

# Anti-worm Dynamics in Distributed Detection

John Mark Agosta, Jaideep Chandrashekar

Intel Research

Santa Clara, CA 95054

{john.m.agosta, jaideep.chandrashekar}@intel.com

August 31, 2006

## 1 Introduction

This paper<sup>1</sup> raises some interesting challenges that have arisen in design of an anomaly detection system in the *Distributed Detection and Inference* (DDI) project. Conventional detection schemes rely on observing traffic at central points, typically at the border of the enterprise. These schemes, while moderately successful, have several limitations, in their visibility into the network, and the power they can bring to detection. It is conceivable that one could do better with a variety of sensors around the network that measure and track different things and correlate the sensor findings. Our approach is to exploit the large number of machines on a network in a completely distributed fashion, both their combined computational resources, and their variety of sensing abilities. Our framework extends this idea to the logical extreme: We consider each end host in the enterprise to have a sensor (or Local Detector) and to communicate with each other host as a peer.

To date our results support the notion that relatively weak, distributed traffic anomaly sensors can combine their beliefs by messaging, and detect “stealthy” day-zero self-propagating worms. ([3][2]). We have explored several variations of dynamic (i.e. multi-stage, temporal) Bayes Networks (DBN) to aggregate individual sensors. In simulation we’ve shown that unacceptable false positive rates for individual detectors can be reduced to acceptable levels while also detecting slow worm traffic significantly below normal background traffic levels. This work is the subject of an exploratory research project, targeted at embedded support for distributed detection widely, on individual platforms throughout the enterprise.

Recently we’ve become aware that our approach still leaves open significant gaps for an adversary to exploit. We discuss approaches to two: First the rigidity of anomaly detector thresholds, which can be made adaptive by machine learning methods; and secondly, the gap due to the wide range of possible worm speeds.

## 2 Worms that hide beneath a threshold

In general, an anomaly detector functions by setting a threshold on a signal, e.g. network traffic levels, above which there’s a higher probability of malicious behavior. Since normal traffic is dynamic and the traffic from the source of the malicious behavior—the worm—adds to the traffic level, the probability of exceeding the threshold depends on the combined worm traffic plus normal traffic level. When normal traffic is little or none it opens up a gap for malicious traffic to sneak under the threshold. The gap is illustrated in Figure 1. The worm, in this case, would simply observe the current traffic level, and only generate its own traffic when the network was relatively quiet. As long as the total traffic remains at a level below the fixed threshold, no detection occurs. Interestingly the worm doesn’t have to know the

---

<sup>1</sup>Acknowledgments: Carlos Diuk-Wasser built the classifier and generated the graphs for the adaptive traffic predictor. We gratefully thank the entire DDI team, including Abe Bachrach, Denver Dash, Branislav Kveton, Alex Newman, Eve Schooler, Micah Sherr and Tomas Singliar, who contributed to the theory and simulation results behind this work. This project is internally funded by Intel Research.

exact threshold to exploit this vulnerability. It can estimate the traffic distribution and assume a likely threshold would fall in the distribution tail. Such evasive behavior would however slow the worm down by limiting its transmission rate.

A response to this threat is to predict a dynamic threshold value that adjusts to track the traffic, thus decreasing the worm’s “headroom.” Predictions of normal network behavior can be derived from a range of sensed values available at individual devices to make it harder for the adversary to squeeze its traffic under the device detection threshold. An adaptive detector opens another vulnerability if the worm can game the inputs to the prediction function. Network traffic could be the result of the worm influencing the predictor. Furthermore it’s known that network traffic by itself is notoriously hard to predict. We have addressed this by including several variables derived from the host internal state, such as user activity in the predictor function. We’ve demonstrated a useful predictor, learned from user activity and its opposite, user idleness, together with related local measures that were shown to be significant, such as connection address entropy. This work has required collection and modeling of enterprise individual machine network traffic, an interesting research area in its own right.

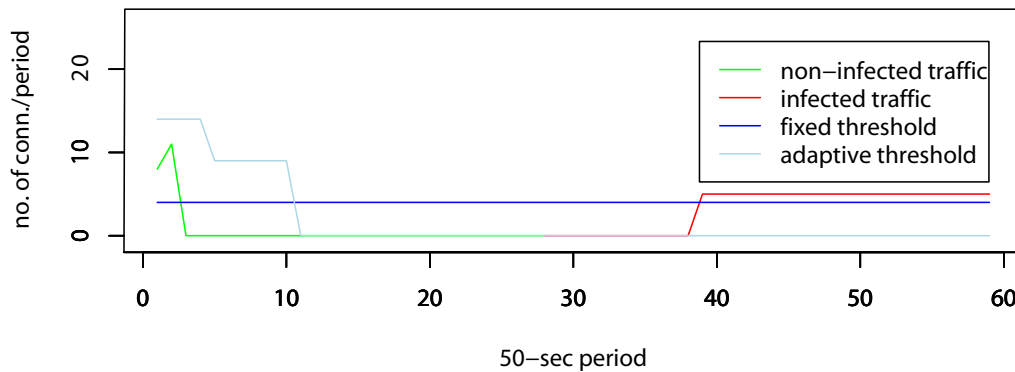


Figure 1: Varying levels of normal traffic create gaps below a fixed detection threshold in which a worm’s traffic can hide. This gap is closed by an adaptive threshold.

What exactly should be the scoring function to optimize for a classifier used for predictive anomaly detection? To the extent that there is any labeled data with which to compute classifier error rates, its not obvious how error rate determines the quality of anomaly detection. The characteristic of an anomaly detector from the point of view of the aggregate detector that receives its signal is its false positive rate. If the local detector’s false positive rate drifts upward, the aggregate detector will be mis-led, and its false positive rate will increase. Therefore one constraint on the adaptive detector is that it maintain a constant false positive rate for a varying threshold.

We’ve implemented such a predictor. First it can be shown that it is a better detector, meaning it has a better ROC curve. The intuition behind this is shown in Figure 2. To the extent that the adaptive threshold can track normal traffic, the adaptive detector detects most of the “band” of anomalous traffic that lies “on-top” of the normal traffic, largely independently of the amount of false positive normal traffic is detects. But this adaptive detector is not perfect and still leaves some gaps that a worm can exploit. The predictor takes a few time-steps to respond to the change in traffic level, and a resourceful worm could transmit in the “shadows” of bursts of traffic. Again, see Figure 2.

Apparently throttling has no significant effect to block slowly spreading worms. The same strategy that a worm uses to evade detection could be used to counter-act throttling and thereby block outgoing worm traffic. A “stealthy” worm could wait for periods of relatively quiet network traffic in which to transmit packets at a rate below the throttling threshold. Of course this would slow down a worm that had this adaptive ability. But the traffic predictor could also drive an adaptive throttling response, making the assumption, as before, that the predictor would not be confused by the traffic that the worm adds.

It remains to be shown that the gains in accuracy achieved by an adaptive local detector make

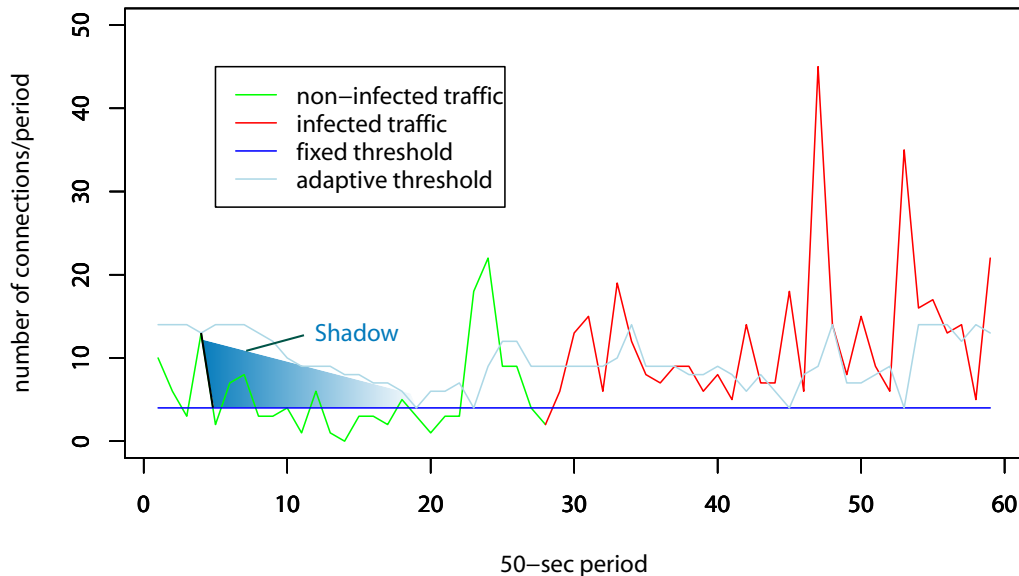


Figure 2: An adaptive threshold follows the traffic level trend, improving accuracy. There are still shadows in which worm traffic can hid since it is not a perfect predictor.

for proportional improvements in system-wide detection. Static thresholds may not be as bad as they sound: The lulls in traffic at one host would not be coincident with those at other hosts, so a worm would always be facing some tight-threshold detectors. Averaging over detectors removes some of the worm’s advantage.

## 2.1 Speed of detection versus optimal sample size for the aggregate detector

Is there a “detection gap” for worm speeds between stealth worm detection and current fast worms—and if so, how does one guarantee that the detectors cover the range of worm speeds?

As mentioned, a major finding of the current work has been that worms with transmission rates deep within normal traffic levels can be detected by combining the signals of a large enough sample of machines. However larger sample sizes quickly lose their value as the worm speeds up, to the point where there is an effective limit on the size of the sample worth collecting. As Figure 3 shows, the benefits in terms of faster detection with larger samples quickly diminishes. This effect is more pronounced with faster worms. Of course in principle the accuracy of detection does always improve with larger samples; however the increase necessary for earlier detection must counteract the smaller signal due to the worm epidemic’s exponential growth curve.

This is a property of inference solely and does not consider degradation from the delays that might accrue from collecting a larger sample: In this analysis the sample is assumed to arrive instantaneously at the detector. It also does not balance the gains in inference against the additional costs of faster messaging to catch the worm. Adding these costs would create an inflection point in graph that effectively bounds the size of the sample.

Thus for slower worms, when messaging speed is less of a constraint, the maximum effective sample size would correspondingly increase. This matches the intuition that for slower worms the detector should be tuned for detection earlier in the infection. Conversely the negative finding, that detection of faster worms cannot be improved by larger collaboration may have the fortunate consequence of moderating the quest for faster, higher capacity messaging schemes.

The question remains open about exactly how to bridge the worm speed detection gap. In response to the tradeoff between detector sample size and worm speed—the slower the worm the larger the optimal sample and vice versa—should one run several versions of distributed detector with different sample “windows”, each tuned to differing worm speeds? This way smaller window detectors could pick

up the faster worms sooner, and larger window detectors be afforded the luxury of waiting for slower worms. A more elegant alternative is to run a sequential detector that can come to a conclusion about the presumed adversary as the sample is being collected, and without needing to see the entire sample. In a sense this is another case of adaptive response: The detector adapts as the sample messages arrive.

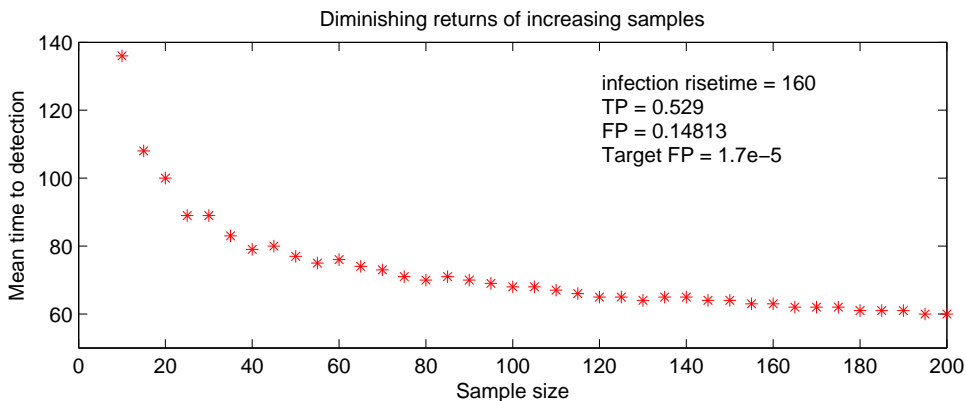


Figure 3: The benefits of including more hosts in a distributed scheme quickly diminishes, costs aside.

### 3 Considerations with multiple defenses

In the larger picture there are likely to be numerous interacting agents, both beneficial and malicious. An interesting case occurred in traces collected on our own enterprise where we detected instances of intentional worm-like behavior that would have generated false positive detections. We learnt that a particular vendor’s installed devices perform patch compliance testing and vulnerability analysis by scanning hosts in the internal address ranges for exposed ports. Such “auto-immune” behavior of security mechanisms makes their co-existence difficult. In this vein we’ve envisioned more general models of random, information-collecting agents. [1] The agents in the model are identified with a distributed, sequential decision calculation, e.g. a DBN that moves from host to host. The agent then takes an action, depending upon the random walk in their belief’s state space, driven by the information updates. Thus two paths are being traced out simultaneously by the agent; its itinerary through the physical network and the trajectory of its belief.

### References

- [1] Agosta, J. M., “Business Network Fellowship Statement of Purpose: Distributed decision agents for managability and security.”
- [2] Cheetancheri , S. G., J. M. Agosta, D. H. Dash, K. N. Levitt, J. Rowe, E. M. Schooler, “A Distributed Hostbased Worm Detection System,” *Sigcomm 2006*.
- [3] Dash, D., B. Kveton, J. M. Agosta, E. M. Schooler, J. Chandrashekar, A. Bachrach, A. Newman, “When Gossip is Good: Distributed Probabilistic Inference for Detection of Slow Network Intrusions,” *AAAI-2006*.