Dependency-based Distributed Intrusion Detection

Ji Li *Dah-Yoh Lim* *Karen Sollins*
MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139
{jli, dylim, sollins}@csail.mit.edu

Abstract

Distributed network intrusion detection has attracted much attention recently. Our main focus in this work is on zero-day, slow-scanning worms, of which no existing signatures are available. We organize end hosts into regions based on network knowledge, which we posit is positively correlated to the dependency structure. Leveraging on this organization, we apply different intrusion detection techniques within and across regions. We use a hidden Markov model (HMM) within a region to capture the dependency among hosts, and use sequential hypothesis testing (SHT) globally to take advantage of the independence between regions. We conduct experiments on DETER, and preliminary results show improvement on detection effectiveness and reduction of communication overhead.

1 Introduction

Traditionally, intrusion detection is carried out at a central point, usually a gateway, as it is a natural position to observe incoming and outgoing traffic. This approach does not scale well, is prone to DoS attacks, and depends on non-local detection of anomalies, prompting a need for new approaches to monitor and respond to security incidents. To that end, host-based distributed intrusion detection has been a promising direction. A key challenge in such a distributed intrusion detection system is that end hosts need to be organized efficiently and intrusion detection techniques applied effectively, so that an intrusion detection decision can be made before the worm infects most of the hosts. Many current mechanisms use simple gossiping protocols or peer-to-peer protocols [10, 6, 5] to aggregate local determinations.

Since the behaviors of zero-day worms are not known a priori, the best location for initial attention is the local host itself, in the context of local behavior and applications [6, 5]. However, at the local node, one loses the aggregation effect of repeated or simultaneous low level anomalies. In addition, it is difficult to make local detectors strong because they see only a small percentage of the global traffic. Thus one must aggregate the results of weak local detectors to get a broader perspective. This work addresses the question of algorithms for effective aggregation. Improving local detection is a separate problem that we do not address here.

New intrusion detection techniques are needed to deal with different dependency structures among hosts more effectively. We postulate an observable causal relationship between the success likelihood of a particular intrusion attempt and network proximity between end hosts. This is based on the observation that enterprise networks are reflected in topological neighborhoods, and also likely to be supporting many similarly configured and managed machines, thus repeating the same weaknesses across an enterprise. Thus, if one host in an enterprise is susceptible in a certain way, it is more likely that its peers are as well. In contrast, random hosts far away from each other in the broad Internet are likely to be independent of each other. Therefore, we can take advantage of different dependency structures between hosts with different detection techniques. In addition, worms often scan consecutive IP addresses, which causes another kind of dependency. For example, Code Red II chose a random IP address from within the class B address space of the infected machine with probability $\frac{3}{8}$; with probability $\frac{1}{2}$ it chose randomly from its own class A; with probability $\frac{1}{8}$ it would choose a random address from the whole Internet [16].

We believe that a good distributed intrusion detection system should satisfy two key requirements: (1) efficient host organization based on network proximity and (2) detection techniques that leverage this host organization and dependency structure. In this work, we propose a dependency-based host organization and message propagation protocol. End hosts are organized into cooperating regions based on their network proximity. Then dif-

ferent detection techniques are applied at different levels. We use a discrete-time Hidden Markov Model (HMM) with unsupervised learning to estimate intrusion within a region (this captures the dependency) [13], and a sequential hypothesis testing (SHT) globally to coordinate findings across regions (this captures the independence) [7]. We implement our mechanism on the DETER testbed [1], and evaluate the performance of our detection techniques and the communication overhead. Preliminary results show that our mechanism can detect intrusion faster, better and cheaper.

As a first step, in this work we only evaluate time-homogeneous first order HMMs (where the transition probabilities between the different states do not vary with time), and use a simple static organization based on both dependency and network proximity. Non-homogeneous higher order HMMs, based on an adaptive organization utilizing various kinds of network knowledge, will be considered in future work.

2 Related Work

2.1 Intrusion detection techniques

Our main focus is on zero-day, slow-scanning worms, as in [16, 5, 3]. Such worms propagate themselves slowly to avoid attention caused by dramatic traffic increase. There are no signatures available as they are completely new.

Many intrusion detection techniques have been developed. Anything based on prior knowledge, such as signature-based approaches [12, 14, 4], cannot be used against zero-day worms since there is no prior knowledge available in a zero-day intrusion.

Bayesian network based techniques are used in [6] to imbue end hosts with probabilistic graphical models. With random messaging to gossip state among the local detectors, they show that such a system is able to boost the weak local detectors to detect slowly propagating worms.

Sequential hypothesis testing (SHT) was first adopted to intrusion detection by Jung et al. in [7]. The original algorithm was centralized, with detection performed at the gateway. It was decentralized in [5], where hosts exchange their information, and perform the inference individually in parallel. We identify two issues with this approach. First, it assumes independence among intrusion attempts and, second, it cannot deal with the case when a worm interleaves the intrusion traffic with non-intrusion traffic. In our work, we assume dependence among hosts within a region, and assume independence between regions. To address this dependence/independence, we use a Hidden Markov Model (HMM) to detect intrusion within a region and SHT globally between regions. The HMM allows us to incorporate our dependency assumption into the regional aggregations, and SHT depends on our assumption of independence between regions.

Machine learning has been applied to intrusion detection in various aspects. For example, Agosta et al. designed an adaptive mechanism that adjusts the threshold of anomaly based on traffic [3]. This does not seem to handle alternating traffic either. Our use of the HMM approach allows us to handle such interleaving, because it learns both transition and emission probabilities from observations, since neither is known a priori.

2.2 Communication protocols

In a centralized intrusion detection system such as [7], all the information is collected and processed at a central point. In a collaborative intrusion detection system, end hosts need to communicate with each other to pool their information together.

Various communication protocols have been applied to distributed intrusion detection systems. One is centralized where all local detectors report intrusion information to a global detector. A recent innovation is to use gossiping protocols between local detectors or multiple global detectors [5, 6].

In [5], decision making is completely distributed. Hosts exchange observations using an epidemic spread protocol without any organizing structure. When a potential intrusion is detected by an end host, it forwards an alert to m randomly selected neighbors, and then each neighbor forwards the alert to its m neighbors together with its own observations, and so on. Each host computes the possibility of intrusion using all the information it has received. This continues unless a decision is made by a host. Usually m equals 1 or 2 for scalability reasons. Each host computes the possibility of intrusion using all the alerts it has received plus its own conclusion. If a host believes that there is an intrusion, it will broadcast its decision to all hosts. In contrast, [6] uses a set of global detectors with a gossiping protocol.

To the best of our knowledge, previous systems have not considered host organization to achieve more effective detection and efficient communication. Therefore, the communication can be inefficient. More importantly, intrusion detection techniques often assume independence in the intrusion attempts amongst all hosts. This is unlikely to be true when nearby hosts are scanned by a worm. In our method, we make use of the network topology and dependency information to organize a region, and consider the dependency among hosts within each region. In this sense, our method can be seen as a hybrid between a centralized and a distributed intrusion detection system.

3 Host Organization

To build an effective intrusion detection system, we propose a host organization based on the concept of regions, and discuss the communication mechanism between hosts [15, 8]. Then we customize the host organization in the intrusion detection scenario.

3.1 Regions

We organize hosts into a two-level hierarchy, using the knowledge from the networks. First, hosts are clustered into regions based on certain criteria. We list three kinds of criteria here: network proximity, including network properties, such as network topology, geographic distance, latency [8]; local host properties, such as operating system types and running services; policy constraints and boundaries, such as enterprise networks. Within each region, hosts elect a regional leader using a distributed leader election algorithm periodically.

Second, the leaders organize themselves into a communication structure. If the number of leaders is small, the leaders form a complete graph; otherwise other organizations such as multiple disjoint trees can be used [9]. Figure 1 demonstrates a region-based organization. It consists of three regions. Nodes close to each other are clustered into the same region. The shaded nodes are the leaders elected in each region.

Corresponding to the region-based structure, we have three kinds of detectors: local detectors, regional detectors, and global detectors. One local detector resides on each host, regional detectors reside on the regional leaders, and global detectors may reside on any hosts. There may be one or more global detectors, depending on the requirement on robustness and the communication structure.

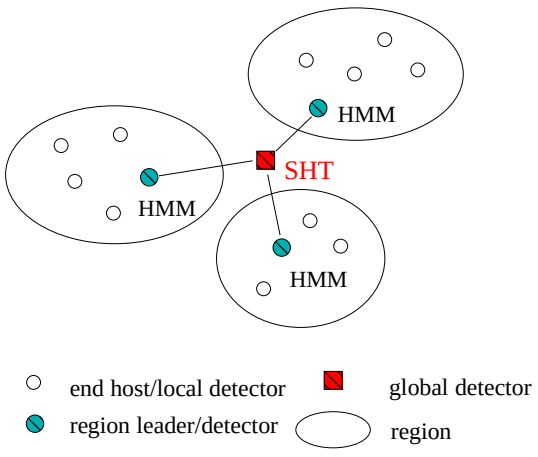

Figure 1: A region-based organization example.

3.2 Communication

Local detectors only communicate with their regional detector. When a local detector detects a potential intrusion attempt, it sends an alert to its regional detector directly. The regional detector collects alerts from local hosts, runs its regional detection algorithm, and then reports to the global detectors. Global detectors wait for reports from multiple regional detectors, and run the global detection algorithm.

Depending on the tradeoff between robustness and overhead, there may be different communication structures between regional detectors and global detectors. For example, we can deploy only one global detector, and all the regional detectors report to it. This centralized method has low communication overhead, but the global detector may become the target of DoS attacks. As another extreme example, we can have one global detector on each regional leader, together with the regional detector. And each regional detector multicasts its report to all the global detectors. Therefore, each region (through its global detector) has a global view of the intrusion situation. Whenever a global detector has enough information to make a decision, it announces its decision to the other global detectors and all the regional detectors. We could also have chosen an intermediate position in which there was more than one global detector, but not as many as one per region.

4 Intrusion detectors

As mentioned above, there are three kinds of detectors in our system: local detectors, regional detectors, and global detectors. Each kind of detector runs the appropriate algorithm, as described in this section.

4.1 Local detector

A local detector resides on each end host. These are weak in their capability of detecting intrusions, and as stated earlier the design of local detectors is a separate problem that we do not address here. The detection criteria may vary, depending on each host. For concreteness, we use the following simple local detector in our experiments: when an end host receives a packet at an un-serviced port, the corresponding local detector triggers an alert to its regional detector; otherwise, it sends a clean signal. There are two things to note. First, there are both false positives and false negatives in the signals the local detector sends. Second, there is a tradeoff between timeliness and detection overhead. If the end host sends one signal upon receiving every packet, the overhead may be too high. If the end host batches signals, this causes a delay in the detection. As alerts are more important than clean signals, we can send out alerts immediately, but

batch clean signals.

4.2 Regional detector

Regional detectors diagnose potential intrusions at the neighborhood level, using discrete-time Hidden Markov Models (HMMs) to detect intrusion for each region. We choose to use HMMs instead of SHTs, because, as discussed above, we believe that the probability of effective intrusion between close neighbors can be dependent on that proximity, and HMMs allow us to reflect that. The second advantage of the HMM approach is the ability to capture a notion of time and therefore multiple connection attempts to the same host. In contrast, SHT systems are particularly easy to game: the worm can make sure that the first connection attempt to any host is always to a servicing port. This is because SHT systems can only handle the first connection attempt to any host, lest the independence assumption breaks down.

Figure 2 demonstrates an HMM for a region. It has four states: *00, 01, 10, 11*, representing a value pair of *(infected?, suspicious?)*. The first bit represents whether there is a worm in the region, and the second bit represents whether there is some host whose behavior is suspicious. This captures the adaptivity of the worm in the sense that an infected host can decide to lay dormant for the time being to avoid detection (similarly, a clean host might accidentally behave suspiciously). Higher order models can be used to capture more of the adaptivity. In general, the model parameters are unknown and have to be estimated. Each regional detector uses the reports (alert or clean) from local hosts to Baum-Welch train the model and to generate the Viterbi path of hidden states [13]. This Viterbi path gives the most likely sequence of hidden states that could have generated the observed sequence of triggering of the local detectors, under the current estimated parameters. Note that the HMM models the current incoming traffic pattern, so it does not matter whether the region is under one worm attack or simultaneous worm attacks.

4.3 Global detector

The global detector uses sequential hypothesis testing (SHT) to determine whether there is an intrusion at the global level, because we believe that under a good organization, different regions can be assumed to be independent of each other in terms of intrusion conditions. Therefore, we use SHT with the independence assumption, and always use the newest information from each region as input to the SHT. The following equation of $L(\bar{Y})$ defines the likelihood ratio from the observation vector $\bar{Y} = \{Y_1, Y_2, ..., Y_n\}$, given two hypotheses H_0 ("no intrusion") and H_1 ("intrusion"), respectively. Y_i indicates whether the regional detector at region *i* believes there is an intrusion (1) or not (0). Note that

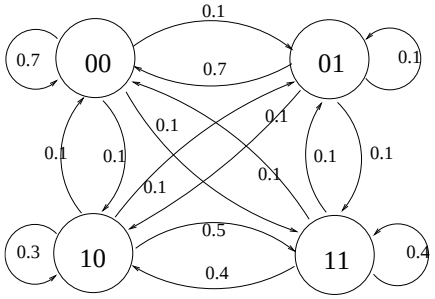

Figure 2: Initial Hidden Markov Model at regional detectors. *00* means the region is clean and its behavior is not suspicious, *01* means clean but suspicious, *10* means infected but not suspicious, *11* means infected and suspicious.

$P[Y_i = 0|H_1]$ is the probability of false negative, and $P[Y_i = 1|H_0]$ is that of false positive.

$$\begin{aligned} L(\bar{Y}) &= \frac{P[\bar{Y}|H_1]}{P[\bar{Y}|H_0]} \\ &= \frac{P[Y_1|H_1] \cdot P[Y_2|H_1] \cdots P[Y_n|H_1]}{P[Y_1|H_0] \cdot P[Y_2|H_0] \cdots P[Y_n|H_0]} \end{aligned}$$

Then $L(\bar{Y})$ is compared with the lower and upper thresholds. The thresholds, T_0 and T_1, are calculated by two parameters: desired detection rate, DD, and desired false alarm rate, DF, as follows:

$$T_0 = \frac{1 - DD}{1 - DF},\ T_1 = \frac{DD}{DF}$$

If $L(\bar{Y})$ is less than T_0, then the global detector accepts hypothesis H_0; if $L(\bar{Y})$ is greater than T_1, then H_1 is accepted; otherwise, i.e., $L(\bar{Y})$ is between T_0 and T_1, no conclusion is made. For the details of SHT, please refer to [17].

5 Experiments on DETER

In this section we present our experiments on DETER. We choose to do our experiments on DETER instead of simulation, as the former is more realistic and may provide more insights. Our evaluation consists of two parts. The first is the effectiveness of our on-line detection mechanism, in which we evaluate the performance of both regional HMM (rHMM) and global SHT (gSHT). The second is the efficiency of region-based host organization, in which we measure the detection speed and communication overhead.

5.1 Experiment Setup

Our experiments run on 88 nodes on DETER. Nodes are clustered into 8 regions, 11 nodes each, and the links between regions are slower than those within a region. Worms are emulated using WormSim [11], and we implement a special worm that scans sequentially within a region and randomly chooses the next region to scan, thus creating dependency with a region and independence between regions. Normal (clean) traffic is generated on each node at a constant rate. There are both false positives and false negatives. That is, normal traffic may be mistaken as intrusion attempts, and intrusion attempts may be viewed as clean. Nodes are divided into two categories: vulnerable and non-vulnerable. Vulnerable nodes will be infected when an intrusion attempt arrives, and then the worm will propagate from the infected host. Non-vulnerable nodes will issue an alert when receiving an intrusion attempt. WormSim and local detectors run on all the nodes except the regional leader nodes. Regional detectors run on the regional leaders, one for each region. rHMM is implemented using the General Hidden Markov Model library (GHMM) [2]. In this experiment, there is only one global detector.

5.2 Intrusion detection performance

In this experiment, we evaluate the performance of our system. The regional detectors run rHMM, and the global detector runs gSHT over all regions. Table 1 lists the parameters used in rHMM and gSHT. As we described in Section 4.2, a regional detector trains the model and infers a Viterbi path. Given the Viterbi path, there is still a question of how to determine whether the region is under intrusion or not. In this work, we use a simple empirical algorithm: if the latest six states contain three consecutive intrusion states (*11* or *10*), then there is an intrusion. Recall that *11* means that the rHMM thinks that (some nodes of) this region is infected and suspicious activity is detected, and *10* means that this region is infected and currently exhibiting normal behavior.

Figure 3 shows the intrusion detection of a region using rHMM. The experiment is divided into alternate clean periods (blank areas in Figure 3) and infection periods (gray areas). *0* means clean and *1* means infected. The solid line (*TRUE* in the figure) shows the true intrusion status (i.e., no false positives or false negatives). The dashed line (*rHMM*) shows the detection result of an rHMM using the reports from local detectors. Note that the time axis is not linear due to data aggregation. We can see that at the beginning of the experiment, rHMM makes a few mistakes due to the false positives and the relatively untrained model. However, it learns to correct the errors very soon. Although there are some noticeable lags, overall, rHMM's performance is very close to the model trained using the true state sequence, despite the false positives and false negatives. We stress again that the training of the rHMM is unsupervised.

Due to the large number of data points, Figure 3 only shows aggregate results. To look at how well an rHMM works in detail, we compare a sequence of true states with the predicted sequence of states from an rHMM in Figure 4, which is between 265 and 433 seconds in Figure 3. Local detectors report an alert (*1*) or a clean signal (*0*) to the regional detector, which may be a false positive or a false negative. Corresponding to that, the solid line shows the true state sequence. Its states labeled *0,1,6,7,8* represents that a host receives the following packets respectively: a clean signal, an alert caused by false positives, a clean signal caused by false negatives, a true alert, a false negative from a vulnerable host who cannot tell intrusion. The dashed line shows the transition of states observed by the rHMM. The states $\{0, 1, 2, 3\}$ correspond to the four states $\{00, 01, 10, 11\}$ in Figure 2. We can see that at the beginning there are two small spikes, and rHMM considers it clean but suspicious. When intrusion really happens at 325 seconds, the true sequence jumps to state 8 and then 7; the inferred Viterbi path first jumps to 1, thinking that it might be just a clean host that accidentally acted suspicious. As more alerts are received, it realizes that the region is under attack, and the state oscillates between states 2 and 3. This in particular means that when the rHMM thinks the region is under attack, but normal packets are received, the rHMM thinks that the worm is laying dormant, as opposed to the region being clean. After that, it remains between 2 and 3 during the infection period, and is not affected by the false negatives and normal traffic. Figure 5 demonstrates the new rHMM model after the experiment.

Figure 6 demonstrates the detection performance of the global detector using Sequential Hypothesis Testing

Regional Hidden Markov Model (rHMM)	
Noise level	0.03
Initial transition matrix	see Figure 2
Initial state probability	$\{0.7, 0.1, 0.1, 0.1\}$
Global Sequential Hypothesis Testing (gSHT)	
False positive	0.10
False negative	0.01
Desire false alarm rate	0.02
Desire detection rate	0.98
Experiment settings	
Number of regions	8
Number of nodes per region	11
Vulnerable nodes	25%
Worm propagation rate	1 scan/second
Normal traffic rate	1 message/second

Table 1: The rHMM and gSHT experiment parameters.

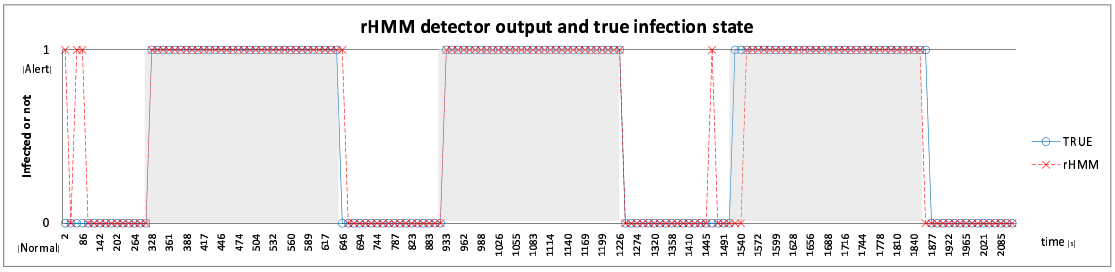

Figure 3: Regional Hidden Markov Model performance. *TRUE* is the states based on the true status (no false positive or false negative), and *rHMM* is that based on the observations of a regional detector. Gray areas represent actual infection periods.

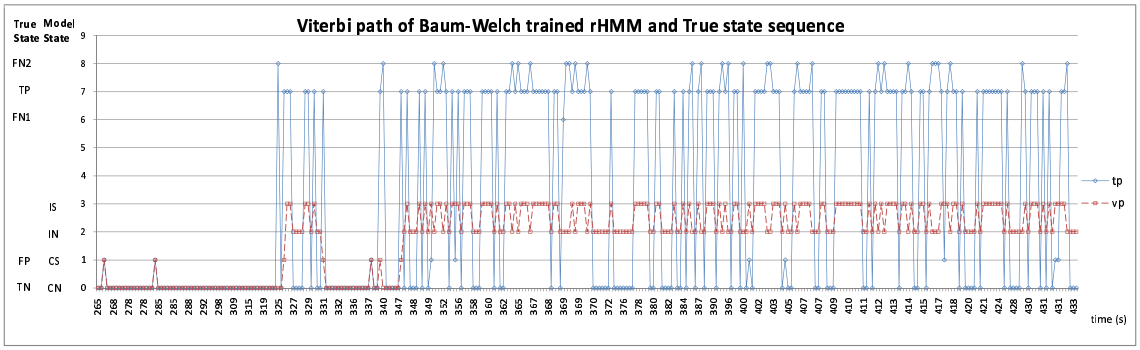

Figure 4: Viterbi path of an rHMM and the true state sequence. *tp* is the true state/event sequence, and *vp* is the inferred state sequence by the rHMM. In the dashed line, *0,1,2,3* in Y-axis correspond to the four states in Figure 2: clean and not suspicious (CN, corresponds to state 00), clean but suspicious (CS, to state 01), infected but not suspicious (IS, to state 10), and infected but suspicious (IS, to state 11). In the solid line, *0,1,6,7,8* represents an event that a host receives the following packets respectively: a normal packet (true negative, TN), an alert caused by false positives (FP), a clean signal caused by false negatives (false negative case 1, FN1), a true alert caused by an intrusion attempt (true positive, TP), a clean signal from a vulnerable host who cannot/would not distinguish intrusion attempts from normal traffic (false negative case 2, FN2).

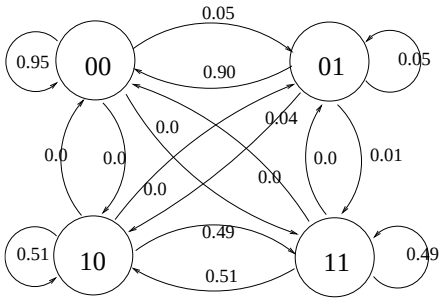

Figure 5: A trained hidden Markov model at a regional detector.

(gSHT): the solid line is the result using the true states and the dashed line is that using rHMM outputs. Since they are quite close, this shows that as far as the gSHT is concerned, the rHMM outputs are almost as good as the truth, lagging a little bit behind.

Compared with Figure 3, gSHT does not have the false positives at the beginning and near 1445 seconds in rHMM. This is because that the global detector collects information from multiple regions, and the independence of regions helps eliminate the false positives.

5.3 Region-based host organization

To evaluate the efficiency of region-based host organization, we compare our method with a gossiping protocol in three aspects: detection speed, communication overhead, and cost. Detection speed measures how fast hosts make a decision on intrusion detection. Communication overhead refers to the number of messages that hosts propagate to reach a decision. Cost is the number of nodes infected by the time of detection.

One set of the experiment results is shown below in Figure 7. *Gossip* refers to the gossiping protocol in [5], where hosts exchange observations using an epidemic spread protocol without any organizing structure, as described in Section 2.2. The gossiping rate is 2. *Region* refers to our region-based protocol. We can see that *Region* outperforms *Gossip* in all the three metrics. *Re-*

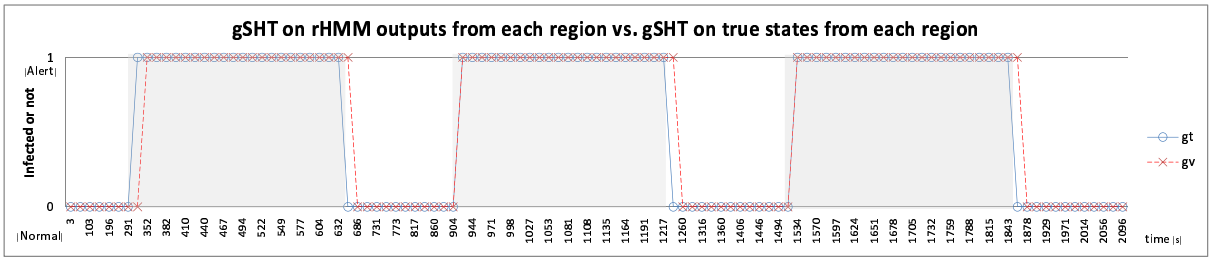

Figure 6: Global sequential hypothesis testing performance. *gt* is gSHT's decision based on rHMM outputs, and *gv* is gSHT's decision based on the true states from each region. Gray areas represent actual infection periods.

gion is faster in detection time, because alerts are aggregated within each region first before being processed at the global detector, while in *Gossip* messages may cross slow links many times between hosts before reaching a decision. Similarly, the number of infected nodes is also smaller in *Region* than in *Gossip*. Finally, the number of messages transmitted in *Region* is significantly smaller than that in *Gossip*. The reason for this is because that the number of messages increases almost exponentially among hosts in *Gossip*, while in *Region* messages from end hosts are only sent to the regional detector, and then processed at the global detector after aggregation.

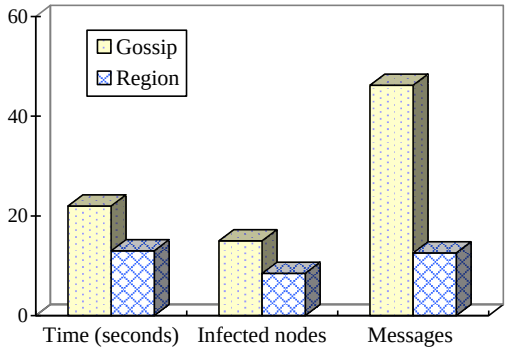

Figure 7: Detection speed and overhead comparison. Note that the messages do not include those for maintaining the cooperation among hosts in their method *Gossip* or messages for training the rHMMs in our method. The number of messages is in units of 10.

6 Discussions

6.1 Robustness and flexibility

Our method is semi-centralized, and can be made more robust to DoS attacks in two ways. First, instead of having only one global detector, multiple widely distributed regional detectors can exchange information so that everyone has an approximate global view and make the decision, thus reducing the vulnerability. Second, leaders (regional detectors) are periodically re-elected distributively, so it is hard for attackers to predict the leaders when attacks happen.

There are two kinds of dependency to be recognized. One is the dependency between end hosts, caused by their proximity, similarity of hardware, software, management, and policy boundaries, etc. Therefore, we assume that network proximity is positively correlated to the dependency structure. The other is the dependency caused by worm scan: for instance, worms may scan an IP block each time, or intentionally scan hosts distant from each other. To deal with this, our approach provides for the flexibility to re-organize regions by considering both kinds of dependency.

6.2 DETER testbed

Our experience with DETER testbed shows that DETER provides a valuable infrastructure for security-related experiments. We suggest several possible improvements here. First, it would be very helpful if DETER incorporated more security-related facilities, such as traffic generator based on real traces, worm simulators, etc. This would greatly simplify the design of experiments and provide the basis for comparison of results among researchers. Second, NS extension commands are important to experiment automation. We hope more commands can be provided in the future. Third, the swap-in process can take a long time when experiments scale up. A way to automatically kill the preloaded experimental programs and reload everything without swap-in or rebooting would significantly reduce the waiting time and speed up the experiment process.

7 Conclusions and Future Work

In this work, we design a dependency-based host organization for collaborative intrusion detection. Hosts are clustered into regions based on network proximity and dependency, and communication among them becomes more efficient. To capture the dependency, we apply different intrusion detection techniques within regions and

across regions. At the regional level, we use a hidden Markov model; at the global level, we use sequential hypothesis testing.

Our experiments on the DETER testbed suggest that dependency-based host organization can improve intrusion detection by providing valuable network-layer and application-layer knowledge to intrusion detection systems. In the future we will follow up with a series of further experiments:

1. Use HMMs across different regions, to confirm that across different regions, there is essentially no loss of effectiveness if we assume independence. Worms with different scanning features will be tested.
2. Use more general HMMs, specifically a non-homogeneous higher order HMM, based on an adaptive organization utilizing different network knowledge. If a strong global clock is available, then continuous time HMMs can be used too.
3. Enhance the reporting scheme to record the signature of a worm. This will provide two improved capabilities. The first is to significantly reduce the impact of false positives. The second is to improve the reporting of a worm, by allowing for reporting of a particular worm signature, thus enabling the disentanglement of simultaneous worm intrusions.

8 Acknowledgments

We would like to thank the DETER operators for help on setting up the experiment, Senthilkumar G. Cheetancheri for providing the original WormSim and defense code. We also thank Rob Beverly and the anonymous reviewers for their comments on the initial draft of this paper. Karen Sollins and Ji Li are supported by a research fund from the Intel Corporation, for which we are grateful.

References

[1] *The DETER Testbed*. http://www.isi.edu/deter/.

[2] *The General Hidden Markov Model library (GHMM)*. http://ghmm.sourceforge.net/.

[3] AGOSTA, J. M., DIUK-WASSER, C., CHANDRASHEKAR, J., AND LIVADAS, C. An adaptive anomaly detector for worm detection. In *Proceedings of Second Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML07)* (2007).

[4] BRUMLEY, D., NEWSOME, J., SONG, D., WANG, H., AND JHA, S. Towards automatic generation of vulnerability-based signatures. In *the 2006 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2006).

[5] CHEETANCHERI, S. G., AGOSTA, J. M., DASH, D. H., LEVITT, K. N., ROWE, J., AND SCHOOLER, E. M. A distributed host-based worm detection system. In *the 2006 SIGCOMM workshop on Large-scale attack defense* (2006).

[6] DASH, D., KVETON, B., AGOSTA, J. M., SCHOOLER, E., CHANDRASHEKAR, J., BACHRACH, A., AND NEWMAN, A. When gossip is good: Distributed probabilistic inference for detection of slow network intrusions. In *Proceedings of the 21st National Conference on Artificial Intelligence* (2006), pp. 1115–1122.

[7] JUNG, J., PAXSON, V., BERGER, A. W., AND BALAKRISHNAN, H. Fast Portscan Detection Using Sequential Hypothesis Testing. In *IEEE Symposium on Security and Privacy* (Oakland, CA, May 2004).

[8] LI, J., AND SOLLINS, K. Exploiting autonomous system information in structured peer-to-peer networks. In *the 13th IEEE International Conference on Computer Communications and Networks (ICCCN)* (October 2004).

[9] LI, J., SOLLINS, K., AND LIM, D.-Y. Implementing aggregation and broadcast over distributed hash tables. *ACM SIGCOMM Computer Communication Review Volume 35, Number 1* (January 2005).

[10] MALAN, D. J., AND SMITH, M. D. Host-based detection of worms through peer-to-peer cooperation. In *the ACM workshop on Rapid malcode* (New York, NY, USA, 2005).

[11] MCALERNEY, J. An internet worm propagation data model. Master's thesis, University of California, Davis, September 2004.

[12] PAXSON, V. Bro: a system for detecting network intruders in real-time. *Computer Networks (Amsterdam, Netherlands: 1999) 31* (1999).

[13] RABINER, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition* (1990), 267–296.

[14] ROESCH, M. Snort - lightweight intrusion detection for networks. In *LISA '99: the 13th USENIX conference on System administration* (1999).

[15] SOLLINS, K. Designing for scale and differentiation. In *ACM SIGCOMM 2003 FDNA Workshop* (2003).

[16] STANIFORD, S., PAXSON, V., AND WEAVER, N. How to own the internet in your spare time. In *the 11th USENIX Security Symposium* (2002).

[17] WALD, A. *Sequential Analysis*. J. Wiley and Sons, New York, 1947.