ITC19/ Performance Challenges for Efficient Next Generation Networks 1727-1736
LIANG X.J. and XIN Z.H.(Editors)
V.B. IVERSEN and KUO G.S.(Editors)
Beijing University of Posts and Telecommunications Press

Emulation of "single-packet" UDP Scanning Worms in Large Enterprises

Lunquan Li*, Soranun Jiwasurat†, Peng Liu‡, George Kesidis§

School of Information Sciences and Technology, The Pennsylvania State University
305 IST Building University Park, PA 16802, USA
Email: lli@ist.psu.edu*

Department of Computer Science and Engineering, The Pennsylvania State University
IST Building University Park, PA 16802, USA
Email: soranun@psu.edu†

School of Information Sciences and Technology, The Pennsylvania State University
313G IST Building University Park, PA 16802, USA
Email: pliu@ist.psu.edu‡

Department of Computer Science and Engineering, The Pennsylvania State University
338J IST Building University Park, PA 16802, USA
Email: kesidis@engr.psu.edu§

Abstract: Worms are a serious threat to Internet security. The past research on worm has been focused on mathematical modeling, numerical analysis, and simulation in addition to proposed defense strategies. We believe a fine-grained, packet-level emulation of worm propagation in enterprise networks is highly beneficial for the deep understanding of worm dynamics and a prerequisite for worm containment analysis. In this paper, we propose a virtual-node approach and an Internet scanning model to run such a worm emulation in a resource-limited testbed. The results from our validation experiments using virtual nodes and other emulation approaches show that our virtual node approach can realize the same level of fidelity while using much fewer testbed nodes. The insights we gained and the lessons we learned in doing worm experiments will be valuable to a variety of enterprise network worm-recreation and defense-evaluation research.
keywords: Emulation, Simulation, Worm, Slammer, Enterprise, Virtualization, Throughput

1. Introduction

Recently, the Internet has experienced a series of self-propagating *worm* attacks (e.g., Code-Red, SQL Slammer, Blaster, etc.) that have caused significant disruption to financial, transportation, and government institutions, among others. The severeness of the experienced worm effects clearly show that worm is a serious threat to Internet security. There are many ongoing research efforts to study the worm dynamics and devise ways to combat worm attacks. And all these efforts have an urgent need on worm experiment methodologies for evaluation and validation.

1.1. Experimental Methodologies for Worm Research

The experimental methodologies for worm research can be classified into the following categories.

- Mathematical analysis and modeling - various models have been proposed to study the worm propagation on the Internet, many based on epidemics models of biology [20], [22]. Normally the results from these modeling analyses match the results from real world monitoring (e.g., [13]), but their underlying assumption of uniform distribution on a simple topology becomes invalid when dealing with worm propagation and defense on a enterprise network where topology and hardware/software configurations vary considerably.
- Simulation - simulation packages such as Network Simulator (NS)[18] and Scalable Simulation Framework (SSF) [17] can be used to reproduce the worm behavior in a more detailed level; but simulation on a simple CPU or multiple CPUs can only handle a rather small topology without losing substantial fidelity.

http://www.ist.edu/deter
This research was funded by the NSF and the DHS under NSF grant ANI-0335241, see http://emist.isi.psu.edu The corresponding author for this paper is Lunquan Li, lli@ist.psu.edu

- Emulation - the fidelity of emulation is highest compared with numerical analysis and simulation. Ideally emulation should be run by one-to-one fashion between real world host and experiment node to get the most accurate results, which makes scalability a big concern for emulation. Emulation can be run on a production network, a testbed network, or an overlay network.
- Hybrid - to balance the fidelity and scalability, various hybrid approaches have been proposed, including the combination of emulation and simulation, simulation with scaling-down, etc.

Although analytical models of worm propagation are well studied in the literature [22], [21], worm simulation and emulation are quite preliminary. In [19], proprietary worm simulations are done but only coarse-grained global worm propagation activities are simulated with very high level abstraction of the worm propagation environment. In [10], SSFNet is used to simulate realistic network worm traffic for worm warning system design and testing, but only at an abstract network level.

1.2. Testbed for Emulation

A testbed supplies a flexible environment with which researchers can configure arbitrary network topologies and test new protocols or security defense technologies. The objectives of DETER (The Cyber Defense Technology Experimental Research) project together with EMIST (Evaluation Methods for Internet Security Technology) project are to "build an effective experimental and testing environment and to develop a corresponding experimental methodology, for Internet security issues and defense mechanisms" [1], [15]. The currently deployed ISI segment of DETER is a 72-node clustered Emulab [3] testbed. All experimental nodes in DETER are connected by one fast switch and node connections are formed by *LAN* or *link* which are actually constructed using the *VLAN* feature of the switch. Basic static routing is supported and more advanced routing can be implemented by running special routing software on the intermediate nodes.

The scale of the testbed makes it possible to emulate worm propagation in large enterprise networks with fidelity so that worm spreading can be recreated and the corresponding detections or defenses can be evaluated and validated. Several worm simulation/emulation experiments have already conducted on DETER. In [7], the need for advanced worm experiment utilities is identified. In [20], the idea of using scale-down to explore worm dynamics is investigated via DETER simulations. In [14], a hybrid quarantine defense is proposed and validated by both NS2 simulations and DETER emulations. The preliminary experiments show that reproducing enterprise worm propagation in a testbed environment using emulation and simulation is feasible and can yield detailed results that simulation alone cannot accomplish.

2. Emulation of UDP Scanning Worms in Large Enterprises

We believe a fine-grained, packet-level emulation or simulation of worm propagation in enterprise networks is highly beneficial for the deep understanding of worm dynamics and a missing link between pure abstract simulation and product testing in a production network. As a pilot study, we want to emulate the propagation of UDP scan worm (Sapphire Slammer [13]) in an enterprise network. In this emulation, we wanted to observe the effect of Internet incoming scans on the enterprise network; the worm propagation within the network; the traffic dynamics under worm propagation; and other fine details that modeling or simulation cannot yield.

2.1. Emulation Challenges on Testbed

There are some challenging issues to run such emulation on a resource-limited testbed, including:

⊙ Scale-down: Large enterprise networks could span multiple class-B IP subnets and may include thousands of nodes. For simplicity, we chose a mid-scale network as the experiment network that has one thousand nodes. However, even this mid-scale network employing a one-to-one emulation approach entails substantial resources that a normal testbed cannot support. The Emulab testbed has fewer than 200 nodes; DETER testbed has 72 nodes. To emulate an enterprise network using such testbeds, we need some kind of scale-down or virtualization while keeping the fidelity level high.

⊙ The Internet interface between the enterprise network and the rest of the Internet: The enterprise network cannot be an isolated island so we have to build an Internet interface on the access link of this enterprise to simulate the communication between the enterprise network and Internet. How to model the Internet is a great challenge because of its vast size and ever-changing nature. Specifically we need to simulate the ingress scanning traffic to the enterprise.

⊙ Synchronization problem: There is an inherent synchronization problem in distributed simulation or integrated simulation/emulation experiment. Though clock synchronization is not so important in an event-driven worm experiment in terms of infection, we still need to consider carefully the work load of a virtual node program to avoid overloading it.

2.2. Virtualization using VMware or Emulab VM, or NS2 simulation

Besides the "virtual node" approach presented in this article, there are alternative virtualization methods that can be utilized, such as VMware [25] and multiplexed virtual nodes in Emulab (Emulab VM) [3].

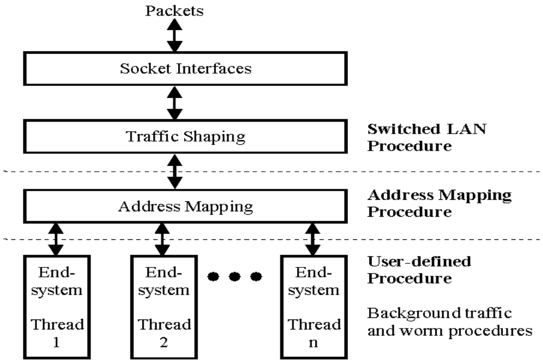
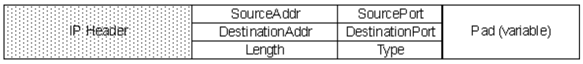


Fig. 1. Programming layout of virtual node model

Network Address	DETER Node Address
10.1.3.0/24	10.1.3.2
10.1.4.0/24	10.1.4.2
10.1.5.0/24	10.1.5.2
10.1.6.0/24	10.1.6.2
default	10.1.8.2

(a) An intra-enterprise address mapping table



(b) A virtual node IP packet structure

Fig. 2. Address mapping and IP packet structure

packet can cause a new infective when it reaches a virtual end-system that is marked as a susceptible end-system. The new infected end-system will itself begin to send the scanning traffic immediately after the infection. In fact, the susceptible end-systems need to be assigned to a virtual node program before we run the virtual node experiment.

3.3. Address Mapping

There are two types of addresses used in the virtual node program, namely a *virtual* IP address of an end-system and an *actual* IP address of a DETER experimental node. The virtual IP address is an identifier of a virtual end-system on a virtual LAN while the actual IP address is a physical IP address of an experimental node in the DETER network. A unique IPv4 address is assigned to each virtual end-system, and the network addresses of virtual end-systems are mapped with the DETER experimental node IP addresses and kept in an intra-enterprise address mapping table. Figure 2(a) shows an example of an intra-enterprise address mapping table. The first column represents the unique network address of virtual end-system and can be used as a table index. The second column is the actual DETER node IP addresses that are associated with the network addresses in the first column. Note that the last row of the intra-enterprise mapping table represents the mapping of inter-enterprise addresses and the address of the inter-enterprise (Internet) gateway.

In our virtual node program, the virtual IP address is used by a virtual end-system to send scanning traffic between virtual end-systems. However, traffic among DETER experimental nodes is communicated by the actual IP address. In other words, the virtual IP address is only known to the virtual node program, not to the other DETER emulated-network devices. Therefore, the virtual IP address has to be translated into the IP address according to the intra-enterprise address mapping table so that the scanning traffic can be sent across the DETER network. In addition, the rest of scanning traffic whose virtual IP address do not match with any particular networks in the intra-enterprise address mapping table (in other words, they are mapped to "default") will be sent to the Internet gateway, and it will be silently dropped here. In the following, we show where both virtual and actual IP addresses are located in an IP packet.

Figure 2(b) shows an IP packet structure used in our virtual node program. The virtual node IP packet, like most IP packets, consists of an actual IP header followed by a number of bytes of payload that consists of a virtual IP header, and a variable length pad. Currently our virtual IP header supports only IPv4 addresses. The length field contains the size of packet, not including the actual IP header. The *Type* field contains a packet information to indicate whether it is a worm packet or a normal packet. Finally, the variable-length *Pad* field is used to fill up the payload to create a packet according to the virtual header length field.

of enterprises in state i at time t. Clearly, for all time $t \geq 0$

$$\sum_{i=0}^{C} y_i(t) \quad = \quad N. \tag{1}$$

Define

$$Y(t) \;\; \equiv \;\; \sum_{i=1}^{C} y_i(t) \; = \; N - y_0(t)$$

as the number of enterprises with one or more worms (infectives); we assume that each such infected enterprise transmits exactly σ scans/s into the Internet irrespective of the degree of its infection. That is, we assume that a single infective saturates the stub-link bandwidth of the enterprise. Finally, an implicit assumption of the following is that "local" infections (between nodes in the same enterprise) are negligible in number. Thus, the total rate of scanning (causing infection) into the Internet at time t is

$$S(t) \quad = \quad \sigma Y(t).$$

The likelihood that a particular susceptible is infected by a scan is $\eta = 2^{-32}$ (purely random scanning in the 32-bit IPv4 address space). The likelihood, therefore, that a scan causes an enterprise in state i at time t to transition to state $i+1$ is $(C-i)\eta$ because there are $C-i$ susceptible but not infected nodes in the enterprise at time t. Thus, define the infection "rate" of an enterprise in state i by

$$\beta_i \quad \equiv \quad \sigma\eta(C-i).$$

The time-evolutions of the states y_i are governed by the following coupled Kermack-McKendrick equations: For times $t \geq 0$,

$$\dot{y}_C(t) \;\; = \;\; \beta_{C-1} y_{C-1}(t) Y(t), \tag{2}$$

$$\dot{y}_i(t) \;\; = \;\; (\beta_{i-1} y_{i-1}(t) - \beta_i y_i(t)) Y(t) \quad \text{for } 1 \leq i \leq C-1 \tag{3}$$

$$\dot{y}_0(t) \;\; = \;\; -\beta_0 y_0(t) Y(t). \tag{4}$$

The total number of worms (infectives) at time t is clearly

$$\sum_{i=1}^{C} i y_i(t). \tag{5}$$

Thus, the scan-rate per worm (per infective) is

$$\frac{\sigma Y(t)}{\sum_{i=1}^{C} i y_i(t)} \quad = \quad \frac{\sigma \sum_{i=1}^{C} y_i(t)}{\sum_{i=1}^{C} i y_i(t)}. \tag{6}$$

Summing equations $i = 1$ to C yields the "standard" Kermack-McKendrick equation $dY/dt = \beta_0 y_0 Y = \beta_0(N - Y)Y$ whose solution is $Y(t) = NY(0)[Y(0) + (N - Y(0))\exp(-\beta_0 N t)]^{-1}$.

It turned out that this model, with its few parameters suitably selected from measurements of Slammer's spread, overestimated the total scan-rate of the worm [20], [8]. A slightly more general version of the model [8], [9] was able to accurately recreate the measured Slammer data (the latter reference effectively exploiting routeview data also used in [20]). Our simulation code "kmsim" for this more general model is documented and publicly available at [15].

4.1. Packet Injector Given Rates

Suppose that the total scan rate, $S(t)$ of a worm is available (either by measured data or by a mathematical model and parameters suitably chosen). Under random scanning, the scan-rate from the Internet directed at the enterprise under simulation could be approximated as $(A/2^{32})S(t)$ where A is the size of the address space of enterprise network under simulation; alternatively, a similar but random thinning of $S(t)$ could be used. Individual scans would be directed to an end-system of the enterprise network that is chosen at random. The scan rate curve generated from the above mathematical model for SQL Slammer is illustrated in Figure 3. As can be seen, the generated scan rate curve closely matches with the measured Slammer curve from the University of Wisconsin.

5. Validation Experiments for Virtualization

To test whether the virtual node model/approach can satisfy the emulation requirement in terms of worm infection, traffic throughput, and packet distribution among LAN nodes etc. [1] We have done a series of worm propagation experiments using three different emulation approaches.

[1]It is not realistic for virtual-node simulation to generate exactly the same network traffic for all LAN nodes, so here the focus of virtual LAN emulation is on the network dynamics viewed from outside the LAN, i.e., on the direct LAN access link and other network segments.

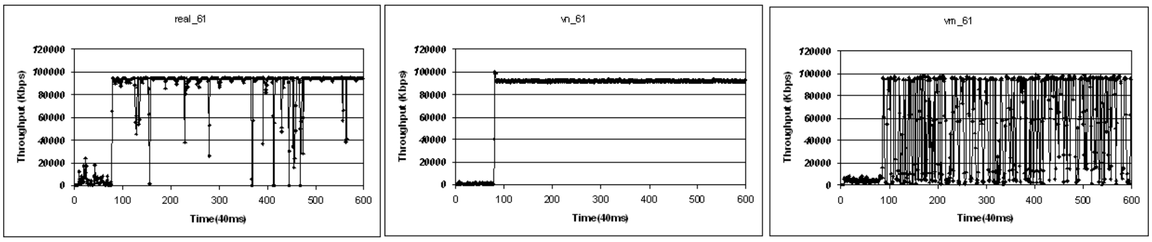


Fig. 5. Throughput of LAN access link under three experiments: on the left real2xx Average=78.8Mb; in the middle vn2xx Average=80.0Mb; on the right vm2xx Average=48.9Mb.

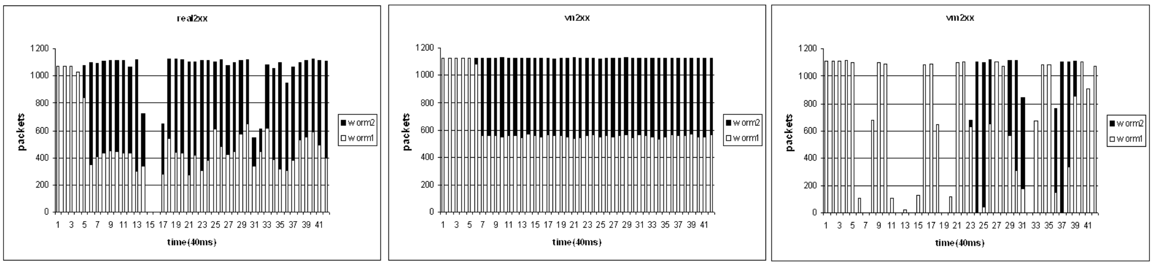


Fig. 6. Packet composition on LAN access link under three experiments. The ratio of packets from the first infected node V.S. the second infected node: on the left real2xx 44:56; in the middle vn2xx 50:50; on the right vm2xx 64:36.

the worst behavior: the throughput fluctuating heavily between 0 and 100Mb with a large variance. One possible explanation for the weak performance of VM multiplexing is that separate process pools for multiple VM nodes needed more OS handling time and higher CPU load on the host machine.

⊙ Packet composition on the LAN Access Link

The distribution of worm scanning packets originating from the two infected nodes on the LAN access link of three experiments are displayed in Figure 6. In vn2xx experiment two infected nodes output the same amount of scanning traffic. The minor difference between the first and second infected nodes in real2xx experiment(44:56) was probably the result of TCPDUMP packet loss. Because of inconsistent traffic generated by VM based nodes, there is no obvious pattern of distribution between the two infected nodes in the vm2xx experiment.

⊙ Throughput on the Internet interface link

Readings of throughput on Internet interface (Figure 7) links are similar with those of the LAN access link: real2xx had the highest throughput at 71.4Mb; vn2xx the second at 63.2Mb; vm2xx the lowest at 36.6Mb. Because two infected nodes were located within the same sub-net and they were the only nodes sending out packets in high speed, the scanning traffic can reach to the Internet interface node without a large amount of packet loss.

5.3. **Discussion**

The experimental results validated our hypothesis that a virtual node can realistically emulate a LAN which has limited susceptible nodes distribution (limited by aggregate outbound bandwidth and CPU load) and background traffic throughput. It is more controllable than Emulab VM multiplexing approach and consumes less resources. While Emulab VM has its advantage over our virtual node approach in that it can save experimenter's work of re-writing programs, you have to carefully plan your multiplexing scheme to avoid overloading the host machine if

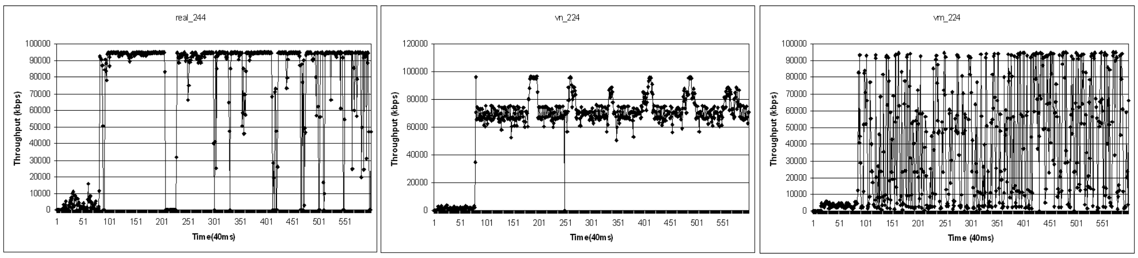


Fig. 7. Throughput at Internet interface node under three experiments: left chart real2xx Average=71.4Mb; in the middle vn2xx Average=63.2Mb; right chart vm2xx Average=36.6Mb.

The virtual LAN program has the similar characteristics as the Honeyd simulation program, which creates virtual hosts/network on one real machine that can be simulated to run various operating systems or network services. In our enterprise emulation experiment, we configured one test virtual node in the topology to simulate a honeypot like network (/24) to passively gather scanning traffic from both outside (scans from Internet interface node) and inside (scans from infected nodes). While its implementation is simple compared with Honeyd's multifarious features (such as ARPd to intercept traffic, NMAP response, etc.), it is sufficient to function as an intrusion detection prototype for testbed emulation experiments. On this honeypot node, we ran a traffic collecting program to receive any traffic to this /24 network segment. Emulation and virtual LAN programs recognize the existence of "dark addresses" by reading the complete topology map file so that the background traffic will skip these addresses and only worm packets will be directed to this node.

7. CONCLUSION

In this work, we have performed a set of emulated experiments on the DETER testbed to study Slammer worm propagation from an enterprise network perspective. We showed that a fine-grained, packet-level worm emulation is feasible using a resource-limited physical testbed. We showed how fidelity and testbed resource requirements can be properly balanced using techniques such as virtualization. The insights we gained and the lessons we learned in doing worm experiments will be valuable to a variety of enterprise network worm re-creation and defenses evaluation experiments.

The experiments we have done are only the first steps to explore the use of testbed for fine-grained, packet-level emulated worm experiments in enterprise networks, and significant future research is needed to make DETER a truly useful testbed for worm recreation and defense evaluation.

REFERENCES

[1] T. Benzel, R. Braden, E. Fraser, A. Joseph, D. Kim, J. Mehringer, C. Neuman, R. Ostrenga, S. Schwab, Dan Sterne. Design of DETER security Testbed.

[2] E. Cooke, M. Bailey, Z.M. Mao, D. Watson, F. Jahanian, D. McPherson. Toward understanding distributed blackhole placement. In *Proc. ACM WORM, Washington DC*, October 2004.

[3] *http://www.emulab.net*

[4] *http://www.isi.edu/deter*

[5] D.J. Daley and J. Gani. *Epidemic modeling, an introduction*. Cambridge University Press, 1999.

[6] J. Hui and M. Devetsikiotis. A Unified Model for the Saturation Throughput and Delay Analysis of IEEE 802.11 EDCF. *submitted to IEEE Trans. on Communications*.

[7] J. Just, G. Kesidis, K. Levitt, J. Rowe, F. Wu and P. Porras. Lack of science for testing the effectiveness of large-scale cyber defenses. In *Proc. ACM WORM, Washington DC*, October 2004.

[8] G. Kesidis, I. Hamadeh and S. Jiwasurat. Coupled Kermack-McKendrick models for randomly scanning and bandwidth saturating Internet worms. In *Proc. QoS-IP, Catania, Sicily*, Feb. 2005.

[9] G. Kesidis, I. Hamadeh, Y. Jin and S. Jiwasurat. A Model of the Spread of Randomly Scanning Internet Worms that Saturate Access Links. Submitted, Dec. 2004.

[10] M. Liljenstam, D. M. Nicol, V. H. Berk and R. S. Gray. Simulating Realistic Network Worm Traffic for Worm Warning System Design and Testing. *ACM CCS WORM Workshop*, 2003.

[11] M.T. Lucas, B.J. Dempsey, D.E. Wrege, and A.C. Weaver. (M, P, S) - An efficient background traffic model for wide area network simulation. *1997 IEEE Global Telecommunications Conference*, Vol. 3, pp. 1572-1576, 1997.

[12] D. Moore, C. Shannon, G. Voelker and S. Savage. Internet Quarantine: Requirements for Containing Self-Propagating Code. In *Proc. IEEE INFOCom 03, San Francisco, CA*, April 2003.

[13] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford and N. Weaver. Inside the Slammer worm. *IEEE Security and Privacy*, 2004.

[14] P. Porras, L. Biesemeister, K. Levitt, J. Rowe, K. Skinner and A. Ting. Hybrid Quarantine Defense. In *Proc. ACM WORM, Washington DC*, October 2004.

[15] Emist Project. *http://emist.ist.psu.edu*.

[16] *available at http://labs.ee.psu.edu/faculty/kesidis/exptl_tech_reports.htm*.

[17] Scalable Simulation Framework. *available at http://www.ssfnet.org*.

[18] The Network Simulator - ns-2. *available at http://www.isi.edu/nsnam/ns*.

[19] A. Wagner, T. Dubendorfer, B .Plattner and R .Hiestand. Experiences with worm propagation simulations. *ACM CCS WORM Workshop*, 2003.

[20] N. Weaver, I. Hamadeh, G. Kesidis and V. Paxson. Preliminary results using scale-down using scale-down to explore worm dynamics. In *Proc. ACM WORM, Washington DC*, October 2004.

[21] C. Zou, L. Gao, W. Gong and D. Towsley. Monitoring and Early Warning for Internet Worms. In *Proceedings of the 10th ACM Conference on Computer and Communication Security*, October 2003

[22] C. Zou, W. Gong and D. Towsley. Code Red Propagation Modeling and Analysis. In *Proceedings of the 9th ACM Conference on Computer and Communication Security*, 2002.

[23] Agilent Technologies (2001). Q&A: How can I generate traffic representative of "Realistic" Internet Traffic? *available at http://advanced.cooms.agilent.com/insight/2001-08/Questions/traffic_gen.htm*.

[24] *available at http://www.honeyd.org*.

[25] *available at http://www.vmware.com/*