Analysis of Attack Graphs using Evolutionary Computation

[Extended Abstract]

Melissa Danforth
California State University, Bakersfield
Bakersfield, CA, USA
mdanforth@csub.edu

ABSTRACT

Attack graphs are a tool to evaluate the security of the network as a whole rather than looking at individual machines. They can discover "foothold" situations where an attacker compromises a series of machines to use as a platform within the network to achieve the final goal(s). Attack graphs are visually complex for all but the smallest networks. Analyzing the graphs to determine a set of actions to take would provide administrators with a plan of action to secure a system. However, determining the minimal set of hardening measures is a reduction of the set cover problem and thus NP. This work explores the use of genetic algorithms to determine a set of hardening measures that maximize the security benefit while minimizing the cost.

Categories and Subject Descriptors

K.6.1 [**Management of Computing and Information Systems**]: Project and People Management; I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

Security

Keywords

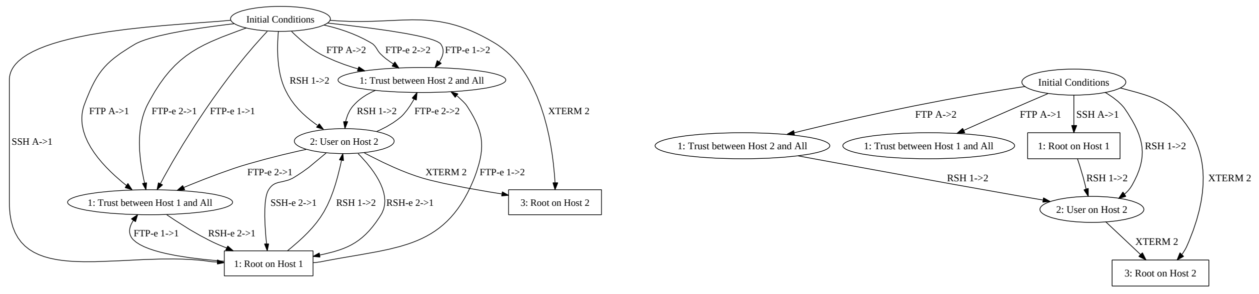
attack graphs, evolutionary computation, patch management, network design

1. INTRODUCTION

Most vulnerability scanners only evaluate individual machines without considering the network as a whole. Attack graphs are a tool to evaluate the composition of vulnerabilities across the entire network. An attacker could use such compositions to achieve further penetration into the network than he might if he were only compromising individual machines. For example, an attacker could penetrate a public server, and use that as a platform to attack internal servers he would otherwise not be able to reach. Attack graphs can show such "foothold" situations and thus are useful at several stages of network planning and management. Attack graphs can be used to evaluate new network designs to see which provides more security, assist patch management by determining the critical set of vulnerabilities with respects to mission resources, identify what steps an attacker most likely took during a forensics evaluation and correlate intrusion detection system (IDS) alerts into an overall attack pattern.

Attack graphs represent exploit paths an attacker can take through the network to achieve a goal such as "root on server x". Originally the graphs were created by hand during network analysis by a team of experts. These hand-created graphs were often quite large and prone to human error. Several methods [9, 3, 8, 7, 1, 6, 10, 2] have been proposed to generate attack graphs automatically. Automatic methods typically have a library of potential exploits, and they match these attack templates to the properties of the network being evaluated. Typically the templates are represented in a "requires/provides" [4] which defines a set of preconditions required for the exploit to be executed and a set of postconditions defining the consequences of the exploit being executed. The automatic methods repeatedly find matches for preconditions of the templates and apply the postconditions until the attacker's goal is met or no more matches are possible.

This method further supports filtering the exploit paths within the graph to assist the analyst in visualizing the attack graph. Specific edges on the attack graph can be hidden in order to reduce the complexity of the visualization of the graph. This can be done automatically via two main viewing modes, "tactical" and "strategic" views, or interactively by the user. The tactical view shows all possible exploit paths through the network. This is useful for situations which require knowledge of all possible ways an attacker might obtain a specific capability. Figure 1(a) shows an example of a tactical view. The strategic view is a "there exists" view which shows all capabilities an attacker can obtain, but not all possible ways in which he can obtain each capability. Thus it shows only that there exists a way for the attacker to obtain a capability. Figure 1(b) shows an example of a strategic view. Both views for any given network will have the same capability nodes. Through a user-defined filter a hybrid view between tactical and strategic views can be generated. An analyst could show all possible exploit edges for certain capability nodes (tactical view) while showing only specific exploit edges for other nodes (strategic view).



(a) Tactical (b) Strategic

Figure 1: Strategic and tactical attack graph views for three node network presented in [9, 3, 8].

2. ANALYSIS PROBLEM

Presenting an attack graph to an administrator is often not helpful as the graph is highly complex for all but the most trivial networks. Even with intricate visualization methods, it is desirable to perform some sort of analysis on the attack graph to give the user some direction to take. The type of analysis is dependent on the use of attack graphs. The two most common uses of attack graphs is for network design and patch management. With these uses, the desired outcome of the analysis is a more secure network. For forensics and intrusion detection correlation, the analysis would be to predict further paths the attacker could have taken in the network and provide a list of items to investigate further. For intrusion response, the analysis should predict the next steps of the attacker and chose the most reasonable responses to block the attacker.

For each of these points of view, different methods of analysis are required. This work focuses on the network design and patch management point of view. The goal of the analysis for network design and patch management is to come up with a set of hardening measures that will prevent the attacker from reaching his goals. A hardening measure is an action which removes a precondition from an atomic attack. Three typical hardening measures are patching a vulnerability, firewalling a port and placing an IDS sensor such that it would detect the exploit. Every hardening measure has a cost associated with it. Typically these costs are determined by the policy of the network. For example, if the policy states the web server on machine A must be publicly acceptable, the hardening measure “firewall port 80 to machine A” should have an infinite cost associated with it so that hardening measure is not used. The costs can also reflect the reality of the network, such as the availability of patches or ease of deploying patches. Even if two networks have the same attack graph, they may have different sets of hardening measures depending on their policies. The analysis algorithm must allow for this site specific customization.

Several prior works [6, 9, 8] have shown that determining the minimum set of hardening measures is a reduction of the set cover problem and thus NP. An approximation method must be used to determine the minimum set of hardening measures. This work explores the use of evolutionary computation to derive the set of hardening measures.

3. IMPLEMENTATION

A genetic algorithm was used to determine a set of hardening measures that maximize the security of the network while minimizing the cost. The security of the network is measured by how well the goal nodes are disconnected from the graph by the hardening measures. This work focuses on deriving hardening measures during the course of patch management or network design. Therefore, the possible hardening measures are:

- Install patch x against an initial condition (vulnerability). This removes a node from the graph corresponding to that vulnerability.
- Firewall connection between hosts a and b on a specified port. This will remove attack edges that depend on network connectivity.
- Place an IDS sensor for attack x occurring between hosts a and b. This will mark edges in the graph as being “watched”. The attack can still succeed but will be detected by the IDS.

Each of these measures has a cost associated with it, as defined by the policy of the network. If the policy does not define a cost, then default costs are assigned depending on the purpose for analyzing the attack graph. If the purpose is to aid the design of a new network, firewalling connections is preferred and gets a lower cost. If the purpose is to determine which patches to install on an existing system, then installing patches is preferred and has a lower cost.

The chromosome is the concatenation of three bitmaps corresponding to each type of hardening measure. The first bitmap contains one bit per node in the initial capabilities of the network. If bit i is 1, this means the patch for the vulnerability in the initial capability node i will be installed. If the bit is 0, no patch will be installed. The second bitmap corresponds to the edges in the attack graph. If bit i is 1, then the network connection required for edge i's attack will be firewalled. If it is 0, then there will be no firewall for that connection. The third bitmap also corresponds to the edges in the attack graph. If bit i is 1, then an IDS rule that will detect the attack between the two hosts in edge i will be enabled. If the bit is 0, no IDS rule for that attack will be enabled.

Some hardening measures will be prevented by the policy of the network or by the nature of an attack. The policy may prohibit the installation of certain patches or it may specify connectivity requirements for a host that would make it impossible to use certain firewall rules. Additionally, not all attacks occur over the network. Some attacks are local to a specific host. For these attacks, a firewall would be ineffective. An IDS would have to be host-based to detect local attacks as well. There are two approaches to handle this. Either the cost associated with such a prohibited action can be set to infinity or the bits corresponding to such actions can be removed from the bitmap or ignored. Both approaches will be explored.

In order to evaluate how the hardening measures indicated in the chromosome affect the security of the network, the fitness function must first apply the actions and then see how the actions affect the goal node(s). For patches, the node corresponding to the patched vulnerability is removed from the graph along with all of its outgoing and incoming edges. For firewall rules, the edge corresponding to the attack that depends on that connectivity is removed. The process of removing edges will affect the in-degree on other nodes in the attack graph. If the in-degree becomes 0, in other words all incoming edges for that node have been removed, then that node is also removed. For IDS rules, the edge is marked as “watched”. Each node contains a bitmap of the incoming edges that are watched. The node to which the edge points sets the corresponding bit in the bitmap to 1. If all incoming edges to a node are watched, the node marks all outgoing edges as watched.

The fitness function determines the sum of the costs for all enabled hardening measures and looks at the goal nodes to determine how well the measures improved the security of the system. The sum of the costs is the total cost for applying those hardening measures. The improvement to the security of the system is the benefit. The ideal result is to have all goal nodes removed from the attack graph. A goal node will only be removed if all its incoming edges are removed. If not all edges can be removed, say due to policy restrictions, then next best result is to have all remaining edges watched. Thus the benefit to the system is based on how many incoming edges have been removed from the goal nodes and how many of the remaining edges are watched. Removing an edge is given a higher weight than watching an edge so that removing all edges will have a higher benefit than having a mixture of removed and watched edges. The fitness function returns $benefit/cost$ so that the more fit individuals are the ones which maximize the benefit and/or minimize the cost.

A genetic algorithm using deterministic rank selection and single point crossover has been selected to evolve the population. For purposes of the crossover operation, all three bitmaps are viewed as a single contiguous bit string. The crossover will occur at some random point along that string. For the mutation operation, a randomly chosen single bit along the string is flipped. This will result in the activation or removal of hardening measures. The initial population is generated by randomly setting the bits in the string.

4. CONCLUSION

Previous analysis methods have ranged from the “try and test” approach using a simulation tool [6], greedy approximation algorithms [9, 8, 3, 1] and expressing the graph as a logical expression of the initial capabilities [2, 5]. The “try and test” method requires extensive interaction by the user. The greedy approximation algorithms of [9, 8, 3] have runtimes that are exponential in terms of the size of the network and their methods do not scale well to larger networks. Ammann’s [1] method is quadratic in terms of the number of attacks, but it only finds the set of minimal critical attacks, not the lowest cost set of hardening measures. The logical expression algorithm of [2, 5] only looks at patching the initial vulnerabilities, not at applying other hardening measures. None of the prior methods consider the policy of the network when determining a set of hardening measures. The genetic algorithm method uses multiple hardening procedures and considers the policy of the network in determining the plan of action.

5. ACKNOWLEDGMENTS

Melissa Danforth’s work was supported in part by a United States Department of Education Government Assistance in Areas of National Need (DOE-GAANN) grant #P200A980307.

6. REFERENCES

[1] P. Ammann, D. Wijesekara, and S. Kaushik. Scalable, Graph-Based Network Vulnerability Analysis. In *Proceedings CCS02: 9th ACM Conference on Computer and Communication Security*, pages 217 – 224, Washington, DC, November 2002. ACM.

[2] S. Jajodia, S. Noel, and B. O’Berry. *Managing Cyber Threats: Issues, Approaches and Challenges*, chapter Topological Analysis of Network Attack Vulnerability. Kluwer Academic Publisher, 2003.

[3] S. Jha, O. Sheyner, and J. Wing. Two Formal Analyses of Attack Graphs. In *IEEE Computer Security Foundations Workshop*, pages 49–63, Cape Brenton, Nova Scotia, Canada, June 2002.

[4] S. J.Templeton and K. Levitt. A Require/Provides Model for Computer Attacks. In *Proceedings New Security Paradigms Workshop*, Cork Island, September 2000.

[5] S. Noel, S. Jajodia, B. O’Berry, and M. Jacobs. Efficient Minimum-Cost Network Hardening Via Exploit Dependency Graphs. In *Proceedings of the 19th Annual Computer Security Applications Conference*, Las Vegas, NV, USA, December 2003.

[6] C. Phillips and L. Swiler. A Graph-Based System for Network-Vulnerability Analysis. In *Proceedings of the New Security Paradigms Workshop*, Charlottesville, VA, 1998.

[7] R. W. Ritchey and P. Ammann. Using Model Checking to Analyze Network Vulnerabilities. In *Proceedings, 2000 IEEE Symposium on Security and Privacy*, pages 156 – 165, Oakland, CA, May 2000.

[8] O. Sheyner. *Scenario Graphs and Attack Graphs*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, April 2004.

[9] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. Automated Generation and Analysis of Attack Graphs. In *Proceedings IEEE Symposium on Security and Privacy*, pages 254 – 265, May 2002.

[10] L. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-Attack Graph Generation Tool. In *Proceedings of DARPA Information Survivability Conference and Exposition II*, June 2001.