univ	3	11,996	47.66	52.0	0.23	0.66	44.56	95.98	95.95

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 λ seconds before the session. If there was no key pressed in the last λ seconds before a session then a void keystroke value of zero is inserted. In the collected traces, λ is set to 10 s. Throughout this paper, 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 6 fields:

```
<session id, direction, src port, dst
 port, proto, timestamp, virtual key
                code>,
```

whose explanation is given below:

- `session id`: 20-byte SHA-1 hash [46] of the concatenated hostname and remote IP address. Hashing preserves privacy, since the collected data are going to be publicly available;
- `direction`: one byte flag indicating outgoing unicast, incoming unicast, outgoing broadcast, or incoming broadcast packets;
- `proto`: transport-layer protocol (i.e., TCP or UDP) of the packet; all transport layer fields are set to zero for network layer packets;
- `src port`: source port of the packet;
- `dst port`: destination port of the packet;
- `timestamp`: millisecond-resolution time of session initiation.
- `virtual key code`: one byte virtual key code, as defined by Microsoft's MSDN library [47], of the last (keyboard or mouse) keystroke that was pressed before the session. In view of our stringent privacy considerations, we only log the very last keystroke that was pressed right before the first packet of a new session. 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.

Some pertinent statistics of the collected benign data are listed in Table 1.[2,3] Diversity of the endpoints used in this study is evident from Table 1, which shows that the endpoints operate in different environments (and hence run different

[2] It should be emphasized that the mean and variance of session rates in Table 1 are computed using time-windows containing one or more new sessions. Since time-windows without network activity are simply ignored, the total profile logging time is not equal to the ratio between the total sessions and the mean session rate. As can be inferred intuitively, time-windows without new network sessions are fairly common on endpoints.

[3] The total profile collection time in Table 1 was computed during result generation for this paper, which was some months back. We now have benign profiles that are up to 21 months long.

types of applications). Also, the total size of the dataset (i.e., total number of sessions) varies from 11, 996 for endpoint 13 to 444, 345 for endpoint 4. In general, we observed that home computers generate significantly higher traffic volumes than office and university computers because: (1) they are generally shared between multiple users, and (2) they run peer-to-peer and multimedia applications. The high traffic volumes of home computers are also evident from the high mean and variance of the number of sessions per second [columns 7 and 8]. Also note that there is little correlation in the amount of TCP and UDP traffic across endpoints; for instance, the amount of UDP traffic [column 6] varies from 1.36% to 82.33% in the benign profiles. This is again a function of the user(s) and the consequent applications running on an endpoint.

Another interesting observation is that, with the exception of home computers, the observed endpoints generally use a small set of source and destination ports very frequently [columns 9 and 10]. (The source and destination port frequencies in Table 1 are computed for outgoing unicast packets.) This observation holds particularly true for destination ports [column 10] because in most cases ten destination ports are used approximately 90% of the times—endpoints 3 and 4 being the exceptions here. This is a preliminary indication that port usage is a statistic that is somewhat consistent across endpoints, and therefore can be leveraged to detect malicious activity. Also, later in the paper it is shown that the different benign behavior of home endpoints poses a considerable challenge to worm detectors.

The last important observation is that without exception all of the observed endpoints use a small set of session initiation keys very frequently [column 11]. (The session initiation key frequencies in Table 1 are computed for outgoing unicast packets with non-zero keys.) In fact, 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 provides us correlated information of keystroke and sessions. In other words, this data can be used to develop a joint session-key probability distribution. In addition to the correlated/joint data, the keystroke-based detectors proposed in this paper also requires marginal distributions of keystrokes. That is, we need a distribution of all the keystrokes that are pressed on an endpoint. The following section describes this data.

3.2 All-keystrokes' profiles

To develop a marginal distribution of keystrokes, we had to log 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, it was not possible to collect such data on all the participating hosts. We installed a custom-developed keylogger on two computers [endpoints 5 and 12] and collected keystroke data for more than a month. Each entry of the keylogger contains two fields: `<timestamp, keystroke>`, which are in the same format as described in the last section.

This dataset is referred to as the *all-keys* data. For the remaining endpoints, an average of the all-keys data of endpoints 5 and 12 is used for the keystrokes' marginal distribution. This *marginal keystroke distribution* 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.

3.3 Worm classification

To generate traffic patterns for each worm, we infected a vulnerable machine with a worm and observed the traffic generated by the worm using the *logger* data utility described in the previous section. (The vulnerable machines used here are different from the operational endpoints used for benign profile collection.) This section details the worms collected and simulated in this study. Before we describe worm data collection, explanation of some terminology is in order.

After compromising a vulnerable host, a worm tries to infect other computers by sending out scan packets with infectious payloads. A vulnerable machine gets infected if it receives and processes a scan packet. Throughout this paper, scan packets generated by a worm after compromising a vulnerable host are referred to as *outgoing scan packets*. Based on the outgoing scan packets, we classify worms into two broad categories:

- *Destination-port worms*: destination ports of scan packets are fixed, but the source ports may be arbitrary;
- *Source-port worms*: source ports of scan packets are fixed, but the destination ports may be arbitrary.

In the former case, we refer to the destination ports of a worm as *attack ports* and the source ports as *non-attack ports*. In the latter case, the roles are reversed and we refer to source ports as *attack ports* and destination ports as *non-attack ports*. All contemporary worms, used in this study, are destination-port worms. Note that a source/destination port worm can be a multi-vector worm [43] targeting multiple vulnerabilities simultaneously. We now describe the worms used in this study.

3.4 Real worms' profiles

A critical aim of our study is to use real and diverse worm data to test our detection technique. To this end, we installed

Table 2 Information of worms used in this study

Worm	Release date	Avg. Scan rate (sps)	Port(s) used
Blaster	Aug 2003	10.5	TCP135,4444, UDP 69
Dloader-NY	Jul 2005	46.84	TCP 135,139
Forbot-FU	Sep 2005	32.53	TCP 445
MyDoom-A	Jan 2006	0.14	TCP 3127–3198
RBOT.CCC	Aug 2005	9.7	TCP 139,445
Rbot-AQJ	Oct 2005	0.68	TCP 139,769
Sdbot-AFR	Jan 2006	28.26	TCP 445
SoBig.E	Jun 2003	21.57	TCP 135,UDP 53
Zotob.G	Jun 2003	39.34	TCP 135,445,UDP 137

original and unpatched releases of Windows 2000 and Windows XP on a computer using Microsoft Virtual PC 2004 [48]. 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 worms used in this paper can be found at [49,50], or [51].) 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 worms from the Internet: Blaster, Rbot-AQJ, and RBOT.CCC.

Table 2 shows the diversity of the worms used in this paper. These worms have different (and sometimes multiple) attack ports and transport protocols. Also, these worms include both high- and low-rate worms; Dloader-NY has the highest scan rate of 46.84 scans per second (sps), while MyDoom-A and Rbot-AQJ have very low scan rates of 0.14 and 0.68 sps, respectively. We show later that the low-rate MyDoom-A and Rbot-AQJ are more difficult to detect than high-rate worms.

All real worms collected for this study fall into the widely prevalent category of destination-port worms. The proposed detection techniques, however, do not rely on source and destination ports of the malware.

3.5 Inserting worm data in the benign traffic profile

We implemented the propagation modules of the simulated worms. A vulnerable VM was then infected with each of the 12 worms. We then used *logger* to log malicious traffic traces from the VM in the same format as the benign data. Armed with this information, we insert T minutes of malicious traffic data of each worm in the benign profile of each endpoint at a random time instance. Specifically, for a given endpoint's benign profile, we first generate a random infection time t_I (with millisecond accuracy) between the endpoint's first and last session times. Given n worm sessions starting at times $t_1, \ldots, t_n$, where $t_n \leq T$, we create a special *infected* profile of each host with these sessions appearing at times $t_I + t_1, \ldots, t_I + t_n$. Thus in most cases once a worm's traffic is completely inserted into a benign profile, the resultant profile contains interleaved benign and worm sessions starting at t_I and ending at $t_I + t_n$. For all worms we use $T = 15$ minutes.

We are now ready to use the infected profiles to characterize traffic perturbations observed when an endpoint is compromised by a worm.

4 Malware detection using joint network-host features

Traditional anomaly detectors are either host- or network-based. We argue that significant improvements can be achieved if both network and host features are correlated and then employed in a joint framework. To that end, in this section we propose two endpoint-based joint network-host anomaly detectors both of which exploit 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 session in a novel entropy-based information-theoretic framework. Like prior endpoint-based studies, we focus solely on outgoing unicast traffic since incoming unicast packets can be easily blocked using firewalls.

4.1 Correlation in the session-key data

As mentioned before, we focus solely on outgoing unicast traffic. 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.

Figure 1 shows normalized plots for the most used session initiation keys in the presence of malware. It can be observed that most-commonly, a few session keys are used to initiate network sessions. However, due to the presence of malware initiated sessions, the histogram appears to be spread out. However still, a few session keys occupy a major percentage of the keys used for session initiation at the

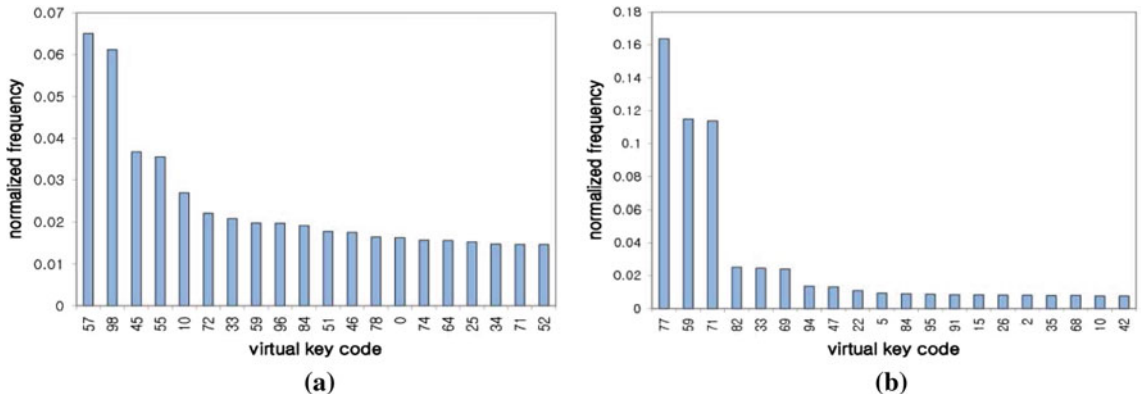

Fig. 1 Normalized histograms of 20 most-used session initiation keystrokes in the presence of malware. Histograms are generated from the session-key data. **a** Endpoint 5. **b** Endpoint 13

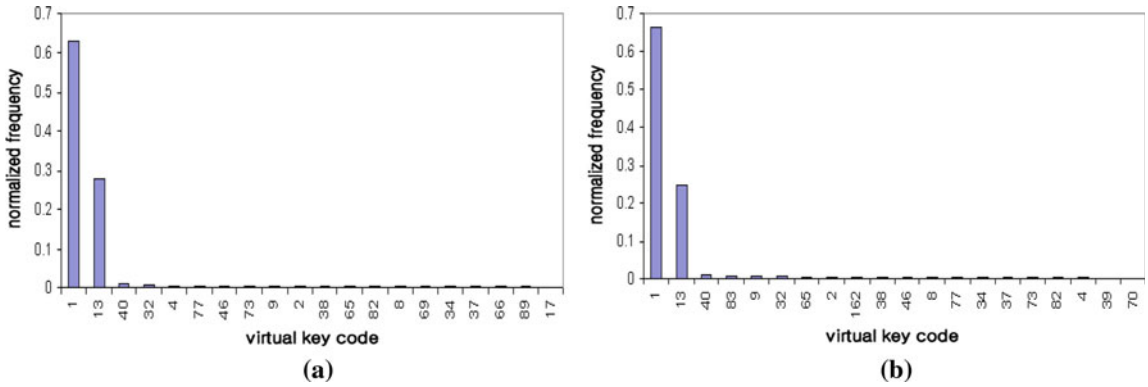

Fig. 2 Normalized histograms of 20 most-used session initiation keystrokes. Histograms are generated from the session-key data. Virtual keys codes 1 and 13 correspond to the `left mouse click` and the `Enter` key, respectively [47]. **a** Endpoint 5. **b** Endpoint 13

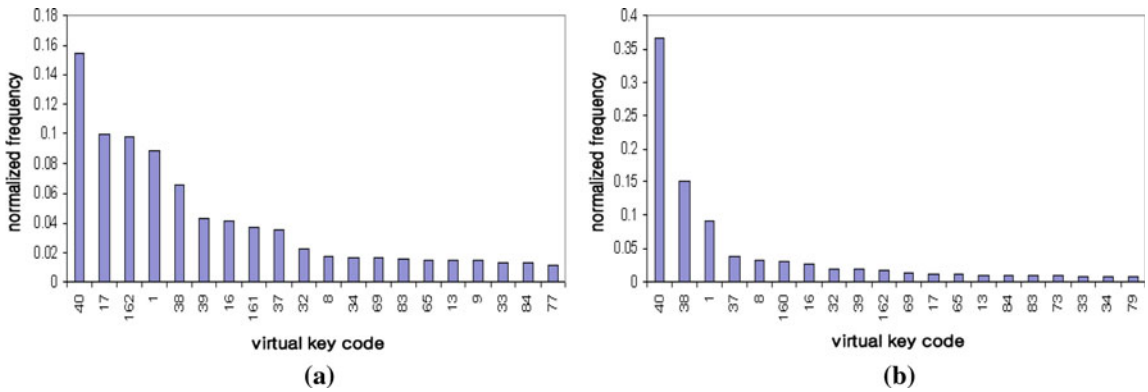

Fig. 3 Normalized histograms of 20 most-used keystrokes. Histograms are generated from the all-keys data. Virtual keys codes 40, 38 and 17 correspond to the `down arrow key`, the `up arrow key` and the `control key`, respectively [47]. **a** Endpoint 5. **b** Endpoint 13

endpoint. Figure 2 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 are observed for the remaining endpoints.) Figure 3 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. 2. For one thing, the all-keys distribution of Fig. 3 is much more spread out than the session-key distribution of Fig. 2. 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. 3(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. There are two information-theoretic measures that can formally leverage this observation for real-time worm detection. The first measure is the entropy of the keystroke histogram observed in a time window. Since entropy quantifies the degree of dispersal or concentration of a probability distribution, according to Fig. 2 the keystroke entropy in a malware-infected window should be higher than the benign windows where only a few keystrokes are being used to initiate sessions. The second information-theoretic measure that we use to quantify the keystroke perturbations is mutual information. From Fig. 3 it can be deduced that in a benign time window mutual information of sessions and keystrokes that are used to initiate the sessions should be very high. On the other hand, in a malware-infected window this mutual information should decrease as the keystrokes will be drawn from the marginal all-keys distributions. The following sections formally describe the entropy and mutual information based detectors.

4.2 Malware detection using keystroke entropy

4.2.1 Definition of keystroke entropy

We define $X_n = \{p_i^n, i \in K_n\}$ as the histogram of keystrokes in a time-window n, where p_i^n is the number of times keystroke i was used in time-window n. Note that due to MSDN's virtual key code definition, $K_n = \{1, 2, \ldots, 255\}$. Let $p_n = \sum_{i \in K_n} p_i^n$ be the aggregate frequency of keystrokes observed in window n. Then sample entropy of the keystroke histogram for window n is

$$H(X_n) = -\sum_{i \in K_n} \frac{p_i^n}{p_n} \log_2 \frac{p_i^n}{p_n} \quad (1)$$

If there is no traffic in a window n (i.e., $p_n = 0$) then malware detection is not performed. Based on previous results, we know that for legitimate sessions, X_n has small variance and therefore the keystrokes' entropy should be low. On the other hand once a self-propagating malicious code starts initiating sessions, the keystrokes will be drawn from the marginal keystroke distribution of the all-keys data. Hence the variance and consequently the entropy of X_n should increase.

We compute keystroke entropy on a window-by-window basis. The results reported in this section use a window size of 60 s. In each window with one or more sessions, we compute the keystroke histogram X_n which is used in equation (1) to compute the entropy. The marginal keystroke histogram is generated from the first 500 entries of the all-keys data.

4.2.2 Entropy perturbations in the infected profiles

We use the infected profiles described in Sect. 3 to evaluate the performance of the entropy-based detector throughout this section. Since the present detector does not rely on source and destination ports, there is no need to evaluate against the simulated malware. Therefore, throughout this section we only focus on detection using the real worms collected for this study. When we used keystroke-entropy 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 results use an order-7 median filter.

The entropies of different endpoints randomly infected with a single infection of a malicious code are outlined in Fig. 4. It can be observed in Fig. 4 that keystrokes' entropy clearly highlights anomalous behavior in all cases. The increase in entropy is revealed for both high- and low-rate malware, and for endpoints with high and low session rates. Thus we conclude that entropy of keystroke histograms is a robust feature that can be leveraged for self-propagating malware detection on network endpoints.

4.3 Malware detection using session-key mutual information

In this section, in addition to the keystroke distribution, we also characterize the session information in a probabilistic framework. We show that the conditional mutual information of the session and keystroke distributions can clearly highlight anomalous behavior.

4.3.1 Mutual information of sessions and keys

Mutual information [45] 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

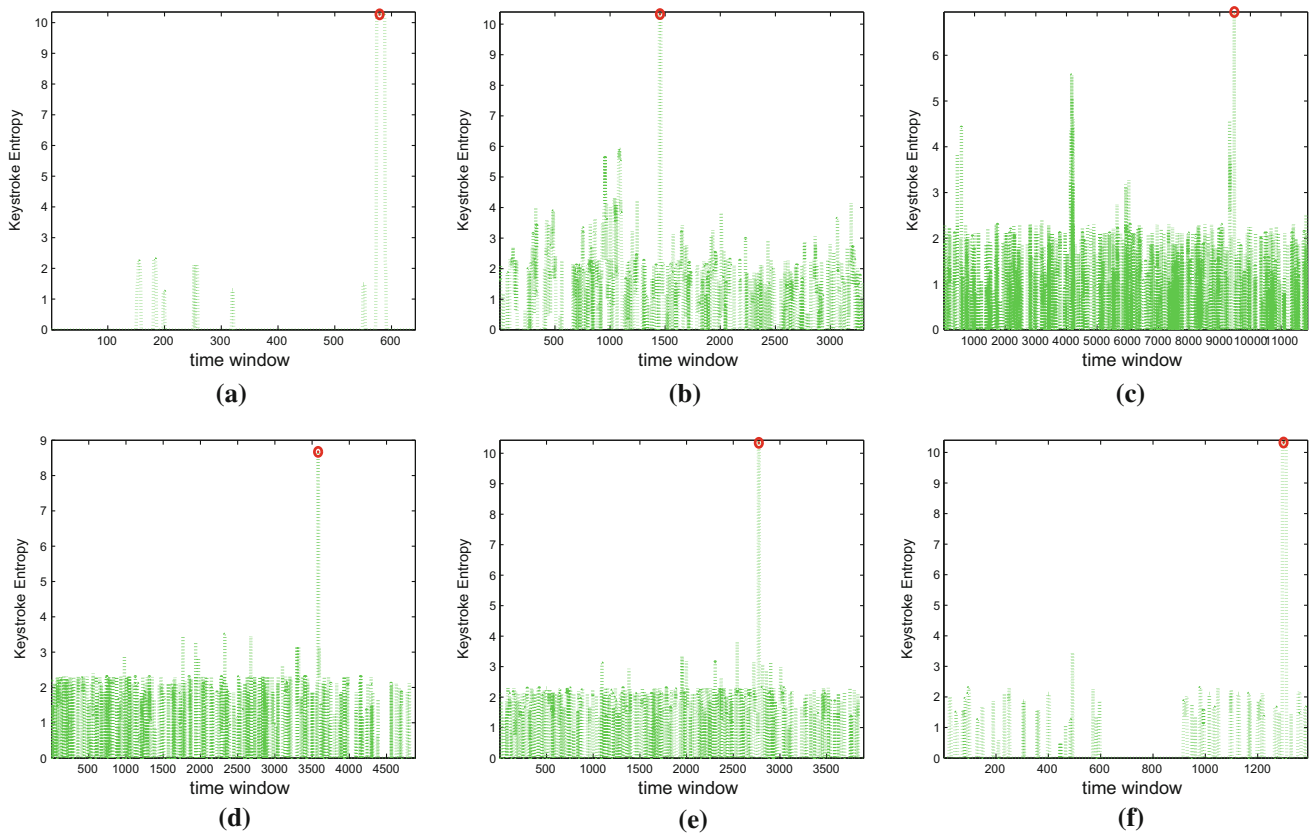

Fig. 4 Entropy of the keystroke histograms at infected endpoints. Each non-overlapping time-window is 60 s. **a** Endpoint 1, `Blaster`. **b** Endpoint 3, `Forbot-FU`. **c** Endpoint 6, `MyDoom-A`. **d** Endpoint 9, `Rbot-AQJ`. **e** Endpoint 11, `SoBig.E`. **f** Endpoint 13, `Zotob.G`

as

$$I(X;Y) = \sum_x \sum_y p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}. \quad (2)$$

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, $X \in \{0 \Rightarrow$ no session inwindow, $1 \Rightarrow$ one or more sessions in window$\}$. Moreover, we define Y as a random variable characterizing the probability of a keystroke, i.e., due to MSDN's keystroke definition [47] $Y \in \{1, 2, \ldots, 254\}$. It can be observed that the marginal $p(Y)$ distribution is simply the normalized all-keys histogram, such as the ones shown in Fig. 3. We derive the marginal $p(X)$ distribution using the first 500 entries of each endpoint's benign session-key profile. Basically, $p(X)$ 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.

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.

4.3.2 Mutual information perturbations in the infected profiles

Similar to the entropy-based keystroke perturbations, we observed some noisy mutual information spikes. Therefore, like the entropy-based technique we use an order-7 median filter to remove these spikes. The mutual information of different endpoints randomly infected with a single infection of a malicious code is outlined in Fig. 5. Clearly, session-keystroke mutual information clearly highlights anomalous behavior for both high- and low-rate malware and endpoints.

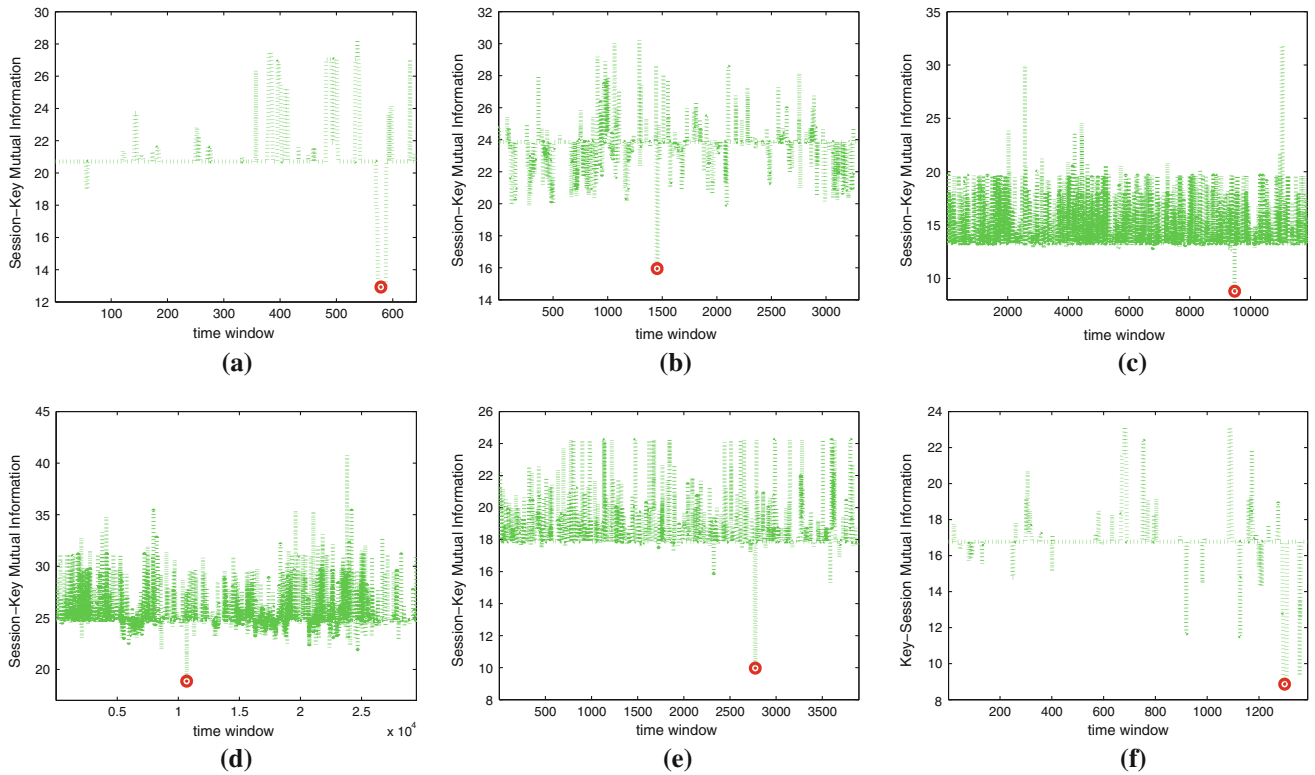

Fig. 5 Mutual information of the session and keystroke random variables at infected endpoints. **a** Endpoint 1, `Blaster`. **b** Endpoint 3, `Forbot-FU`. **c** Endpoint 6, `MyDoom-A`. **d** Endpoint 9, `Rbot-AQJ`. **e** Endpoint 11, `SoBig.E`. **f** Endpoint 13, `Zotob.G`

In the benign data, the mutual information is consistently high because only a few keys are used to initiate most of the sessions. Once compromised, the endpoint's marginal keystrokes get flagged as session initiation keys. The mutual information drops in Fig. 5 are because the marginal all-keys distribution has very little correlation with network sessions.

The keystroke-based measures proposed in this section are fairly independent of the rate of session initiation. This is a unique attribute of the present techniques because other network-based anomaly detectors implicitly or explicitly use this rate for detection. Consequently detection and false alarm rates of such detectors are dependent on the scanning rate of the malicious code. The techniques proposed in this section jointly consider sessions and keystrokes and are therefore not entirely dependent on the session rate.

In the following section, we develop an automated tool that uses keystroke entropy and mutual information values for real-time malware detection.

4.4 Automated detection using keystroke perturbations

As mentioned in previous sections, we use an order-7 median filter to filter out the noise in the keystroke entropy and mutual information values. To leverage the filtered entropy values in a real-time and automated fashion, we train the entropy detector using the first benign keystroke entropy values and the mutual information based detector is trained using the first benign mutual information values of an endpoint. We find the sample mean and sample standard deviation of the entropy values of an endpoint. An alarm is raised when the filtered entropy value observed in a window is more than the mean plus three standard deviations. Similarly, we find sample mean and sample standard deviation of the mutual information values. An alarm is raised when the filtered mutual information value in a window is less than the mean plus one standard deviation.

5 Performance evaluation and comparison with existing techniques

In this section, we evaluate the performance of the proposed worm detection techniques with three existing techniques proposed in [14,21,20]. Following is a brief description of these techniques. We majorly focus on the algorithm adaptation and parameter tuning for the datasets under consideration. Readers are referred to [14,20,21] for details of the algorithms.

5.1 Rate limiting

Rate limiting [19,21] detects anomalous connection behavior by relying on the premise that an infected host will try to connect to many different machines in a short period of time. Rate limiting detects portscans by putting new connections exceeding a certain threshold in a queue. An alarm is raised when the queue length, η_q, exceeds a threshold. Threshold values for each endpoint were generated as $\eta_q = \mu + k\sigma$, where μ and σ represent the sample mean and sample standard deviation of the connection rates in the training set, and $k = 0, 1, 2, \ldots$ is a positive integer. Large values of k will provide low false alarm and detection rates, while small values will render high false alarm and detection rates.

5.2 TRW with credit-based rate limiting (TRW-CB)

The original TRW algorithm [5] detects incoming portscans by noting that the probability of a connection attempt being a success should be much higher for a benign host than for a scanner. To leverage this observation, TRW uses sequential hypothesis testing (i.e., a likelihood ratio test) to classify whether or not a remote host is a scanner. A hybrid solution to leverage the complementary strengths of Rate Limiting and TRW was proposed by Schechter et al. [20]. Reverse TRW is an anomaly detector that limits the rate at which new connections are initiated by applying the sequential hypothesis testing in a reverse chronological order. A credit increase/decrease algorithm is used to slow down hosts that are experiencing unsuccessful connections. The right threshold value for each endpoint is selected by varying η_0 and η_1.

5.3 Maximum entropy method

This detector estimates the benign traffic distribution using maximum entropy estimation [14]. Training traffic is divided into 2,348 packet classes and maximum entropy estimation is then used to develop a baseline benign distribution for each packet class. Packet class distributions observed in real-time windows are then compared with the baseline distribution using the Kullback-Leibler (K-L) divergence measure. An alarm is raised if a packet class' K-L divergence exceeds a threshold, η_k, more than h times in the last W windows of t seconds each. Thus the Maximum Entropy method incurs a detection delay of at least $h \times t$ seconds.

Hence the rate-limiting detector is the only other technique that is designed specifically for endpoints, the maximum-entropy detector is one of the only two information-theoretic anomaly detection techniques and the credit-based threshold random walk detector is a hybrid solution to leverage the complementary strengths of the rate limiting detector and the soundness of the likelihood ratio test in original TRW detector. We use the same parameters values and learning/detection algorithms that were employed in [14,21] and [20]. We also tried to compare the performance of the proposed detector with the entropy-based technique by Lakhina et al. [13]. However, we observed that it was impractical to migrate the detector of [13] to endpoints because the detector required projection of high-dimensional feature metrics into benign and anomalous subspaces at a border router. On an endpoint, the same technique will result in only 3 possible subspaces, and in most cases it is not possible to classify them as benign and anomalous using the thresholding technique of [13].

We use the infected profiles for performance evaluation of the present malware detectors. Thus there are non-overlapping random infections of each malicious code in every endpoint's benign profile. As discussed earlier, each infection is approximately $T = 15$ minutes. Hence, all results provided in this section are averaged over one hundred experiments per endpoint per malicious code. We compute detection and false alarm rates for each experiment as follows. For 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 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 of the keystroke-entropy and mutual information based detectors are shown in Fig. 6. Figure 6(a) shows that the detection rate of the keystroke-entropy based technique is 100% for all endpoints and all malware. Detection rate of the mutual information detector is 100% for all endpoints except endpoint which has an average detection rate of 99.66%. Thus both the proposed detectors provide very high detection accuracy. Figure 6(b) shows that the mutual information detector has negligible false alarm rates. The keystroke-entropy detector has slightly higher false alarm rates than the mutual information detector; the highest false alarm rate of 2.39% was observed at endpoint 12. Hence, overall the both malware detector proposed in this section provide very high accuracy for the diverse set of endpoints and malware considered in this study.

Let us now compare the proposed detector and the maximum-entropy detector of [14]. From Fig. 6(a) it can be seen that the proposed keystroke-entropy as well as mutual information based techniques provide much higher detection rates than the maximum-entropy detector. Also, for the maximum-entropy detector, the false alarm rates for the home endpoints [endpoints 3 and 4] are extremely high. We believe that the high false alarm rates are due to peer-to-peer applications running on the home endpoints of this study. Moreover, maximum-entropy detector was designed for deployment at the perimeter, where even in a short period of time most of the

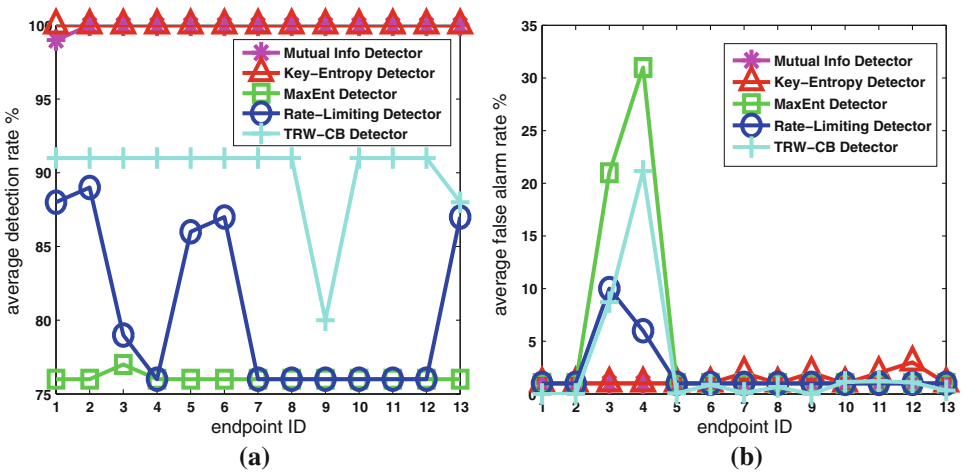

Fig. 6 Comparison of detection and false-alarm rates of the mutual information based and keystroke-entropy based malware detectors with maximum-entropy, rate-limiting and TRW-CB detectors. Each point is averaged over 9 malicious codes with 100 random infections per malicious code per endpoint. **a** Detection rate. **b** False-alarm rate

2,348 packet classes of [14] were observed. On an endpoint, many of these classes are not present in the benign training data. We observed that even if the maximum-entropy training is performed using a lot of benign data, the performance still does not improve. (The maximum-entropy model was trained using 100 and 1000 benign sessions, but the performance in both cases was identical.) Also note that due to the use of a sliding window, the maximum-entropy detector has higher training complexity and incurs an inherent detection delay that is not present in our detectors. The run-time complexities of the two techniques are comparable as the maximum-entropy technique requires frequent computation of K-L divergence over a large sample space of 2,348 outcomes, whereas our techniques train on small sample spaces.

For the rate-limiting detector, the detection rates of all endpoints except endpoint 2 are much lower than the proposed detection techniques. Also, much like the maximum-entropy detector, the false-alarm rates for home endpoints are quite high. (A false alarm is raised when the rate-limiter reports an anomaly, but the session queue of the rate limiter has no malicious sessions.) Thus the performance of the rate-limiting detector, although better than the maximum-entropy detector, is still much worse than the entropy as well as the mutual information based detection techniques proposed in this paper. The inferior performance of the rate-limiting detector shows that simply monitoring traffic volume at an endpoint is not sufficient. In addition to session volume, the actual characteristics of the traffic must also be taken into account for accurate detection.

The credit-based TRW approach (TRW-CB) achieves nearly 90% detection rate for all endpoints except for a university endpoint [endpoint 9]. This is considerably higher than the detection rates of both the maximum entropy detector as well as the rate limiting detector. A probable reason for this can be that the endpoint attack traffic contains mostly outgoing scan packets and the credit-based variant of TRW leverages outgoing scans for portscan detection. This is unlike the TRW detector, which makes use of incoming scans for portscan detection. Thus, TRW-CB combines the complementary strengths of rate limiting and TRW to provide a more accurate portscan detector for endpoints as compared to maximum entropy and the rate limiting detectors. This result agrees with earlier results in [59]. The false alarm rate pattern of TRW-CB matches those of maximum entropy and the rate limiting detectors. On the two home endpoints [endpoints 3 and 4] TRW-CB, however, performs worse than the rate limiting detector. However, the detection rates of the TRW-CB detector though being much better than the maximum entropy and the rate limiting detectors, are still less than the proposed techniques.

Based on the results of this section, we conclude that the worm detection techniques proposed in this paper provides significantly better performance than the techniques of [14,20,21].

6 Attacks and countermeasures

In this section, we discuss attacks that can circumvent the proposed worm detection techniques, and possible countermeasures to mitigate these attacks.

6.1 Mimicry attack

In a mimicry attack [53], a worm tries to hide its traffic inside benign traffic to avoid detection.

A mimicry attack can be launched against the keystroke-based detectors 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 keystroke- based detectors. To mitigate this attack, the time threshold for logging the session

initiation keystroke can be made adaptive. Also, we are currently investigating the efficacy of the keystroke-based detectors in a scenario when the last keystroke is always logged irrespective of the time elapsed since that keystroke.

6.2 Attack by acquiring system-level privileges

On an endpoint where security policies and user privileges are not appropriately defined, a worm after compromising the endpoint can gain system-level privileges and can in then disable the worm detector [38]. This vulnerability is a consequence of the design of contemporary operating systems and the lack of appropriate user rights management. All endpoint-based worm detectors suffer from this vulnerability. This attack can be mitigated by appropriate security policing and user management. To completely defeat this attack, a trusted computing platform [54] or a virtual machine [48] must be employed. Design of such operating systems is presently an area of active research [55–58].

7 Conclusion

In this paper, we proposed two information-theoretic malware detection techniques for network endpoints. These techniques made use of entropy and mutual information of keystrokes that are used to initiate network sessions to detect malware propagation. The proposed techniques were also compared with a few prominent anomaly detectors, namely the maximum entropy detector, the rate limiting detector and the TRW-CB based detector. Both the proposed techniques were highly accurate and provided significant improvements over existing methods.

References

1. Khayam, S.A., Radha, H.: Using session-keystroke mutual information to detect self-propagating malicious codes. In: IEEE ICC, June 2007
2. Ellis, D., Aiken, J.G., Attwood, K.S., Tenaglia, S.D.: A behavioral approach to worm detection. In: ACM Workshop on Rapid Malcode (WORM), October 2004
3. Zou, C.C., Gao, L., Gong, W., Towsley, D.: Monitoring and early warning of Internet worms. In: ACM Conference on Computer and Communications Security (CCS), October 2003
4. Wu, J., Vangala, S., Gao, L.: An effective architecture and algorithm for detecting worms with various scan techniques. In: Network and Distributed System Security Symposium (NDSS), February 2004
5. Jung, J., Paxson, V., Berger, A.W., Balakrishnan, H.: Fast portscan detection using sequential hypothesis testing. In: IEEE Oakland Symposium on Security and Privacy, May 2004
6. Weaver, N., Staniford, S., Paxson, V.: Very fast containment of scanning worms. In: Usenix Security Symposium, August 2004
7. Lakhina, A., Crovella, M., Diot, C.: Diagnosing network-wide traffic anomalies. In: ACM SIGCOMM, August/September 2004
8. Lakhina, A., Crovella, M., Diot, C.: Characterization of network-wide traffic anomalies in traffic flows. In: ACM Internet Measurement Conference (IMC), October 2004
9. Barford, P., Kline, J., Plonka, D., Ron, A.: A signal analysis of network traffic anomalies. In: ACM Internet Measurement Conference (IMC), November 2002
10. Krishnamurthy, B., Sen, S., Zhang, Y., Chen, Y.: Sketch-based change detection: methods, evaluation, and applications. ACM Internet Measurement Conference (IMC), October 2003
11. Soule, A., Salamatian, K., Taft, N.: Combining filtering and statistical methods for anomaly detection. In: ACM/Usenix Internet Measurement Conference (IMC), October 2005
12. Kim, Y., Lau, W.C., Chuah, M.C., Chao, H.J.: PacketScore: statistics-based overload control against distributed denial-of-service attacks. In: IEEE INFOCOM, March 2004
13. Lakhina, A., Crovella, M., Diot, C.: Mining anomalies using traffic feature distributions. In: ACM SIGCOMM, August 2005
14. Gu, Y., McCullum, A., Towsley, D.: Detecting anomalies in network traffic using maximum entropy estimation. In: ACM/Usenix Internet Measurement Conference (IMC), October 2005
15. Moore, D., Shannon, C., Voelker, G.M., Savage, S.: Network telescopes. CAIDA technical report. http://www.caida.org/outreach/papers/2004/tr-2004-04/
16. Cooke, E., Bailey, M., Mao, Z.M., Watson, D., Jahanian, F., McPherson, D.: Toward understanding distributed blackhole placement. In: ACM Workshop on Rapid Malcode (WORM), October 2004
17. Bailey, M., Cooke, E., Jahanian, F., Nazario, J., Watson, D.: The Internet motion sensor: a distributed blackhole monitoring system. In: Network and Distributed System Security Symposium (NDSS), February 2005
18. Dagon, D., Qin, X., Gu, G., Lee, W.: HoneyStat: local worm detection using honeypots. In: International Symposium on Recent Advances in Intrusion Detection (RAID), September 2004
19. Twycross, J., Williamson, M.M.: Implementing and testing a virus throttle. In: Usenix Security Symposium, August 2003
20. Schechter, S.E., Jung, J., Berger, A.W.: Fast detection of scanning worm infections. In: RAID (2004)
21. Williamson, M.M.: Throttling viruses: restricting propagation to defeat malicious mobile code. In: Annual Computer Security Applications Conference (ACSAC), December 2002
22. Sellke, S., Shroff, N.B., Bagchi, S.: Modeling and automated containment of worms. In: International Conference on Dependable Systems and Networks (DSN), June/July 2005
23. Whyte, D., Kranakis, E., van Oorschot, P.C.: DNS-based detection of scanning worms in an enterprise network. In: Network and Distributed System Security Symposium (NDSS), February 2005
24. Gupta, A., Sekar, R.: An approach for detecting self-propagating email using anomaly detection. In: International Symposium on Recent Advances in Intrusion Detection (RAID), September 2003
25. Xiong, J.: ACT: attachment chain tracing scheme for email virus detection and control. In: ACM Workshop on Rapid Malcode (WORM), October 2004
26. Me, L., Michel, C.: Intrusion detection: a bibliography. Tech. Rep. SSIR-2001-01, September 2001
27. Cui, W., Katz, R.H., Tan, W.-T.: BINDER: an extrusion-based break-in detector for personal computers. In: Usenix Security Symposium, April 2005
28. Ilgun, K., Kemmerer, R.A., Porras, P.A.: State transition analysis: a rule-based intrusion detection approach. IEEE Trans. Softw. Eng. **21**(3), 181–199 (1995)
29. Jha, S., Tan, K., Maxion, R.A.: Markov Chains, classifiers, and intrusion detection. In: IEEE CSFW, June 2001
30. Ye, N.: A Markov Chain model of temporal behavior for anomaly detection. In: IEEE Workshop on Information Assurance and Security, June 2000

31. DuMouchel, W.: Computer intrusion detection based on bayes factors for comparing command transition probabilities. Tech. Rep. 91, National Institute of Statistical Sciences (1999)
32. Lazarevic, A., Ozgur, A., Ertoz, L., Srivastava, J., Kumar, V.: A comparative study of anomaly detection schemes in network intrusion detection. In: SIAM Conference on Data Mining, May 2003
33. Lippmann, R.P., et al.: The 1998 DARPA/AFRL off-line intrusion detection evaluation. In: RAID, September 1998
34. Lippmann, R.P., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. ACM Comput Netw **34**(4), 579–595 (2000)
35. Endpoint Security Homepage. http://www.endpointsecurity.org/
36. Symantec Internet Security Threat Report XI. Trends for July–December 07. March 2007
37. Raschke, T.: The new security challenge: endpoints. IDC/F-Secure, August 2005
38. Weaver, N., Ellis, D., Staniford, S., Paxson, V.: Worms vs. perimeters: the case for hard-LANs. In: IEEE Symposium on High Performance Interconnects (Hot Interconnects), August 2004
39. Wong, C., Wang, C., Song, D., Bielski, S., Ganger, G.R.: Dynamic quarantine of Internet worms. In: International Conference on Dependable Systems and Networks (DSN), July 2004
40. Wong, C., Bielski, S., Studer, A., Wang, C.: Empirical analysis of rate limiting mechanisms. In: International Symposium on Recent Advances in Intrusion Detection (RAID), September 2005
41. Li, Q., Chang, E.-C., Chan, M.C.: On effectiveness of DDOS attacks on statistical filtering. IEEE Infocom, March 2005
42. Kuzmanovic, A., Knightly, E.W.: Low-rate TCP-targeted denial of service attacks. In: ACM SIGCOMM, August 2003
43. Staniford, S., Paxson, V., Weaver, N.: How to own the Internet in your spare time. In: Usenix Security Symposium, August 2002
44. Panjwani, S., Tan, S., Jarrin, K.M., Cukier, M.: An experimental evaluation to determine if port scans are precursor to an attack. In: International Conference on Dependable Systems and Networks (DSN), June/July 2005
45. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley-Interscience, New York (1991)
46. SHA-1. The Secure Hash Algorithm. FIPS PUB 180-1, April 1995
47. MSDN Library. http://msdn.microsft.com
48. Microsoft Virtual PC 2004. http://www.microsoft.com/Windows/virtualpc
49. Sophos Virus Info. http://www.sophos.com/virusinfo/
50. Symantec Security Response. http://securityresponse.symantec.com/avcenter
51. TrendMicro Virus Encyclopedia. http://au.trendmicro-europe.com/smb/vinfo
52. Kumar, A., Paxson, V., Weaver, N.: Exploiting underlying structure for detailed reconstruction of an Internet-scale event. In: ACM/Usenix Internet Measurement Conference (IMC), October 2005
53. Wagner, D., Soto, P.: Mimicry attacks on host-based intrusion detection systems. In: ACM CCS, November 2002
54. Trusted Computing Alliance. https://www.trustedcomputinggroup.org
55. Dunlap, G., King, S., Cinar, S., Basrai, M., Chen, P.: ReVirt: enabling intrusion analysis through virtual-machine logging and replay. Usenix OSDI, December 2002
56. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: a virtual machine-based platform for trusted computing. ACM SOSP, October 2003
57. Lampson, B.W.: Computer security in the real world. IEEE Comput. **37**(6), 37–46 (2004)
58. Rosenblum, M., Garfinkel, T.: Virtual machine monitors: current technology and future trends. IEEE Comput. **38**(5), 39–47 (2005)
59. Wong, C., Bielski, S., Studer, A., Wang, C.: Empirical analysis of rate limiting mechanisms. In: RAID (2005)