A Model of Diversity and Resistance to Attack in an Adaptive Software Network

Neal Holtschulte

Abstract

The feasibility of automatically evolving software patches has been demonstrated by Forrest and Weimer [2]. In this paper we describe a model of a network of computers, each capable of evolving in response to an attack, bug, or vulnerability. Such a network may develop a diversity of software over time. We model such a network in order to explore the effects of different attack patterns and patch-sharing paradigms on the software diversity across the whole network. We wish to eventually investigate whether software diversity throughout a network confers increased resistance to novel attacks.

Introduction

This project is still in an early stage of development. This paper will describe the goal of the project, design of the model, parameters of interest, key questions, and a brief glimpse into some of the output.

The model consists of a graph, representing nodes in a computer network. Each node is capable of evolving its own software patches. Nodes also have a choice of whether or not to share patches they have developed wth their neighbors and whether or not to incorporate patches from their neighbors. Assuming that there are a finite number of maximally resistant programs then there is a tradeoff between diversity and resistance.

If nodes are selfish actors then it is not clear what incentives they have for using valuable CPU's to evolve a patch if they can simply incorporate a patch shared by a neighbor instead. This tradeoff and other interactions will be investigated using the model.

Our goal is to create a network system that will quickly develop resistance to exploits by automatically patching vulnerabilities and that will spread this resistance rapidly, with low overhead, to other computers in the network while maintaining software diversity.

Design / Setup

Initial mock up:

- 16-node ring
- Identical initial software
- Sequential and increasingly severe attacks
- Resistance via local evolution or sharing of resistant patches
- Choice to incorporate or reject shared patches with consequences for diversity and overhead

For now we are using a simple mock-up network as a proof of concept. The sixteen computers (nodes) in the network are arranged in a ring. Each node can share resistance only with its clockwise neighbor. Sixteen was chosen so that we could also analyze a 4 x 4 grid.

Nodes are initialized with identical, low-quality "software" represented by a 20-bit bit string encoding a number in the range zero to one. The quality of the software is determined by its resistance to exploitation or attack. A bit string with floating point value b resists an attack x if $sin(32\pi b) > x$.

$sin(32\pi b)$ has global minima at every $3/64 + n/16$. There are 16 minima between zero and one, enough for one distinct optimal fitness per node.

At time step one, each node in the network is simultaneously subjected to attack. For now each node is subjected to an attack of equal severity. Attack severity is a number in the range zero to one with zero being the least severe and one being the most severe. The closer to one the severity gets, the more time and cpu cycles it takes each node to evolve a patch that confers resistance to the attack. The attack severity keeps rising over the course of a run and the nodes are tasked with "keeping their heads above water", but as time goes on there are fewer values that are resistant until only sixteen resistant values remain.

A new attack will occur only after all nodes are resistant to the current attack. Subsequent attacks will be increasingly difficult to evolve solutions for. In the current implementation, a node will never evolve resistance to one attack and by doing so become susceptible to an earlier attack.

In response to attack, each node runs a local genetic algorithm to evolve a patch (bit string) that resists the attack. Once found, a node propagates the patch to neighbors. Nodes do not wait around to optimize the patch, but will stop evolving as soon as a "good enough" patch is discovered. Nodes will only propogate patches once per attack.

In the current implementation the model waits to proceed to the next attack phase until all nodes have evolved, or adopted from a neighbor, "software" resistant to the attack. That is, each node must acquire a bit string with resistance greater than the attack severity.

Resistance, when acquired, is always passed forward to neighbors, but is incorporated probabilistically when received. A node receiving a patch has a $p\%$ chance of replacing its own bit string with the patch if it has not yet evolved its own. If the node accepts the patch then it will also distribute the patch to its neighbor. If not, it will evolve its own patch and distribute that. Resistance is currently only shared once per attack, and is accepted or rejected without regard to the attack severity.

Parameters of interest

- Attack frequency
- Attack range
- Frequency of resistant patch sharing
- Frequency of resistant patch incorporation
- Network structure

Attack frequency: The current model is sharply delineated into attack phases during which all nodes attempt to aquire resistance to a uniform attack. No new attacks are generated until the current phase ends. In the future, we will do away with the notion of an attack phase. Instead attacks against the network will occur at irregular intervals, though they may only target a subset of the total network.

The three-way interaction between network attack-resistance, attack frequency, and network diversity is of particular interest. We believe there is a tradeoff between diversity and the rapidity with which an individual node can resist an attack, but network-wide diversity itself may serve as a potent defense against frequent attacks directed against a subset of nodes.

Attack range: One of the many assumptions inherent in the current model is that there exists a set of "ideal" (maximally resistant) programs. It is possible instead that patching one hole always exposes the software to another. This assumption could be removed by creating periodic attacks that only exploit programs in a certain range of values, but for which no value in that range is resistant.

Attacks are currently implemented as uniformly increasing and one dimensional in that resistance to an attack of severity x implies resistance to all attacks of severity $\leq x$ and the attack applies equally

across the entire interval zero to one. We will investigate the effect of attacks that very not only in severity, but also in the range they effect. For example, currently it is impossible to evolve software resistant to an attack with severity greater than one, but if such an attack only affected software with bitstring values in the range 0.3 to 0.5 then resistance would be possible, but all the software in the range 0.3 to 0.5 would be eliminated. Such an attack would immediately reduce diversity. We will investigate how the range is "re-populated" in the wake of such attacks.

Frequency of patch sharing: Frequency of patch sharing will impact the cpu time other nodes must spend to evolve resistance as well as the overall network diversity. Additionally, if nodes are selfish actors then it is not clear what incentives they have for using valuable CPU's to evolve a patch if they can simply incorporate a patch shared by a neighbor instead.

Frequency of patch incorporation: It may not be in the best interest of the network as a whole for nodes to adopt shared patches 100% of the time. In the future a more sophisticated decision process may be implemented with respect to patch incoporation. This decision may depend on factors such as the time already spent attempting to evolve a patch, the criticality of the node in question, and the severity of the attack.

Network structure: The current ring layout is both uninteresting and un-representative of real world networks. We will investigate the efficacy of our system on both real-world networks and attempt to discover the ideal network structure to achieve our goals.

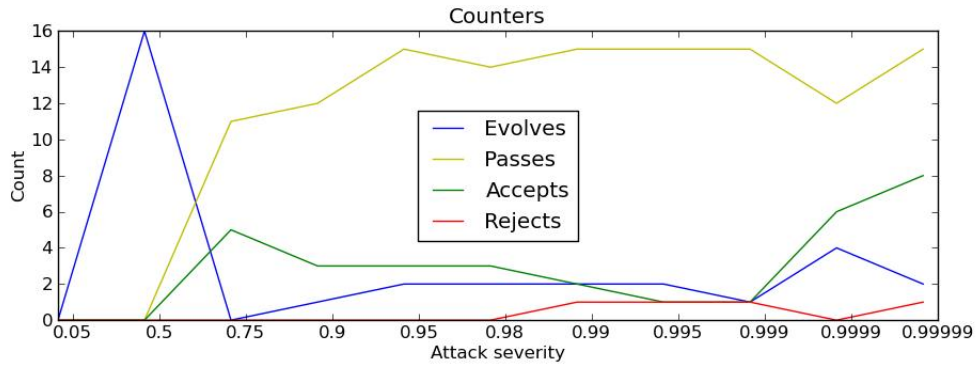
Key Questions

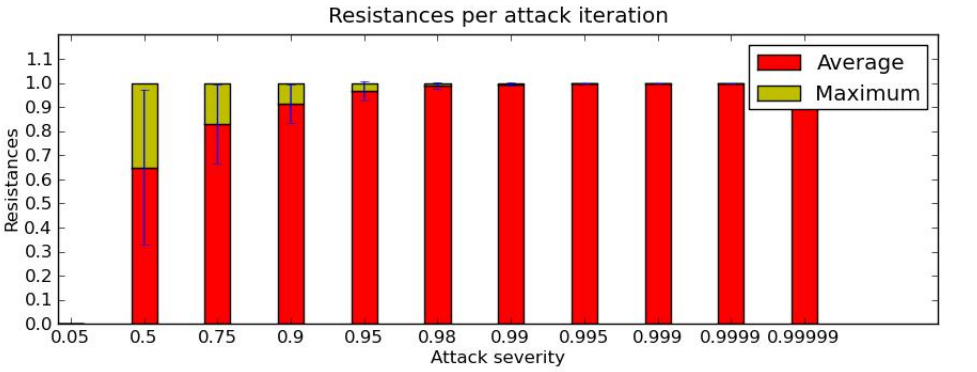
- Does network-wide diversity protect the whole against novel attacks?
- What is the optimal level of diversity to maintain?
- How can this level be maintained network-wide without sacrificing local response times?

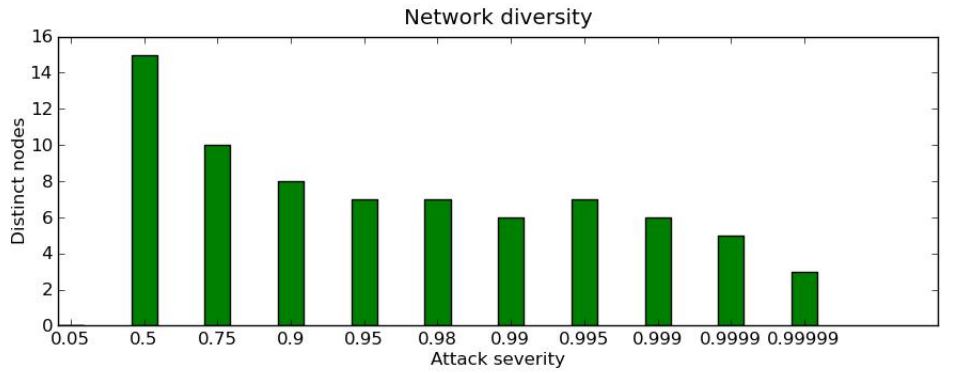
Trade-offs to quantify:

- Rapid distribution of resistance vs. diversity
- Rapid distribution of resistance vs. network traffic overhead
- Evolution of new resistant software vs. cpu overhead

Figures



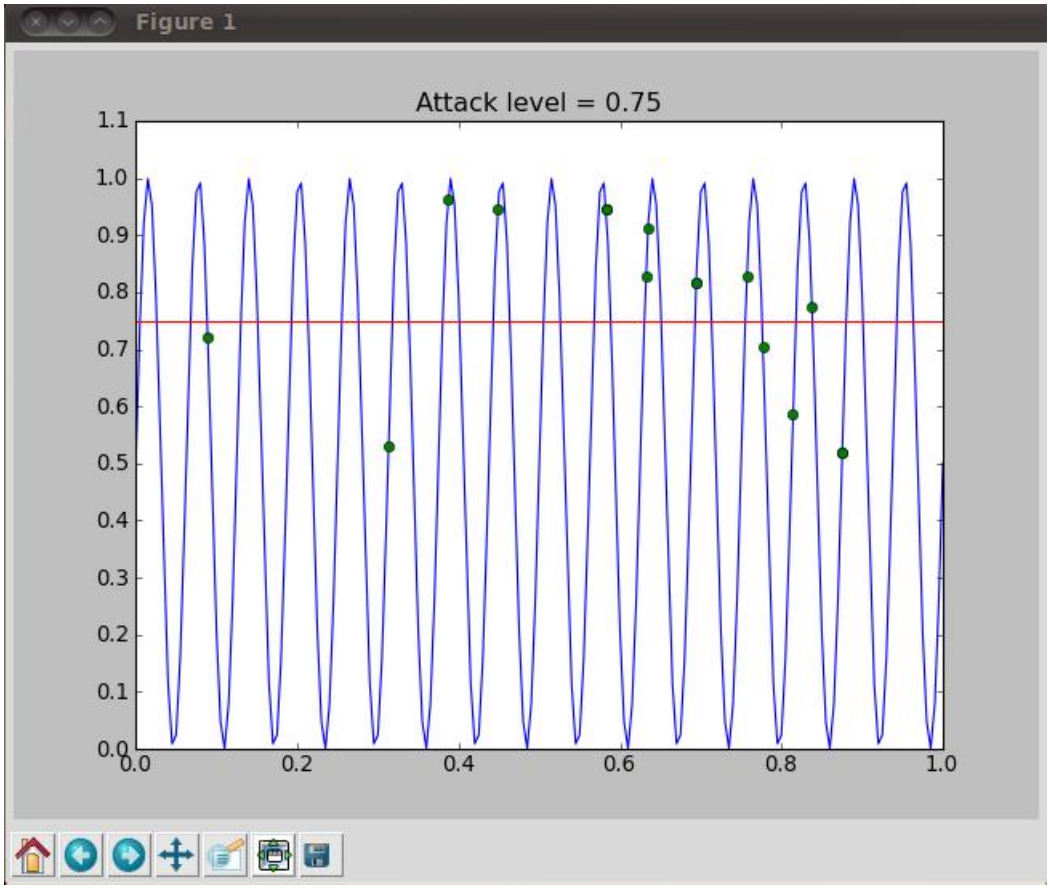




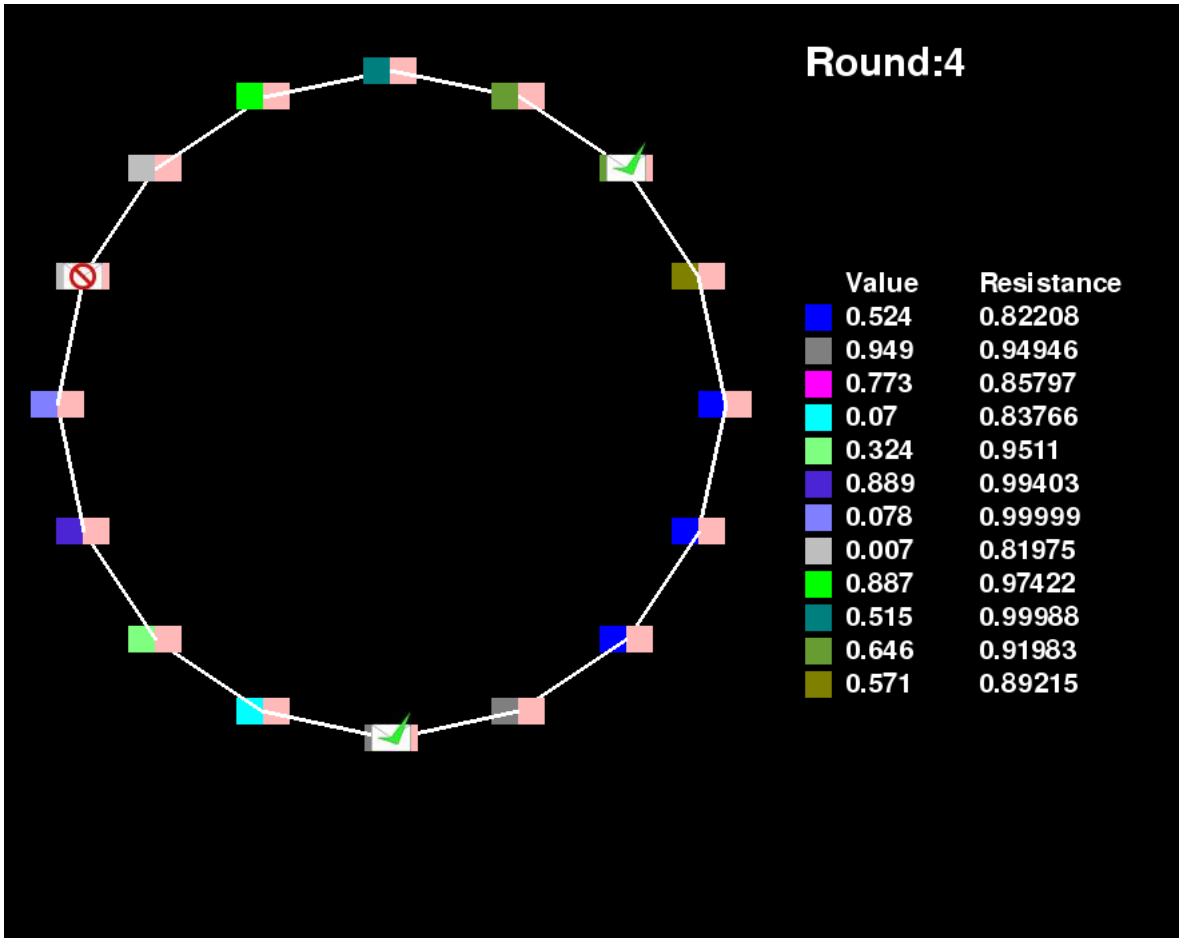
The first graph above shows a count of evolved patches, passed (ie shared patches), and of the shared patches how many were accepted and how many rejected in favor of a computer evolving its own patch. Accepts + Rejects does not equal Passes because patches that were shared with computers that were already resistant were simply ignored.

The second graph above shows the population resistance as the attack severity rose. As you can see, some individuals rapidly reach peak resistance.

The third graph shows diversity of "software" on the network at each attack severity.



Above is a snapshot of a dynamic graph showing nodes (green dots) and where their "software" falls on the fitness landscape (sine curve) as the attack severity (red line) rises.



The above is a screenshot of the visualization of patch sharing on the ring network.

References

[1] "Automatic program repair with evolutionary computation" W. Weimer, S. Forrest, C. Le Goues, T. Nguyen. Communications of the ACM Research Highlight 53:5 pp. 109-116 (2010).

[2] "Automatically finding patches using genetic programming" W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest. ICSE '09: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering, pp. 364–374. IEEE Computer Society, Washington, DC. (2009). PDF ACM SIGSOFT Distinguished Paper and IFIP TC2 Manfred Paul Award for Excellence in Software: Theory and Practice.