A HEURISTIC APPROACH TO NETWORK HARDENING USING ATTACK GRAPHS

TANIA ISLAM

A THESIS

IN

THE CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE IN INFORMATION SYSTEMS SECURITY

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

MAY 2009

Your file Votre référence
ISBN: 978-0-494-63037-2
Our file Notre référence
ISBN: 978-0-494-63037-2

ABSTRACT

A Heuristic Approach to Network Hardening Using Attack Graphs

Tania Islam

In defending against multi-step attacks, network hardening answers the following important question: Which vulnerabilities must be removed from a network in order to prevent attackers from compromising critical resources while minimizing the implied cost in terms of availability or administrative efforts. Existing approaches to network hardening derive a logic proposition to represent the negation of the attack goal in terms of initially satisfied security conditions. In the disjunctive normal form (DNF) of the logic proposition, each disjunction then provides a viable solution to network hardening. However, such solutions suffer from an exponential time complexity. In this thesis, we study heuristic methods for solving this important problem with reasonable complexity. We evaluate our proposed solutions through extensive experiments. The results show that our solution can achieve reasonably good network hardening results in significantly less time than the optimal solution would require. Also, for scenarios where additional cost constraints may render a perfectly secure network hardening solution impossible, we extend our heuristic methods to partial hardening solutions. Such solutions can provide best possible improvement in terms of security under given cost constraints.

Acknowledgments

I would like to thank all those who provided help for this thesis.

I would like to express my sincere thanks to my supervisor, Dr. Lingyu Wang, for his help, guidance and advices to my study at Concordia University, especially to the published papers and this manuscript.

I would like to thank my parents, relatives and all my friends for their kind support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Protecting computer networks in enterprises and critical infrastructures against malicious intrusions is crucial to the economy and national security. Various intrusion detection systems (IDSs) and vulnerability scanners have been developed to protect networks against malicious attacks. Nonetheless, we have seen an increasing threat from network intrusions in terms of both scale and severity despite over twenty years of research in vulnerability analysis and intrusion detection.

One of the reasons to the increasing threat from network attacks is that most existing tools can only identify vulnerabilities or attacks in isolation. On the contrary, attackers usually exploit multiple correlated vulnerabilities to evade detection or firewalls or to gradually elevate their privilege in a network. The isolated alerts or vulnerabilities reported by IDSs and vulnerability scanners thus only provide a partial picture about multi-step sophisticated attacks. The sheer number of alarms raised by those tools will usually render them difficult to analyze and eventually lead them to be completely ignored by security administrators.

Moreover, a vulnerability analysis often assumes that all identified weaknesses will be immediately removed. However, in practice, this is usually not the case. The removal of vulnerabilities is often complicated by environmental factors (the availability of software patches or hardware upgrades), cost factors (in terms of money or administrative efforts), or mission factors (the demand for availability and usability). In many cases, a security administrator may have to live with some of the discovered vulnerabilities. A critical question thus naturally arises: Which vulnerabilities should be removed first in order to prevent attacks while reducing the implied costs? This question is usually named the *network hardening* problem.

There exist solutions for network hardening using attack graphs [53,61] (the next chapter will give a detailed review of the literature). Attack graph is a well accepted model for correlated vulnerabilities. By correlating vulnerabilities through common pre- or post-conditions, an attack graph can help a security administrator to understand the threat of potential multi-step attacks and to determine potential attack paths, which may be regarded as an automated penetration testing. However, attack graph does not directly provide a solution for removing identified threats. Finding the solution by hands for large networks whose attack graphs are usually very complicated is generally not feasible.

On the other hand, an automated hardening method regards correlated vulnerabilities as combinations of Boolean variables [61]. To secure critical resources represented as goal conditions in an attack graph, a logic proposition is derived based on the attack graph.

The logic proposition is then converted to its disjunctive normal form (DNF) in the set of initially satisfied conditions. Then a least-cost network hardening options can be chosen among the conjunctive clauses in the DNF. However, finding a minimum-cost solution in this way is expensive, because the length of a derived logic proposition may be exponential in the number of initial conditions or in the size of an attack graph. Nonetheless, considering the practical impact of the problem, heuristic solutions are clearly desirable for obtaining low-cost hardening solutions within a reasonable time limit.

1.2 Thesis Contribution

This thesis focuses on finding feasible solutions for network hardening when given time or cost constraints do not allow for an optimal solution to be derived. More specifically, our main contributions are the following.

- We point out limitations of existing attack graph-based network hardening approaches. We show that such methods have an exponential worst case complexity which prevents them from scaling to large networks. We also point out that given constraints on the cost of hardening solutions may lead such methods to produce no solution at all, which is clearly not desirable.
- We study a series of heuristic algorithms as candidate solutions. The algorithms all aim to find a set of initially satisfied security conditions for disabling such that potential attacks on critical resources may be prevented while the cost for disabling such conditions remains reasonably low. We show the worst case scenario of each

such candidate algorithm and point out limitations of the algorithms.

- We propose the main heuristic approach based on insights obtained while studying the candidate solutions. This approach sorts conditions based on not only cost but also the number exploits requiring that condition, and it includes a forward search in the attack graph with a limited degree of backtracking. We instantiate the approach as two heuristic algorithms with slightly different ways of sorting the conditions.

- We also study situations where a perfectly secure hardening solution cannot be derived due to given cost constraints. We extend our heuristic approaches to a partial hardening solution based on a probabilistic security metric. This solution reduces the risk of potential multi-step attacks to the least possible degree with respect to given cost constraints.

- We conduct comprehensive experiments to compare the performance of our solutions with the optimal solution in terms of the cost of resulted hardening solutions and the time taken to compute such solutions. We vary different parameters in generating random attack graphs, such as the size of attack graphs, the number of initial conditions, the ratio between exploits and conditions, and the ratio between different relationships among exploits and conditions. All results show that our solution can produce reasonably good hardening solutions while taking significant less time limit than exhaustive search does.

1.3 Organization of the Thesis

In Chapter 2, we discuss related work on attack graph and its construction, analysis and different applications where it has been successfully applied. In Chapter 3, we review the background knowledge of our work. We describe attack graphs and related concepts, and we formally define the network hardening problem and the partial hardening problem. In Chapter 4, we first study candidate solutions which provide us critical insights to our main heuristic solutions. We then present our main heuristic solutions in details and analyze potential difficulties in reducing the cost and how our proposed algorithms can overcome them. We also discuss the proposed partial hardening solutions in this chapter. In Chapter 5, we show experimental results on the comparison between our heuristic methods and the optimal method for network hardening with respect to both the cost of computed solutions and the time taken. Finally, in chapter 6 we conclude the thesis and discuss potential future work.

Chapter 2

Literature Review

In this chapter, we review the literature on the generation, analysis, and application of attack graphs. We also discuss other work that are relevant to our research.

2.1 Attack Graph

For protecting systems against malicious attacks, there exist various intrusion detection systems (IDSs) and vulnerability scanners, such as Nessus [16], Nmap [35], Snort [52], Cisco security scanner, SATAN [20], System Scanner by ISS [26], CyberCop [10] and Computer Oracle and Password System (COPS) [19]. Those solutions are being applied in real world networks and they can render attacking such networks much more difficult than without such solutions in place. However, existing solutions usually identify vulnerabilities or attacks in isolation, which only provides a partial picture about securing a network since today's attackers typically employ sophisticated multi-step attacks.

Phillips and Swiler [45] propose the concept of attack graph and they also present a graph-based approach for generating attack graphs. In their model, the inputs of an attack graph include configuration files, attacker profiles, and a database of attack templates, which must be manually created. The nodes of the attack graph are attack templates instantiated with particular users and machines, whereas edges are labeled by probabilities of success or cost of attacks. The graphs can be analyzed to find the shortest paths between given start and end nodes. The idea of grouping similar nodes is mentioned although the correctness critically depends on identical configuration among such nodes.

Another model [58] expresses attack graphs with the require and provide approach using the precondition and postcondition of each exploit. For each successful attack, the attacker can obtain the ability to perform more attack steps so each successful exploitation increases his/her capabilities in launching new attacks. An attack specification language JIGSAW is used in describing attack steps. The language requires low level details of capabilities and requirements of attacks which may be hard to obtain. This require and provide approach brings flexibility in discovering potentially new attack scenarios. Using this language and the given specifications, an IDS and attack analysis system can be created.

In [50], model checking is applied to the analysis of multi-step network attacks. Known vulnerabilities on network hosts, connectivity between hosts, initial capabilities of the attacker are described as states and exploits as transitions between states. This model is given to a model checker as its input and the reachability in terms of given goal states is given as a query. The model checker then produces a counterexample if a sequence of exploits can lead to goal states. Such a sequence of exploits indicates a potential attack that must be avoided to secure the network. The term *topological vulnerability analysis* is coined in [51] which provides more details on how connectivity should be modeled at different layers.

In [29, 53], model checking is used for a different purpose, that is to enumerate all attack paths. A modified model checker is used to take as input the finite-state machine created from network information. The model checker provides all counterexamples to a query about the safety of the goal states. Those are essentially the possible attack paths. Other types of analysis are also discussed by the authors, including how to find a *cut set* in the attack graph, such that goal conditions can no longer to reached. The problem of finding the minimum possible attack that leads to the given goal conditions is shown to be intractable. One apparent limitation of this approach is that all attack paths are explicitly enumerated in its result,.which leads to a combinatorial explosion.

A *monotonic assumption* is adopted in [1] to address the scalability of model checking-based approaches. It states exploits will never cause the attacker to relinquish any previously obtained privileges. Attack paths can then be implicitly modeled as paths in a directed graph including exactly one copy of each exploit and its pre- and post-conditions; edges interconnect exploits to their conditions. The assumption thus reduces the complexity of attack graph from exponential to polynomial in the number of hosts. However, it also makes some attacks impossible if they disable services or invalidate vulnerabilities. Attack graphs are generated using a two-pass search that first links exploits by starting from the attacker's initial state and then removes those irrelevant states by searching backward from the goal state.

In [39] the authors proposed a logic-based approach to attack graph generation to improve the efficiency. Using this approach, the generated attack graph always has a polynomial size in the size of analyzed network. A Network security analyzer MulVAL [40] is used to build the attack graph generation tool. Here, each node of the attack graph represents a logical statement and the edges define the relationship between network configuration and what the privileges the attacker potentially could gain. Here, the main focus is on the root causes of the attack. Using this logical attack graph representation, one can obtain all the possible attack scenarios using a simple depth-first search. Using this representation, it can also be ensured that an attack graph will always have a polynomial size in the size of the network.

Attack graph has many different applications, such as the proactive detection of potential multi-step attacks, the realtime detection of anomalies, the network hardening, and computer forensic [29]. As attack graphs can reveal potential intrusions beforehand, it can be used to incorporate security policies and IDS models into vulnerability scanning and then to perform analysis and upgrade the defense system accordingly. It can be used to do a cost/benefit analysis by finding the least cost way for hardening an insecure network. In Forensics, attack graph is also use to find probable attacks and to assess damages of the system [56]. For taking legal action against the attacker, the analyst needs to show attack steps as evidence. Using attack graph, the whole attack paths can be matched to data extracted from IDS logs. In [22] the authors showed informal attack graphs are helpful in the iterative design of a system used to protect sensitive data at a customer site.

2.2 Network Hardening

In [53] a *minimum critical set* is computed which is basically the minimum set of exploits in the attack graph by disabling which we can harden the network. The minimum critical attack set is essentially the concept of a cut set in graph theory. In [1], the authors propose an algorithm called findMinimul to find the attack that takes the least number of steps from the initial state to the goal condition to launch an attack. Another such approach proposed in [29] introduces the minimum critical attack set in a more scalable way. However, all these solutions cannot be directly used by system administrators as the set of exploits to be disabled are unavoidable consequences of other exploits, which must be disabled in the first place.

The concept of network hardening with respect to initially satisfied conditions is first introduced in [36]. It is argued that disabling such initial conditions is a better choice than minimum critical sets with respect to the need of security administrators. To disable exploits in the critical set, we have also disable the causes of such exploits, which ultimately leads to a set of security conditions that are initially satisfied and do not depend on others. Such initial conditions are only the pre-condition of some exploits but they are not the post condition of any exploit. This indicates that these conditions can be disabled independently. Therefore, an effective hardening measure is to find the set of initial conditions disabling which can disable the goal conditions.

This thesis is mainly inspired by the approach in [36] of finding a set of initial conditions that can disable the given goal conditions and has the minimum cost. In that work, the authors represent given critical network resources as a logic proposition of initial conditions. To build the logic proposition, each vulnerability is first viewed as a Boolean variable. The two connectives AND and OR are used where AND is used between conditions required by the same exploit, and OR is used between the exploits implying the same condition. A true condition means a condition is satisfied and false condition means it is disabled for hardening the network. The hardening option is made explicit by transforming the logic proposition into its disjunctive normal form (DNF). Each of the disjunctive form is then treated as a solution to network hardening. The minimum-cost solution can be chosen among these options. However, as we shall show, the procedure has an unavoidable exponential worst-case complexity, because its result is exponential. So, for larger networks, enforcing this approach will be costly or even impossible depending on the number of hosts and their connectivity. Our work addresses this issue with a heuristic approach that yields reasonably good results in significantly less time .

2.3 Security Metric

The concept and overview of security metrics is given in [28] with methods of defining, creating, and utilizing security metrics in enterprises. Also, relevant issues of security metrics are given in the 2001 Workshop on Information Security System Scoring and Ranking [2]. Existing standardization efforts include the assessment of methods for measuring the level of computer security [37], the security metrics guide for information technology systems [57]. Some properties of security metric are also discussed in [11, 12, 38]. They use MTTF and Markov model to measure the security of a network, which is based on the success rate for an attacker that is distributed exponentially. In [3] the authors discussed using minimum efforts for executing exploits as a metric. Another approach uses the weakest attack model showing the least number of conditions for an attack as a metric [44]. The attack surface concept is a metric for measuring the vulnerability in software against potential attacks [41–43].

Due to its capability of modeling correlated vulnerabilities, attack graph is used as the basis of developing network security metric in [62]. A framework for a security metric is proposed which includes several principles and methodologies. There are several general requirements that any security metric should satisfy. That is, a longer path leading to the attack goal means better security; multiple attack paths are less secure than any of the paths alone; executing an exploit may change the difficulty of executing another exploit, even if the two do not directly depend on each other in the attack graph; the security of a network should be measured against all relevant resources with different weights and each reconfiguration may incur a certain cost that must be considered together with security.

The framework is instantiated as an attack resistance metric in [4]. The proposed attack resistance metrics is based on two compositions assuming the attack resistant either as a real number or a set of initial security conditions. This metric helps to quantify the comparison among the security of different network configurations. Another work defines a probabilistic security metric based on the same framework [60]. The goal is to calculate an overall score for the network and use it to measure how secure the given critical resources in a network are. To define the basic metric, each exploit and condition is associated with a probability to measure the intrinsic likelihood of the exploit. The value assignments for these individual probabilities are based on some basic measurements, such as, how difficult the exploit is, which are available in standards like CVSS [9]. Then, a cumulative probability score is calculated as the probability of an exploit to be successfully exploited in an attack graph based on the probability of other exploits. To calculate the cumulative scores, the conjunction and disjunction relationships among exploits must be distinguished. Cycles are handled during calculating the probabilities of exploits.

2.4 Other Related Work

A parallel research topic is alert correlation, which reconstructs multi-step attack scenarios from isolated alerts. They may employ prior knowledge about attack strategies [8,14,15,18] or the causal relationships between attacks [7,32,33]. Those methods may either aggregate alerts with similar attributes [6,13,55,59] or statistical patterns [30,46]. Hybrid approaches also exist that combine different techniques for better results [33, 47, 63]. Alerts missed by IDSs can be tolerated by clustering alerts with similar attributes [34], and incomplete knowledge can be pieced together through statistical analyses [46,47].

Alert correlation techniques are also used for other purposes such as to relate alerts to the same thread of attacks [25]. The privacy issue of intrusion detection and in particular alert correlation is investigated in [64]. Alert correlation is a potential method for dealing with insider attacks in [48, 49]. Existing efforts on integrating information from different sources exist, such as the model in *M2D2* [31] and the Bayesian network-based approach [65]. Some commercial products claim to support realtime analyses of alerts such as Tivoli Risk Manager [23].

Chapter 3

Preliminary

In this section, we briefly review some important concepts that are relevant to our further discussions. First, we introduce attack graph and its related concepts. Second, we formalize the network hardening problem. Third, we introduce the probabilistic security metric in order to make our later discussion on partial hardening self-contained. Finally, we review standard heuristic approaches and explain how they can potentially be integrated with our heuristics.

3.1 Attack Graph

An *attack graph* is a graphical model of inter-dependent vulnerabilities on networked hosts. An attack graph is a directed graph whose set of nodes is partitioned into two classes, namely, *exploits* and *security conditions* (or simply *conditions*). An exploit is typically represented as a predicate $v(h_s, h_d)$, where h_s and h_d represent two connected hosts and v a vulnerability on the destination host h_d. A security condition is a predicate $c(h)$, indicating the host h satisfies a condition c relevant to one or more exploits. Notice that h_s, h_d, and v are abstract notations that could in practice possess different semantics, for example, h_s and h_d can be host names, IP addresses, and so on, and v can be the name of a vulnerability or its ID in a vulnerability database. These are formalized in Definition 1.

Definition 1 *An attack graph G is a directed graph $G(E \cup C, R_r \cup R_i)$ where the set of nodes include E, a set of* exploits, *and C, a set of* conditions, *and the set of edges include the* require *relation $R_r \subseteq C \times E$ and the* imply *relation $R_i \subseteq E \times C$.*

Corresponding to the inter-dependency between exploits and conditions, the two types of edges in an attack graph have different semantics. First, the *require* relation R_r is a directed edge pointing from a condition to an exploit, which means the exploit cannot be executed unless the condition is satisfied. For example, an exploit $v(h_s, h_d)$ requires following two conditions, that is the existence of the vulnerability v on h_d and the connectivity between h_s and h_d. Second, the *imply* relation R_i pointing from an exploit to a condition means executing the exploit will satisfy the condition. Notice that there is no edge directly connecting two exploits (or two conditions).

Figure 1 shows a small example of attack graphs. We assume a simple scenario where a file server (host 1) offers the File Transfer Protocol (ftp), secure shell (ssh), and remote shell (rsh) services; a database server (host 2) offers ftp and rsh services. The firewall only allows ftp, ssh, and rsh traffic from a user workstation (host 0) to both servers. In the attack graph, exploits of vulnerabilities are depicted as predicates in ovals and conditions as predicates in clear texts. The two numbers inside parentheses denote the source and destination host, respectively. The attack graph represents three self-explanatory sequences of attacks (attack paths).

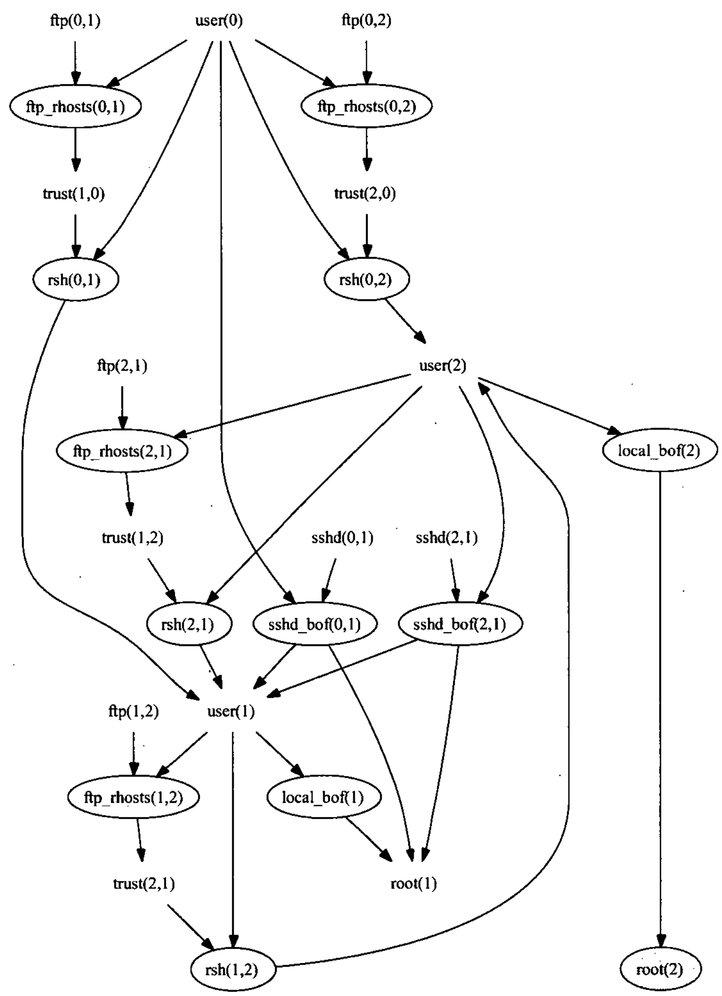

Figure 1: An Example of Attack Graph

Two important semantics of attack graphs are as follows. First, the require relation is always *conjunctive* whereas the imply relation is always *disjunctive*. More specifically, an exploit cannot be realized until *all* of its required conditions have been satisfied, whereas a condition can be satisfied by any *one* of the realized exploits. Second, the conditions are further classified as *initial* conditions (the conditions not implied by any exploit) and *intermediate* conditions. An initial condition can be independently disabled to harden a network, whereas an intermediate condition usually cannot be [36].

To generate an attack graph, two types of inputs are necessary, namely, *type graph* and *configuration graph*. Type graph represents expert knowledge about the dependency relationship between vulnerabilities. On the other hand, configuration graph represents hosts and their connectivity and vulnerability information. We assume the domain knowledge required for type graph is available from tools like the Topological Vulnerability Analysis (TVA) system, which covers more than 37,000 vulnerabilities taken from 24 information sources including X-Force, Bugtraq, CVE, CERT, Nessus, and Snort [27]. On the other

can be obtained using available network scanning tools, such as the Nessus scanner [17].

3.2 The Network Hardening Problem

though Figure 1 is a relatively simple scenario with three hosts and four vulnerabilities, because multiple interleaved attack paths can lead to the goal condition, an optimal solu-

such a solution by hand may not be trivial, either. As an example of attack paths, the

Input: An attack graph $G(E \cup C, R_r \cup R_i)$ and the goal conditions $C_g \subseteq C$
Output: A solution set L which needs to disable
Method:

1. **Calculate** the smallest cost of the initial conditions
2. **Sort** the initial conditions IC according to the cost and sort same cost elements according to their effective cost
3. **Make** the cost of condition = 0 and cost of exploit = infinity
4. Set S = GenSet(ic)
5. **Make** each element of S explored
6. **Traverse** G in BFS manner
7. **For** each condition c do
8. **If** c is explored
9. **If** all the exploits that imply c are explored
10. **Make** solution of c = combination of all solutions of exploits implying c
11. **Make** cost of c = combination of all the cost of the exploits that imply c
12. **Enqueue** all the exploits that require c
13. **Else**
14. **Put** c back in queue
15. **For** each exploit e do
16. **If** e is explored
17. **If** all the conditions required by e are explored
18. **Make** solution of e = smallest cost solution of the condition that required by e
19. **Make** cost of e = cost of the smallest cost condition
20. **Enqueue** all the conditions that implied by e
21. **Else**
22. **Put** e back into queue
23. **If** the Goal is not disabled
24. **Goto** step 4
25. **Return** solution of Cg

Figure 10: The Second Heuristic Approach

conditions are disabled) or condition (when all of the exploits implying it are disabled).

More specifically, a condition should be marked as disabled only if all of the exploits implying it are explored and disabled; it should be marked explored (but not disabled) if all of the exploits implying it are explored but at least one of them is not disabled. Otherwise, we cannot make any decision since the unexplored exploit may or may not be disabled in the future when we explore it. For an exploit, it should be marked as disabled if at least one of the required conditions is disabled; it should be marked explored (but not disabled) if all of the required conditions are explored but not disabled. We cannot make any decision if none of the required conditions is explored.

However, cycles in attack graphs may bring additional difficulties to network hardening approaches. For example, consider the attack graph shown in Figure 11, which includes a cycle $c5 \rightarrow e3 \rightarrow c6 \rightarrow e4 \rightarrow c5$.

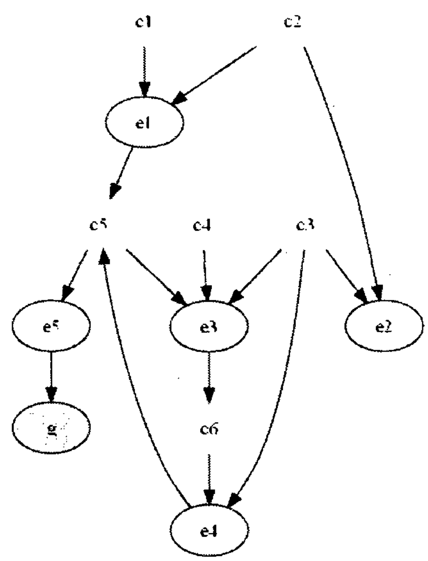

Figure 11: An Example of Cycle in Attack Graph

In traversing the graph, after we disable $e1$ and reach $c5$, we need to make the decision whether $c5$ should be marked as disabled and explored. As aforementioned, we cannot make any decision here since the unexplored exploit $e4$ implies $c5$ and $e4$ may or may not be disabled in the future when we explore it. Similarly, when we proceed from $c3$ and $c4$ to $e3$, since $e3$ depends on $c5$ that is still not explored, we cannot make any decision. Such an inter-dependency between $c5$ and $e4$ thus will cause the algorithm to stop. However, we can easily see that in this particular case, $e3$ and $e4$ can never be exploited if $e1$ is disabled.

Therefore, we should simply regard all nodes inside the cycle as disabled.

More generally, the following enumerates the different cases we may have in handling cycles.

- As illustrated in the left-hand side of Figure 12 (0 indicates disabled and 1 not disabled), if we reach a cycle through a condition and if all the exploits required by this condition are not disabled, and if no other exploit from outside the cycle can satisfy the required conditions inside the cycle, then all the nodes will be considered disabled.

- As illustrated in the right-hand side of Figure 12, if we reach a cycle through a condition and if all the exploits required by this condition are not disabled, and if any other exploit from outside the cycle can satisfy the required condition inside the cycle, then all the nodes will be considered not disabled.

- As illustrated in the left-hand side of Figure 13, if we reach a cycle through an exploit and if all the conditions required by this exploit are not disabled, and if no other exploit from outside the cycle can satisfy the required condition inside the cycle, then all the nodes will be considered disabled.

- As illustrated in the right-hand side of Figure 13, if we reach a cycle through an exploit and if all the conditions which required by this exploit are not explored, and if any other exploit from outside the cycle can satisfy the required condition inside the cycle, then all the nodes will be considered not disabled.

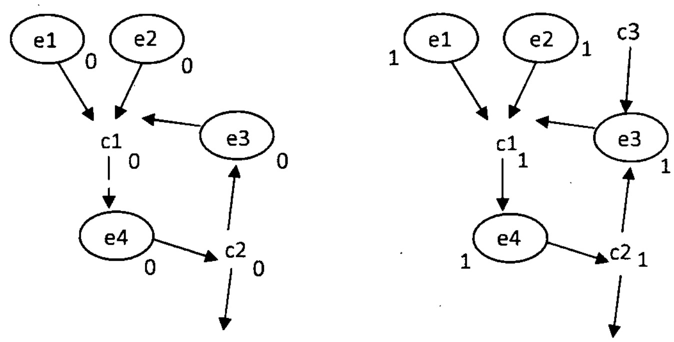

Figure 12: Case 1 and 2

4.4 Partial Network Hardening

In hardening a network, fully harden the network to disable the goal condition is not always an option. Sometimes this may require such a high cost that it outweighs the security risk. We consider a simple model where a constraint is given as the highest amount of cost that can be accepted for any hardening solution. With such a constraint, a network may not be fully hardened since the optimal solution may have a cost higher than the given constraint. The only remaining choice is a best effort approach in reducing the likelihood of attacks. We employ the security metric introduced in previous sections to measure such a likelihood. We then extend our previous approach to a partial hardening solution for reducing the risk of, instead of eliminating, potential attacks on the goal condition.

In Figure 14, we annotate each exploit in the attack graph with its individual score. All the conditions have the individual score of 1, which is omitted. Assume the cost constraint is given as 12. We can easily see that the network cannot be fully hardened, as the least-cost

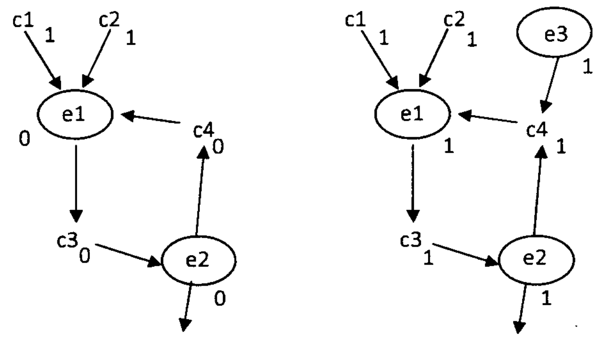

Figure 13: Case 3 and 4

solution is to disable both $c1$ and $c2$, which has a cost of 14.

Table 2 and Table 3 show the calculations of the cumulative scores for conditions and exploits, respectively.

Table 2: Cumulative Scores of Conditions

Condition	p(c)	P(c)
c1	1	1
c2	1	1
c3	1	1
c4	1	0.9
c5	1	0.6
c6	1	0.4
c7	1	0.24
g	1	0.048

We extend our algorithm in the previous section to provide partial hardening solutions for reducing the cumulative score of the goal condition. More precisely, we modify the algorithm in such a way that by disabling a node, we change its individual score as zero, indicating that the node cannot be reached any longer. We then update the cumulative

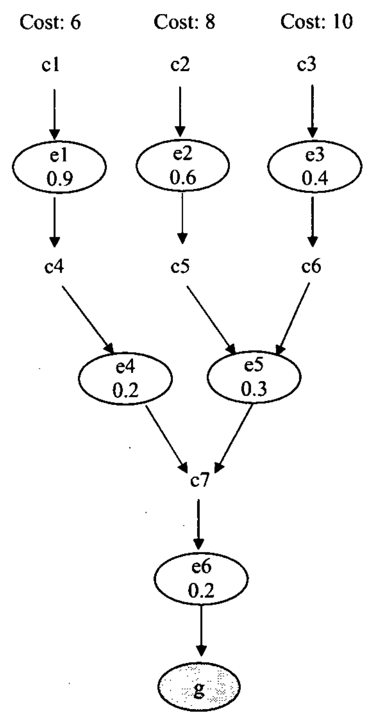

Figure 14: An Attack Graph with Probability Scores

scores based on the new individual scores. In doing so, we attempt to search for a solution that can yield the least cumulative score for the goal condition, with the cost of the solution below the given constraint.

The algorithm will thus starts with $c1$, which has the least cost. By disabling $c1$, we change the cumulative score of $e1$, $c4$, and $e4$ to be zero (which in this particular case basically eliminates the left branch of the attack graph). Table 4 and Table 5 show the updated cumulative scores after this partial hardening. We can see that the goal condition now has a cumulative score of 0.014, which is smaller than its original value.

The algorithm will then proceed to disable $c2$ and $c3$, respectively (since $c1$ and $c2$ together will not satisfy the cost constraint). The cumulative score for the goal condition in those two cases is 0.278 and 0.328, respectively. Clearly, the best solution is to disable $c1$.

Table 3: Cumulative Scores of Exploits

Exploit	p(e)	P(e)
e1	0.9	0.9
e2	0.6	0.6
e3	0.4	0.4
e4	0.2	0.18
e5	0.3	0.072
e6	0.2	0.048

Table 4: Cumulative Scores of Conditions after Partial Hardening

Condition	p(c)	P(c)
c1	1	1
c2	1	1
c3	1	1
c4	1	0
c5	1	0.6
c6	1	0.4
c7	1	0.072
g	1	0.014

The above procedure is more formally described in Figure 15.

Table 5: Cumulative Scores of Exploits after Partial Hardening

Exploit	p(e)	P(e)
e1	0.9	0
e2	0.6	0.6
e3	0.4	0.4
e4	0.2	0
e5	0.3	0.072
e6	0.2	0.0144

Input: An attack graph $G(E \cup C, R_r \cup R_i)$,the goal and Afford cost ac
conditions $C_g \subseteq C$
Output: A solution set L which needs to disable
Method:
1. **Calculate** the smallest cost of the initial conditions
2. **Sort** the initial conditions IC according to effective cost
3. **Make** the cost of condition = 0 and cost of exploit = infinity
4. Set S = GenSet(IC)
5. **Make** each element of S explored
6. **Traverse** G in BFS manner
7.   **For** each condition c do
8.   **If** c is explored
9.   **If** all the exploits that imply c are explored
10.   **Make** solution of c = combination of all the solutions of the exploits that imply c
11.   **Make** cost of c = combination of all the cost of the exploits that imply c
12.   **If** cost $c > ac$
13.   **Put** c back to traverse again
14.   **Else**
15.   **Enqueue** all the exploits that require c
16.   **For** each exploit e do
17.   **If** e is explored
18.   **If** all the conditions required by e are explored
19.   **Make** solution of e = smallest cost solution of the condition that required by e
20.   **Make** cost of e = cost of the smallest cost condition
21.   **If** cost $e > ac$
22.   **Put** e back to traverse again
23.   **Else**
24.   **Enqueue** all the conditions that implied by e
25.   **If** the Goal is not disabled
26.   **Goto** step 4
27. **Return** solution of Cg

Figure 15: The Procedure for Partial Network Hardening

Chapter 5

Simulation Results

All experiments are based on a PC equipped with one Intel Core 2 Duo 1.86 GHz CPU, 1 GB of RAM, Microsoft Windows XP Professional with Service Pack 2. For graph rendering we use the GraphViz visualization package [54]. For development, Netbeans 5.5 and jdk1.6.0_01 is used. The main objective of the experiments is to compare the performance of the proposed heuristic algorithms and that of the optimal solution, in terms of both time taken and quality of the hardening result (that is, the cost of the hardening solution).

All the synthetic attack graphs were randomly generated using a Java program we developed. The generation is based on a set of adjustable parameters. These parameters control the size of the network, the number of abstract exploits, the number of conditions, the relationships between exploits and conditions, the way exploits and conditions are assigned to hosts, the choice of goal conditions, and so on. We shall show simulation results based on varying choices of such parameters in the rest of this section to demonstrate the effectiveness of our algorithms when applied to different types of attack graphs.

5.1 Performance Comparison Based on Time

Figure 16 compares the performance of our heuristic approach 1 with that of the optimal solution (that is, exhaustive search) in terms of the time taken by the two methods (in millisecond). The size of attack graphs varies from 10 to 50 nodes. For attack graphs with 50 nodes, the optimal cost calculation takes approximately 6000 ms for calculating the network hardening solution and our proposed algorithm takes approximately 80 ms to generate the heuristic solution. We can see the time taken by the optimal solution increases very fast, which is as expected, whereas the heuristic algorithm increases significantly slower (not quite observable in this figure). The heuristic method thus provides administrators with a reasonably efficient solution.

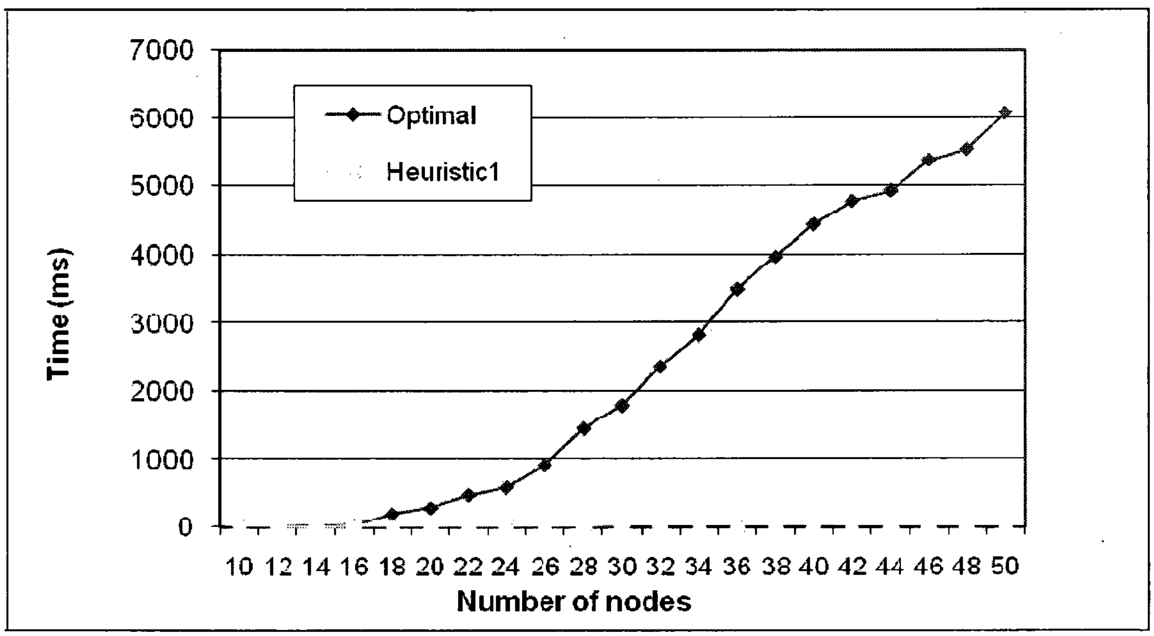

Figure 16: Comparison of Time Taken by the Heuristic Approach 1 and the Optimal Solution

Figure 17 compares the performance of the two heuristic approaches in terms of the time taken. The size of attack graphs also varies from 10 to 50 nodes. We also vary the distribution of exploits among hosts. One set of the curves correspond to uniform distribution and the other set normal distribution. We can see that different distributions of exploits has only a small effect on the performance of both approaches. Also, the second heuristic approach (that is, to sort initial conditions first based on cost then on effective cost) performs slightly better than the first approach.

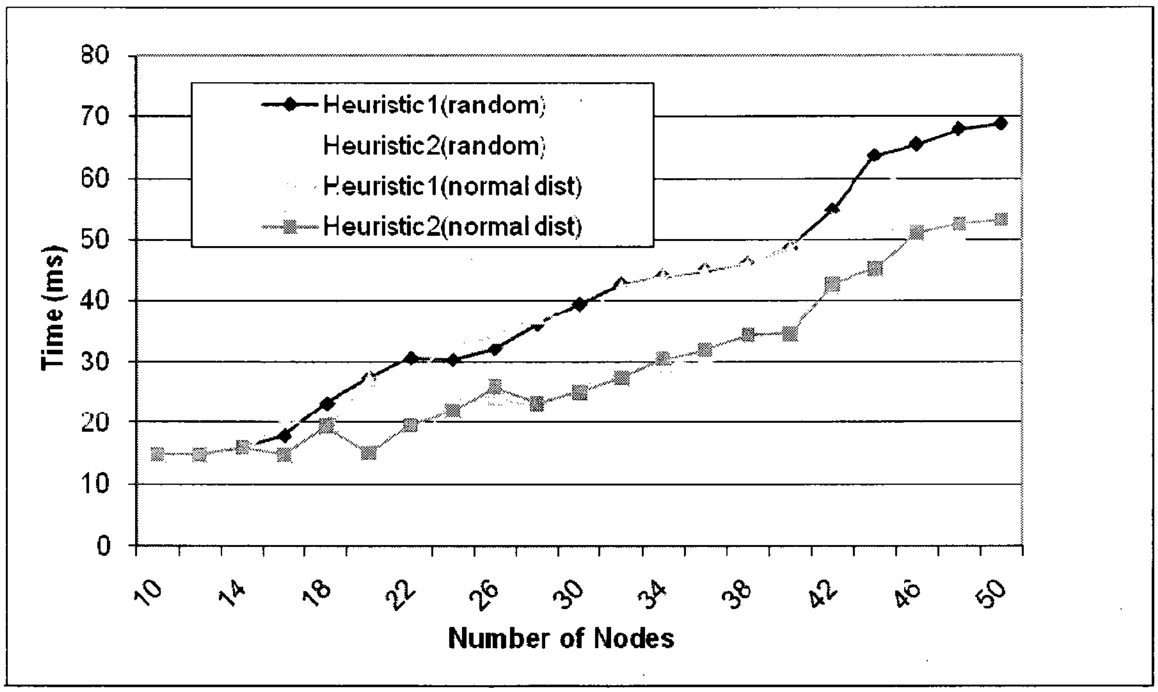

Figure 17: Comparison of Time Taken by the Two Heuristic Approaches with Different Distributions of Exploits

Figure 18 compares the cost of solutions produced by the optimal solution and the heuristic approach. Although the cost of our solution is relatively higher than the optimal solution, the difference is small. Considering the significant gain in terms of performance, our solution has clear advantages over exhaustive search. Also, we can see that the difference between the two solutions stays relatively stable when the attack graph size increases.

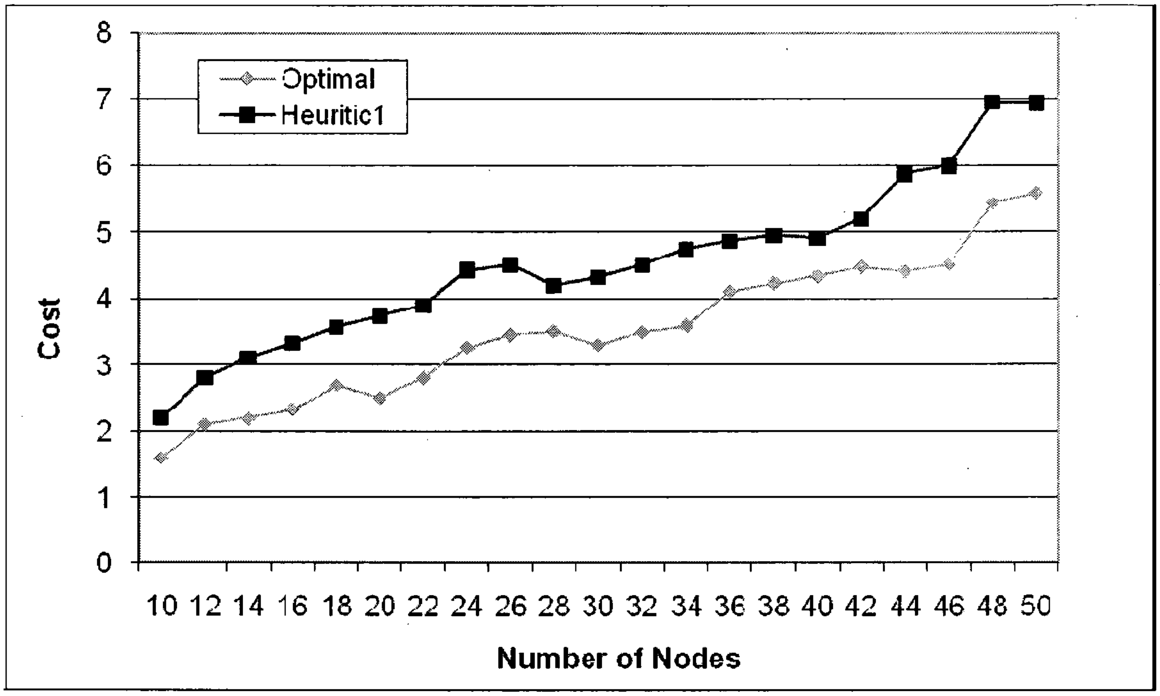

Figure 18: Comparison of the Cost of Solutions Produced by the Heuristic Approach 1 and the Optimal Approach

Figure 19 compares the cost of different heuristic solutions with exploits assigned according to uniform and normal distributions. We can see that, although the cost produced by both approaches is slightly lower with the normal distribution than uniform distribution, the difference is almost negligible. This indicate both approaches work well for different types of attack graphs.

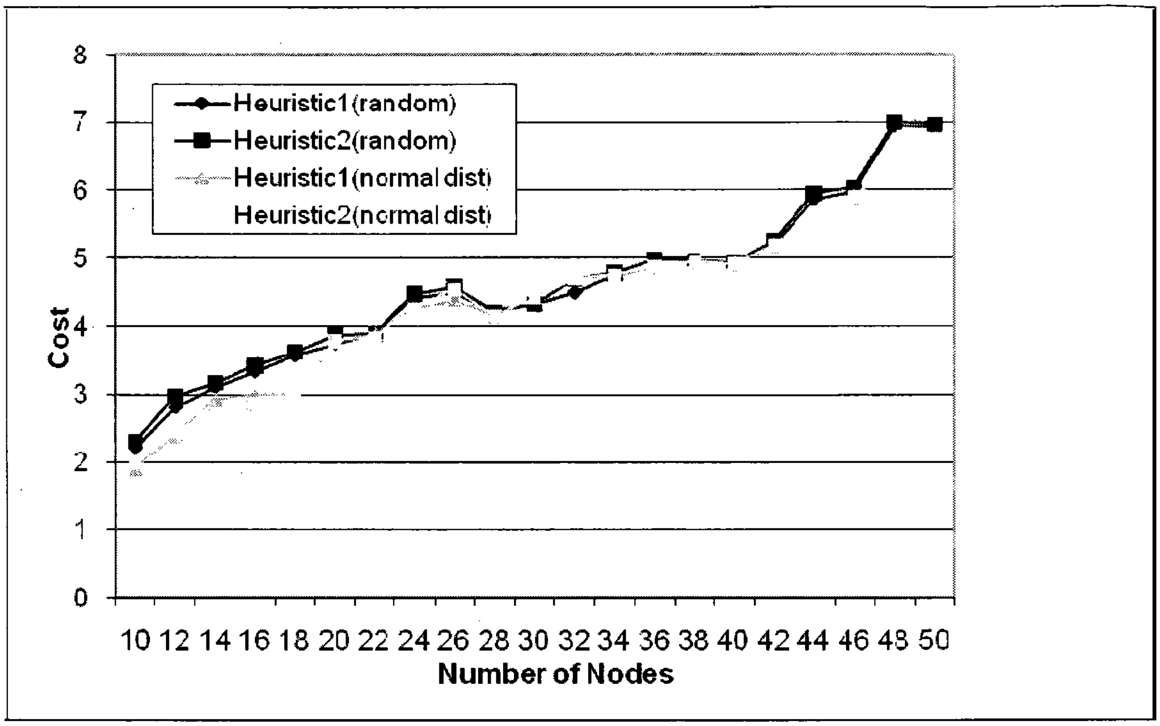

Figure 19: Comparison of the Cost of Solutions Produced by the Heuristic Approaches

5.2 Performance Comparison with Varying Number of Initial Conditions

Since the cost of network hardening critical depends on the number of initial conditions, we study the effect of such number on the performance of our algorithms. Figure 20 compares the time taken by heuristic approach 1 and the optimal solution. The trends are very similar to Figure 16. Clearly, the heuristic solution has a much better scalability than exhaustive search.

Figure 21 compares the time taken by the two heuristic approaches under different distributions of exploits. We can see the difference is minor. Both algorithms scale roughly

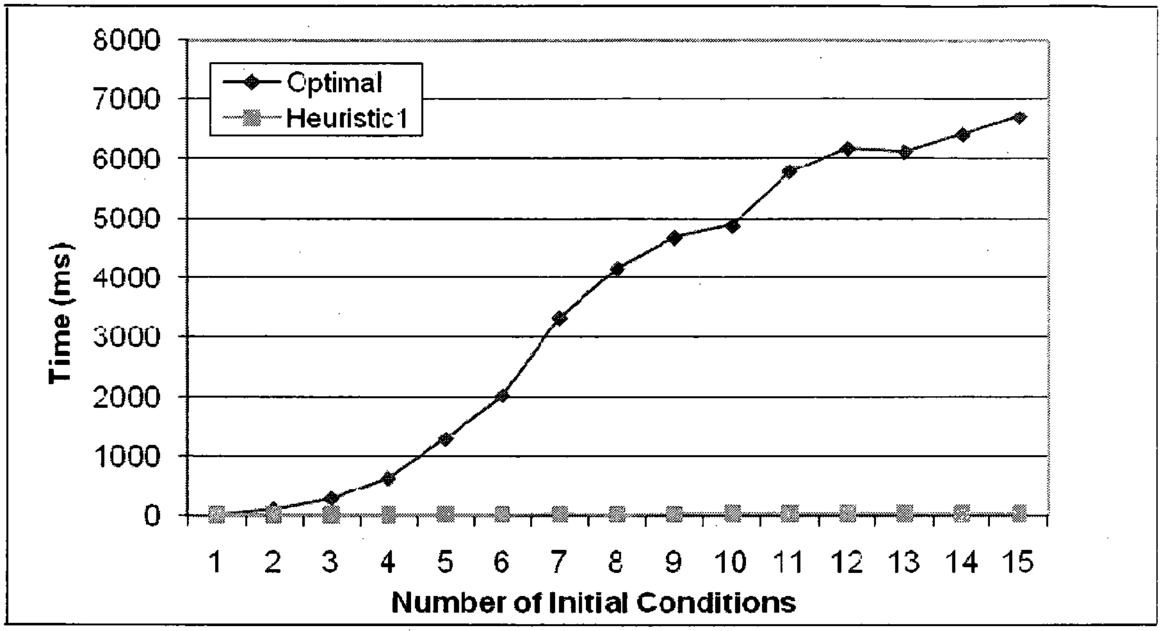

Figure 20: Comparison of Time Taken by the Heuristic Approach 1 and the Optimal Solution

the same when the number of initial conditions increases.

Figure 22 compares the cost of the solutions produced by the heuristic approach and that by the exhaustive search. We can see that difference is relatively small, which indicates the effectiveness of our approach. Also, we can see that although the absolute value of the difference increases as the number of initial conditions increases (due to the fact that both solutions have a higher cost), the relative difference does not change as much.

Figure 23 compares the cost of the solutions produced by the two heuristic approaches. We can see that the difference is negligible. Both approaches produce roughly the same result, which indicates that the effectiveness of our solutions are not affected much by the distribution of exploits among hosts.

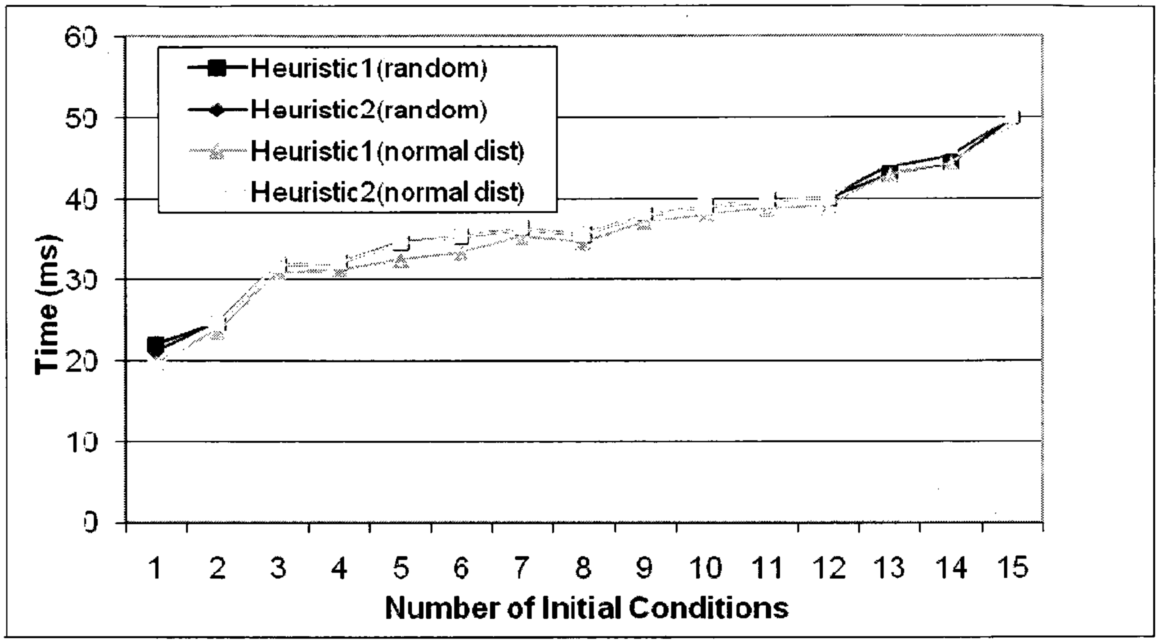

Figure 21: Comparison of Time Taken by the two Heuristic Approaches

5.3 Performance Comparison with Ratio between Imply/Require Relations

We have mentioned that there exists an important semantic difference between the imply relation over an exploit and its post-conditions, which is always disjunctive, and the require relation between an exploit and its pre-conditions, which is always conjunctive. Therefore, the ratio between those two types of relations may affect the way our algorithms search the attack graph. To measure such effect, we conduct experiments while varying the ratio of require/imply relations. In Figure 24, we can see that the time taken by both our heuristic approach and the exhaustive search increases when the ratio increases. This is because when each initial condition is required by more exploits, the search starting from the

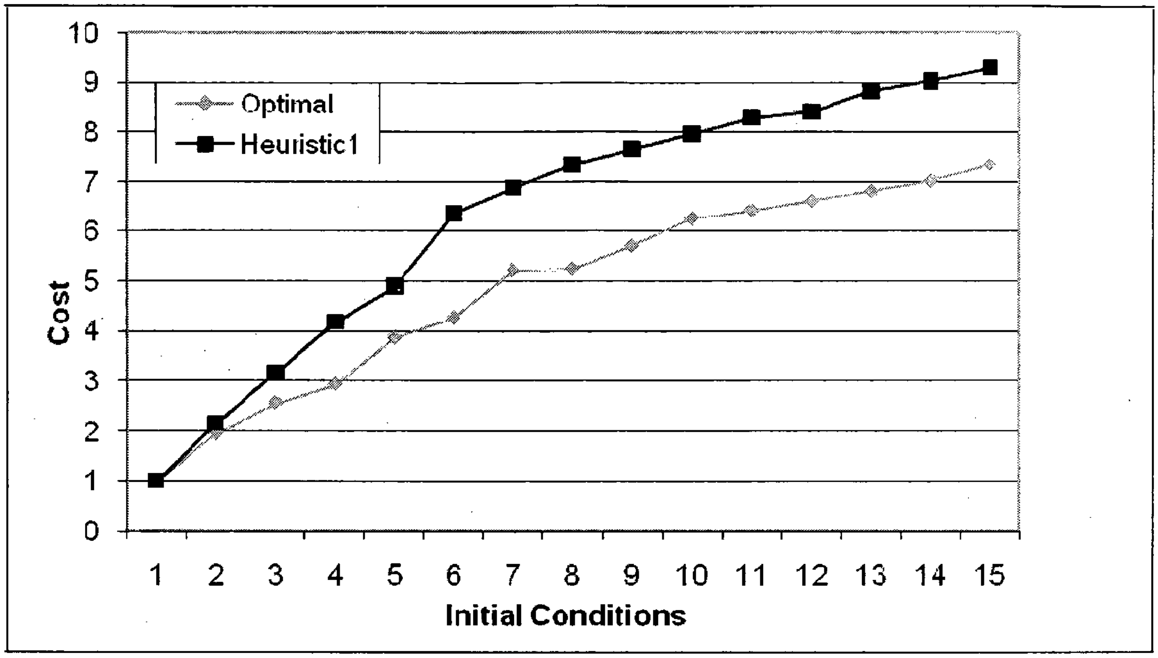

Figure 22: Comparison of the Cost of Solutions Produced by the Heuristic Approach 1 and the Optimal Approach

same number of initial conditions will involve a larger portion of the attack graph and thus takes more time. However, the heuristic approach clearly outperforms exhaustive search regardless of the ratio.

In Figure 25, we compare the time taken by the two heuristic approaches under varying ratios of require and imply relationships. We can see that the difference is marginal. The two heuristic approaches both performs well regardless of the ratio.

Figure 26 compares the cost of solutions produced by the exhaustive search and that by the heuristic approach 1 under varying ratios of require and imply relationships. We can see that the cost decreases with both approaches when the ratio increases. This is because when each exploit requires more conditions, it is easier to harden the network since disabling one

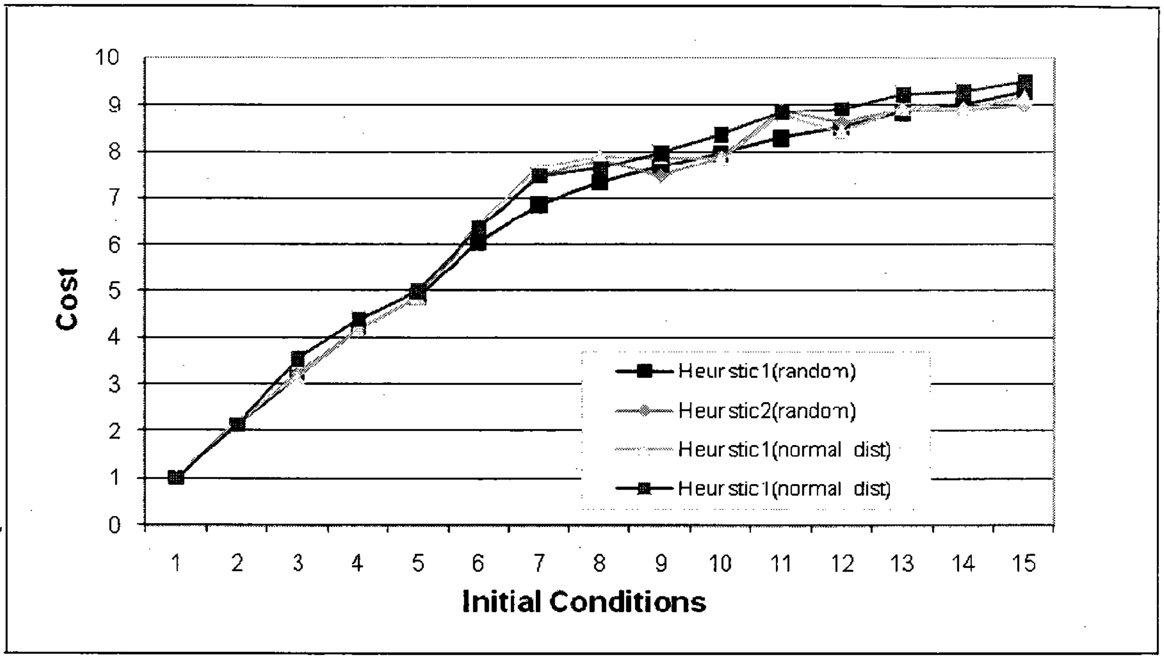

Figure 23: Comparison of the Cost of Solutions Produced by the Heuristic Approach 1 and the Optimal Approach

condition will prevent more exploits. We can see also that the difference between the heuristic approach and optimal solution is small, indicating our approach is effective.

Figure 27 compares the cost of solutions produced by the two heuristic approaches under varying ratios of require and imply relationships. We can see that the difference between the two heuristic approaches is marginal indicating that both approaches perform roughly the same regardless of the ratio.

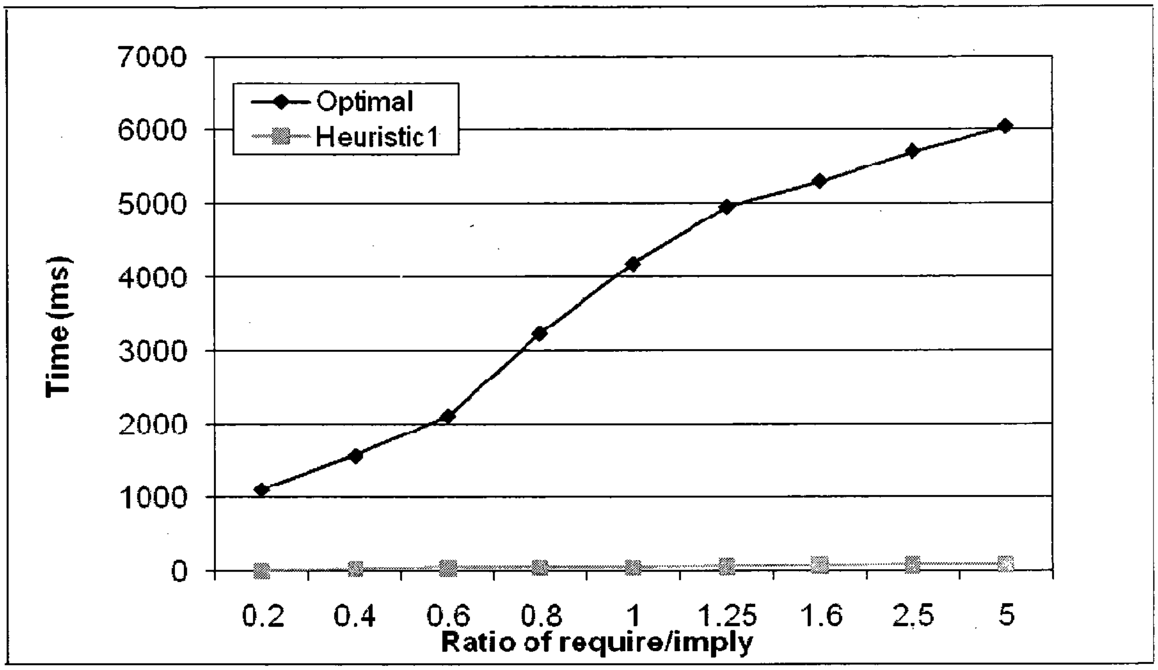

Figure 24: Comparison of Time Taken by the Heuristic Approach 1 and the Optimal Solution with Varying Require/Imply Ratio

5.4 Performance Comparison with Ratio between Exploits and Conditions

We now study the effect of the ratio between exploits and conditions on the performance of our approaches. Figure 28 compares the time taken by the heuristic approach and the optimal solution under varying ratio between exploits and conditions. We can see that when the number of conditions increases both solutions require more running time. The reason lies in the fact that the number of initial conditions also increases. However, we can see that the heuristic solution outperforms the exhaustive search regardless of the ratio.

Figure 29 compares the time taken by both heuristic solutions. We can see that the

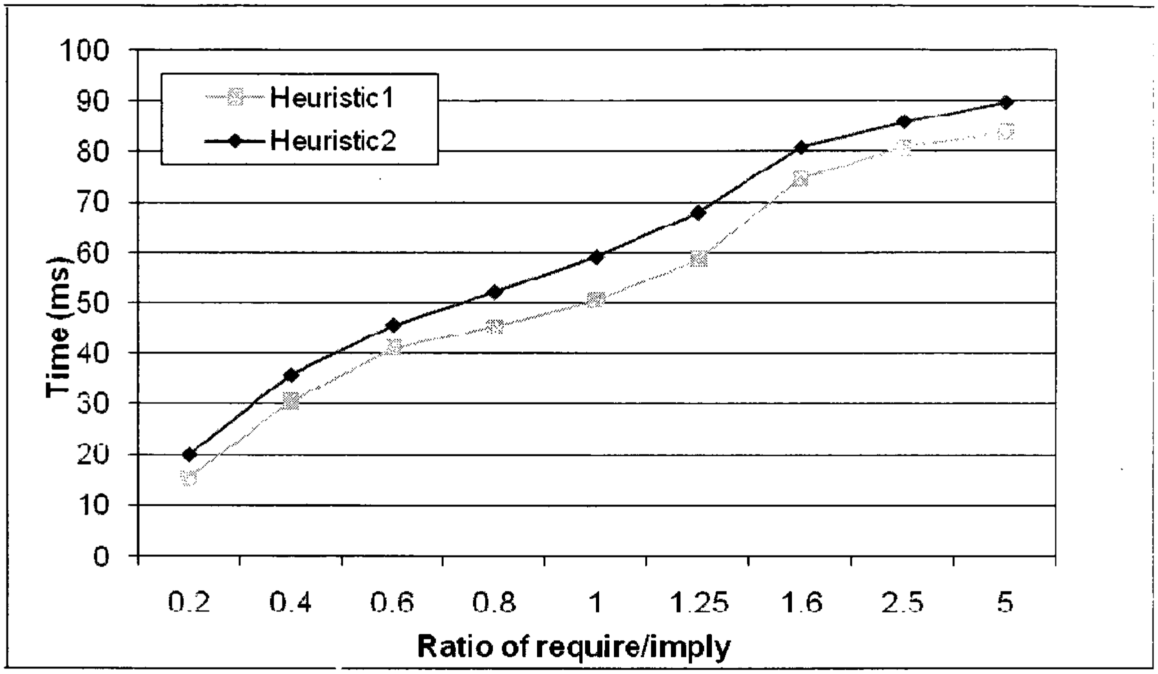

Figure 25: Comparison of Time Taken by the two Heuristic Approaches with Varying Require/Imply Ratio

second heuristic approach takes slightly more time but the difference is small, so both approaches perform well regardless of the ratio.

Figure 30 compares the cost of solutions produced by the heuristic solution 1 and that by the optimal solution. We can see that the cost decreases when the ratio increases. This is because when there are more conditions, it is relatively easier to disable each exploit so the cost of a solution decreases. We can also see that the difference is small, indicating our approach is effective regardless of the ratio.

Figure 31 compares the cost of solutions produced by the heuristic solutions. We can see that the difference is marginal, indicating both approaches perform roughly the same regardless of the ratio.

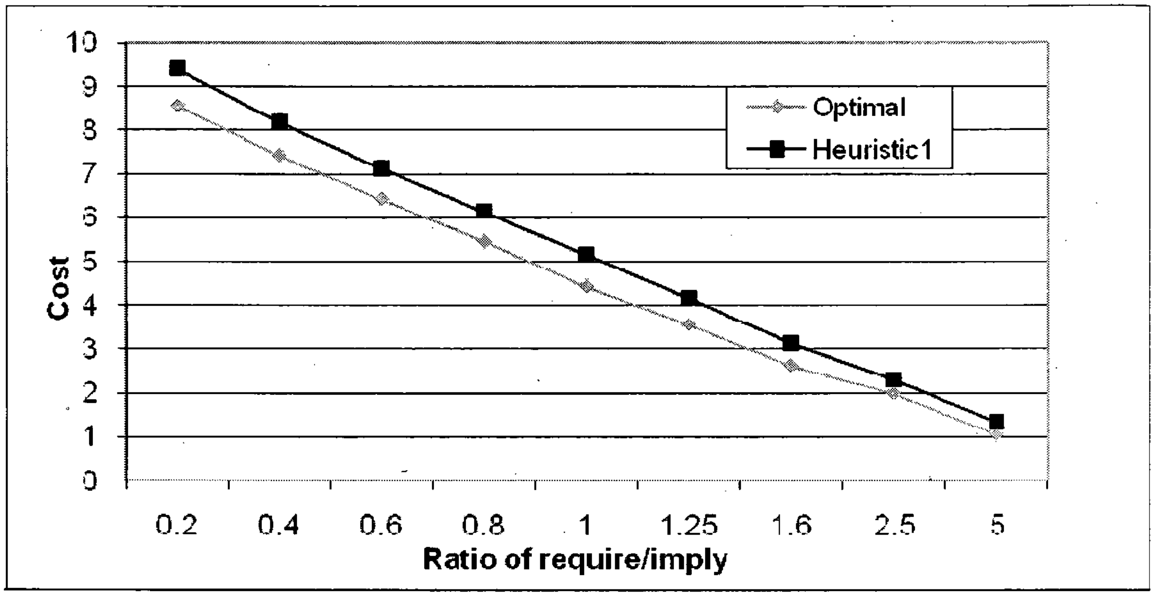

Figure 26: Comparison of the Cost of Solutions of the Heuristic Approach 1 and the Optimal Solution with Varying Require/Imply Ratio

5.5 Performance Evaluation for Partial Network Hardening

To evaluate our performance of partial network hardening approaches we chose attack graphs which have individual probability assigned for each of the nodes then calculate the cumulative probability for them. Let all the cost constrain is the 20% of the optimal cost solution which is showed in the second figure. Then we calculate the cumulative probabilities for the goal condition and hence compare them to show the reduced value. We denote the original cumulative probability of goal condition as P(g) and our calculated one as PŠ(g) in the graph.

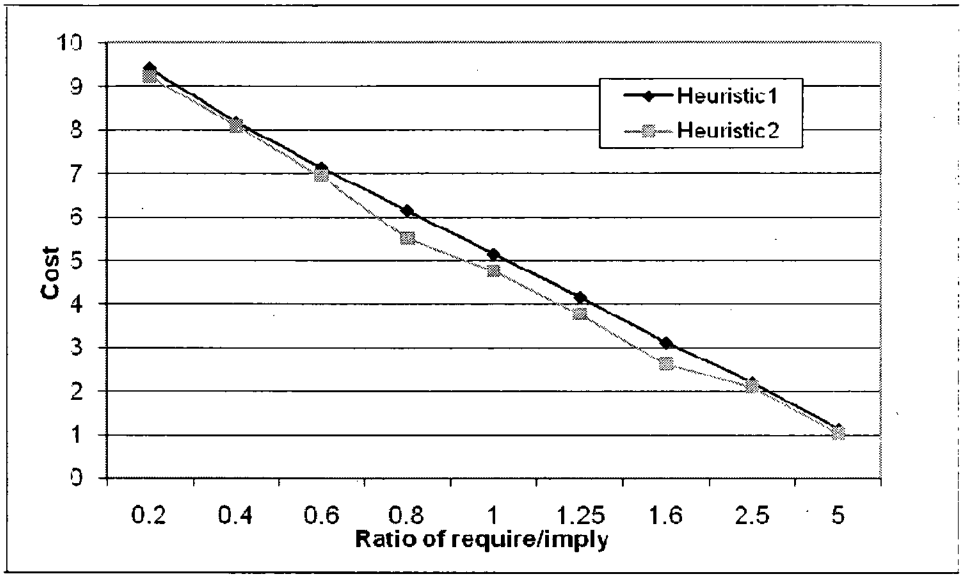

Figure 27: Comparison of the Cost of Solutions of the Heuristic Approaches with Varying Require/Imply Ratio

5.6 Summary

From all the experiment results, we can clearly see that the heuristic approaches can produce a reasonably good solution in significantly less time then exhaustive search. The variation in different aspects of attack graphs has only a minor effect on the performance of our solutions both in terms of time taken and the quality of solutions (that is, the cost).

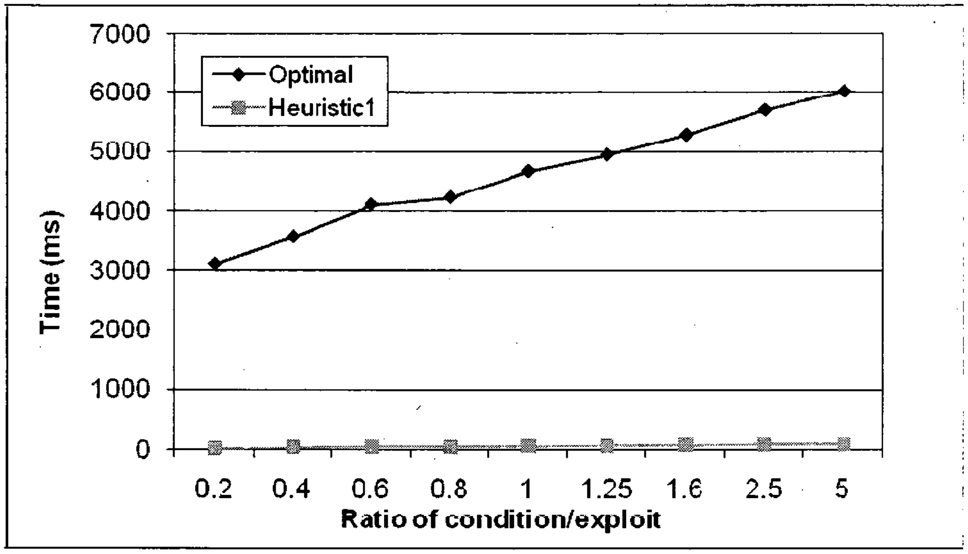

Figure 28: Comparison of Time Taken by the Heuristic Approach 1 and the Optimal Solution with Varying Exploit/Condition Ratio

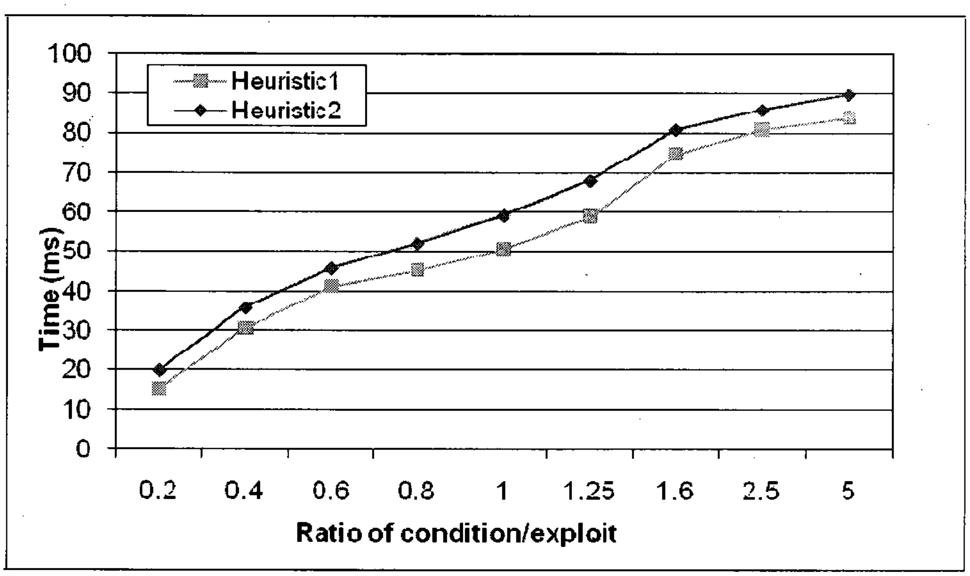

Figure 29: Comparison of Time Taken by the Heuristic Approaches with Varying Exploit/-Condition Ratio

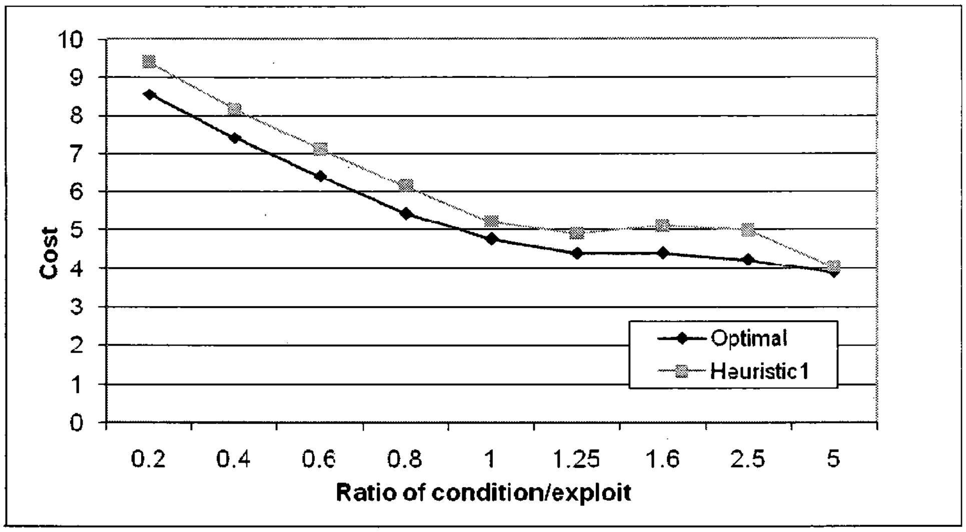

Figure 30: Comparison of the Cost of Solutions of the Heuristic Approach 1 and the Optimal Solution with Varying Exploit/Condition Ratio

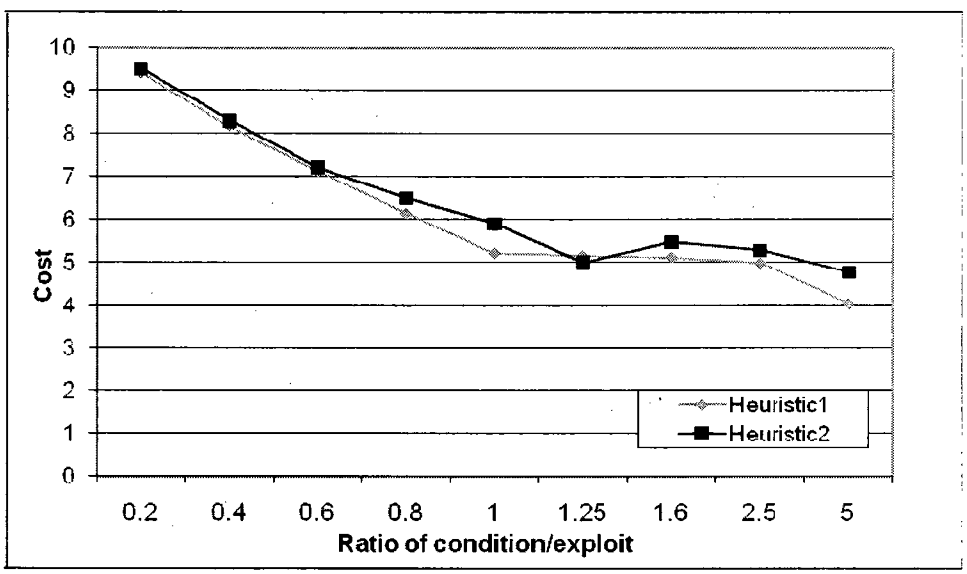

Figure 31: Comparison of the Cost of Solutions of the Heuristic Approaches with Varying Exploit/Condition Ratio

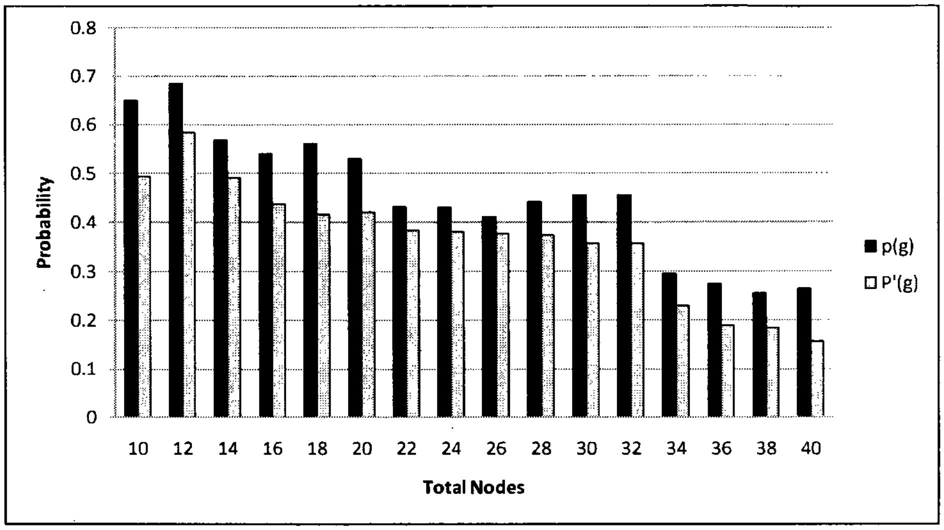

Figure 32: Comparison of Probability of Goal Condition for without and with Partial Network Hardening

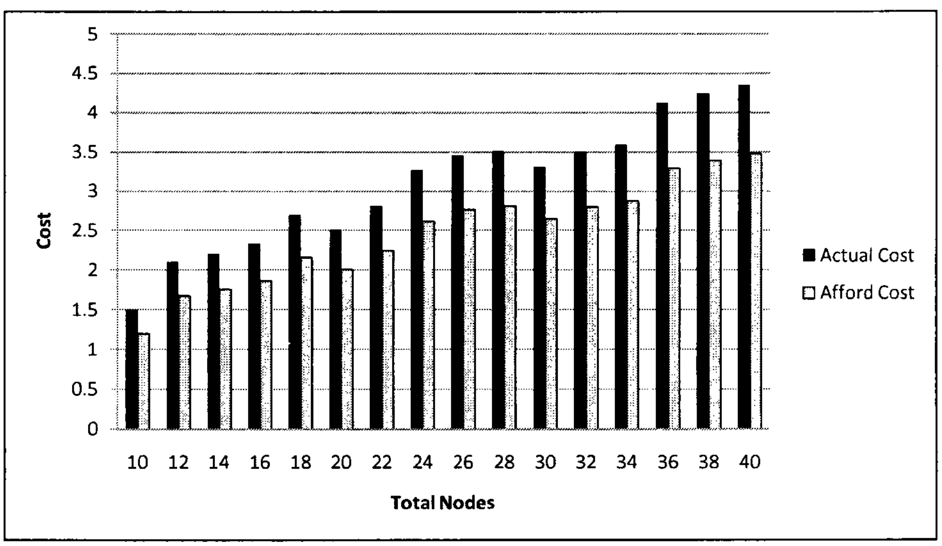

Figure 33: Comparison of Actual Cost and Afford Cost for Partial Network Hardening

Chapter 6

Conclusion

In this thesis, we have pointed out that existing approaches to network hardening by deriving a logic proposition and converting it to its DNF suffer from an exponential time complexity. In this thesis, we have studied heuristic methods for solving this important problem. We evaluated our proposed solutions through extensive simulations. All results have shown that our solution can achieve reasonably good results in significantly less time than the exhaustive searches. For scenarios where additional cost constraints may prevent a full hardening, we have extended our heuristic methods to a partial hardening solution based on probabilistic security metric. Such solutions could provide best possible improvements in terms of security. Our future work include integrating other heuristics to future improve the performance, especially for the case of partial hardening.

Publication

Publication related to this thesis:

- Tania Islam, Lingyu Wang, "A Heuristic Approach to Reducing the Cost of Network Hardening Using Attack Graph," Proc. The 2nd IFIP International Conference on New Technologies, Mobility and Security (NTMS 2008), November 5 - 7, 2008, pages 1-5.

- Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia, "An Attack Graph-Based Probabilistic Security Metric," Proc. 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC 2008) , Springer-Verlag Lecture Notes in Computer Science (LNCS), Vol. 5094, July 13-16, 2008, pages 283-296.

Bibliography

[1] 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 (CCS'02)*, pages 217–224, 2002.

[2] Applied Computer Security Associates. Workshop on. In *Information Security System Scoring and Ranking*, 2001.

[3] D. Balzarotti, M. Monga, and S. Sicari. Assessing the risk of using vulnerable components. In *Proceedings of the 1st Workshop on Quality of Protection*, 2005.

[4] Steve Barker and Gail-Joon Ahn, editors. *Measuring the Overall Security of Network Configurations Using Attack Graphs*, volume 4602 of *Lecture Notes in Computer Science*. Springer, 2007.

[5] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003.

[6] F. Cuppens. Managing alerts in a multi-intrusion detection environment. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC'01)*,

2001.

[7] F. Cuppens and A. Miege. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'02)*, pages 187–200, 2002.

[8] F. Cuppens and R. Ortalo. LAMBDA: A language to model a database for detection of attacks. In *Proceedings of the 3rd International Symposium on Recent Advances in Intrusion Detection (RAID'01)*, pages 197–216, 2001.

[9] Common vulnerability scoring system - SIG. available at: http://www.first.org/cvss/, Accessed May 2008.

[10] Network associates,cybercop scanner. Available at http://www.nss.co.uk/grouptests/va/edition2/nai_cybercop_scanner/nai_cybercop_scanner.htm.

[11] M. Dacier. Towards quantitative evaluation of computer security. Ph.D. Thesis, Institut National Polytechnique de Toulouse, 1994.

[12] M. Dacier, Y. Deswarte, and M. Kaaniche. Quantitative assessment of operational security: Models and tools. Technical Report 96493, 1996.

[13] O. Dain and R.K. Cunningham. Building scenarios from a heterogeneous alert system. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, 2001.

[14] O. Dain and R.K. Cunningham. Fusing a heterogeneous alert stream into scenarios. In *Proceedings of the ACM Workshop on Data Mining for Security Applications*, pages 1–13, 2001.

[15] H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *Proceedings of the 3rd International Symposium on Recent Advances in Intrusion Detection (RAID'01)*, pages 85–103, 2001.

[16] R. Deraison. Nessus scanner, 1999. Available at http://www.nessus.org.

[17] R. Deraison. Nessus scanner, 1999. Available at http://www.nessus.org.

[18] S.T. Eckmann, G. Vigna, and R.A. Kemmerer. STATL: An attack language for state-based intrusion detection. *Journal of Computer Security*, 10(1/2):71–104, 2002.

[19] D. Farmer and E.H. Spafford. The COPS security checker system. In *USENIX Summer*, pages 165–170, 1990.

[20] Dan Farmer and Wietse Venema. Satan:security administrator's tool for analyzing networks, 1995.

[21] Manfred GILLI and Peter WINKER. A review of heuristic optimization methods in econometrics. Swiss Finance Institute Research Paper Series 08-12, Swiss Finance Institute.

[22] Suvajit Gupta and Joel Winstead. Using attack graphs to design systems. *Security & Privacy, IEEE*, 5(4):80–83, July-Aug. 2007.

[23] IBM. IBM tivoli risk manager. Available at http://www.ibm.com/software/tivoli/products/risk-mgr/.

[24] G. Laporte I.H. Osman. Metaheuristics: A bibliography. *Annals of Operations Research*, 63(5):511–623, 1996.

[25] SRI International. Event monitoring enabling responses to anomalous live disturbances (EMERALD). Available at http:// www.sdl.sri.com/projects/emerald/.

[26] System Scanner Internet Security Systems. Internet security systems, system scanner. Available at http://www.iss.net.

[27] S. Jajodia, S. Noel, and B. O'Berry. Topological analysis of network attack vulnerability. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, *Managing Cyber Threats: Issues, Approaches and Challenges*. Kluwer Academic Publisher, 2003.

[28] A. Jaquith. In *Security Metrics: Replacing Fear, Uncertainty, and Doubt*. Addison Wesley, 2007.

[29] S. Jha, O. Sheyner, and J.M. Wing. Two formal analysis of attack graph. In *Proceedings of the 15th Computer Security Foundation Workshop (CSFW'02)*, 2002.

[30] Klaus Julisch and Marc Dacier. Mining intrusion detection alarms for actionable knowledge. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 366–375, 2002.

[31] B. Morin, L. Mé, H. Debar, and M. Ducassé. M2D2: A formal data model for IDS alert correlation. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID'02)*, pages 115–137, 2002.

[32] P. Ning, Y. Cui, and D.S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*, pages 245–254, 2002.

[33] P. Ning and D. Xu. Learning attack strategies from intrusion alerts. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*, 2003.

[34] P. Ning, D. Xu, C.G. Healey, and R.S. Amant. Building attack scenarios through integration of complementary alert correlation methods. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS'04)*, pages 97–111, 2004.

[35] Nmap-network mapper. Available at http://nmap.org/index.html.

[36] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. Efficient minimum-cost network hardening via exploit dependency grpahs. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03)*, 2003.

[37] National Institute of Standards and Technology. Technology assessment: Methods for measuring the level of computer security. NIST Special Publication 500-133, 1985.

[38] R. Ortalo, Y. Deswarte, and M. Kaaniche. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Trans. Software Eng.*, 25(5):633–650, 1999.

[39] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 336–345, 2006.

[40] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. Mulval: a logic-based network security analyzer. In *SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium*, pages 8–8, Berkeley, CA, USA, 2005. USENIX Association.

[41] J. Wing P. Manadhata. Measuring a system's attack surface. Technical Report CMU-CS-04-102, 2004.

[42] J. Wing P. Manadhata. An attack surface metric. Technical Report CMU-CS-05-155, 2005.

[43] J. Wing P. Manadhata. An attack surface metric. In *First Workshop on Security Metrics (MetriCon)*, 2006.

[44] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup. A weakest-adversary security metric for network configuration security analysis. In *Proceedings of the 2nd ACM workshop on Quality of protection*, pages 31–38, New York, NY, USA, 2006. ACM Press.

[45] Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *NSPW '98: Proceedings of the 1998 workshop on New security paradigms*, pages 71–79, New York, NY, USA, 1998. ACM.

[46] X. Qin and W. Lee. Statistical causality analysis of INFOSEC alert data. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, pages 591–627, 2003.

[47] X. Qin and W. Lee. Discovering novel attack strategies from INFOSEC alerts. In *Proceedings of the 9th European Symposium on Research in Computer Security (ESORICS 2004)*, pages 439–456, 2004.

[48] A. R. Chinchani andIyer, H. Ngo, and S. Upadhyay. Towards a theory of insider threat assessment. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks (DSN'05)*, 2005.

[49] I. Ray and N. Poolsappasit. Using attack trees to identify malicious attacks from authorized insiders. In *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS'05)*, 2005.

[50] R. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. In *Proceedings of the 2000 IEEE Symposium on Research on Security and Privacy (S&P'00)*, pages 156–165, 2000.

[51] R. Ritchey, B. O'Berry, and S. Noel. Representing TCP/IP connectivity for topological analysis of network security. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC'02)*, page 25, 2002.

[52] M. Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 1999 USENIX LISA Conference*, pages 229–238, 1999.

[53] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'02)*, pages 273–284, 2002.

[54] AT&T Labs. Graphviz Graph Visualization Software. Graphviz - graph visualization software. Available at http://www.graphviz.org/.

[55] S. Staniford, J.A. Hoagland, and J.M. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1/2):105–136, 2002.

[56] P. Stephenson. Using formal methods for forensic analysis of intrusion events- - a preliminary examination. white paper. available at http://www.imfgroup.com/Document Library. html.

[57] M. Swanson, N. Bartol, J. Sabato, J. Hash, and L. Graffo. Security metrics guide for information technology systems. NIST Special Publication 800-55, 2003.

[58] Steven J. Templeton and Karl Levitt. A requires/provides model for computer attacks. In *NSPW '00: Proceedings of the 2000 workshop on New security paradigms*, pages 31–38, New York, NY, USA, 2000. ACM.

[59] A. Valdes and K. Skinner. Probabilistic alert correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 54–68, 2001.

[60] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. In *DBSEC'08: Proc. 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2008.

[61] L. Wang, S. Noel, and S. Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, 11 2006.

[62] L. Wang, A. Singhal, and S. Jajodia. Toward measuring network security using attack graphs. In *QoP '07: Proceedings of the 2007 ACM workshop on Quality of protection*, pages 49–54, 2007.

[63] D. Xu and P. Ning. Alert correlation through triggering events and common resources. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, pages 360–369, 2004.

[64] D. Xu and P. Ning. Privacy-preserving alert correlation: A concept hierarchy based approach. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC'05)*, 2005.

[65] Y. Zhai, P. Ning, P. Iyer, and D. Reeves. Reasoning about complementary intrusion evidence. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, pages 39–48, 2004.