ISSN-2230-9659

Validating and Attacking Distributed Software Diversity

Amandeep Kaur
Research Scholar, CMJ University,
Shillong, Meghalaya

ABSTRACT:-

The field of viral propagation modeling has garnered a great deal of attention in recent years as computer security researchers attempt to find ways of mitigating rapid malcode propagation. A variety of techniques have been suggested which can delay the spread of a worm, including rate-limiting network cards , targeted immunization of highly connected nodes and a combination of address blacklisting and content filtering. In complementary work, researchers have been focusing on the software monoculture on the Internet and its relationship to viral epidemics. The value of software diversity to computer security comes from the fact that an attack written for one piece of software rarely works for a different but functionally equivalent software package. By increasing the number of diverse software packages present on the network, the research argues, the chances that an attack will be effective against a randomly selected node will decrease.

The research literature in software diversity suggests that the introduction of different software packages is an effective method of disrupting the activities of an attacker or a worm, particularly one which repeatedly utilizes a pre-written and unchanging attack to compromise machines. However, there have been no quantitative evaluations of the impact of software

ISSN-2230-9659

diversity on malcode propagation in real network topologies. These technologies serve as an effective method for preventing worm epidemics.

INTRODUCTION :

While the number of monochromatic edges is an effective metric as an optimization goal, it does not directly express the ability of the diversity assignment algorithm to limit the virulence of a worm. This section, on the other hand, quantifies the quality of a software diversity assignment by focusing on the effect that network assignments of diverse software has upon the propagation of worms. Given a worm whose rate of propagation from an infected node to each of its vulnerable neighbors is b and the rate at which infected nodes are disinfected is a, we study the epidemic threshold, or the ratio of a/b below which an infectious agent will burn itself out i.e., the ratio below which there will be no infected nodes in the network at steady state.

One of the goals of any virus mitigation technique should be to increase the epidemic threshold of the network. In this chapter, our goal is to study:

1. The epidemic threshold with a randomized distribution of diverse software packages to nodes in a real network (an IPv6 BGP topology) as well as a synthetically generated an Erd¨os-R´enyi random graph network topology.

2. The relationship of the above results to the number of different software packages available to distribute among the nodes.

3. The epidemic threshold on the same networks with a topology-sensitive algorithm driven distribution of diverse software packages.

As before, we represent a network of computers by graph G and a set of diverse software packages which can be assigned to nodes on the network by C. We consider a contagion which can infect only a single software package in C. Assume that the number of software packages available in C is greater than or equal to the chromatic number of the graph (G). If the software packages are randomly distributed to the network, then a portion but not all of the nodes will be rendered immune to the infection. However, if a graph coloring algorithm is used to assign the software in C to the nodes in G, then no edges will be left to spread the infection, and the infection is guaranteed to die out.

The remainder of this section is organized as follows. We validate these models using simulations of virus propagation on both synthetic and real network topologies. The simulations show that the improvement in the epidemic threshold experienced under an algorithm-driven diversity assignment algorithm is significantly higher than that predicted by the bounds generated by our models for real-world graphs.

VIRAL PROPAGATION AND SOFTWARE DIVERSITY

It is possible to show that, regardless of the underlying viral propagation model, an assignment of software packages to a graph such that the assignment forms a perfect graph coloring will force the epidemic threshold to infinity. Consider a perfect coloring, where there are no edges across which a virus can propagate. The only infected hosts that exist are those which are initially infected by a virus. Because this set cannot increase, the disinfection rate of systems will continually decrease the number of infected systems until all systems are uninfected.

It may not be possible to guarantee that a sufficient number of software systems will be available to perfectly color the network. It would then be more appropriate to assign the limited amount of diversity so as to limit the number of monochromatic edges and thus increase the epidemic threshold. To achieve this goal, we use the COLOR FLIPPING algorithm. The distributed algorithm has each node choose an initial software package, or color, and, at random intervals, communicate with their immediate neighborhood of nodes to discover their current color. The node initiating the communication will then switch to the neighborhood's minority software package if it finds a majority of its neighbors are running the same software package.

It is important to note that it is not necessary to examine every variation of the coloring algorithms for their effect on the viral propagation metrics. We are interested in showing the trends that a decreasing defective edge count has upon the studied viral propagation characteristics, which will be provided by any of the distributed algorithms.

ISSN-2230-9659

The introduction of a graph coloring algorithm removes some of the assumptions of randomness that underpin the statistical models which results in loose bounds on the epidemic threshold on networks colored using the COLOR FLIPPING algorithm. Rather than providing only loose bounds, we examine the effect of algorithm-driven color assignments on the epidemic threshold primarily through the use of simulation.

STATISTICAL MODELS

We can consider nodes which run software packages which are different from their neighbor to be relatively immune to attack from their neighbor. Assuming a randomized distribution of diverse software packages, if there are c software packages available for n nodes, it is expected that $n - n/c$ nodes will be relatively immune to the n/c vulnerable nodes.

A similar analysis can be done for the Pastor-Satorras and Vespignani model, which shows an increase in the epidemic threshold by a similar factor. In order to test the utility of diversity assignments for increasing the epidemic threshold, it is necessary to either generate or measure a network topology for simulation study.

Our first network was generated by collecting a list of the BGP peers present in the IPv6 network by accessing the routing table from IPv6 capable Looking Glass routers. A second network was created using an Erdos-Renyi random graph generator. Both graphs contain 266 nodes and approximately 7, 500 edges. While both graphs have similar average degree, the degree distribution for both graphs is dramatically different.

While we utilized larger networks to study the coloring algorithms , the computational load associated with executing the viral propagation simulations makes this option infeasible. The rest of the simulation studies presented in the section follow a standard methodology; a single color is tagged as being vulnerable to infection, and the graph is assigned an initial coloring. A high percentage of the nodes assigned the vulnerable color are randomly chosen to be the nodes which initially contain the infection. We experimentally determine the epidemic threshold by progressively changing relative to a fixed until a persistent infection is not seen over numerous simulation runs with both the same initial infection set and with alternate initial infection sets.

GRAPH THEORY DERIVED MODELS

ISSN-2230-9659

In a fashion consistent with Wang's model, we are able to restate the goal of the software assignment in terms of graph partitions and the subsequent eigen values of the sub graphs. We denote our software assignment as

f : V (G) 7→ C,C = {1, 2, ..., c},

where C is the set of available software packages.

Loose bounds for general graphs and hard bounds on regular graphs can be determined for the largest eigen value of the adjacency matrix of a diversified network. Rather than relying upon the loose bounds, we directly measure the eigen value of a network which is actively undergoing diversification to predict the epidemic threshold.

To examine the impact the number of monochromatic edges has upon the epidemic threshold, we simulate a homogeneous network of systems, then allow each system to minimize its number of monochromatic neighbors by executing the COLOR FLIPPING algorithm. At each time-step, we compute the epidemic threshold predicted by the Pastor-Satorras and Vespignani model and Wang's eigen value model. The Kephart and White model is inappropriate for use with networks using an algorithm-driven diversity assignment as the application of the algorithm to the network removes the homogeneous degree distribution on the network.

It is clear from the simulation studies that reducing the number of monochromatic edges in the network is an extremely effective method of increasing the epidemic threshold. The simulation studies confirm the utility of recomputing the eigen value-derived epidemic threshold with each step of the graph coloring operation is an effective method of approximating the epidemic threshold. Furthermore, the experiment shows that decreases in the number of defective edges go hand in hand with increases in the epidemic threshold.

While a wide variety of techniques for mitigating rapid malware propagation have been analyzed and simulated using standard virus modeling techniques, the contributions of the software diversity community have not yet been fit into this framework. In this section, we make the first contributions toward analyzing viral propagation modeling in the presence of software diversity. We use both models and simulations to show that on both simulated and real networks of systems, a natıve, randomized software diversity assignment is able to increase the epidemic threshold.

ATTACKING NETWORK DIVERSITY ASSIGNMENTS

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.

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.

REFERENCES

- R. Albert, H. Jeong, and A. L. Barab´asi. Error and Attack Tolerance of Complex Networks. Nature, 406:378–382, July 2000.
- S. Alexander. Defeating compiler-level buffer overflow protection. ;login:, 30(3):59—71, June 2005.
- 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.
- W. A. Arbaugh, W. L. Fithen, and J. McHugh. Windows of vulnerability: A case study analysis. IEEE Computer, 33:52–59, December 2000.
- Giorgio 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.
- Aviˇzienis. Fault-tolerance and fault-intolerance: Complementary approaches to reliable computing. In Proceedings of the international conference on Reliable software, pages 458–464, Los Angeles, California, 1975. ACM Press.
- P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 164–177, New York, NY, USA, 2003. ACM Press.
- E. G. Barrantes, D. H. Ackley, T. S. Palmer, D. Stefanovi´c, 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.
- S. M. Bellovin. Distributed firewalls. ;login:, pages 39–47, November 1999.
- S. Bhatkar, D. C. DuVarney, and R. Sekar. Address obfuscation: An efficient approach to combat a broad range ofmemory error exploits. In Proceedings of the 12th USENIX Security Symposium, pages 105–120, Washington D.C., USA, August 2003.
- Bulba and Kil3r. Bypassing StackGuard and StackShield. Phrack Magazine, 0xA (0x38), May 2000.