Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 29 (2006) 3812–3824
computer communications
www.elsevier.com/locate/comcom

Minimum-cost network hardening using attack graphs

Lingyu Wang *, Steven Noel, Sushil Jajodia

Center for Secure Information Systems, George Mason University, Fairfax, VA 22030-4444, USA

Received 11 May 2006; received in revised form 22 June 2006; accepted 26 June 2006
Available online 18 July 2006

Abstract

In defending one's network against cyber attack, certain vulnerabilities may seem acceptable risks when considered in isolation. But an intruder can often infiltrate a seemingly well-guarded network through a multi-step intrusion, in which each step prepares for the next. *Attack graphs* can reveal the threat by enumerating possible sequences of exploits that can be followed to compromise given critical resources. However, attack graphs do not directly provide a solution to remove the threat. Finding a solution by hand is error-prone and tedious, particularly for larger and less secure networks whose attack graphs are overly complicated. In this paper, we propose a solution to automate the task of hardening a network against multi-step intrusions. Unlike existing approaches whose solutions require removing exploits, our solution is comprised of initially satisfied conditions only. Our solution is thus more enforceable, because the initial conditions can be independently disabled, whereas exploits are usually consequences of other exploits and hence cannot be disabled without removing the causes. More specifically, we first represent given critical resources as a logic proposition of initial conditions. We then simplify the proposition to make hardening options explicit. Among the options we finally choose solutions with the minimum cost. The key improvements over the preliminary version of this paper include a formal framework of the minimum network hardening problem, and an improved one-pass algorithm in deriving the logic proposition while avoiding logic loops.

Keywords: Intrusion detection; Vulnerability analysis; Intrusion prevention

1. Introduction

Attackers typically employ multiple attacks to evade security measures and to gradually gain privileges and approach the final goal. Such a multi-step network intrusion can often infiltrate even a seemingly well guarded network. Isolated vulnerabilities reported by vulnerability scanners, such as Nessus [3], may not seem to be a serious threat until they are cleverly combined by attackers. The completeness of a penetration testing usually heavily depends on techniques of the red team, and is prone to human errors.

Existing approaches build *attack graphs* to represent attack paths, i.e., the possible sequences of vulnerabilities that attackers may exploit during a multi-step intrusion. However, while attack graphs reveal the threats, but they do not directly provide a solution to harden the network against them. Removing vulnerabilities usually incurs different costs, and in practice it is usually infeasible to remove all identified vulnerabilities. A critical but unanswered question in defending against multi-step intrusions is thus: *which of the vulnerabilities should be removed, such that none of the attack paths leading to given critical resources can be realized, where such removal incurs the least cost?* Finding an answer to this question manually is error-prone and tedious, and becomes infeasible for larger and less secure networks whose attack graphs are too complicated.

One recent effort aims to compute a minimal set of vulnerabilities as the solution to harden the network [15,6]. However, such a solution is not directly enforceable, because some of the vulnerabilities are consequences of exploiting other vulnerabilities, and the consequences cannot be removed without first removing the causes. For example, the solution may require an FTP-related

* Corresponding author. Tel.: +1 703 993 3931.
E-mail address: lwang3@gmu.edu (L. Wang).

0140-3664/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2006.06.018

vulnerability to be removed. The vulnerability depends on the existence of the vulnerable FTP service on the destination host and the FTP access privilege for source hosts, and the latter may further depend on other vulnerabilities on the source hosts. Clearly, there are multiple choices with different costs in removing this single vulnerability. This shows that a minimal set of vulnerabilities is not necessarily a minimal solution, considering the vulnerabilities they may implicitly depend on.

In this paper, we propose a different method that takes into account the dependency relationships among vulnerabilities in deriving hardening solutions. More specifically, we view each vulnerability as a Boolean variable, and we derive a logic proposition to represent the negation of given critical resources in terms of initially satisfied security-related conditions (or *initial conditions* for short). This proposition is thus the necessary and sufficient condition for protecting the critical resources. To make hardening options explicit, we transform this logic proposition into its disjunctive normal form (DNF). Each disjunction in the DNF provides a different option in hardening the network. We then choose options with the minimum costs based on given assumptions on the cost of initial conditions.

Our solution removes the previously mentioned limitation of existing approaches, because the hardening options require disabling initial conditions only. Each initial condition can be independently disabled because they do not depend on other vulnerabilities or conditions. For example, instead of requiring the removal of an FTP vulnerability, our solution may required disabling the vulnerable FTP service or denying FTP accesses to certain hosts, which are both readily enforceable. In the simplification of the logic proposition, we can identify seemingly relevant initial conditions whose removal does not really help to protect the critical resources. Such insights are important in keeping the cost of network hardening minimal, but they are also impossible to obtain in previous approaches.

The preliminary version of this paper has outlined the basic idea and method [9]. The key improvements in the current paper are as follows. First, we formalize the notation of attack graph and clearly define the minimum-cost network hardening problem. Second, instead of depending on an extra forward search to remove cycles in attack graphs, we propose a different algorithm that searches the attack graph and removes cycles all in one-pass. This approach removes the difficulty of the previous method in dealing with cycles that cannot be easily removed in the forward search. As a side benefit, it also improves the performance by approximately 50% through saving the preprocessing step of forward search.

The rest of this paper is organized as follows. The next section reviews related work. Section 3 provides a formal framework of the attack graph and an example to motivate our study. Section 4 states the problem of network hardening and derives a solution based on graph searching. Section 5 provides a case study to illustrate the proposed method. Finally, Section 6 concludes the paper.

2. Related work

A number of tools are available for scanning network vulnerabilities, such as Nessus [3], but most of them can only report isolated vulnerabilities. On the research front, attack graphs are constructed by analyzing the inter-dependency between vulnerabilities and security conditions that have been identified in the target network [4,18,11,2,10,13,16,14,1,15,5]. Such analysis can be either forward starting from the initial state [11,16] or backward from the goal state [13,15]. Model checking was first used to analyze whether the given goal state is reachable from the initial state [13,12] but later used to enumerate all possible sequences of attacks between the two states [15,6].

The explicit attack sequences produced by a model checker face a serious scalability issue, because the number of such sequences is exponential in the number of vulnerabilities multiplied by the number of hosts. To avoid such combinatorial explosion, a more compact representation of attack graphs was proposed in [1]. The *monotonicity assumption* underlies this representation, i.e., an attacker never relinquishes any obtained capability. This newer representation can thus keep exactly one vertex for each exploit or security condition, leading to an attack graph of polynomial size (in the total number of vulnerabilities and security conditions). In this paper, we shall assume such a compact representation of the attack graph.

Algorithms exist to find the set of exploits from which the goal conditions are reachable [1]. This eliminates some irrelevant exploits from further consideration because they do not contribute to reaching the goal condition. However, as we show in Section 5, this result may still include many irrelevant exploits, even though the goal condition is reachable from them. The reason lies in that the reachability is only a necessary but not sufficient condition for an exploit to actually contribute to reaching the goal condition. On the other hand, our solution is necessary and sufficient for a goal condition to be satisfied.

Closest to our work, the *minimal critical attack set* is a minimal set of exploits in an attack graph whose removal prevents attackers from reaching any of the goal states [15,6,1]. The minimal critical attack set thus provides solutions to harden the network. However, their method ignores the critical fact that consequences cannot be removed without removing the causes. The exploits in their solutions usually depend on other exploits that also need to be disabled. The solution is thus not directly enforceable. Moreover, after taking into account those implied exploits the solution is no longer minimum. Our method fixes this problem by including only initial conditions in the solution. The initial conditions can be independently disabled, leading to a readily deployable solution.

Attack graphs have been used for correlating intrusion alerts into attack scenarios [8,17]. Such alert correlation methods are parallel to our work, because they aim to employ the knowledge encoded in attack graphs for detecting and taking actions against actual intrusions, whereas