A New General Graph-based Model for Non-Monotonic Protection Systems

Mohammad Ebrahim Rafiei[1], Hamid Mousavi[1],
Hamid Reza Shahriari[2], Reza Sadoddin[1], Rasool Jalili[3].
Network Security Center, Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran.
[1]{rafiei, h_mousavi, saededdi}@ce.sharif.edu
[2]shahriari@mehr.sharif.edu
[3]jalili@sharif.edu

Abstract: *One of the most challenging problems in security is the safety problem in which we should determine whether a subject can gain access to an object or not. Many approaches have been proposed to address this problem. Nevertheless, most of them lack the ability to model real-world systems or suffer from efficiency problems. In this paper, we propose a general graph-based protection system. In addition to monotonic rules, both non-monotonic rules and rules which may check for absence of rights as their preconditions are included in our model. Moreover, broad range of vulnerabilities including most of DoS vulnerabilities can be modeled via these general rules easily. It is proved that the safety problem in general form of our proposed model is NP-Complete. However, we introduce some simplified cases of the model, such as monotonically increasing systems and systems which contain only permanent rules, in which the safety problem can be answered in polynomial time.*

Keywords: Safety Problem, Protection System, Vulnerability Analysis, Access Control Models, NP-Completeness, 3-SAT Problem.

1. Introduction.

Safety is one of the most important problems in many systems especially computer-based ones. Determining if a system is in a safe state or not is usually considered as the *safety problem*. Safety problem is important mainly because of its extensive use in many security issues such as access control systems, intrusion detection systems and etc. Another important issue, that safety problem can also help in, is the vulnerability analysis. A specific failure of the controls is called *vulnerability* which may allow unauthorized access to attackers [1]. Thus, *vulnerability analysis* deals with specifying, designing, and implementing a computer system without vulnerabilities, discovering unknown vulnerabilities and detecting possible exploits of vulnerabilities. To solve the safety problem, usually a model is needed which is called *Protection System*. Knowing all of these, we can define the *safety problem* more formally as the following question: "Given an initial configuration of a protection system, whether subject *s* can obtain access right *r* on object *o* or not." In fact, we desire to prevent an unwanted or unreliable subject to achieve a right which it did not already have [2].

As already mentioned, the first step to solve the safety problem is to construct a protection system. Ideally, the proposed protection system should address all aspects of real-world systems with some reasonable assumptions and the safety problem also should be solvable in a reasonable time. Till now, some protection systems were proposed [1][2][3][4][5][6][7][8], however simplifications and restrictive assumptions have made most of these models far from real systems.

In almost all of these models, there are only some limited rules. Many of them contain only increasing rules that is the rules which can only add rights or other entities to the model. We call these models

monotonic. A few numbers of existing models contains decreasing rules in which only some accesses (or other entities) will be removed. However, there are many cases in which some goals or new access rights will not be achieved unless some other access rights are removed (denial of service attacks are good examples of this.) Thus, some rules are needed which are both increasing and decreasing at the same time. In other word, we need rules which can add to and delete from the model simultaneously. Another important shortcoming of most existing models is the lake of rules which check for non-existence of rights (or other entities) in their pre-condition parts. Some of the existing models are also weak in simulating nodes properties. Many of them can note define a specific property or vulnerability for a node. Attribute-based models are of the few ones which have tried to solve this shortcoming [6]. Based on the knowledge of the authors, none of the previously proposed models contains all of the mentioned properties at the same time. Almost all of them either are not general enough or can not be simplified to more practicable models.

Obviously there is a trade-off between the generality of the model and the complexity of the safety problem in it. This is confirmed in [9] too. Thus, in this paper, we propose a new general graph-based protection system which contains all the mentioned properties at once with the least possible complexity. We also show how our model contributes to analyze some vulnerabilities more easily. It is proved that the safety problem in this general model is an NP-Complete problem. Also, we show how the model can be simplified to some polynomially solvable sub-problems. As an example, it is shown how to solve the safety problem in polynomial time for monotonically increasing systems. The model actually is designed in a way that one can simplify it for his/her own use. It is worthy to note that simplifications in some models, such as HRU [5], make it weak in modeling real system efficiently. However, we try to preserve important properties in simplified models. This may lead to more practicable cases.

The rest of this paper is organized as follows: at first, we summarize related works in section 2. In section 3, we define our protection system and provide the formal definition of the safety problem in the model. The polynomially solvable sub-problems are discussed in section 4. In section 5, we prove the NP-completeness of the general problem. Finally, conclusion and future works are included in section 6.

2. Related Work.

In [5], Harrison *et al.* proved that the safety problem in its general form is undecidable. However, in addition to considering some unnecessary assumptions in their model, the negation of a condition is not allowed too. That is, one can not test for the absence of an access right as a command's (rule's) precondition [1]. Later works showed that in some constrained cases, not only the safety problem is decidable but it can also be answered in polynomial time (and in some cases in linear time.)

Jones *et al.* in [4] proposed Take-Grant model in which the safety problem is decidable in linear time. However, the Take-Grant model is not close enough to the real-world systems [2]. Several applications of this model with some extensions have been explored in [2][3][7][8][10]. Frank and Bishop in [10] presented the notion of *cost* to generate the most likely paths for information and rights transfer. Shapiro in [2] introduced a new access model based on Take-Grant called *diminish-take* including a new fundamental rights-weakening operation called *diminish.* It was shown that safety in diminish-take model is also decidable in linear time. Shahriari *et al.* in [3] extended the Take-Grant model and added some rules, which are standing for the vulnerabilities explosion, to be used in network vulnerability analysis. This way, their model which is called Vulnerability Take-Grant (VTG) would be able to model some vulnerabilities and their exploits such as buffer overflow, weak password, and etc.

Graph-based techniques have been used in several works for designing protection system [3][11][13][14][15][16][17][18]. Polynomial time solutions to safety sub-problems were also proposed in some of them [3][15][16][18]. Nevertheless, most of the previous researches on protection system have been focused only on monotonically increasing systems. In a monotonically increasing system, state of the system can only be changed by adding new access rights. This makes the safety problem much simpler and that is why all the mentioned works have considered monotonically increasing systems. However, considering only monotonically increasing rules in a model makes it weak in dealing with real systems. To overcome this shortcoming, Sandhu and Suri in [19] proposed a formal model called *Non-*

Monotonic Transformation (NMT). They showed how to implement the model in a distributed environment, using the client-server architecture. Later, Ammann and Sandhu used the NMT model to analyze the safety problem in [20], but their model is not general enough since it contains few numbers of rules. In [21], another graph-based non-monotonic model has been proposed in which rules either remove or add graph structure but does not do both simultaneously.

Maybe the only models which have completely considered non-monotonic systems are those in which model checking techniques are used [11][12][14][28]. In this approach, an abstract model of the designated system is usually constructed, and then security constraints are specified formally. Finally, a model checker checks if the model meets the specified security constraints. Ramakrishnan and Sekar in [11] used this approach to analyze a UNIX operating system configuration vulnerabilities. Jha *et al.* followed a similar approach in [12] using automatic generation tools to construct attack graph model.

These proposed methods have the advantage of being independent of the rules' types and their complexities. That is, they can support all kinds of rules in the same way. Nevertheless, the major problem with using model checkers is the scalability problem. At the time of the writing this paper, model checkers are not even able to check a model of middle-size real-world systems in a reasonable time.

Another part of related works contains the theoretical aspects of more general models such as complexity of the related problems to the safety matter. Harrison *et al.* in [5] showed that the safety problem is undecidable in its general form. They also proved that the safety problem will be decidable in a *mono-operational* protection system or in systems in which creation of a node is not permitted. Lipton and Snyder in [22] proved that safety will be decidable if the number of subjects is finite. Other special cases of their model were introduced later which were not only decidable but some of them also had polynomial solutions. Koch *et al.* showed that safety will be decidable in their graph-based model for access control if each rule either deletes or adds graph structure but does not do both [21]. Considering an access control system as a state-transition system, Tripunitara and Li introduced an idea to compare the expressive power of access control models in [23].

3. Model Specification and the Safety Problem.

Harrison *et al.* in [5] proposed a formal specification of a protection system and tried to define a generalized form of the commands (rules). The following specification is inspired by their work and the work done in [3], but the model proposed here is more general in some aspects than theirs. We have also made some relaxations to simplify the model and its related algorithms.

The protection system PS is defined as a triple (G, R, P), where G is the initial modeling graph, R is the set of rules, and P is the set of predicates which specify the security policy. Next few paragraphs are devoted to discuss these three parameters.

For the sake of simplicity, we define edges as triple (*v, u, l*) in the graph G, where v and u are source and destination vertices, respectively, and *l* denotes the edge's associated label. For example, the edge (*v, u, read*) differs from the edge (*v, u, write*). This way, we do not need to deal with labeling issues in the algorithms. Similar to VTG, vertices may have vulnerabilities. To handle this, for each vulnerability *v* associated to node *x*, we use a loop edge on *x* labeled with *v*. Similarly, we can use a loop edge labeled with the nodes' type. Thus, there is no need for another structure to handle nodes' types. For example to show that the type of node *n* is *object*, we use the loop edge (*n*, *n*, *object*). Another simplification is that there is no need to remove a node; instead we can remove all its adjacent edges. We also assume that there is no need to create new nodes. This assumption makes the safety problem decidable in the model (Analogous to Theorem 3 of [5].)

As it can be seen in the previous paragraph, we are trying to eliminate any unnecessary structure from our model, to keep the structure simple, and to deal with different concepts such as vulnerabilities of nodes and type of nodes in the same way. All of these make the model more flexible to be used in more particular protection system.

To define a rule, we need to introduce edge pattern notation. An *edge pattern* is a triple of (*a*, *b*, *t*) where *a* and *b* can be matched any vertices of G, and *t* is a label.

Definition 1. We say edge pattern (*a*, *b*, *t*) will *match* edges (*v*, *u*, *l*) if *l* and *t* are the same labels. In this case, we say *a* and *b* match *v* and *u* respectively.

For example, (*a*, *b*, *r*) matches all edges labeled *r*, or (*a*, *a*, *o*) matches all loop edges labeled *o*. We can

have also star (*) notation in an edge pattern. Edge pattern (*, *a*, *m*) matches some edges if there exists vertex *v* which every other vertex has an edge to *v* labeled *m*. In this case we say *a* match *v*. (*a*, *, *m*) can be defined in a similar way. Accordingly, the general form of the rules is defined as follows:

Definition 2. Every rule in R is defined as a tuple (E_e, E_n, E_a, E_d) where E_e and E_n are two sets of edge patterns that indicate which edges should exist/not exist in order that the rule can be applied, respectively. E_a and E_d are two other set of edge patterns. E_a represents new edges which will be added to graph G after the rule application. On the other hand, E_d indicates the edges which will be removed from graph G after the rule application.

To apply a rule, there should be a match for all edge patterns in its E_e and there should not be any match for any edge patterns in its E_n. If this is the case, the edges produced by patterns in E_a and E_d respecting to the matches found for E_e, will be added to and removed from G, respectively.

This definition is more general than the one proposed in [5]. This is mainly because of that the latter one considers the patterns set E_n which addresses the edges which should not exist as a precondition of the rule. This makes our protection system capable of analyzing new types of attacks.

For example, consider a system running an Intrusion Detection System (IDS) to monitor activities of some entities. Also assume that the attacker *A* can gain write access to a given object if it is not being monitored by IDS, Fig. 1. As shown in Fig. 1, the system can be represented in our model by a new vertex in the graph in place of the IDS alongside with its associated edges, labeled m, to the nodes which it is monitoring. Adding the following rule can show how an attacker may reach her goal formally:

$$R = (\varnothing, \{(*, o, m)\}, \{(a, o, w)\}, \varnothing)$$

The new rule implies that if the IDS is not monitoring a specific object, called *o*, then the attacker can gain write access to *o*.

Based on the knowledge of authors, none of the existing models of protection systems proposed before are able to model such an exploit, whereas too many of such exploits can be addressed in the real world. The last part of our protection system is the set P which defines the safety problem. The safety problem was defined informally before. Here, we provide the formal definition.

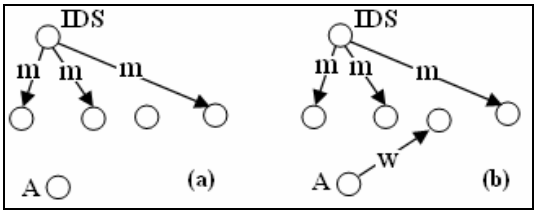

Fig. 1. A simple IDS model. a) Before applying the IDS rule. b) After applying the IDS rule.

Definition 3. Having a protection system (G, R, P), a *witness* is a sequence of rules, $r_1, r_2 \ldots r_n$, ($r_i \in R$, $1 \leq i \leq n$) which are applicable to the initial modeling graph respectively.

Definition 4. Let *r* be an access right and, *A* and *B* be two distinct vertices in the modeling graph G. Predicate can●share(*A, B, r*, G) is true if and only if there is a witness whose application to modeling graph G generates a new graph with an edge from *A* to *B* labeled *r*.

Security policy of the protection system PS is specified by predicates of P which may be violated by applying a witness. We define P to be a set of can●share predicates. Note that one can add other predicates to this set, but for our intention in this paper, can●share is enough. Completing the definition of our protection model, we can define the Safety Problem as what follows:

Definition 5. Having the protection system PS = (G, R, P), the *Safety Problem* is the problem of finding witness *w* whose application to graph G violates at least one of the predicates included in P.

Obviously, there are other predicates which can be included in set P. However in our general model, they are not required since we can answer most of the other predicates by answering the basic predicate can●share. As an example, we show how to answer can●revoke predicate using can●share. The predicate can●revoke can be defined as follows:

Definition 6. Let *A* and *B* be two distinct vertices in the modeling graph G and there is an edge between *A* and *B* labeled *r*. Predicate can●revoke(*A*, *B*, *r*, G) is true if and only if there is a witness whose application to G generates a new graph in which there is not any edge from *A* to *B* labeled ***r*.**

To answer this predicate, we can add a new rule with following arguments to the set of rules R:

$$E_e = \varnothing, E_n = \{(A, B, r)\},$$

$E_a = \{(A, B, r')\}, E_d = \varnothing$

Provided that r' has not been used in any other rules, predicate can●share(A, B, r', G) will not be satisfied without using the new rule, which involves non-existence of an edge labeled r between A and B. Thus, predicates can●revoke(A, B, r, G) and can●share(A, B, r', G) can be used interchangeably. Despite all of these, one may add its own defined predicates to the security policy P. obviously in special cases of the model, can●share can not describe the whole policy and it is obligatory to add some new predicate to it.

4. Polynomially Solvable Sub-Problems.

In this section, to show the flexibility of the proposed model and its capability to model special systems with less cost, we provide some instances of the safety problem which can be solved in polynomial time. Let's call the rules which can decrease access rights from the modeling graph in the protection system *decreasing rules*. Initially, we can divide decreasing rules into two main classes; the rules which do not add new access rights to the model, and those which add new access rights in addition to removing some other access rights. Later we will show that we can eliminate the former rules in some cases.

Definition 7. *Monotonically decreasing* rules are rules which only may decrease access rights, that is, $E_a = \varnothing$.

Most of DoS attacks [24] and also DDoS attacks [25] are clear examples of monotonically decreasing rules, which may cause some services not to be accessible, without adding new access rights to the system. For example, consider the SynFlood attack which exploits a vulnerability in the TCP protocol. In a TCP SYN flooding DoS attack, an attacker sends out many SYN messages with forged IP addresses (this type of attack is called spoofing). The server replies with SYN/ACK messages, but the attacker never acknowledges these messages, thereby leaving many half-open connections on the server. The intruder can continue sending SYN messages until the server reaches its half-open-connection limit and can not respond to any new incoming requests. Thus in the next steps of its attack scenario, the attacker can easily forge himself to be the node, which is attacked, using its IP address and its flooded port number.

Definition 8. Simple rules are the rules in which $E_n = \Phi$ and each of the sets E_e, E_a, and E_d contains O(1) number of edges pattern.

```
1. Let list F be the set of all edge of the
   modeling graph P.G.
2. while (! isEmpty(F))
3. e = head (F)
4. Check for all applicable monotonically
   increasing rules which e is involved in.
5. foreach (resulting edge like f)
6.     add f to P.G
7.     if (f has not been in F before)
8.           Add f to the list F
9. Delete e from F
10. return P
```

Fig. 2. *Gen_Closure_4MIR*: A polynomial time algorithm which answers can●share predicate when the protection system contains only simple and monotonically increasing rules.

In other words, applying a simple rule just involves checking existence of some edges. The second condition makes the application of simple rules take O(1) time. Recall that in its general form, application of a rule may require the graph not to include some edges. Almost all previously proposed graph-based models are restricted to simple rules. Whereas, there are some kinds of exploits in which we have to be sure that some edges do not exist in the modeling graph such as the IDS exploit example described in section 3.

Theorem 1. In a protection system PS = (G, R, P), the predicate can●share can be answered in polynomial time, if R contains only rules that are both simple and monotonically increasing.

Proof. An algorithm like the one proposed by Frank and Bishop in [10] will answer can●share as shown in Fig. 2. The proof of polynomial time complexity is similar to theirs as well. ■

Theorem 2. In a protection system, the predicate can●share can be answered in polynomial time, if R contains only simple rules, and the rules which decrease access rights act monotonically.

Proof. We prove that monotonically decreasing rules are useless in satisfying predicate can●share. We show that monotonically decreasing rules can not cause to new edges to be added to the modeling graph. First of all, a decreasing rule may not add new edges directly. Thus, the only possibility is that a monotonically decreasing rule causes to addition of new edges indirectly. This may occur when the precondition of another increasing rule becomes true. However, monotonically decreasing rules can only

affect the second part of the other rules' conditions which will not be considered in the simple rules at all.

Therefore in such situations, no new edge will be added because of an edge removal and we can simply eliminate all monotonically decreasing rules and use the closure-based algorithm described above to answer the predicate can●share in polynomial time.■

Corollary 1. In a similar way it can be shown that if all defined rules were simple, then monotonically decreasing rules could be removed from the protection system. But this time, since there still may exist some decreasing rules in the model, we can not use the same closure-based algorithm to answer the safety problem.

Definition 8. An edge in a modeling graph is *permanent*, if and only if its associated access right will never be removed because of deletion of any other edges.

The definition implies that no matter whether the conditions which have caused adding a permanent edge still hold or not, it will continue to exist in the graph permanently. As an example of this type of access right, suppose the attacker *A* wants to use the passwords stored in a file *f* on a host in its attack scenario. As soon as *A* achieves read access to *f*, it has reached its goal. Even encrypting the password file *f*, will not hide the achieved information from *A*, because *A* has already read what it wanted. Thus, the read access is permanent. Therefore, the only way to removing a permanent edge is to delete it directly by a rule.

We will refer to edges which are not permanent as *impermanent* edges**.** As an example, consider that an attacker wants to use a service which needs authentication. Suppose the attacker has acquired the information of an account for the service. The attacker can use the service as long as the promised account has not been disabled. Therefore, the attacker's access to the service is impermanent.

Definition 9. A rule is *permanent* if it generates only permanent edges; otherwise it is *impermanent*.

Next theorem deals with an interesting property of the systems which use only permanent and simple rules:

Theorem 3. Let that PS = (G, R, P) is a protection system in which all the initial access rights are permanent and only the permanent and simple rules are allowed. In such a system, the predicate can●share can be answered in polynomial time.

Proof. The main idea is to construct the closure of the modeling graph using a conflict graph. The *conflict graph* has one vertex in association with every possible edge in the modeling graph and is initially empty. We say that two edges in the closure have conflict (according to the conflict graph) if and only if there exists a directed path between their related vertices in the conflict graph. The rule R = (E_e, Φ , E_a, E_d) is applicable according to the conflict graph if and only if the match found for E_e contains non-conflicting edges according to the conflict graph.

Initially the closure is identical to the graph G. In each step of the algorithm, we will apply an applicable rule according to the conflict graph and update two graphs (closure and conflict graphs) as follows; Let R = (E_e, $\varnothing$, E_a, E_d) be the selected rule. To apply R, the edges produced by E_a will be added to the closure and all their conflicts will be removed from the conflict graph (That is, for each edge *e* produced by E_a, we will remove all the edges outgoing from *e*'s related vertex in the conflict graph). The edges included in E_d (obviously if there exists any) will not be removed from the closure; instead we will add directed edges from E_d's associated vertices in the conflict graph to those of E_a.

We will repeat this step until there is no applicable rule according to the conflict graph. Thus, if there is an edge from vertex *A* to vertex *B* labeled *r* in the computed closure, the answer to the predicate can●share(*A*, *B*, *r*, G) will be yes. Obviously, the algorithm is polynomial because in each step at least one edge will be added to the closure. (Note that there is no need to consider monotonically decreasing rules according to corollary 1.) Since the closure contains polynomial number of edges and each step of the algorithm needs polynomial time, the time complexity of the algorithm will be polynomial too.■

5. NP-Completeness Results.

To show the NP-Completeness of the general problem we use reduction from 3-SAT [26]. 3-SAT is itself a special case of Boolean satisfiability (SAT) problem. SAT is the first established NP-complete problem [27] and many typical NP-Complete problems can be directly reduced from it.

Basically, SAT problem is either to find a satisfying truth assignment of all variables or to prove there is no satisfying assignment for a given Boolean formula (usually in Conjunctive Normal Form

(CNF).) 3-SAT is a special case of SAT in which each clause has exactly three literals.

Theorem 4. The safety problem in the protection system PS = (G, R, P) is NP-complete if the initial edges and rules are not necessarily *permanent*.

Proof. As we mentioned**,** the safety problem can be interpreted as finding a witness which makes one or more predicates in P true. Therefore, it is clear that if we have a witness, it is possible to verify whether the witness violates security policy or not in polynomial time. That is, the problem is in NP. Note that since a witness includes only different rules it size is polynomial. To prove NP-Completeness, we use reduction from 3-SAT. Let φ be an instance of 3-SAT problem. We construct protection system PS = (G, R, P) such that φ is satisfiable if and only if there exists a witness in PS which violates at least one of the predicates included in P.

Corresponding to each clause C_i in φ, G has a node C_i. There are three vertices y_i, x_i and $\bar{x}_i$ in G, corresponding to each literal x_i in 3-SAT problem. Graph G also contains two other vertices *T* and *T'*. *T* has a specific type called *m* which is shown by a loop labeled *m* on it. For each literal x_i, we add directed edges from y_i to both x_i and $\bar{x}_i$ labeled *r*. If the *i*th clause contains x_j ($\bar{x}_j$), we will put a directed edge from C_i to x_j ($\bar{x}_j$) labeled *s* and *true*, and another directed edge to $\bar{x}_j$ (x_j) labeled *false*. For each *i*, there is an edge from y_i to *T* labeled *notok*. This means that yet we do not know whether x_i has consistent values in all of its occurrences. The initial graph which is constructed from a sample 3-SAT problem is shown in Fig. 3.

The rule set R contains 5 rules:

1. If clause *c* contains literal *x* (a variable or its negation), we will assign the *true* value to *x* and *false* to its negation:

$$E_e = \{(c, x, s), (y, x, r), (y, x', r), (c, x, false)\},$$
$$E_n = \varnothing,$$
$$E_a = \{(c, x, true), (c, x', false)\},$$
$$E_d = \{(c, x, false), (c, x', true)\}$$

2. If clause *c* contains literals x_1 and x_2 in which both have the *true* value, we can assign the *false* value to one of them and *true* value to its negation:

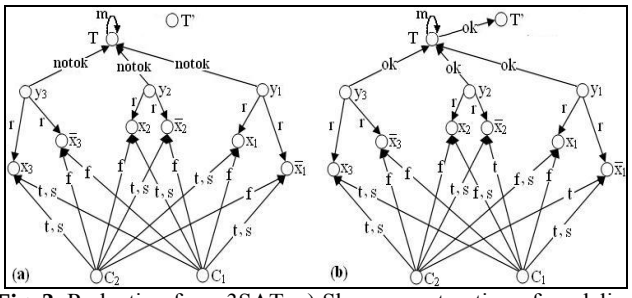

Fig. 3. Reduction from 3SAT. a) Shows construction of modeling graph for formula $f = (!x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee !x_2 \vee x_3)$. b) Shows a scenario which violate security and also is an answer to 3SAT problem ($x_1 = x_2 =$ false and $x_3 =$ true.)

$$E_e = \{(c, x_1, s), (y, x_1, r), (y, x'_1, r), (c, x_1, true), (c, x_2, s), (c, x_2, true)\},$$
$$E_n = \varnothing,$$
$$E_a = \{(c, x_1, false), (c, x'_1, true)\},$$
$$E_d = \{(c, x_1, true), (c, x'_1, false)\}$$

3. If literal *x* has got true value in all clauses which contain *x*, a new edge can be added to the modeling graph G from *y* to *T* labeled *ok*, where *y* is the vertex having an edge to *x* labeled *r*:

$$E_e = \{(y, x, r)\}, E_n = \{(a, x, false)\},$$
$$E_a = \{(y, T, ok)\}, E_d = \{(y, T, notok)\}$$

4. Just like the previous rule for *x*'s false value:

$$E_e = \{(y, x, r)\}, E_n = \{(a, x, true)\},$$
$$E_a = \{(y, T, ok)\}, E_d = \{(y, T, notok)\}$$

5. If all literal get consistent values in their related clauses, a new edge can be added from *T* to *T'* labeled *ok* which shows that φ is satisfiable (*m* is the type of node *T*):

$$E_e = \{(T, T, m)\}, E_n = \{(a, T, notok)\},$$
$$E_a = \{(T, T', ok)\}, E_d = \varnothing$$

The construction will be completed by defining the set P in protection system PS to contain just the predicate can•share(*T, T', ok,* G). It is not so hard to show that if there exists a satisfying truth assignment for φ, there will be a witness in SP which violates the security policy and vice versa. ■

6. Conclusions and Future Works.

In this paper, a general graph-based protection system was proposed. Although we showed that the safety problem in its general form in the proposed model is NP-Complete, but there are still cases in which the safety problem can be answered in

polynomial time. We introduced ideas of simple, permanent, and monotonically decreasing rules alongside with some simplified sub-problems of the safety problem. We also provided polynomial algorithms to solve these problems. We showed how the monotonically decreasing rules can be eliminated when a system contains only simple rules. We also showed that the safety problem can be answered polynomially in a system which is restricted to rules which are both permanent and simple.

Investigating more polynomially solvable problems can be considered as an interesting future work. Especially, relevant problems to impermanent rules are worthy of considering more thoroughly. In addition, our main goal in this paper was to propose the general model itself, but the generality of the proposed model may make it impracticable to be used in the real world directly. Thus, some adjustments should be made on the model to simplify it for one's specific application while preserving its useful properties. Simplifying the model by bounding the number of access rights, rules, vulnerabilities of the nodes, and their types is yet another area of research.

References

[1] M. Bishop. "Computer Security: The Art and Science," Addison-Wesley, 2003.

[2] J.S. Shapiro. "The practical application of a decidable access control model", Technical Report SRL-2003-04, John Hopkins University, 2003.

[3] H.R. Shahriari, R. Sadoddin, R. Jalili, R. Zakeri, and A.R. Omidian. "Network Vulnerability Analysis through Vulnerability Take-Grant Model (VTG)," To appear in Proceedings of 7th International Conference on Information and Communications Security (ICICS'05).

[4] A.K. Jones, R.J. Lipton, and L. Snyder. "A linear time algorithm for deciding security," *Proceedings 17th Annual FOCS Conference*, pages 33-41, Houston, October 1976.

[5] M.A. Harrison, W.L. Ruzzo, and J.D. Ullman. "Protection in operating systems," *Communications of the ACM*, 19(8), pages 461–471, August 1976.

[6] X Zhang, Y Li, and D Nalla. "An attribute based access control matrix model," *Proceedings of the 2005 ACM symposium on Applied computing*, 2005.

[7] M. Bishop. "Conspiracy and information flow in the Take-Grant protection model," *Journal of Computer Security*, vol. 4(4), pages 331-360, 1996.

[8] M. Bishop. "Practical Take-Grant Systems: Do They Exist?" Ph.D. Thesis, Purdue University, 1984.

[9] R. Sandhu. "The Typed Access Matrix Model," *IEEE Symposium on Security and Privacy*, pages 122-136, 1992.

[10] J. Frank and M. Bishop. "Extending the Take-Grant protection system," Technical Report, 1996.

[11] C. Ramakrishnan and R. Sekar. "Model-based vulnerability analysis of computer systems," *Proceedings of the 2nd International Workshop on Verification, Model Checking and Abstract Interpretation*, September 1998.

[12] S. Jha, O. Sheyner, and J. Wing. "Two formal analyses of attack graphs," *Proceedings of the 2002 Computer Security Foundations Workshop*, pages 45-59, Nova Scotia, June 2002.

[13] C. Phillips and L. Swiler. "A graph-based system for network-vulnerability analysis," *Proceedings of the New Security Paradigms Workshop*, pages 71-79, Charlottesville, VA, 1998.

[14] R.W. Ritchey and P. Ammann. "Using model checking to analyze network vulnerabilities," *Proceedings of the 2000 IEEE Symposium on Security and Privacy (Oakland 2000)*, pages 156-165, Oakland, CA, May 2000.

[15] P. Ammann, D. Wijesekera, and S. Kaushik. "Scalable, graph-based network vulnerability analysis," *Proceedings of 9th ACM Conference on Computer and Communications Security,* Washington, DC, November 2002.

[16] S. Noel, B. O'Berry, C. Hutchinson, S. Jajodia, L. Keuthan, and A. Nguyen. "Combinatorial analysis of network security," *Proceedings of 16th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, Orlando, Florida, April 2002.

[17] D. Zerkle and K. Levitt. "NetKuang - A multi-host conuration vulnerability checker," *Proceedings of 6th USENIX Security Symposium,* San Jose, California, pages 195-204, July, 1996.

[18] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. "Efficient minimum-cost network hardening via exploit dependency graphs," Proc. *19th Annual Computer Security Applications Conference,* pages 86-95, December 2003.

[19] R.S. Sandhu and C.S. Suri. "Non-monotonic transformation of access rights," *Proceedings of the IEEE Symposium on Security and Privacy,* pages 148-161, 1992.

[20] P.E. Ammann and R.S. Sandhu. "One-representative safety analysis in the non-monotonic transform model," *proceedings of 7th IEEE computer security Foundations Workshop,* Franconia, NH, USA, pages 138-149, June 1994.

[21] M. Koch, L.V. Mancini, and F. Parisi-Presicce. "Decidability of safety in graph-based models for access control," *Proceedings of 7th ESORICS,* volume 2502 of LNCS, pages 229-243, 2002.

[22] R.J. Lipton and L. Snyder. "On synchronization and security," Demillo *et al*., editor, *Fundamental of Secure Computation,* Academic Press, pages 367-385, 1978.

[23] M.V. Tripunitara and N. Li. "Comparing the expressive power of access control models," *Proceedings of 10th ACM Conference on Computer and Communications Security,* Washington, DC, USA, pages 62-71, October 2004.

[24] A.R. Sharafat and M.S. Fallah. "A framework for the analysis of denial of service attacks," *The Computer Journal*, Oxford University Press, Vol. 47, No. 2, pages 147-162, 2004.

[25] J. Mirkovic, and P. Reiher. "A taxonomy of DDoS attack and DDoS defense mechanisms," ACM SIGCOMM Computer Communication Review, 34(2), April 2004.

[26] M.R. Garey, and D.S. Johnson. "Computers and intractability: A guide to the theory of NP-completeness," W.H. Freeman, New York, NY, USA, 1979.

[27] S.A. Cook. "The complexity of theorem proving procedures," *Proceedings of 3rd Annual ACM Symposium on the Theory of Computing,* New York, pages 151-158, 1971.

[28] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. "Automated generation and analysis of attack graphs," *Proceedings of the 2002 IEEE Symposium on Security and Privacy (Oakland 2002)*, Oakland, CA, May 2002.