ContagAlert: Using Contagion Theory for Adaptive, Distributed Alert Propagation *

Michael Treaster
Google, Inc
Mountain View, CA 94043
treaster@google.com

William Conner, Indranil Gupta, Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{wconner, indy, klara}@cs.uiuc.edu

Abstract

Large-scale distributed systems, e.g., Grid or P2P networks, are targets for large-scale attacks. Unfortunately, few existing systems support propagation of alerts during the attack itself while also suppressing disruptive alerts from faulty or malicious sources. This paper proposes the "ContagAlert" protocol, which uses contagion spreading behavior to spread alerts. ContagAlert rapidly propagates alerts during attacks while also suppressing disruptive alerts. The core contagion protocols in the system are completely localized, but result in desired behavior at the network scale. We analyze and evaluate our protocol with synthetic simulations and in both Internet worm and DoS attack scenarios.

1 Introduction

In recent years, there has been a substantial increase in the deployment of large-scale distributed systems such as Grid networks [11], federated testbeds such as PlanetLab [4], and storage networks such as HP's Federated Array of Bricks (FAB) [14]. However, this increase in the availability of such valuable computing resources has led to a rise in the occurrence of large-scale attacks on these systems. These attacks may come in the form of worms like Code Red [23] or Sapphire [9], or as explicit hacking, such as the recent Cisco attacks [5] or intrusions of TeraGrid systems [20].

Although there exist a variety of single-host and cooperative automatic intrusion detection systems [10, 13, 29, 31], distribution mechanisms for alert data have typically been regarded as secondary to detection capabilities. This lack of attention to alert distribution can allow adversaries to insert false alerts into a detection network, resulting in false positives or false negatives.

This paper proposes the ContagAlert protocol, a new technique for automatic, peer-to-peer alert propagation that is tolerant of disruptive alerts. We define a *legitimate* alert as an alert derived from an event that will be observed everywhere or almost everywhere. An example of such an event would be an Internet worm spreading very quickly throughout the entire network. We define a *disruptive* alert as an alert related to an event that will not be observed everywhere or almost everywhere. For example, a malicious or faulty node might report a non-existent attack. From the perspective of ContagAlert, this event would be considered a false positive because it will not be observed throughout the entire network and, therefore, does not interest all of the nodes in the network. We assume that ContagAlert will only be used to spread alerts about events that will be observed at almost every node in the network. All other alerts should be suppressed. Due to the design of the ContagAlert protocol, all network peers receive legitimate alerts with high probability and only a small minority of the network receives disruptive alerts. Although we focus this discussion on intrusion alerts, the approaches in this paper can be used for any sort of signal that one might want to propagate throughout a distributed network. Thus, "alert" can be taken to mean "any message or signal propagated using the ContagAlert protocol."

The ContagAlert protocol is based on a variation of signal propagation ideas from contagion theory[1]. Contagion theory studies the spread of influences, such as social fads, power grid failures, biological epidemics, or, in our case, network messages, across interconnected populations.

*This work was supported in part by NSF ITR grant CMS-0427089, in part by NSF CAREER grant CNS-0448246, and in part by the AT&T Labs Fellowship Program. Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

[1] Standard contagion theory employs a fractional threshold model, where a node becomes alerted after a certain fraction of its neighbors are alerted. In order to simplify our implementation and analysis, we use a variation on this model in which a node becomes alerted after some fixed number of its neighbors are alerted.

These populations are modeled as a graph, describing the relationships (edges) between individuals (vertices) in the population. The influence originates in some number of seed individuals in the graph, and spreads across the graph using a threshold-based activation rule [30]. A *contagion* has occurred if all nodes eventually become infected. Without resorting to central control, the ContagAlert protocol carefully adjusts this alert[2] threshold on individual nodes, thereby changing the critical number of seeds required to trigger a contagion. It sets the threshold to a point that exceeds the expected number of disruptive alerts, while still remaining below the expected number of legitimate alerts.

2 Related Work

The goal of our work is to provide a protocol that prevents malicious messages and false positives from interfering with the well-behaved nodes in the network. Most systems accomplish this by attempting to keep malicious nodes out of a system entirely using a trust mechanism. There are two basic approaches to instituting trust relationships in a network: certificate authorities [8, 17, 25, 33], where each node can be held accountable for the messages it sends, and "webs of trust" [1, 18, 21, 27]. Both approaches have problems if a malicious agent is able to attain a trusted status.

The ContagAlert protocol avoids these issues by discarding the idea of trust entirely, and instead relying on the premise that legitimate messages will be generated by a larger fraction of the network than an adversary would be able to subvert to his own uses.

The "friend protocol" described by Nojiri, et al. [24] bears some similarity to the ContagAlert protocol. As in the ContagAlert protocol, each node has some number of neighboring peers, and if the number of alerted neighbors exceeds a certain threshold, the node becomes alerted itself and notifies its neighbors of the new alert. The differences are significant, however. In the Friend Protocol, all nodes use the same, fixed threshold setting. Furthermore, the Nojiri paper does not fully explore or exploit the threshold behavior of the protocol they describe. By comparison, in ContagAlert, the node threshold settings are adaptively set to accommodate any network structure. Additionally, ContagAlert allows for the tolerance level for disruptive messages to be specified, and the threshold behavior is automatically adjusted to meet the specification.

The Vigilante [6] system uses an alternative approach to propagating alert information. In Vigilante, self-certifying alerts (SCAs) are distributed to contain Internet worms. When a host receives an SCA, it is able to inexpensively prove to itself that the vulnerability is real and thus take steps to counter it. The detection engines used in Vigilante are assumed to be highly accurate and capable of producing SCAs that can be verified at other hosts. However, many detection engines are less accurate than those assumed by Vigilante, and thus are more susceptible to producing false positives. For example, a detection engine monitoring the incoming traffic on a range of port numbers might mistakenly report a worm attack that it has inferred by observing a port scan unrelated to any actual attack. Networks using such detection engines would benefit from the ContagAlert protocol, which can tolerate such disruptive alerts. Another difference between Vigilante and our work is that while Vigilante requires a small dedicated group of machines to act as the nodes for alert propagation, ContagAlert has more of a true peer-to-peer design, where all individual nodes run detectors.

[2]We use "alerted" instead of "infected" to refer to a node that has adopted an alerted state. We reserve the term "infected" to refer to a node under the influence of a malicious interference such as a worm or other intrusion.

3 Protocol Design

The primary purpose of the ContagAlert protocol is to rapidly distribute security alerts of a widespread attack throughout a peer-to-peer network, while suppressing uncommon erroneous messages and messages generated by malicious adversaries. We refer to such messages as *disruptive*.

The protocol is intended to be used in applications where the sources of legitimate alerts have a high degree of redundancy. We assume that a threshold exists, stated as a fraction of the number of peers in the system, which separates the legitimate alerts from disruptive alerts based on the number of sources of the alert. In other words, disruptive nodes cannot compose an arbitrarily large fraction of the network, and any legitimate message must be generated at multiple peers. Finally, it is important that benign nodes be willing to cooperate in the execution of the protocol. This assumption should be valid for any network where peers have any interest in the survival of the network.

3.1 Basic Protocol

The basic approach of the ContagAlert protocol is very simple. Peers in the cooperative network each have a small list of neighbors whose network locations are known. These peer relationships need not be reflexive. At various times, certain peers will observe an event (e.g., an attempted intrusion) and will respond by generating an alert, which is sent to its neighbors (e.g., to warn of the possibility of imminent intrusion). These neighbors each independently decide whether to act on the alert, taking appropriate responsive actions and forwarding the alert to their respective neighbors. This decision procedure of whether or not to act represents the core of the ContagAlert protocol.

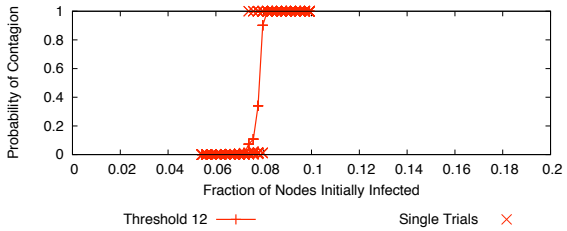


Figure 1: Contagion Threshold Behavior – 4096 Nodes with Average Degree 64, Threshold 12

It should be noted that ContagAlert supports a variety of alert message types. For example, alerts could be simple alert messages with no other payload, or could contain port numbers being attacked in a DoS attack, or could perhaps even contain code that was found to be malicious. The implementation details of such alert messages are beyond the scope of this paper, as our primary concern is focused on the actual propagation characteristics of the alerts themselves.

Each peer in a system using ContagAlert has its own, independent threshold to determine whether or not it will forward an alert message to all of its neighbors. This threshold is expressed as a constant number of messages. For example, if a node has a threshold of 2, it will forward a message to all of its neighbors if it first receives a copy of the message from at least two different nodes. We refer to this piece of the protocol as the *basic* ContagAlert protocol.

This type of decision procedure results in very clear threshold behavior emerging from the network as a whole, as shown in Figure 1. This figure shows the probability of a contagion occurring given the fraction of nodes seeding the alert. A 4096-node network with an average of 64 directed edges per node was used for these trials[3]. The X-axis represents the number of instances of a simulated alert that were used to seed the propagation. The Y-axis represents the fraction of the network that accepted the alert as legitimate and forwarded the data to neighboring nodes. Thirty trials were run for each seed value. The trend line in the graph shows the average value of all thirty trials for each seed point. The 'X' points show the results of each individual trial in the data for the threshold of 12.

The frequency of contagion is 0% at the left of the series and 100% at the right. In the middle is a region where, for certain numbers of seeds, a contagion may or may not occur. Note that the single-trial points always occur at the very bottom edge or the very top edge of the plot. This indicates that there is never a "half-contagion" where a cascade spreads to a large number of non-seed nodes but does not complete the contagion. Also, although a contagion is more likely if there are more seed nodes, a contagion starting from few seeds appears identical to a contagion starting from many seeds. We call the steeply sloped region of the average line the *threshold region*.

It is a crucial observation: given a single set of seed signals, after evaluating the alert propagation it is impossible to assess where the trial lies with regard to the threshold region. The result indicates only whether or not a contagion occurred, but it is the probability of contagion for the number of seeds that is most important. We call this observation "Axiom A".

Other data we collected shows that this type of threshold behavior observed in Figure 1 occurs regardless of the network size, topology, and average node degree. However, these characteristics will cause the network's threshold region to shift to the left or right, or to stretch wider or narrower. In order to shift the threshold region to a specified point, the activation thresholds on individual nodes must be adjusted to account for changes in these traits. Since these traits are difficult to ascertain for a decentralized, peer-to-peer system, and since they can change over time, the protocol parameters cannot be specified ahead of time or computed offline. Instead, the protocol must *adaptively* set the threshold setting for each node in the system at runtime.

3.2 Adaptive Protocol

The goal of the ContagAlert protocol as a whole is to generate a network-level threshold behavior, with the threshold region centered on the fraction of nodes that must be seeded for a contagion to be possible. We call this fraction the *target tolerance*, because it describes the approximate number of disruptive source nodes that can be tolerated by the protocol before the message is propagated throughout the rest of the network. Signals with too few seeds are implicitly assumed to be disruptive and will not spread, while signals with enough seeds are implicitly assumed to be legitimate and will result in contagion. Messages are never explicitly classified as legitimate or disruptive.

The principal difficulty in constructing such a system is the adjustment of the contagion threshold to the target tolerance. Even with an omniscient, centralized controller it is not clear how this would be accomplished. It is difficult to assess how the many network parameters will affect the threshold region, and due to Axiom A, it is not possible to estimate the current position of the threshold region with a single evaluation of the network.

The difficulty is compounded when each node in the system is responsible for adjusting its own threshold based on its limited, local view. Additionally, a node has control only over itself. If it should decide that a neighbor node is incor-

[3]Unless otherwise noted, all networks have random connectivity. If a disconnected graph was generated, it was discarded and a new network was created. We generated data that showed that the ContagAlert protocol was effective for a variety of other network topologies, but these results are outside the scope of this paper.

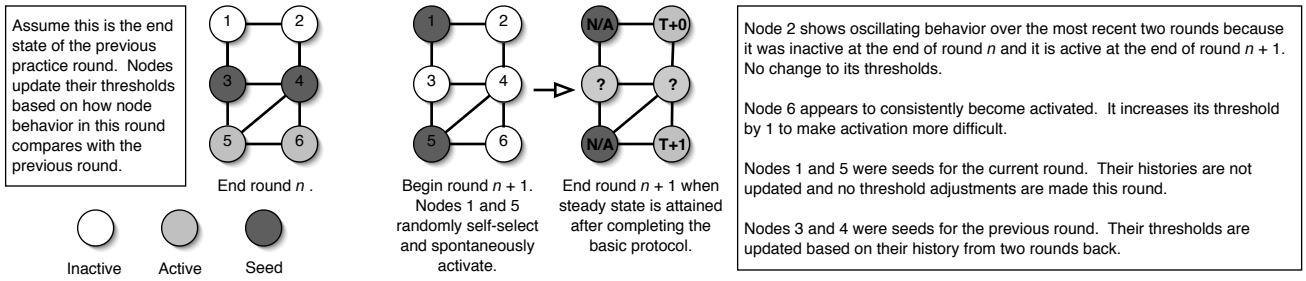


Figure 2: Adaptive Procedure

rectly adjusted, it has no way to force that node to listen. Finally, although a node n possesses a list of neighboring nodes, these are not necessarily the nodes that are sending alerts to n. A node has no direct means of knowing how many other nodes it should expect to hear from or what local conditions exist for those nodes.

The ContagAlert protocol addresses these difficulties by having the network periodically execute "practice" runs of the basic ContagAlert protocol to allow nodes to experiment with different threshold settings. Nodes in the network periodically generate synthetic test messages and attempt to spread these messages as though they were real.[4] Messages are generated at the target tolerance level of the protocol. The goal of the protocol is for each node to adjust its threshold such that a contagion occurs one-half of the time. If this goal is attained, the threshold region is shifted to be approximately centered on the target tolerance fraction. This behavior is illustrated in Figure 2, and is described in additional detail below.

During the adaptive phase, each node periodically generates a practice message with probability p, where p is the target tolerance fraction for the network. For a network composed of N nodes, this inserts approximately $p \times N$ simulated signals into the system. If all node thresholds are set correctly, this is exactly the number of signals that should result in the most uncertain behavior from the protocol. Therefore, the desired behavior is that a contagion occur exactly one-half of the time. Attaining this alternating behavior is the goal of the adaptive aspect of the ContagAlert protocol.

After deciding whether to generate a message, each node adheres to the basic ContagAlert protocol, forwarding the practice messages (or not) according to their own threshold T. This continues for a period of time, until no messages are received for a timeout period and it can be assumed that the contagion would have completed if a contagion were going to occur. Each node decides locally when to time out and end the practice round.

After the time period expires, each node would like to assess whether or not a contagion has occurred in the network. However, since nodes possess only local information, they cannot make such a determination for the network as a whole without layering a consensus protocol on top of the existing ContagAlert system. For now, assume that each node possesses an oracle that can determine whether or not the network has experienced a contagion.

If a contagion has occurred, each node assumes (perhaps incorrectly) that the network as a whole is somewhere to the right of the threshold region of the graph. Otherwise, it assumes the network is left of the threshold region. The node then adjusts its personal threshold to inhibit or facilitate future contagions. Since the node is guessing the most probable state of the threshold region, it will be correct the majority of the time. As a result, the threshold region should approach the target tolerance level after a large number of practice runs.

After each practice round, each node attempts to guess whether the threshold region of the network is left or right of the target tolerance. It examines the contagion history for the previous two rounds to attempt to determine the present network state. This two-round time window allows the node to classify itself into one of three categories:

1. Both of the last two rounds resulted in contagions. This suggests that the node's threshold is set too low and that the node is becoming alerted too easily. In this case, the node responds by increasing its threshold by 1.

2. Both of the last two rounds resulted in no contagion. This suggests that the node's threshold is set too high and that the node is having too much difficulty becoming alerted. In this case, the threshold is reduced by 1.

3. The last two rounds have produced differing contagion results. This indicates that the node has attained the desired alternating behavior. No changes are necessary.

Each node that has not been a spontaneously activated

[4]These test messages are entirely separate from the normal messages propagated by the network. These messages have no direct relationship to real signals, and vice versa.

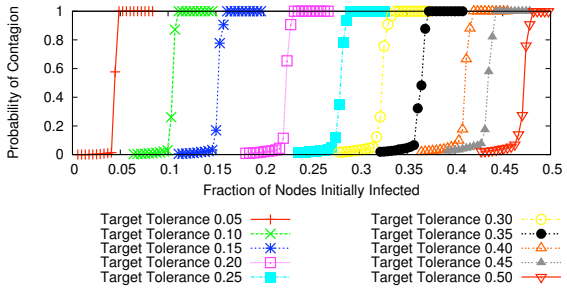


Figure 3: Threshold Behavior of the ContagAlert Protocol Adapted to Different Target Tolerances – 4096 Nodes of Average Degree 64

seed in the current round adjusts its threshold as dictated by its self-assessed categorization[5]. Over many test rounds, its threshold approaches the point where, when considered with all other nodes in the network, the network's tolerance threshold for malicious nodes is shifted to the specified level. Signals with a number of sources below the threshold will be suppressed, while signals exceeding the threshold will be distributed throughout the network.

Since individual nodes do not actually possess an oracle, there remains the issue of determining whether or not a contagion has occurred. As mentioned previously, peers possess information only about themselves unless they explicitly query neighboring nodes for their states. No individual node has information about the global network state.

However, it is actually very simple for a node to make an accurate guess about the network state based only on its own information. Recalling Figure 1, when a signal is able to propagate at all it usually results in a complete contagion. As a result, a node can simply look at its own activation state at the conclusion of the practice round and it will be correct with high probability.

4 Protocol Simulation Results

We implemented a simulator to examine the adaptivity performance of the protocol. The simulator generates a network graph of a specified size and degree. The adaptive protocol is executed for a large number of rounds with a specified target tolerance to allow the node thresholds to stabilize.

Once the adaptive phase is completed, the simulator runs a large number of trials to assess the behavior of the network when subjected to "real" signals with various numbers of starting seeds. The simulation for each seed value was repeated 30 times in order to assess the probability that a contagion would occur for any given number of seeds. The large number of different seed values, combined with the large number of repetitions for each seed value, allows the probability curve of the network's behavior to be closely estimated.

The simulation proceeds in rounds. On each round, each node counts the number of alerts it has received from any neighbors. If this number exceeds its threshold, the node becomes alerted and broadcasts the alert to all of its neighbors. This round procedure repeats until the network state achieves a stable, unchanging state.

Figure 3 shows the behavior of simulated networks after the adaptive period has completed. Each trend line represents a different target tolerance. This data indicates that the adaptation and various target tolerance settings definitely have an effect on the network behavior. Although the adaptation is not exactly perfect, each trend line appears in the correct region of the plot. The ordering of the lines is correct, indicating that setting a higher target tolerance for the protocol will result in a higher contagion threshold for the network.

5 Protocol Analysis

The ContagAlert protocol views a peer-to-peer network as being akin to a cellular automaton on an irregular topology. In particular, the basic form of the protocol bears a close resemblance to *bootstrap percolation*, a family of deterministic cellular automata using a class of activation rules on an n-dimensional lattice.

Although ContagAlert is inspired by and draws ideas from contagion theory, due to the modification of the decision procedure on a node to use a constant threshold instead of a fractional one, it is more straightforward to analyze the system from the angle of *bootstrap percolation*. Bootstrap percolation describes a specific deterministic cellular automaton rule on a two-dimensional lattice of cells. Each cell is activated with probability p at time $T = 0$. On each subsequent time step, inactive cells become active if at least two of their neighbors are active. Cells never deactivate. More general, but less common models of bootstrap percolation allow for d dimensional lattices, and cells activate when l neighbors are active.

The study of bootstrap percolation focuses on determining the probability that the entire lattice will become activated, given the fraction of lattice nodes that are active at the beginning of the process. It has been shown that this probability shows a sharp threshold behavior. The probability shifts from nearly 0 to nearly 1 over a very small increase in the fraction of seed nodes. The ContagAlert protocol leverages this threshold effect to suppress malicious signals and false positives, while allowing legitimate signals to spread normally.

The basic aspect of the ContagAlert protocol maps closely onto this definition of the bootstrap percolation sce-

[5]Making threshold adjustments based on spontaneous activation prevents convergence to the appropriate value.

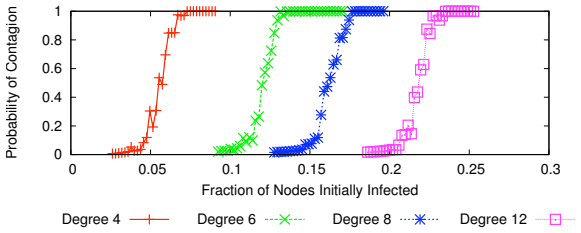


Figure 4: Threshold Behavior of the Lattice Topology – 4096 Nodes, Activation Threshold = Node Degree / 2

nario.Each peer participating in the protocol corresponds to a cell in bootstrap percolation. The neighbors of the peer correspond to the neighboring cells in the lattice. The threshold number of alert messages that a peer must receive before becoming alerted corresponds to l, the threshold number of activated neighbors required to cause a bootstrap percolation cell to become activated. The fraction of peers acting as the source of an alert or other signal corresponds to the activation probability p. The only difference between the two systems is in the topology of the neighbor relationships. While the ContagAlert protocol can function on any network topology, bootstrap percolation is limited to only d-dimensional lattices.

In both bootstrap percolation and the ContagAlert protocol, it is easy to see that if p is too small, few or no nodes will be initially activated. Any spreading of the activation state that does occur will stall after a small handful of activations, and the remainder of the lattice will remain inactive. Inversely, if p is high, most of the nodes will be initially activated, and the remaining inactive nodes will become active after just a few time steps.

The probability of the active state completely filling the network (a contagion) can be considered as a function of p. Although one might expect this probability to begin at 0 when p is small and gradually increase to 1 as p nears 1, this is not actually the case. In fact, this probability remains close to 0 up to some threshold fraction p, at which point it rapidly increases to 1 as p increases just a small amount. When p exceeds this threshold point, the lattice is almost guaranteed to achieve 100% activation.

It is exactly this threshold behavior in bootstrap percolation systems that the ContagAlert protocol exploits. The nature of this threshold point is the common topic of research regarding bootstrap percolation. If it is assumed that the ContagAlert protocol is running on a d-dimensional lattice topology, the results of bootstrap percolation research can be directly applied to this work.

In bootstrap percolation, it has been proven that for an $L \times L$ lattice, as L becomes large, the probability that the entire lattice becomes alerted approaches 1 if the fraction of seed nodes is greater than $(\pi^2/18)/\ln(L)$, and it approaches 0 if the fraction of seed nodes is less than this value [16]. Therefore, for a 4096 node network with a 2-dimensional lattice topology and an activation threshold of 2 on every node, we expect the contagion threshold to occur at $(\pi^2/18)/\ln(\sqrt{(4096)}) = 0.1318$, or 540 seed nodes. Although the degree 4 trend line in Figure 4 shows that this is not actually the case, this can be attributed to the fact that the convergence to this threshold is extremely slow as the number of nodes increases. In other words, this limit is approached only for extremely large networks. [2].

A similar result has been proven for n-dimensional bootstrap percolation for $n > 2$ dimensions with arbitrary activation threshold l [3]. In this case, however, the result is more complex and less precise.

For $2 < l \leq d$, there exist two constants, $\alpha_-(d,l)$ and $\alpha_+(d,l)$, such that

$$0 < \alpha_-(d,l) \leq \alpha_+(d,l) < \infty$$

independent of p, such that if

$$L_\pm(d,l,p) := exp^{\circ(l-1)}(\alpha_\pm p^{-\frac{1}{d-l+1}})$$

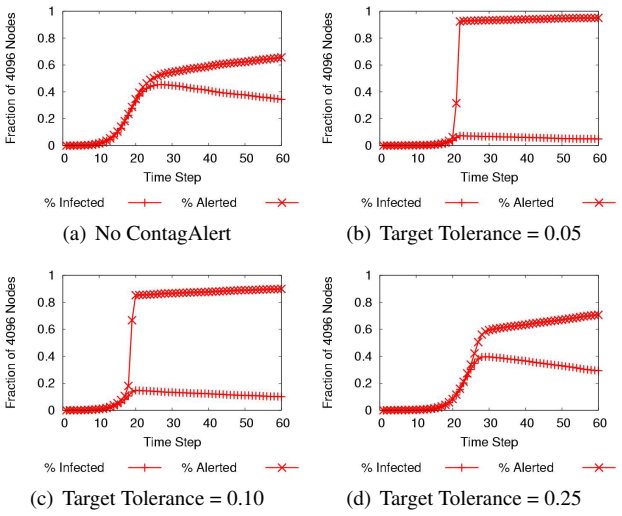
then

$P(contagion) \to 1$ if $(p, L) \to (0, \infty)$ with $L \geq L_+(d,l,p)$
$P(contagion) \to 1$ if $(p, L) \to (0, \infty)$ with $L \leq L_-(d,l,p)$

This result is proven by induction, showing how scenarios with arbitrarily large d and l can be reduced down to the 2-dimensional, threshold 2 case, one step at a time. However, they are unable exactly to specify the threshold. Instead, they are able to prove that the threshold exists, and they are able to reduce the uncertainty to the values α_+ and α_-, which are functions of only d and l.

The results presented above correspond only to n-dimensional lattices. Obviously, very few peer-to-peer overlays will have this type of topology. As topologies become more irregular, these thresholds will vary in a manner that may be impossible to quantify. The theoretical results above are largely based on the predictable way in which an activation state will propagate across the lattice. This predictability is not shared by networks with any kind of random component, and it can be hard to see even in topologies with regular structures.

However, we can qualitatively consider the behavior we would expect to observe from these more common topologies. We approach the problem by considering the clustering coefficient of each structure. If nodes A and B are neighbors, and if nodes B and C are neighbors, the clustering coefficient of the topology is the probability that node A and C are neighbors. A high clustering coefficientsuggests that large cascades are easier to trigger, because if any



(a) No ContagAlert (b) Target Tolerance = 0.05

(c) Target Tolerance = 0.10 (d) Target Tolerance = 0.25

Figure 5: Comparison of Alerted Nodes to Infected Nodes as Simulated Worm Attack Progresses – 4096 Nodes, Average Degree 64

one node in a cluster becomes activated it is likely that the others in the cluster will follow. The lattice topology has a clustering coefficient of 0, since there exist no triangular loops of edges in this type of topology.

6 Application Data

We studied the behavior of the protocol in two applications to assess its effectiveness outside of an entirely abstract environment.

6.1 Internet Worm Attack

We modified our existing simulator by replacing the fixed number of seeds with an increasing number of seeds modeling the spreading behavior of a worm. The spreading behavior of the worm is based on a variant of the model presented by Kephart in [19]. Alerted nodes spread warning information to other nodes using the ContagAlert protocol in an attempt to inhibit the further spread of the worm.

The speed of the spread of an alert across the network is primarily dependent on the target tolerance of the network. If the tolerance is set higher, more seeds are required to tip the network past the contagion threshold and the propagation of warnings will be delayed. This allows the worm to attain a stronger foothold in the network before defensive measures are raised. If the target tolerance is set lower, the opposite behavior will occur, but it may open the possibility for disruptive messages to spread.

The simulator generated a network of a size, and average degree. As with the earlier simulations, the network was

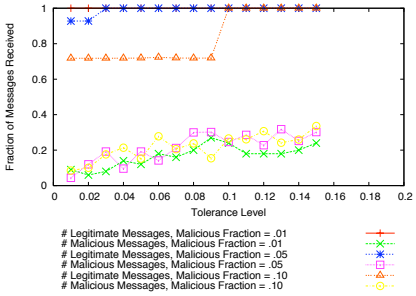


Figure 6: Fraction of Messages Received in Pastry Application – 100-Node Network

adapted to a specific target tolerance with interference before testing its performance. After the adaptive period was completed, a simulated worm was released on the network. The initial infection was seeded at a single, randomly selected node. After the initial infection, we used an infection probability of $p = .5$ and a cure rate of $c = .01$. A trace of the network state was collected, recording how many nodes were in an infected state and how many nodes were in an alerted state at each time step.

Figure 5 compares the number of infected nodes (dashed lines) to the number of alerted nodes (dotted lines) in the network at each time step for a typical simulation run. The figure shows that a target tolerance of 0.05 or 0.10 results in a dramatic reduction in the number of nodes that become infected by the worm. With a large target tolerance of 0.25, however, ContagAlert is of limited effectiveness when trying to compete against a rapidly spreading worm.

6.2 DoS Attack Against a P2P Network

We also implemented a P2P application, called DoS-Alert, as a testbed for the ContagAlert protocol. In DoS-Alert, each peer monitors for attack patterns in the traffic passing through its node. If a node detects that a target destination is being overwhelmed by a particular source, then that node will start dropping messages from that source and will propagate an alert to other nodes identifying that source as an attacker. Also, if a node receives enough distinct alerts identifying a particular source as an attacker, then that node will begin dropping requests from that attacker and will propagate the alert to all of its neighboring nodes.

We evaluated the effectiveness of the ContagAlert protocol in this environment with a trace-based simulation of 100 nodes. Using the FreePastry API, we implemented DoS-Alert on top of the Pastry P2P routing substrate [12], [26]. The application underlying the DoS detection in the simulation was a peer-to-peer cooperative Web caching scheme where each well-behaved peer requests Web objects based on a randomly chosen, single machine trace [7]. Malicious nodes bombard targets with a large number of re-

quests and also inject false alerts into the system identifying well-behaved peers as attackers.

Figure 6 shows that lower tolerance levels often lead to malicious requests being dropped more aggressively. Unfortunately, lower thresholds also make it easier for attackers to inject false alerts into the system that identify benign nodes as attackers. However, in the above results, when the percentage of malicious nodes is less than or equal to the tolerance level, no legitimate messages are dropped (i.e., false alerts from attackers are completely ineffective) and over 60% of the attackers' requests are dropped.

7 Conclusion

We introduced the ContagAlert protocol, a new protocol oriented towards disseminating alerts across a peer-to-peer network. The protocol leverages the threshold behavior studied in contagion theory to suppress messages of malicious intent while causing legitimate messages to spread. The protocol is able to adaptively set the contagion threshold of the network to specify a target tolerance level for malicious messages. This is accomplished by adjusting the activation threshold of individual nodes using an adaptive distributed algorithm.

References

[1] A. Abdul-Rahman and S. Hailes. A distributed trust model. In *Proceedings of the 1997 workshop on New security paradigms*, pages 48–60. ACM Press, 1997.

[2] J. Adler, D. Stauffer, and A. Aharony. Comparison of bootstrap percolation models. *Journal of Physics A*, 22:L297–L301, 1989.

[3] R. Cerf and F. Manzo. The threshold regime of finite volume bootstrap percolation. In *Mathematical Physics Preprint Archive*, July 2001.

[4] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3), July 2003.

[5] Breach that hit cisco wider than thought. *CNN/Money*, May 2005.

[6] M. Costa, J. Crowfort, M. Castro, A. Rostron, L. Zhou, L. Zhang, and P. Barham. Vigilante: End-to-end containment of internet worms. In *Symposium on Operating Systems Principles*, 2005.

[7] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of www client-based traces. Technical Report TR-95- 010, Boston University Department of Computer Science, 1995.

[8] D. Dhillon, T. S. Randhawa, M. Wang, and L. Lamont. Implementing a fully distributed certificate authority in an OSLR MANET. In *Wireless Communications and Networking Conference (WCNC)*, 2004.

[9] D. M. *et al*. The spread of the Sapphire/Slammer worm. http://www.caida.org/outreach/papers/2003/sapphire/, 2003.

[10] J. S. B. *et al*. An architecture for intrusion detection using autonomous agents. In *ACSAC*, pages 13–24, 1998.

[11] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 2001.

[12] FreePastry. http://www.cs.rice.edu/CS/Systems/Pastry/FreePastry/.

[13] D. Frincke, D. Tobin, J. McConnell, J. Marconi, and D. Polla. A framework for cooperative intrusion detection. In *21st National Information Systems Security Conference*, pages 361–373, October 1998.

[14] S. Frolund, A. Merchant, Y. Saito, S. Spence, and A. Veitch. FAB: Enterprise storage systems on a shoestring. In *9th Workshop on Hot Topics in Operating Systems (HOTOS)*, June 2003.

[15] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse. Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.

[16] A. E. Holroyd. Sharp metastability threshold for two-dimensional bootstrap percolation. *Probability Theory and Related Fields*, 125(2):195–224, 2003.

[17] R. Hunt. PKI and digital certification infrastructure. In *Networks*, 2001.

[18] The International PGP Homepage. www.pgpi.org.

[19] J. O. Kephart and S. R. White. Directed-graph epidemiological models of computer viruses. In *IEEE Computer Society Symposium on Research in Security and Privacy*, 1991.

[20] B. Krebs. Hackers strike advanced computing networks. *Washington Post*, April 2004.

[21] A. D. R. Marc Waldman and L. F. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. 9th USENIX Security Symposium*, pages 59–72, August 2000.

[22] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2):226–251, 2001.

[23] D. Moore, C. Shannon, and J. Brown. Code-red: A case study on the spread and victims of an Internet worm. In *Proceedings of the Internet Measurement Workshop (IMW)*, 2002.

[24] D. Nojiri, J. Rowe, and K. Levitt. Cooperative response strategies for large scale attack mitigation. In *DARPA Information Survivability Conference and Exposition*, pages 293–302, 2003.

[25] J. S. Park and R. Sandhu. Binding identities and attributes using digitally signed certificates. In *Computer Security Applications (ACSAC)*, 2000.

[26] A. Rowstron and P. Druschel. Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.

[27] B. Sniffen. Trust economies in the Free Haven project, 2000.

[28] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[29] V. Vlachos, S. Androutsellis-Theotokis, and D. Spinellis. Security applications of peer-to-peer networks. *Comput. Networks*, 45(2):195–205, 2004.

[30] D. J. Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences (PNAS)*, 99(9), 2002.

[31] G. White, E. Fisch, and U. Pooch. Cooperating security managers: A peer-based intrusion detection system. *IEEE Network*, 10(1), 1994.

[32] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE Journal on Selected Areas in Communications*, 2003. Special Issue on Service Overlay Networks.

[33] L. Zhou, F. B. Schneider, and R. V. Renesse. Coca: A secure distributed online certification authority. *ACM Trans. Comput. Syst.*, 20(4), 2002.