This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2008 proceedings.

Worm Detection at Network Endpoints Using Information-Theoretic Traffic Perturbations

Syed Ali Khayam*
NUST Institute of IT (NIIT)
National Univ. of Science & Tech. (NUST)
Rawalpindi, Pakistan
khayamsy@niit.edu.pk

Hayder Radha†
Dept. of Electrical & Computer Engg.
Michigan State University
East Lansing, MI 48824, USA
radha@egr.msu.edu

Dmitri Loguinov
Dept. of Computer Science
Texas A&M University
Station, TX 77843, USA
dmitri@cs.tamu.edu

Abstract— **In this paper, we propose an endpoint-based anomaly detection scheme that detects computer worms by comparing the current traffic patterns of each host to the corresponding *benign* traffic profile of the host. To detect deviations in the traffic patterns, we employ the information-theoretic Kullback-Leibler (K-L) divergence measure which estimates the distance between the distribution of source/destination ports engaged in current communication and that observed in the legitimate host traffic collected earlier. We use a small subset of traces obtained from endpoints in home, university, and office environments to build benign traffic profiles of studied endpoints. Endpoint traces are then infected with both real and simulated worms to examine the performance of our detection mechanism. To perform automated, real-time worm detection, we use Support Vector Machines (SVMs) that are trained using the K-L divergence values. Our results show that the proposed worm detector provides almost 100% detection with negligible false-alarm rates and significantly surpasses the accuracy of existing anomaly detectors.**

I. Introduction

Effective containment of rapidly evolving worms and viruses requires real-time defense mechanisms that can detect *novel* (i.e., previously unknown) attacks. To this end, network-based anomaly detectors attempt to flag behavior that is anomalous for a network or a networked entity [4], [6], [7], [13], [17]. Recent statistics show that increasingly *network endpoints*[1], comprising client machines at homes and offices, are serving as extremely potent and viable launch pads and carriers for worm and virus infections [10]. Thus it is important that real-time defenses be developed specifically for network endpoints.

Network-based anomaly detection on an endpoint requires a model of benign/legitimate behavior, characterizing features that will get perturbed if the endpoint is compromised by any past, present, or future worm. Malicious activity can then be detected by observing deviations from the benign behavior in the traffic transmitted by the host. This paper quantifies such behavioral features using information-theoretic tools and then leverages these features for real-time worm detection at network endpoints.

To obtain benign profiles 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. For malicious activity, we use real and simulated worms. These worms vary in their propagation rates and scanning techniques. In this dataset, we observe that the vulnerabilities targeted by all tested worms are associated with a small number of source or destination ports. Thus, on a compromised machine, the distribution of source or destination ports on which a host communicates is perturbed after infection and is easily quantified using information-theoretic measures. We propose to use the Kullback-Leibler (K-L) divergence measure [2] to characterize perturbations in source and destination port profiles as a means of detecting attempts of worm propagation. Our results show that K-L divergence of port histograms is perturbed significantly on compromised endpoints, which allows very accurate detection of malicious activity by simply observing each host's traffic.

To create an automated detection tool based on the proposed technique, we use the K-L divergence values to train Support Vector Machines (SVMs) [1]. The trained SVMs are then tested by embedding malicious traffic at multiple random instances in the benign traffic profiles. For all our experimental evaluations, we observe that the proposed detector provides almost 100% detection accuracy and negligible false-alarm rates, and easily surpasses the accuracy of existing maximum-entropy [4] and rate limiting [17] detectors.

II. Related Work

There is significant research literature on network- and host-based anomaly detection. However, in this paper we focus on and compare performance with endpoint-based and information-theoretic anomaly detectors that are directly related to the present work. This section provides brief description of these existing detectors.

The only endpoint-based network-level worm detection technique that the authors are aware of involves rate limiting. This technique proposed by Twycross and Williamson [13],

*This work was supported in part by the Pakistan National ICT R&D Fund.
†This work was supported in part by NSF Awards CNS-0721550, CCF-0515253, and CNS-0430436.

[1] We define a network endpoint as "an individual computer system or device that acts as a network client and serves as a workstation or personal computing device." [3]

978-1-4244-2075-9/08/$25.00 ©2008 IEEE

[17] limits the rate of an endpoint's network traffic to curb and detect worm propagation. Sellke *et al.* [7] extend rate limiting by proposing a branching worm propagation model and using this model develop a window-based rate limiting mechanism.

There are two recent studies that have proposed information theoretic measures for anomaly detection. The first technique proposed by Lakhina *et al.* [6] detects anomalies using entropies of the distributions of source ports, destination ports, and origin/destination pairs. The second study by Gu *et al.* [4] uses maximum-entropy parameter estimation to quantify a baseline distribution of benign activity at a network gateway or router. In [4], packets are classified into $2,348$ distinct classes based on their destination ports and protocol information, where the probability of each class is learned from benign data using maximum-entropy estimation.

III. Data Collection and Simulation

In this section, we explain the two main (benign and malicious) datasets collected for this study.

A. Benign Traffic Profiles

We collected network profiles of a diverse set of 13 endpoints for over 12 months. Users of these endpoints included home users, research students, and technical/administrative staff with Windows 2000/XP laptop and desktop computers. Data were collected by a multi-threaded windows application called `argus` which runs as a background process storing network activity. `argus` only logs session-level information, where a *session* corresponds to bidirectional communication between two IP addresses. Communication between the same IP address. 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.

Each entry of the log file has the following 6 fields:

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

whose explanation is given below:

- `session id`: 20-byte SHA-1 hash of the concatenated hostname and remote IP address;
- `direction`: one byte flag indicating outgoing unicast, incoming unicast, outgoing broadcast, or incoming broadcast packets;
- `proto`: transport-layer protocol of the packet;
- `src port`: source port of the packet;
- `dst port`: destination port of the packet;
- `timestamp`: millisecond-resolution time of session initiation.

We evaluate a total of $1,881,235$ sessions for the 13 endpoints. The total number of sessions per endpoint vary from $11,996$ sessions for endpoint 13 to $444,345$ sessions for endpoint 4. The mean session rate varies from 0.19 sessions per sec to 5.28 sessions per sec. 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.

B. Worm Classification

Before describing the worm traffic data, we define terminology that will be used throughout the paper.

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 call the destination ports of a worm *attack* and source ports *non-attack*. In the latter case, the roles are reversed and we call source ports *attack* and destination ports *non-attack*. With the exception of the `Witty` worm [11], all worms used in this study are destination-port worms.

C. Real Worms

We installed original and unpatched releases of Windows 2000 and Windows XP on a computer using virtual machines (VMs). The advantage of using VMs was that once a virtual host was infected, we could reinstall it by overriding just a few key files. These VMs were then compromised by `Zotob.G`, `Forbot-FU`, `Sdbot-AFR`, and `Dloader-NY`. (Details of the worms used in this paper can be found at [9], [11], or [12].) Through our research collaborators, we acquired `SoBig.E@mm` and the C source code of `MyDoom.A@mm`. Finally, we downloaded binaries or source codes of the following worms from the Internet: `Blaster`, `Rbot-AQJ`, and `RBOT.CCC`.

The worms used in this work 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. While these worms provided us with a good base for evaluating our proposed technique, we also aimed at testing our method against an even broader class of attacks. Consequently, we simulated three additional worms which are described next.

D. Simulated Worms

We first simulated the source port `Witty` worm [8], [11]. To simulate this worm, we use the pseudo random number generator parameters and pseudo code provided in [5]. We test the worst-case scenario with $20,000$ scan packets at the average scan rate of 357 sps.

We also simulate the HTTP-based CodeRed II worm using an average scan rate of 4.95 sps [11] .We acknowledge that it is unlikely that an endpoint will be running a service that can be infected by an HTTP worm. Nevertheless, we simulate an HTTP worm because its scan packets use destination port 80, which is a very common port in the benign profile of an endpoint.

Finally, we also simulate a source-port worm that sends scan packets with a fixed TCP source port of 1500 at an average scan rate of 3.57 sps; note that this scan rate is exactly 100 times less than Witty's average scan rate, which makes this simulated worm challenging to detect.

E. 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 argus 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 except Witty, we use $T = 15$ minutes and to simulate the worst-case behavior of Witty, we insert only $20,000$ scan packets (i.e., approximately $T = 1$ minute of malicious traffic) in each Witty-infected profile.

IV. Information-Theoretic Traffic Perturbations

Since worms target vulnerabilities on services running on some fixed ports, it can be intuitively argued that the usage frequency of an attack port should get perturbed on a compromised host. Similarly, every new outgoing scan packet originating at a compromised endpoint is assigned a distinct source port. Consequently, the number of source ports that are actively communicating with other hosts on the Internet should differ between normal and infected hosts. Thus number and usage frequency of communication ports can potentially serve as very effective discriminant features for worm detection.

In this section, we first generate benign distributions of source and destination port usage from training data. Then we show that the port distributions change considerably once a host is compromised by a worm.

A. Comparison of Port Distributions using Information Divergence

Frequency histograms of source and destination ports of outgoing packets are generated using a 20-second window (other window sizes produce qualitatively similar results). Source and destination port histograms for each window are computed by counting the number of times a particular port is used during the window. Throughout this paper, we focus solely on outgoing unicast traffic since incoming unicast packets can be easily blocked using firewalls.

Lakhina *et al.* [6] in a recent work showed that sample entropy of traffic feature distributions at a border router can reveal traffic anomalies. We observed that port distributions' entropy is not an appropriate feature to detect traffic anomalies at an endpoint because entropy does not take port values into consideration. Therefore, we propose to use the information-theoretic Kullback-Leibler (K-L) divergence measure to quantify perturbations in port distributions. K-L divergence [2] is an information-theoretic measure of the similarity or dissimilarity between two probability distributions.

Let us denote the *benign* source and destination port histograms derived from an endpoint's benign profile as $X = \{p_i, i \in S\}$ and $Y = \{q_j, j \in D\}$, where S and D respectively denote the sets of source and destination ports observed in the benign profile. Then the K-L divergence between the benign and currently observed port histograms can be expressed as:

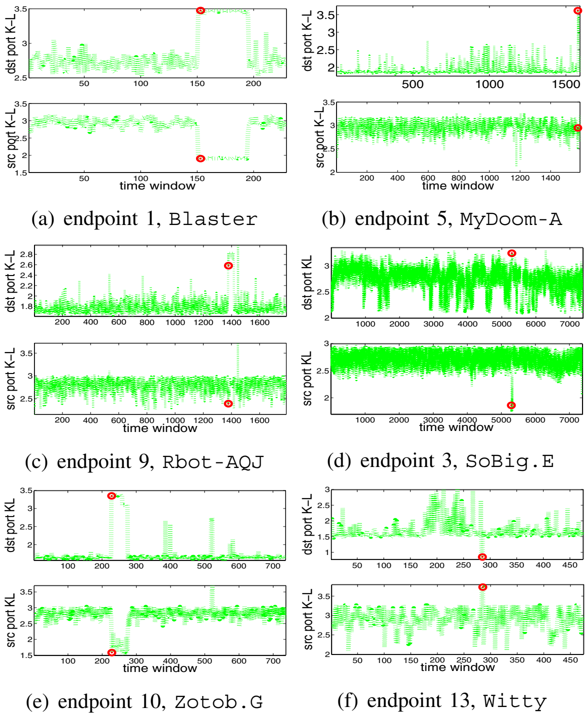
$$\begin{aligned} D(X_n||X) &= \sum_{i \in S_n} \frac{p_i^n}{p_n} \log_2 \frac{p_i^n/p_n}{p_i/p}, \\ D(Y_n||Y) &= \sum_{j \in D_n} \frac{q_j^n}{q_n} \log_2 \frac{q_j^n/q_n}{q_i/q}, \end{aligned} \tag{1}$$

where $p = \sum_{i \in S} p_i$ and $q = \sum_{j \in D} q_j$ respectively represent the aggregate source and destination port frequencies observed in the benign profile. Note that K-L divergence is an asymmetric measure. The advantages of using window-based metrics X_n and Y_n as primary distributions of the K-L divergence are twofold: 1) fewer sessions are observed in a window as opposed to the benign profile, $|S_n| < |S|$ and $|D_n| < |D|$, which reduces the complexity of real-time detection; and 2) better detection accuracy can be achieved if we focus on the specific ports engaged in communication during the current window n.

We generate port histograms of benign profiles using the first 100 sessions on an endpoint. The training time for the endpoints of this study ranged between 12 hours to 5 days, with an average of approximately 2 days. We train with only 100 sessions to quantify worst-case performance of the proposed detector.

To effectively leverage K-L divergence in the present endpoint-based anomaly detector, we introduce the following provisions. First, in (1), if $p_i = 0$ and $p_i^n > 0$, for any i, then $D(X_n||X)$ is set to ∞. This disparity also holds for $D(Y_n||Y)$ in (1). In other words, X and Y must be continuous with respect to X_n and Y_n, respectively. To achieve this, before training we initialize the benign histograms with $p_i = 1$ and $q_i = 1$, for $i = 0, \ldots, 65535$, which assigns never-used ports very small, non-zero frequencies.

Second, it is well-known that scaling of training data improves the performance of learning tools by making the training process better behaved and by mitigating the bias



(a) endpoint 1, `Blaster`

(b) endpoint 5, `MyDoom-A`

(c) endpoint 9, `Rbot-AQJ`

(d) endpoint 3, `SoBig.E`

(e) endpoint 10, `Zotob.G`

(f) endpoint 13, `Witty`

Fig. 1. Source and destination ports' Kullback-Leibler divergences at infected endpoints.

towards larger input values. Therefore, we normalize the K-L divergence values by a constant factor.

Finally, to reduce complexity and to filter out noise due to benign data, we introduce a provision to ignore overtly benign behavior. From the training data, we generate a histogram of session volume (i.e., total number of sessions) in a window. After normalization, we compute the histogram's mean μ_e and variance σ_e^2 for each endpoint e. We invoke worm detection only when the total number of sessions observed in a window is greater than $\gamma = \lceil \mu_e + \sigma_e \rceil$. The value of γ varied between 3 and 13 sessions per 20 second window, with an average of 6.6 sessions per 20 second window, for the endpoints considered in this study.

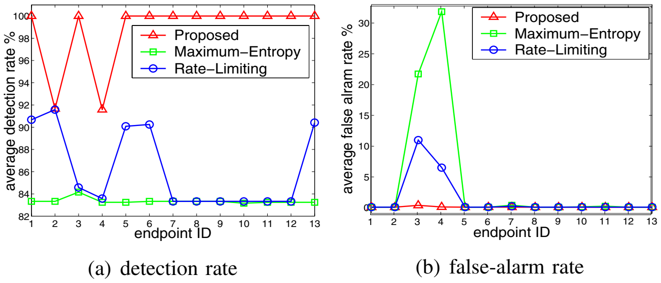
B. K-L Perturbations on Infected Hosts

The K-L divergences of different endpoints randomly infected with a single infection of each worm are outlined in Fig. 1. From Fig. 1 it is clear that the K-L divergence highlights anomalous behavior in the port distributions. For high scan rate worms (`Blaster`, `SoBig`, `Zotob`, `Witty`), perturbations in both attack and non-attack ports are identified, while for low-rate worms (`MyDoom`, `Rbot`) only the attack ports' distribution is perturbed.

In the following section, we train a machine learning tool using the K-L divergence of benign and malicious data, which is then used for automated worm detection.

V. Detection Using Support Vector Machines

In this section, we train Support Vector Machines (SVMs) [1] for real-time worm detection. We first train the SVMs using K-L divergence values derived from a subset of the benign profiles and worm data. The SVMs are then used to detect worms in the infected profiles. We also compare the performance of the proposed detector with existing techniques.



(a) detection rate

(b) false-alarm rate

Fig. 2. Comparison of detection and false-alarm rates of the proposed K-L/SVM-based worm detector with maximum-entropy and rate-limiting detectors. Each point is averaged over 12 worms with 100 random infections per worm per endpoint.

A. SVM Training

We used a small subset of malicious and benign data to train a C-SVM with a degree-3 radial basis kernel function [1]. We use the source and destination ports' K-L divergence values to train two SVMs for each endpoint. To train the SVMs, we take ten K-L divergence values from the benign traffic profile. These values comprise the positive examples. We then take a total of 13 negative examples by computing K-L divergence of benign traffic windows with `Blaster`- and `Witty`-infected windows.

B. Performance Evaluation and Comparison with Existing Techniques

In this section, we evaluate the performance of the proposed worm detector with two existing techniques proposed in [4] and [17].

We inserted 100 random and non-overlapping infections of each worm in every endpoint's benign profile. We compute detection and false alarm rates for each experiment as follows. For 100 infections of a particular worm on an endpoint, the percentage detection rate for that worm is computed by simply counting the number of infections that are detected by the worm 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. 2. It can be seen in Fig. 2(a) that the detection rate of the proposed K-L/SVM-based detector is 100% for all endpoints except endpoints 2 and 4; for endpoints 2 and 4, some instances of the low-rate `MyDoom-A` and `Rbot-AQJ` worms were not detected. Nevertheless, even for endpoints 2 and 4, the average detection rate is above 90%. Except for these two endpoints, detection rates of the maximum entropy and the rate limiting detectors are significantly $(10-17\%)$ lower than the proposed detector.

Moreover, the proposed detector has negligible false alarm rates at all endpoints. The highest false alarm rate we observed was approximately 0.45%, with endpoints 1, 6, 7, and 8 exhibiting almost zero false alarms. The false alarm rates of the maximum-entropy and rate limiting detectors are much higher than the proposed detector, especially in the case of endpoints 3 and 4. Both of these endpoints are home endpoints which were running peer-to-peer and multimedia applications. Since these endpoints generate significant traffic under benign conditions, detectors that rely solely on traffic rate (e.g., the rate limiting detector) fail to detect anomalous activity on these endpoints. Similarly, the maximum-entropy detector is designed for deployment at the perimeter, where even in a short period of time most of the $2,348$ packet classes of [4] are 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.)

VI. Attacks and Countermeasures

A. Mimicry Attack

Three mimicry attacks [15] can be launched against the worm detector proposed in this paper. Under the first attack, a worm can use non-attack ports that are frequently used by an endpoint. While this attack can mimic non-attack ports, mimicry of attack ports is not possible because vulnerabilities targeted by a worm are associated with fixed ports, and consequently the attack ports of outgoing scan packets are fixed. Thus, even with mimicked non-attack ports, the proposed detector can detect perturbations in the attack port distribution, as shown by the `CodeRed II` results in Section V.

Another type of mimicry attack can be launched by generating deceptive packets in order to maintain an attack port distribution that is similar to the benign port distribution. While the attack port perturbations can be hidden using this attack, the non-attack port distribution will get perturbed because the worm must send each scan packet using a distinct non-attack port.

An effective mimicry attack can be launched by a very low-rate worm which can hide its traffic within benign traffic, while keeping the total number of sessions under γ, where γ [defined in Section IV-A] is the threshold number of sessions below which worm detection is not invoked. As mentioned in Section IV-A, for the endpoints of this study the values of γ were very small; ranging between 0.15 and 0.65 sessions per minute, with an average of 0.33 sessions per minute. A mimicking worm with less than γ sessions per time-window will have a very slow propagation rate, and hence will allow human countermeasures.

1) 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 turn disable the worm detector [16]. 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 [14] or a virtual machine must be employed.

VII. Conclusion

We used actual network and worm traffic to show that source and destination ports' distributions get substantially perturbed when an endpoint is compromised by a worm. We demonstrated that an endpoint's benign profile of source and destination ports' distributions can be developed using very little training data. Anomalies can then be detected using information divergences of the traffic in the current window and the benign profiles of source and destination ports. We trained support vector machines using K-L divergence values, which provided very high detection rates and extremely low false-alarm rates, thus providing considerably better performance than existing detectors.

References

[1] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min and Know Disc*, 2(2) 121–167, 1998.

[2] T. M. Cover and J. A. Thomas, "Elements of Information Theory," Wiley-Interscience, 1991.

[3] Endpoint Security Homepage, `http://www.endpointsecurity.org/`.

[4] Y. Gu, A. McCullum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," *ACM/Usenix IMC*, 2005.

[5] A. Kumar, V. Paxson, and N. Weaver, "Exploiting underlying structure for detailed reconstruction of an Internet-scale event," *ACM/Usenix IMC*, 2005.

[6] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *ACM SIGCOMM*, 2005.

[7] S. Sellke, N. B. Shroff, and S. Bagchi, "Modeling and automated containment of worms," *International DSN*, 2005.

[8] C. Shannon and D. Moore, "The spread of the Witty worm," *IEEE Sec & Priv*, 2(4) 46–50, 2004.

[9] Sophos Virus Info, `http://www.sophos.com/virusinfo/`.

[10] "Symantec Internet Security Threat Report XI – Trends for July – December 07," March 2007.

[11] Symantec Security Response, `http://securityresponse.symantec.com/avcenter`.

[12] TrendMicro Virus Encyclopedia, `http://au.trendmicro-europe.com/smb/vinfo`.

[13] J. Twycross and M. M. Williamson, "Implementing and testing a virus throttle," *Usenix Security Symp*, 2003.

[14] Trusted Computing Alliance, `https://www.trustedcomputinggroup.org`.

[15] D. Wagner and P. Soto, "Mimicry Attacks on Host-Based Intrusion Detection Systems," *ACM CCS*, Nov. 2002.

[16] N. Weaver, D. Ellis, S. Staniford, and V. Paxson, "Worms vs. Perimeters: The case for Hard-LANs," *IEEE Hot Interconnects*, 2004.

[17] M. M. Williamson, "Throttling viruses: Restricting propagation to defeat malicious mobile code," *ACSAC*, 2002.