On Achieving Software Diversity for Improved Network Security using Distributed Coloring Algorithms

Adam J. O'Donnell*
adam@ece.drexel.edu

Harish Sethu
sethu@ece.drexel.edu

ECE Department
Drexel University
3141 Chestnut St.
Philadelphia, PA, USA

ABSTRACT

It is widely believed that diversity in operating systems, software packages, and hardware platforms will decrease the virulence of worms and the effectiveness of repeated applications of single attacks. Research efforts in the field have focused on introducing diversity using a variety of techniques on a system-by-system basis. This paper, on the other hand, assumes the availability of diverse software packages for each system and then seeks to increase the intrinsic value of available diversity by considering the entire computer network. We present several distributed algorithms for the assignment of distinct software packages to individual systems and analyze their performance. Our goal is to limit the ability of a malicious node to use a single attack to compromise its neighboring nodes, and by extension, the rest of the nodes in the network. The algorithms themselves are analyzed for attack tolerance, and strategies for improving the security of the individual software assignment schemes are presented. We present a comparative analysis of our algorithms using simulation results on a topology obtained from e-mail traffic logs between users at our institution. We find that hybrid versions of our algorithms incorporating multiple assignment strategies achieve better attack tolerance than any given assignment strategy. Our work thus shows that diversity must be introduced at all levels of system design, including any scheme that is used to introduce diversity itself.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; K.6.5 [**Management of Computer and Information Systems**]: Security and Protection—*Invasive software*

General Terms

Algorithms, Security, Management

Keywords

Network security, survivability, software monoculture, software diversity, graph coloring, viruses and worms

1. INTRODUCTION

A great deal of attention in computer security research has recently been devoted to the security implications of the software monoculture present in the Internet. Researchers who espouse the belief that the current lack of software diversity is troublesome assert that security can only be achieved in a real network if a multitude of software packages is utilized. It is reasoned that by increasing the number of different software systems deployed the effectiveness of a single system-specific attack can be minimized. Position papers that assess the inherent value of a heterogeneous population of software packages have been published in both peer-reviewed conferences [33] and in more public forums [13, 28]. Work has been done to introduce diversity at the system level through a variety of techniques, including both source [22] and instruction set [4, 20] randomization. Researchers have yet to examine the problem of distributing diversity from a network-aware perspective that would decrease the rate at which an attacker can progress across the network.

In this paper, we show that randomization of individual systems is insufficient for increasing the diversity of the network as a whole. We show that it is possible to distribute software packages to systems across a network topology to increase the inherent effectiveness of software diversity at slowing an oncoming worm or hacker. We describe a series of distributed algorithms which, through the systematic introduction of diversity into the network, reduce the ability of an attacker to move from system to system. The algorithms are analyzed from the standpoint of the quality of diversity introduced into the network and the tolerance of the algorithm to attack. Such a topologically aware distribution of heterogeneous software would achieve the stated goals of software diversity. Rather than being able to leap-frog from one identical system to the next across the network, hackers would be limited to clusters of similar systems by the size of their toolkit. The rest of the network, however, would only be reachable by traversing systems which are dissim-

*The author's work was supported by the NSF Graduate Research Fellowship and the Koerner Family Fellowship.

CCS'04, October 25-29, 2004, Washington, DC, USA.

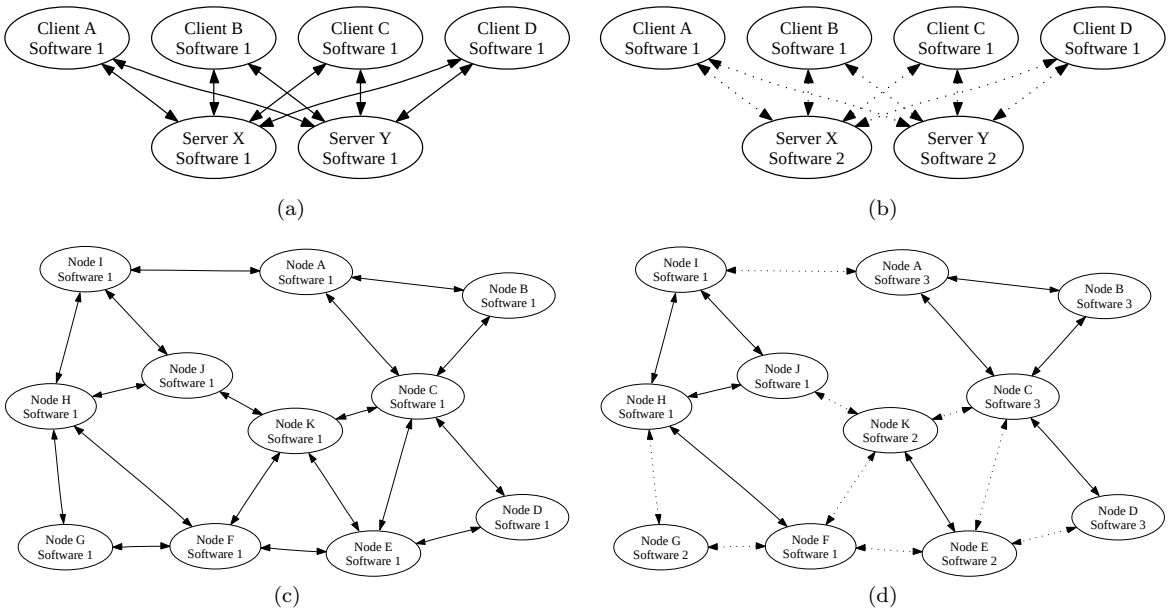


Figure 1: Comparison of network topologies utilizing either a single software package or a diverse software distribution. The effect of optimally distributing two software packages on a bipartite network is clear in (a) and (b). Bipartite network such as these are often found in client-server file sharing topologies. Likewise, a random network topology clearly benefits from a random distribution of three heterogeneous software packages (d) as compared to a uniform distribution of a single package (c). While the assignment is sub-optimal, the number of edges which exist between nodes running similar software packages is clearly reduced.

ilar, from a vulnerability standpoint, as compared to the node from which the attack is launched.

1.1 Applications of Secure Diversity

E-Mail Topologies: Any individual that utilizes e-mail has become a target of self-propagating code. Vulnerabilities associated with the default configurations of MIME handlers [15] have given rise to client-side computer viruses [14]. Errors in the parsing code in major mail transfer agents have resulted in server-side attacks that are also propagated via e-mail traffic [21]. Secure diversity can be implemented in the stated situation through the utilization of interchangeable MIME and e-mail header parsers which are selected by the application based upon a topology-sensitive algorithm. Replacing one parser library with another would have no user-discernible impact on the software's behavior and performance.

Client-Server File Shares: Network-accessible file shares have become a popular target for platform-dependent worm propagation [17]. In many office environments, the file shares are partitioned into the client and server groups as shown in Figure 1(a), where communication links between similar systems are represented by a solid line. This partitioning can be enforced using firewalls and ACLs. A worm infection on a client system would be able to self-propagate to any machine in the file-sharing topology by first attacking a server machine; likewise, a worm infection on a server would have to first attack a client before propagating further.

The secure diversity principle can be quite effectively applied to such a network with only two different software packages. All previous communication links between similar systems are replaced by links between dissimilar computers, represented by the dotted lines in Figure 1(b). By utilizing a second software package for file sharing on the server systems, it is possible to prevent a client system from propagating a worm that attacks a vulnerability in the file sharing subsystem.

Sensor Networks: The networking field that would benefit greatly from the secure diversity principle is sensor networks [9]. Enforcing a diversity policy in a sensor network is less of an administrative challenge, since these large networks of relatively simple computational and environmental monitoring nodes are usually controlled by a single entity, be it a military commander or a building supervisor. Because the hardware is characterized as being relatively simple, it is not a major technical challenge to recreate their comparatively small software suite for the purposes of introducing variation between individuals in the population.

Consider the possibility of a system-wide vulnerability that allows for an attacker to take over a single networked sensor. A single attack can be used to leap-frog from node to node across the entire network, as indicated by the bidirectional links in Figure 1(c). Sensor networks can be distributed with multiple operating systems in ROM. After being dropped into the operational location, a node can load up one of a multiple set of OSes. By constructing a network that contains a multiplicity of operating systems, a single operating system-specific attack will not be able to propagate across the entire breadth of the network. Such a randomized distribution of software packages, as shown in 1(d), can reduce the number of possible node-to-node movements by an attacker.

1.2 Contributions

Our goal in this work is to increase the value of system level diversity through the introduction of a topology-aware

software assignment scheme. This paper is inspired by the philosophy described in [33], but provides a series of distributed algorithms to achieve the goals laid down therein. Unlike [18], our work does not require centralized and complete knowledge of the topology; the algorithms are designed to utilize information available locally to a node. Individual nodes work to reduce the ability of an attacker to utilize any given node to launch an attack on any of its neighbors, and by extension, any other node reachable in the network. Our objectives become:

1. A minimization of the number of neighbors running the same software packages

2. A maximization of the number of disconnected "islands" of nodes running the same software packages

These objectives, referred to as the defective edge count and the connected component count, are not orthogonal. A local reduction in the number of neighbors running the same software package globally reduces the number of edges an attacker can use to propagate an attack. A global increase in the number of disconnected components increases the number of initial nodes that must be taken by an attacker if he or she wishes to compromise every node on the network.

Our algorithms are based on examining local information and making local decisions. They work by directly decreasing the defective edge count and indirectly improving the connected component count. We have examined these algorithms through analysis and simulation, as shown in Sections 4 and 6.1.

Given the purpose of the software distribution algorithm, it is logical to explore the vulnerability of the coloring algorithms themselves from the standpoint of an attacker. Based upon this reasoning, we have developed a series of attacks against our own algorithms and explored their effectiveness through simulation. These attacks do not rely upon attacking implementation flaws in the algorithms, but instead are based on malicious nodes attempting to deceive well-behaving nodes running the algorithm. The results of this simulation work are presented in Section 6.2.

In Section 6.3, we draw several conclusions from our examination of the simulation results. Our explorations of the attacks' effects on the coloring algorithms presented give rise to the observation that *there exists a tradeoff between an algorithm's tolerance to attack and the quality of the software assignment created by the algorithm.* Furthermore, we show that revisiting the initial thesis on the value of diversity is applicable in the design of software assignment scheme when an algorithm designer wishes to increase the algorithm's tolerance to a directed attack. More precisely stated, we conclude that *diversity must be introduced at all levels of the system design, including any scheme that is used to introduce diversity itself.*

This paper does not try to introduce heterogeneity at a system level, as does the work presented in [4, 5, 10, 11, 20, 22]. We are taking a more network-oriented view of the problem, which is applicable in situations where off-the-shelf technologies are required. Because our work depends upon a topological consideration of the communication environment and distributing heterogeneous applications correspondingly, we view our work as a complementary effort.

1.3 Organization

A survey of research related to software diversity and improving a given network's attack tolerance is presented in Section 2. A more formalized statement of the diversity problem is provided in Section 3. Each of our algorithms is presented and discussed in Section 4. In order to test the security of the algorithms themselves, we present a series of attacks against the algorithms in Section 5. A simulation-based analysis of these algorithms is presented in Section 6. In Sections 6.1 and 6.2, we examine the behavior of the algorithms discussed both in the absence of malicious nodes and after malicious nodes have been inserted into the network. We derive principles based upon our simulation results in Section 6.3. Finally, we state our conclusions in Section 7.

2. RELATED WORK

Evidence corroborating the inherent value of heterogeneity in a population can be found across a variety of fields, including the field of biology and organic systems. The American farmer, for example, learned of the disastrous consequences of sowing a limited number of genetic strains and its subsequent vulnerability to an infectious agent of limited capability. In the 1970's, the U.S. corn crop was destroyed when the *Bipolaris Maydis* pathogen ate through the genetically similar plantings. This single event destroyed over $1 billion of harvestable corn, or about 15% of the crop [16].

Inspiration for the examination of a network from the standpoint of an attacker's progress in conquering multiple connected computer systems is drawn from attack graph research [27]. In general, an attack graph is a graph theoretic representation of an attacker's ability to attain attack states, represented by nodes, and the techniques used to attain those states, represented by edges. Much of this research has concentrated on efficient ways of generating these graphs [2, 18]. Suggestions on how to improve the security of an attack graph relies upon having absolute knowledge of vulnerabilities on each node.

Researchers working on problems related to virus propagation, which is the automated version of the attacker problem, have suggested several interesting methods that would delay the propagation of network-based worms. The use of secure network interface cards [12] and connection rate throttling [30, 31] would reduce the number of systems that can be attacked and the rate of infection propagation, respectively. The former requires active administration and an anomaly detection engine to be a functional system, while the latter would still require active human intervention to prevent a worm from compromising every accessible machine on the Internet.

The similarities between the topological properties of human social relations and the Internet allow us to examine research originally intended for preventing human epidemics in the context of computer hackers and viruses [7, 8, 24, 25, 26]. It has been shown that in certain classes of network topologies, any infection, under standard models, would become an epidemic. Additionally, they state that an epidemic can be stopped by conducting selective immunization of nodes based on their node degree. High-degree nodes are essential for the connectivity of the network, and removing even a small fraction of them can quickly disconnect the graph [1]. While it would be possible to install different software based solely upon node degree, unequal protection

against an attack would occur. A worm that would attack the software population's low-degree nodes would have difficulty in spreading and would not compromise the network. An attack against the software assigned to the high-degree nodes would be able to rapidly propagate and disconnect the network.

The fault-tolerance community has been applying techniques developed for detecting defective systems and code to the security problem. Joseph and Avižienis [19] suggest the use of N-version programming for the prevention of computer viruses. This system-level diversity has been extended through the introduction of randomization techniques. The stack memory allocation work proposed in [11] has been extended through the use of "canary values" for detecting buffer overflow attacks in StackGuard [5]. Randomized stack offset tools have been combined with code reordering strategies in the latest versions of GCC [10]. The principle has even been applied to instruction set randomization, which can be performed with [20] or without [4] intrinsic hardware support. As stated in Section 1.2, this paper takes a network-oriented view of the problem and attempts to maximize the impact of diversity through arrangement of already diverse systems.

There have been a small number of position papers that extend the notion of security through diversity through the deployment of differing applications, operating systems, and communications protocols on a computer network [13, 28, 33]. Most notably, Zhang *et al.* [33] discussed philosophical rationales and several possible strategies for measuring and delivering a diverse computer network for the purpose of improved security. We provide algorithms and simulations that speak to the philosophy laid down in the cited work.

3. PROBLEM STATEMENT

As stated in Section 1.2, we want to provide a class of algorithms which assigns software packages to nodes on a communication network in order to limit the total number of nodes an attacker can compromise using a limited attack toolkit. The primary optimization goal would be to reduce the number of neighboring nodes running the same software package on the network. The secondary goal is to increase the number of disconnected islands formed by communication links between nodes running the same software packages.

In more formal terms, we represent a communication network using a graph $G = (V, E)$, where V is the set of all nodes on a network and E is the set of all communication links on the network. The number of nodes and edges in the network are denoted by n and m, respectively. The number of neighbors of any given node $v \in V$ is $d(v)$. The set of software packages is denoted by S, and the number of software packages is denoted by $k = |S|$. We wish to devise an assignment of software packages, $V \mapsto S$, such that the ability of the attacker to compromise the entire network is significantly reduced.

The assignment of k software packages to the graph G is what graph theoreticians would call a *coloring* of graph G. The assignment of colors in such a way that the number of *defective* edges, or communication links that exist between two nodes of the same color, is minimized is called an *optimum coloring*. A *perfect coloring* is an assignment of the minimum number of colors necessary to color a graph such that no two neighboring nodes share the same color. The minimum number of colors required for a perfect coloring is denoted by $\chi(G)$. When $k < \chi(G)$, any color assignment will induce at least one edge where both endpoints are similarly colored. A coloring where such an edge, referred to as a defective edge, is present is called a *defective coloring*.

We use the terms *colors* and *software packages* interchangeably throughout the rest of the paper.

Determining a minimum number of colors required to achieve a perfect coloring is, in the general case, an NP-Hard problem [3]. Aside from a handful of special cases, determining an optimum coloring with a minimum number of defective edges is also NP-Hard [6].

4. DISTRIBUTED ALGORITHMS

As stated previously, we have designed and analyzed a series of distributed algorithms which seek to minimize the number of defective edges present on a communication graph. The algorithms are presented in order of increasing complexity of implementation. The RANDOMIZED COLORING algorithm presented in Section 4.1 requires each node to randomly select its color and not change it throughout the duration of the network's operation. The second algorithm allows a node, at random intervals, to examine its local neighborhood and choose a new color for itself if a large number of its neighbors have the same color. We refer to this algorithm as the COLOR FLIPPING algorithm, and it is presented in Section 4.2. The next pair of algorithms, referred to as the COLOR SWAPPING algorithms, allows pairs of nodes, again at random intervals, to swap their colors in order to reduce the number of defective edges. These are presented in Section 4.3. Finally, a pair of algorithms which combine both color flipping and color swapping strategies are presented in Section 4.4.

4.1 Randomized Coloring

The first, and most basic, algorithm discussed is the RANDOMIZED COLORING algorithm. This provides, on average, m/k defective edges. Proving this is a simple exercise: after randomly coloring every node on the graph, select a single edge. The probability that both endpoints have the same color is $1/k$. Summing across all edges, the average number of defective edges is m/k. The algorithm requires $O(1)$ time to run on each node, and zero communication between the nodes is required. Because of the lack of inter-node communication, the algorithm can be considered extremely secure against attack.

The graph coloring provided by the algorithm, however, is sub-optimal. In the worst case, this algorithm performs poorly. A randomized algorithm may lead to every link forming a connection between two identical systems. While the probability of this event occurring is $(1/k)^{n-1}$, the result would have a significant impact on system security.

4.2 Color Flipping Algorithms

In the COLOR FLIPPING algorithm, nodes initialize themselves by executing the randomized coloring presented in Section 4.1. After a random delay, each node performs a local search amongst its immediate neighbors to determine if switching to a new color would decrease the number of locally defective edges. Since each node must now poll its immediate neighbors to discover their current color, the algorithm requires $O(\Delta(G))$ time to poll the neighbors per cycle, where $\Delta(G)$ is the maximum degree of the graph. Af-

ter the data is collected, $O(\Delta(G) + k)$ operations must be done to generate a census of the local colors and determine the minority color.

If it is discovered that switching to the minority color would decrease the local defect to below $d(v)/k$, then the flip is instantiated. It can be easily shown that the COLOR FLIPPING algorithm will converge. Each color flip reduces the number of defective edges by at least 1. The number of edges present in the graph is m. The maximum number of color flips that can therefore be conducted is m. Similar proofs can be found throughout the literature; Vazirani leaves the proof as an exercise to the reader in [29]. By the time the algorithm has converged, total number of defective edges is provably decreased below the average number of defects in the RANDOMIZED COLORING algorithm:

THEOREM 1. *The upper bound on the number of defective edges produced by* COLOR FLIPPING *is no more than the average number of defective edges produced by* RANDOMIZED COLORING.

PROOF. At the point of convergence, each node is connected to at most $\lfloor d(v)/k \rfloor$ defective edges. The number of defective edge endpoints is $\sum_v \lfloor d(v)/k \rfloor$. The number of defective edges is therefore $1/2 \sum_v \lfloor d(v)/k \rfloor$. In comparison to the randomized algorithm:

$$\frac{1}{2} \sum_v \left\lfloor \frac{d(v)}{k} \right\rfloor \leq \frac{1}{2} \sum_v \frac{d(v)}{k} = \frac{m}{k}$$

□

4.3 Color Swapping Algorithms

The following pair of algorithms are extensions of the Kernighan-Lin heuristic [3] for computing balanced cuts. In both algorithms, each node attempts to reduce its number of defective edges by negotiating for a color "swap" between itself and its neighbors. After collecting the number of defective edges which would be removed from the neighbor node and itself by conducting a swap from each neighbor, the initiating node executing the algorithm chooses a neighbor which it views to be optimal and proposes a color swap. If the neighbor agrees to the swap, the initiating node takes the color of the neighbor and the neighbor takes the color of the initiating node.

For a swap to take place in the first algorithm, known as MUTUALLY BENEFICIAL SWAPPING, the exchange of colors must reduce the defective edge count for both nodes involved. The second algorithm, referred to as GREATER GOOD SWAPPING, will incur a swap if the total number of defective edges between both nodes is reduced by the exchange. The greater number of nodes that are available for a GREATER GOOD SWAPPING execution means the quality of the solution associated with the GREATER GOOD SWAPPING algorithm is expected to be better than that associated with the MUTUALLY BENEFICIAL SWAPPING algorithm. Correspondingly, the increased number of swap partners increases the vulnerability of the algorithm to attack. This phenomenon is discussed further in Section 5.

4.4 Hybrid Algorithms

The final set of algorithms are hybrids of the color swapping and color flipping schemes presented in Sections 4.2 and 4.3, respectively. The RANDOMIZED HYBRID algorithm requires that a node which wishes to change its color to randomly choose to execute either the GREATER GOOD SWAPPING algorithm or the COLOR FLIPPING algorithm. The selection between the GREATER GOOD SWAPPING algorithm and the COLOR FLIPPING algorithm does not need to be unbiased; on the contrary, it may be beneficial from a convergence rate or attack tolerance standpoint for the algorithm to prefer one coloring scheme over the other. Determining the optimal point between conducting a flip or a swap can be done through the use of game theoretic analysis.

The BEST CHOICE HYBRID algorithm allows pairs of nodes to examine the defective edge reduction that is possible by either doing a color swap as a pair or independently doing a color flip. If each node in a swap can eliminate a greater number of defective edges by cooperating and performing a swap as compared to individually performing a flip, a swap is conducted. If either of the two nodes finds it can better serve itself by conducting an independent color flip, then a swap is not conducted. If the node that initiates the recoloring attempt finds that a swap is not feasible, it attempts to conduct an independent color flip.

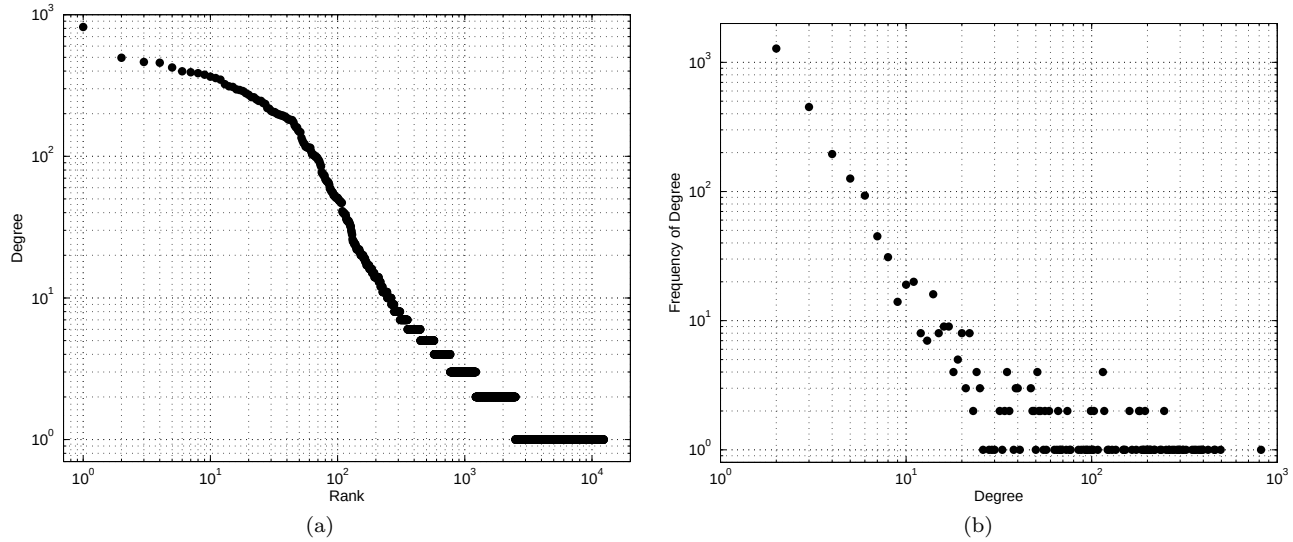
5. ATTACK DESIGN

Given that the algorithms discussed are being used to decrease the ability of an attacker from compromising the network, it is likely that an attacker would be interested in affecting the performance of the coloring algorithm itself. Therefore, we propose a set of primitive behaviors exhibited by a malicious node from which any attack can be created.

Spreading: Upon inspection, instead of looking to flip its color, a node that is malicious will look to subvert a neighboring node that is of its own color.
Misrepresentation: A node may falsely report its current color when it is queried for its color by neighboring nodes. Additionally, a node may falsely report its defective edge reduction to neighboring node wishing to conduct a color swap.
Inertia: A node will not change its color regardless of external stimulus.

The first algorithm analyzed is robust against attacks directed toward the algorithm itself. The RANDOMIZED COLORING algorithm requires nodes to set their color without examining their environment. In turn, any network implementing the algorithm is not affected by the last two attacks, and can only be affected by the spreading attack.

The COLOR FLIPPING algorithm introduces an inherent security flaw. Any node looking to flip its color must trust that their neighbors will be truthful in reporting their own color assignment. If a malicious node decides to lie about its own color, it can influence a querying node's color choice, but not force a color assignment upon the querying node. For example, a malicious node can falsely report to a node that its color is the same as a querying node, which would contribute to the querying node's defect count. If the malicious node is fortunate, the defective edge count observed by the querying node would become greater than $\lfloor d(v)/k \rfloor$. This will cause the querying node to flip to a new color. The goal of the malicious node is to push the querying node to flip to a specific vulnerable color. If a flip takes place, the malicious node has no way of being certain the querying node will flip to a vulnerable color.



(a) (b)

Figure 2: Log-Log Plots of E-Mail Graph Statistics. The properties of the collected data are statistically similar to many other topologies, including the AS topology seen in BGP routing.

Both the Mutually Beneficial Swapping and Greater Good Swapping algorithms introduce a security flaw due to the inherent trust associated with a color swap. If a malicious node either proposes or agrees to a swap with a participating neighbor, it can keep its own color even after the neighbor has completed switching to the new color. The action would create a defective edge that the malicious node can use to propagate an attack. In the case of the mutually beneficial swap algorithm, a swap would never be acceptable to a node unless the defective edge count of the node decreases. Even if a malicious node wants to "push" a vulnerable color onto a node, it would only be able to do this to the subset of its neighbors which would stand to gain from an honest swap. The Greater Good Swapping algorithm, however, has a larger security vulnerability associated with it. A malicious node can force a color change onto a neighboring node by claiming an extremely high defect improvement. To the neighbor, it would appear that the proposed swap is globally beneficial, regardless of its own increase in the number of defective edges. Therefore, a single compromised node can spread a chosen color across an entire network, one node at a time.

There does not exist a single optimal attack that works against both algorithms, however. If the network implements a swapping algorithm, lying about a malicious node's own color would lead a querying node to swap to a random, non-vulnerable color. Rather than increasing the number of nodes that can be attacked in the network, running the optimal swapping algorithm attack on a network running the color flipping algorithm would actually *decrease* the number of vulnerable nodes. Vulnerable nodes, which were previously unable to swap their color to one which would induce less defective edges because of a lack of potential swapping partners would find nodes with a previously unseen color in their neighborhood. Therefore, not only would the number of vulnerable nodes decrease, the number of defective edges present across the network would decrease as well. Likewise, a network running the color flipping algorithm would not be impacted by the contract-breaking attack mentioned above. No inter-node contracts are involved in the algorithm, and correspondingly, there is no opportunity to break a color-changing agreement.

Based upon this analysis, the behavior of the hybrid algorithms discussed in Section 4.4 under attack can be expected to be a synthesis of the reactions of both the color swapping and color flipping algorithms to the stated attacks.

6. SIMULATION

In order to test our algorithms, it was necessary to acquire a topology that is representative of the networks that our distributed coloring algorithm would expect to encounter. As many researchers consider generation of a simulated, representative network topology to be an open research problem [23, 32], we have decided to capture an *actual* topology for our algorithm simulation.

For our simulation experiments, we examine a topology generated by e-mail traffic inside the ECE Department at Drexel University. We captured a sample of the logs created by e-mails as they passed through the `ece.drexel.edu` server. The raw data consisted of 1,038,939 log entries for each e-mail sent and received by 278,435 unique accounts handled by `ece.drexel.edu`'s `sendmail` server from January 13th to September 19th of 2003. Of the original 1,038,939 e-mails recorded, there are 337,532 unique {to, from} e-mail address pairs. This means, strictly according to the logs, there are 337,532 unique pairs of individuals using the mail server to communicate.

To reduce the impact of spam on our data set, we preserve those edges where, for each sender and receiver, at least one e-mail is sent from the initial message receiver to the initial message sender. This represents a complete communication between the two e-mail entities. Our data set is then reduced to 37,618 {to, from} address pairs, or 18,809 undirected edges. These edges exist between 12,408 nodes, or unique e-mail ID's, in 14 separate connected components, where the largest connected component consists of 12,354

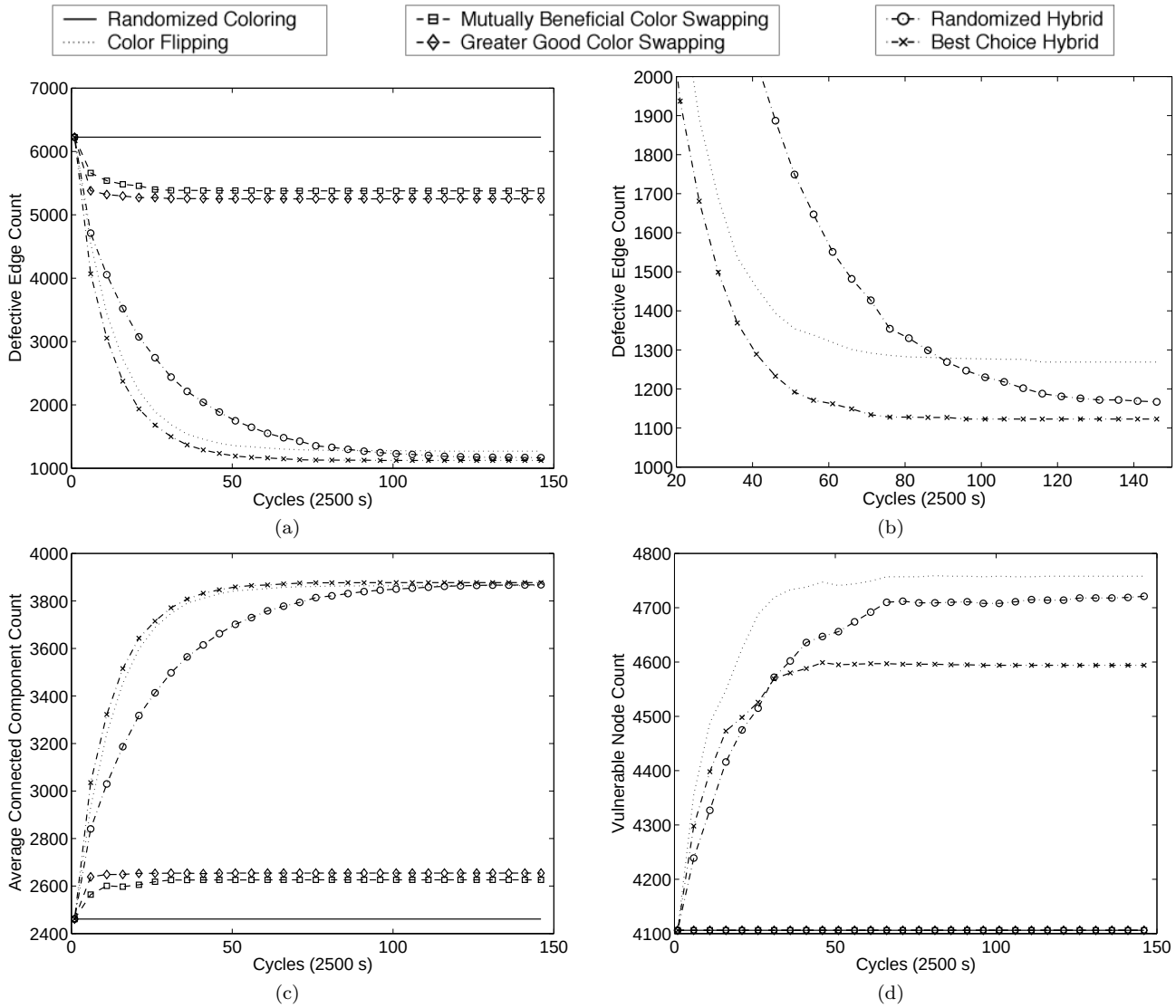


Figure 3: Comparison of coloring algorithms with no malicious nodes present

nodes and 18,768 undirected edges. Our simulation studies use this largest connected component.

It is customary in the study of large-scale network topologies to examine the distribution of node degrees on a log-log plot. Accordingly, we have plotted the degree of each node versus its rank in a sorted list along with the frequency of degree versus the degree of the node. These plots, whose distribution is consistent with the work of [8, 24], are shown in Figures 2(a) and 2(b), respectively.

6.1 Algorithm Simulation

The coloring algorithms presented in Sections 4.1, 4.2, 4.3, and 4.4 are provided with three distinct colors, and are each executed by the 12,354 nodes at intervals determined by a Poisson process running at each node. The Poisson rate λ is set to $1/n$ algorithm executions per cycle for each node in order to normalize the execution rate of the algorithm by each node with respect to graphs that differ in node count, allowing for an unbiased comparison of the algorithm's performance across varying networks. By the end of every 100,000 cycles, each node would have executed its coloring algorithm an average of 8.09 times.

In accordance with the design goals laid out in Section 1.2, we monitor the number of defective edges present in the graph, the average number of connected components induced by each color, and the number of nodes which have been defined as being "vulnerable". The first metric is our primary optimization goal and corresponds to the number of edges that exist in the graph that can be traversed by a node-hopping attack. The second metric indicates the minimum number of separate infections that must take place for all vulnerable nodes to be compromised given an attack that is unable to change the color assignment. Since a separate curve exists for each color, we average the number of connected components across all colors for each algorithm analyzed. The final metric provides a baseline of the number of vulnerable nodes in the network. In the absence of an external agent, namely an attack that is aware of the

coloring algorithm, this value should be affected only by the coloring algorithm itself.

Figure 3(a) shows the improvement in the number of defective edges as the three classes of dynamic algorithms converge to their local optimums. The difference in the quality of the solutions provided at convergence is shown in Figure 3(b). In Figure 3(c), a comparison of the number of average connected components for each color is presented. Figure 3(d) shows the evolution of the population of nodes of a single color; these nodes are later tagged as being vulnerable to attack and, if attacked, become malicious. The upward bias in the number of nodes of the specific color being examined is relatively small in comparison to the number of nodes on the graph and is an artifact of the simulation run. Not surprisingly, the number of nodes in the one color being examined is approximately the same for all three classes of algorithms.

In Figures 3(a) through (c), both the Mutually Beneficial Swapping and the Greater Good Swapping algorithms provide an improvement as compared to the Randomized Coloring algorithm. The two swapping algorithms provide a solution which is inferior to the Color Flipping algorithm. The marked difference in the quality of the coloring solutions observed between the swap-based algorithms and the flip-based algorithm can be attributed to the availability of colors to any given node. In the swap algorithms, a node can only change its color to one that is present amongst its neighbors, and then only if the outcome of the swap is mutually beneficial to the nodes or globally beneficial to the graph. The flip algorithm places no restrictions upon a node's potential color choices if the node is exposed to a large number of monochromatic edges. As a result, the Color Flipping algorithm allows for a greater fraction of nodes to change their color assignment when the distributed algorithm is executed.

It is clear from Figure 3(b) that the Randomized Hybrid and Best Choice Hybrid algorithms produce a better coloring than either the swap-based or the flip-based algorithms alone. The hybrid algorithms generate a better solution by simultaneously drawing on the swap algorithm to eliminate deadlocks that may occur in a neighborhood and the flip algorithm to provide a wider range of colors that a node can assign itself.

6.2 Attack Simulation

A second series of experiments is conducted to test each algorithm's tolerance to attack. One color is selected and labeled as *vulnerable*, meaning an attacker can compromise that color and only that color. It then becomes the goal of the attacker to switch every node in the network to the vulnerable color. After the coloring algorithms have converged, 1% of the vulnerable nodes are infected with a worm, which is able to carry out any combination of the attacks described in Section 5.

Figures 4(a), 4(b), and 4(c) show the effect of malicious nodes on the number of defective edges present, the average number of connected components for each color, and the number of vulnerable nodes, respectively. These malicious nodes are introduced to the network after the distributed algorithm has largely converged. They begin to attack the network by lying about their color and breaking swapping contracts, but respond honestly when asked about their own improvement with respect to the number of similarly colored neighbors when queried about a proposed color swap.

Figures 4(d), 4(e), and 4(f) show the effect on the metrics studied in Figures 3 and 4(a)–(c) when nodes that lie about the quality of a proposed swap and break swapping contracts are introduced into the network some time after convergence. It should be noted that the Color Flipping algorithm is not vulnerable to this attack, since it does not propose swaps with neighboring nodes.

Figures 4(g), 4(h), and 4(i) show the effect of completely dishonest nodes upon the network. This "brute force" attack is not designed to attack any one particular algorithm, nor are the malicious nodes cognizant of the coloring algorithm that is being executed by their neighbors. Instead, it is designed to examine the effects of completely uncooperative nodes upon the network.

As stated in Section 5, color liars increase the number of defective edges in a network when the network is executing the Color Flipping algorithm, but decrease the number of defective edges present in a network executing the Color Swapping algorithms. The introduction of color liars in Figure 4(a)–(c) experimentally confirms this analysis. The behavior of the hybrid algorithms indicates a bias in both algorithms towards the use of the Color Flipping strategy as opposed to the Color Swapping strategy, as evidenced by the similarity between the number of defective edges experienced by the Color Flipping, Randomized Hybrid, and Best Choice Hybrid algorithms in 4(a). Furthermore, the experiment has shown that even after convergence is achieved, it is possible to disrupt the color assignment of the graph.

The behavior of a network that is being attacked via defect liars is dramatically different, as shown in Figure 4(d)–(f). While the network implementing Mutually Beneficial Swapping algorithm appears to not be affected by the malicious behavior, the network utilizing the Greater Good Swapping is completely compromised. The two algorithms, while exceedingly similar, exhibit markedly different tolerance to attack. The rationale for this phenomenon resides in the relative "voting power" of swapping partners. In the Mutually Beneficial Swapping algorithm, both neighbors have equal input for the swap decision. Regardless of the input of one's neighbor, a swap will not take place unless the action can benefit both nodes. After the distributed coloring converges, no node operating under this algorithm can further improve the quality of its coloring by conducting a swap. Nodes in networks implementing the Greater Good Swapping algorithm, however can *always* conduct a swap that the node *believes* would increase the quality of the network's coloring. Under this algorithm, a swap partner can have an unbounded contribution to the swap decision. A malicious node can use this to force a coloring upon any neighboring node with whom a swap is being negotiated. Since the hybrid algorithms depend upon the swapping algorithm, they are both vulnerable to this form of attack.

In the plots contained in Figure 4, the performance of the Randomized Hybrid and the Best Choice Hybrid algorithms under attack appear to be rather similar. As stated in Section 4.4, the Randomized Hybrid algorithm contains a tunable parameter, however, which forces the algorithm to utilize the Color Flipping algorithm at a higher or lower frequency compared to the Greater Good Swapping algorithm. Deriving the optimal balance between the

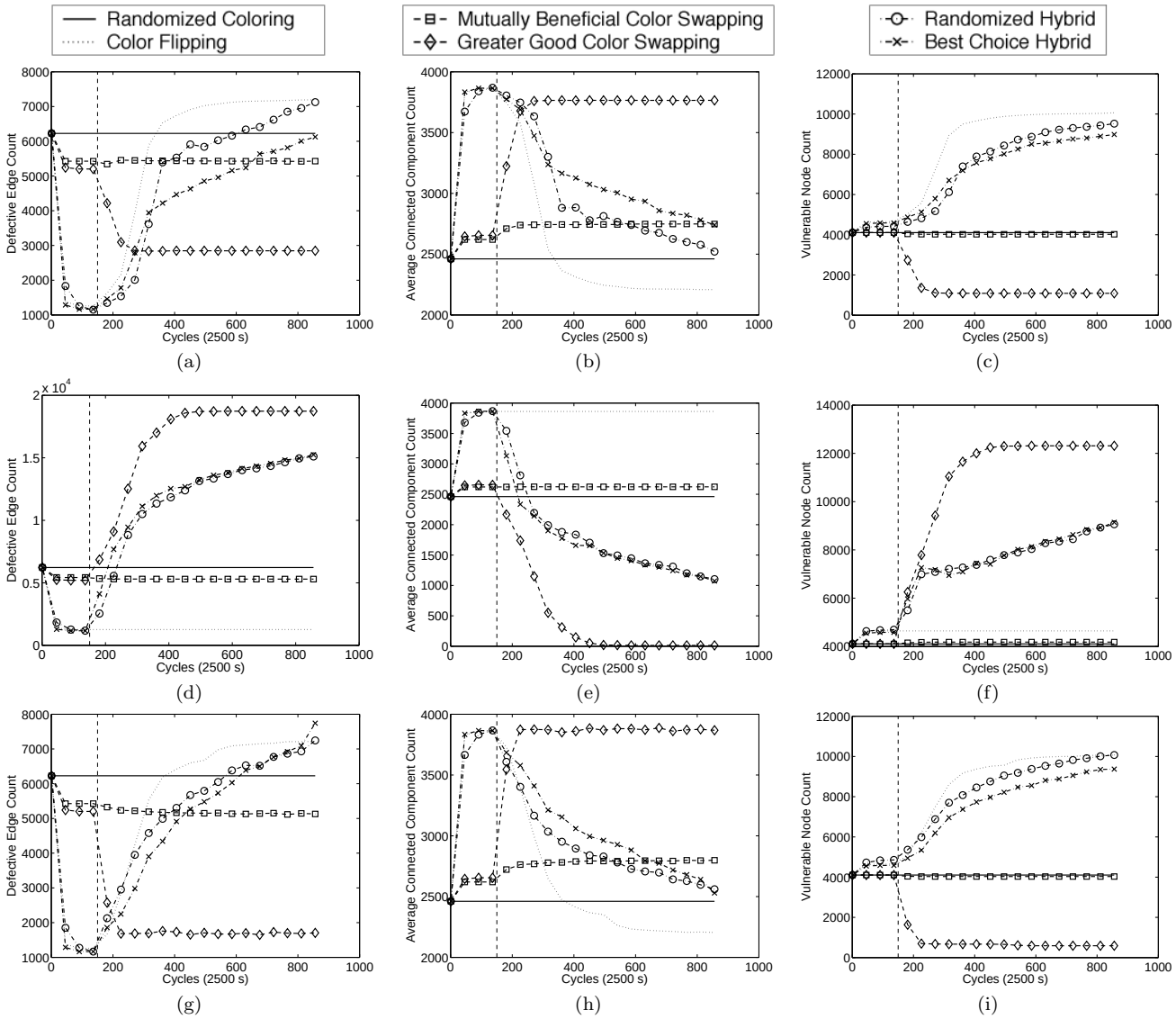


Figure 4: Comparison of the performance of coloring algorithms under attack. Plots (a) through (c) examines the impact of nodes that lie about their color on the algorithms, (d) through (f) examines the impact of nodes that lie about defect improvements. Plots (g) through (i) examines the impact of nodes that lie about both defect improvements and their color. The vertical line indicates the time when malicious nodes are added to the network.

two algorithms for the purpose of minimizing the effects of an attack against the algorithm can be accomplished using game theory, but the equilibrium point would be unique to the topology of the graph. In the formulation, payoffs experienced by either the network operator or the attacker would be derived from the rate at which non-vulnerable nodes can be convinced to change to a vulnerable color because of input from malicious neighbors. The usage rate of either the Color Flipping or the Greater Good Swapping algorithms would be selected to balance out the risk of executing either of the two algorithms over the long term.

6.3 Analysis of Simulation Results

Two important conclusions can be drawn from the analysis of the coloring algorithms and their tolerance to a tailored attack. The Mutually Beneficial Swapping algorithm converges to the largest number of defective edges of any algorithm which allows for re-coloring of individual nodes. After convergence, though, attacking this algorithm has shown to be extremely difficult. A slight modification to the Mutually Beneficial Swapping algorithm was presented in the Greater Good Swapping algorithm, which relaxes the guidelines for an acceptable swap. While this allows for more color swaps to take place and in turn reduces the number of defective edges in the graph, the algorithm becomes far more vulnerable to a directed attack. Algorithms which allow a node to undergo a local and independent color flip, while extremely effective at reducing the total number of defective edges, have been shown to be heavily impacted by malicious nodes which lie about their

color. Given enough time for convergence and a small but finite set of moderately connected nodes, the malicious nodes would likely be able to compromise the entire network. The only algorithm which is not vulnerable to a directed attack is the randomized algorithm, which, not coincidentally, provides the worst defective coloring performance. Based upon these results, we believe that *there is a fundamental tradeoff between the quality of the diversity achieved by an algorithm and the algorithm's tolerance to attacks.*

Both of the hybrid algorithms allow for a node to choose between the two coloring algorithms at each instant of operation. The ability to switch between the two algorithms removes the attacker's ability to know which coloring algorithm a targeted node is intending to execute. In the absence of precise knowledge of the currently running coloring algorithm, an attacker would have some difficulty crafting an optimal attack. As discussed in Section 5, the most effective attack against the COLOR FLIPPING algorithm would be the introduction of color liars to the network, and the most effective attack against the swapping components of both the MUTUALLY BENEFICIAL SWAPPING and the GREATER GOOD SWAPPING is the introduction of contract breakers to the network. For a contract-breaking node to work correctly, however, it must be completely honest about its color. Otherwise, the swap partner would swap to a different color from that of the malicious node, which would halt the proceeding attack. Through similar reasoning it is easy to see why introducing a set of honest contract breakers would be counterproductive for attacking all coloring algorithms.

The above rationale is no different from the motivation for security through diversity itself. The COLOR FLIPPING, MUTUALLY BENEFICIAL SWAPPING and GREATER GOOD SWAPPING algorithms are vulnerable to attack simply because the same algorithm is running on every node, and every node is vulnerable to the same form of attack. Introducing diversity at the diversity assignment layer would mean an attacker would not be able to use a single attack strategy to take over the network. The RANDOMIZED HYBRID and the BEST CHOICE HYBRID algorithms are vulnerable to all forms of misrepresentation and contract-breaking attacks, but the existence of a mixed coloring strategy increases the algorithm's tolerance to attack. Experimental evidence has shown that both hybrid algorithms fare better when presented with both forms of attack, than the COLOR FLIPPING and GREATER GOOD SWAPPING algorithms when each are presented with their appropriate attack strategies. The increased tolerance to attack is due to the lack of knowledge on the part of the malicious nodes; since the malicious nodes are unaware of which algorithm is being executed by the targeted nodes, choosing an effective attack becomes a game of chance. It is based upon this observation that we state that *the most effective way of achieving attack tolerance in our algorithms is to reapply the fundamental thesis of the paper, and implement diversity strategies into the algorithms themselves.*

7. CONCLUSION AND FUTURE WORK

Research in improving network security through the disruption of software monocultures has garnered considerable attention in recent years. The literature details a variety of solutions, including introducing heterogeneous software to systems through randomization, N-version programming, and various other techniques. However, for both business and technical reasons, the limited number of functionally equivalent yet distinct software packages makes heterogeneity a less effective strategy than one may like.

In this paper, we have provided a series of algorithms for increasing the effectiveness of system-level heterogeneity on a network. Even though the computation of an optimally diverse software allocation is believed to be intractable, the distributed algorithms presented here reduce the number of links that can be utilized for propagating an attack. Furthermore, our algorithms effectively cluster the network, which helps to isolate infected systems from the rest of the topology.

Any methodology for increasing the attack tolerance of a network is destined to come under attack itself. We have shown that there exists a trade-off between the ability of an algorithm to reduce the number of defective edges present in the network and the ability of the algorithm to tolerate a directed attack. The algorithm which exhibits the best worst-case performance against attack was a hybrid of our basic algorithms, which itself highlights the principle of security through diversity.

Based upon our observations, simulations, and analysis we are left with a confirmation of our thesis; not only is diversity critical for improving the attack tolerance of a network, but the inherent value of diversity can be increased through an algorithmic distribution of diverse systems. Furthermore, these principles must be applied to all levels of system design, including any scheme which introduces diversity itself.

In the future, we would like to extend the study of diversity through graph theoretic techniques to incorporate new metrics and methods. Preliminary research has shown that the epidemic threshold would be an appropriate metric for measuring the quality of a diverse software assignment. Furthermore, we would like to explore the use of game theory in the context of optimizing our hybrid algorithm's attack tolerance.

8. ACKNOWLEDGMENTS

First and foremost, we would like to thank Jonathan Hoult, the ECE Department's UNIX administrator, for providing the raw `sendmail` logs from which we extracted the e-mail network topology.

We would also like to thank several individuals for the numerous discussions shared over the course of the research, including Dr. Jose Nazario of Arbor Networks, Dr. Vassilis Prevelakis of Drexel's CS Department, and Jeff Abrahamson and Trip Denton of Drexel's Applied Algorithms Laboratory.

Finally, we extend their gratitude to the anonymous reviewers for their labor. Their comments have contributed greatly to making this a better piece of work.

9. REFERENCES

[1] R. Albert, H. Jeong, and A. L. Barabási. Error and Attack Tolerance of Complex Networks. *Nature*, 406:378–382, July 2000.

[2] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 217–224. ACM Press, 2002.

[3] G. Ausiello, M. Protasi, A. Marchetti-Spaccamela, G. Gambosi, P. Crescenzi, and V. Kann. *Complexity and Approximation: Combinatorial Optimization*

Problems and Their Approximability Properties. Springer-Verlag New York, Inc., 1999.

[4] E. G. Barrantes, D. H. Ackley, T. S. Palmer, D. Stefanović, and D. D. Zovi. Randomized instruction set emulation to disrupt binary code injection attacks. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 281–289. ACM Press, 2003.

[5] C. Cowan, C. Pu, D. Maier, H. Hinton, P. Bakke, S. Beattie, A. Grier, P. Wagle, and Q. Zhang. Automatic detection and prevention of buffer-overflow attacks. In *Proceedings of the 7th USENIX Security Symposium*, January 1998.

[6] L. J. Cowen, W. Goddard, and C. E. Jesurum. Coloring with defect. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 548–557. Society for Industrial and Applied Mathematics, 1997.

[7] Z. Dezső and A.-L. Barabási. Halting viruses in scale-free networks. *Physical Review E*, 65(055103), 2002.

[8] H. Ebel, L.-I. Mielsch, and S. Bornholdt. Scale-free topology of e-mail networks. *Physical Review E*, 66(035103(R)), 2002.

[9] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, Utah, May 2001.

[10] H. Etoh. GCC extension for protecting applications from stack-smashing attacks, 2004. http://www.trl.ibm.com/projects/security/ssp/.

[11] S. Forrest, A. Somayaji, and D. Ackley. Building diverse computer systems. In *Proceedings of the 6th Workshop on Hot Topics in Operating Systems (HotOS-VI)*, pages 67–72. IEEE Computer Society, 1997.

[12] G. R. Ganger, G. Economou, and S. M. Bielski. Self-securing network interfaces: what, why and how. Technical Report CMU-CS-02-144, Carnegie Mellon University, May 2002.

[13] D. Geer, R. Bace, P. Gutmann, P. Metzger, C. P. Pfleeger, J. S. Quarterman, and B. Schneier. Cyberinsecurity: The cost of monopoly. Technical report, CCIA, 2003. http://www.ccianet.org/papers/cyberinsecurity.pdf.

[14] A. Gudmundsson and E. Chien. Security response: W32.klez.a@mm. Technical report, Symantec, 2001. http://securityresponse.symantec.com/avcenter/venc/data/w32.klez.a@mm.html.

[15] J. S. Havrilla and S. V. Hernan. Advisory CA-2001-06: Automatic execution of embedded mime types. Technical report, CERT, 2001. http://www.cert.org/advisories/CA-2001-06.html.

[16] T. G. Horsfall. *Genetic vulnerability of major crops*. National Academy of Sciences, 1972.

[17] A. Householder and R. Danyliw. Advisory CA-2003-08: Increased activity targeting windows shares. Technical report, CERT, 2003. http://www.cert.org/advisories/CA-2003-08.html.

[18] S. Jha, O. Sheyner, and J. Wing. Two formal analyses of attack graphs. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, page 49. IEEE Computer Society, 2002.

[19] M. K. Joseph and A. Avižienis. A fault tolerance approach to computer viruses. In *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pages 52–58. IEEE Computer Society Press, April 1988.

[20] G. S. Kc, A. D. Keromytis, and V. Prevelakis. Countering code-injection attacks with instruction-set randomization. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 272–280. ACM Press, 2003.

[21] J. P. Lanza and S. V. Hernan. Advisory CA-2003-07: Remote buffer overflow in sendmail. Technical report, CERT, 2003. http://www.cert.org/advisories/CA-2003-07.html.

[22] R. C. Linger. Systematic generation of stochastic diversity as an intrusion barrier in survivable systems software. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*, volume 3, page 3062, 1999.

[23] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: An approach to universal topology generation. In *Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01)*, page 346. IEEE Computer Society, 2001.

[24] M. E. J. Newman, S. Forrest, and J. Balthrop. Email networks and the spread of computer viruses. *Physical Review E*, 66(035101), 2002.

[25] R. Pastor-Satorras and A. Vespignani. Epidemics and immunization in scale-free networks. In S. Bornholdt and H. G. Schuster, editors, *Handbook of Graphs and Networks: From the Genome to the Internet*, pages 113–132. Wiley-VCH, May 2002.

[26] R. Pastor-Satorras and A. Vespignani. Immunization of complex networks. *Physical Review E*, 65(036104), 2002.

[27] C. Phillips and L. Painton Swiler. A graph-based system for network-vulnerability analysis. In *Proceedings of the 1998 workshop on New security paradigms*, pages 71–79. ACM Press, 1998.

[28] M. Stamp. Risks of monoculture. *Commun. ACM*, 47(3):120, 2004.

[29] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlang New York, Inc., 2001.

[30] Y. Wang and C. Wang. Modeling the effects of timing parameters on virus propagation. In *Proceedings of the 2003 ACM workshop on Rapid Malcode*, pages 61–66. ACM Press, 2003.

[31] M. M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. In *Proceedings of the 18th Annual Computer Security Applications Conference*, page 61. IEEE Computer Society, 2002.

[32] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *IEEE Infocom*, volume 2, pages 594–602, San Francisco, CA, March 1996. IEEE.

[33] Y. Zhang, H. Vin, L. Alvisi, W. Lee, and S. K. Dao. Heterogeneous networking: a new survivability paradigm. In *Proceedings of the 2001 workshop on New security paradigms*, pages 33–39. ACM Press, 2001.