A Mathematical Framework to Assess the Security of an Information Infrastructure

F.Baiardi, S.Suin, C.Telmon

Dipartimento di Informatica, Università di Pisa
L.go B.Pontecorvo 3, 56125 - PISA
{f.baiardi, s.suin} @unipi.it, claudio@next-hop.it

1 Introduction

We propose a mathematical framework to assess the security of an information infrastructure focused on the analysis of rights acquired by a threat through attack sequences. The framework supports distinct abstraction levels to focus the assessment on the most critical components.

The most critical problem to define the framework is how to deduce all the rights acquired by a threat through a successful attack. These rights depend upon the relation among infrastructure components. As an example, a successful attack to control a component enables the control of any component that trusts the attacked one. To describe the relation, we model components as objects, each defining of a set of methods invoked by either a user or other components and describe relations among components through a labelled hypergraph where a hyperarc denotes a dependency of the destination node from the source ones. Relations that are considered concern the ability of invoking, controlling or managing a method as well as that of determing a security attribute of a component. The hypergraph is the input of a logic program to compute the attack sequences that may be implemented against the infrastructure. Lastly, the framework may be used to define an optimal ordering to remove vulnerabilities and some programming tools can be defined to support the assessment.

2 Infrastructure Hypergraph

The framework describes the infrastructure as a set of related components. A component consists of some internal state and methods, each an operation the component implements. Legal users own the rights to invoke some methods while threats are interested in gaining some rights according to their goal. In the following, both legal users and threats will be denoted simply as user. Each user is paired with a set of rights, a set of pairs that defines the methods it can invoke. Some of these rights may have been achieved through a sequence of attacks. For each infrastructure component, we determine a set of methods such that anyone that has the rights of invoking all the methods in the set, can control some security attribute of the component. A further relation among the components describes how a security attribute of a component depends upon those of other components. Taking into accounts the rights of a user U, i.e. the methods U can invoke, and the relations among methods and the security attributes of components we compute the transitive closure of the rights of U. This closure includes any attribute of a component that U can control because of its rights. Assume, as an

example, that a method M of C determines the integrity of a component C and that the confidentiality of a distinct component D depends upon the integrity of C. Hence, any user that can invoke M controls the integrity of D. In another case of interest, a method M updates the state of a component ACL used to grants or revokes the right of invoking F. Obviously, if U can invoke M, then it can also grant or revoke the right of invoking F. Here, any user that can invoke M manages, i.e. controls the availability of F. In the most general case, the hypergraph includes three kinds of nodes that describe, respectively, users, components and methods. The following kinds of hyperarcs are introduced:

1. right hyperarc: from a user node to a non empty set of method nodes. It is not labelled and it describes a method a user can invoke;
2. source hyperarc: from a set of method nodes to a component node. It is labelled by one attribute of the component and it describes the methods that control the corresponding attribute of the component;
3. component hyperarc: from a set of component nodes to a component node. It is labelled by one attribute for each source component and one attribute for the destination one. It describes a dependence of the attribute in the destination component from those of the source ones;
4. destination hyperarc: from a set of components nodes to a method node. It is labelled by one attribute for each source node and by an attribute for the destination one. It describes a dependency of a method from a set of components.

To compute a transitive closure, rights are propagated through the hyperarcs.

3 Vulnerability and Attacks

The framework characterizes an attack in terms of the vulnerabilities that enable it, the resources and the rights it requires and the rights achieved due to its success. If A is an attack, V(A) is the set of the component vulnerabilities that enables A and C(A) is component that is the target of A. A user U can execute A if it can access all the resources in R(A), i.e. information, tools and know how about A. Furthermore, U can attempt A only if it satisfies pre(A), the precondition of A. Pre(A) is a set of rights and U satisfies pre(A) if it owns all the rights in pre(A). If U owns all the resource in R(A) and satisfies pre(A) then it can execute A and, if A is successful, U will acquire the set of rights post(A).

Initially, U owns the rights in Init(U) and it executes sequences of attacks. U can execute the sequence A1...An if it owns the resources for any Ai $1 \leq i \leq n$ and if, after executing the sequence A1 ... Aj, it satisfies pre(Aj+i). Hence, the transitive closure of Init(T) includes pre(A1) and, after any Ai, $i \in 1..n-1$, the transitive closure of $Init(T) \cup post(A1) ... \cup post(Ai)$ includes pre(Ai+1). Each sequence satisfying these conditions is **feasible** for U. For simplicity sake, in the following we neglect the dependency of feasible sequences from the initial set of rights.

Since U is rational, it executes only those feasible sequences that enable it to achieve one of its goals. Each goals of U is a set of rights and U achieves a goal when and if it owns the corresponding rights. A feasible sequence that enables U to achieve one of its goals is an **evolution useful** for T. Useful evolutions depend upon:

1. the infrastructure hypergraph,

2. the vulnerabilities in the infrastructure components and the attacks they enable,
3. the attacks U can implement and its goals.

4 Security Evolutions of the Infrastructure

Security evolutions describe how the n users in a set SU can achieve n goals, i.e. sets of rights, SR1, ..., SRn, through a sequence of attacks SA = A1 ... Am, where each attack has been executed by just one user. To define evolutions, we define the projection PR(SA, U) of the sequence SA onto a user U. PR(SA, U) includes the subsequence of SA with the attacks implemented by U. This is a subsequence useful for U. If PR(SA, U) is empty, U is not involved in the evolution due to SA. Since each attack in SA is implemented by a user in SU as a step to reach a goal, there is a set $\{U1, ..., Uk\} \subseteq SU$ and a corresponding set of projections $\{PR(SA, U1), ..., PR(SA, Uk)\}$ where

1. each attack in SA belongs to one projection
2. distinct projections are disjoint,
3. after the execution of attacks in PR(SA, Ui), Ui achieve its goal SRi.

Hence, each evolution results from the interleaving of useful sequences for the considered users. Two equivalent evolutions result in the same rights for any user in the considered set. Since user rights are never revoked, we can model monotonic evolutions only, where the set of user rights never decreases. Given a set of user, each with a goal, they can achieve their goals iff there is at least one corresponding evolution. Evolutions can be computed taking into account the infrastructure hypergraph and useful attacks sequences for the users.

5 Ranking of Vulnerabilities

The proposed framework may exploit alternative metrics to rank vulnerabilities according to the attack sequences they enable, the impact of these attacks and so on. Here we present a metric based upon the smallest sets of countermeasures to prevent any user from achieving its goals or, alternatively, to stop all evolutions. A countermeasure for a vulnerability V is any strategy that removes V, i.e. it makes an attack that exploits V ineffective. In the simplest case, a patch that removes V is applied. Other countermeasures may change the dependencies among components so that even if a user owns a set of rights, it cannot control or manage a component. A set of countermeasures is complete if after applying its countermeasures, no evolution is possible so that no user can achieve any of its goals. A set of countermeasures is minimal if it is complete and none of its subsets is complete. Minimal sets define the smallest sets of countermeasures to be applied to stop all the evolutions

To rank a vulnerability V, we consider the percentage of minimal sets with a countermeasure that removes V. If no minimal set removes V, then evolutions can be stopped even if V is not removed. On the other hand, if any minimal set includes a countermeasure that removes V, then the only way to stop evolutions is to remove V. The percentage of minimal sets removing V estimates how critical V is for the infrastructure security.

Till now we have assumed that any two goals of any user are equivalent. Instead, in several cases, each goal may be paired with a weight proportional to its impact, i.e.

to the corresponding loss for the infrastructure owner. In the same way, we can define a minimal set in terms of the cost of the countermeasures rather than of their number.

The framework has been implemented through a logic program that computes

1. the sets of rights each user may achieve
2. alternative evolutions due to a set of users
3. minimal sets of countermeasures
4. the set of rights a user may achieve after applying a set of countermeasures.

References

1. P.Ammann, D.Wijesekera, S. Kaushik, *Scalable, Graph-based Network Vulnerability Analysis*, Proc. of the 9th ACM conference on Computer and communications security, November 18-22, 2002, Washington, DC, USA
2. F.Cuppens, A. Mie'ge, *Alert Correlation in a Cooperative Intrusion Detection Framework*,IEEE Symp. on Security and Privacy, p.202, May 12-15, 2002
3. J. Dawkins, C. Campbell, J. Hale, *Modeling Network Attacks: Extending the Attack Tree Paradigm*, Workshop on Statistical and Machine Learning in Computer Intrusion Detection, Johns Hopkins University, June 2002.
4. R. P. Goldman, W. Heimerdinger, and S. A. Harp. *Information Modeling for Intrusion Report Aggregation*,DARPA Information Survivability Conference and Exposition (DISCEXII), June 2001.
5. S. Jajodia, S. Noel, B. O'Berry, *Topological Analysis of Network Attack Vulnerability*, in *Managing Cyber Threats: Issues, Approaches and Challenges*, V. Kumar, J. Srivastava, A. Lazarevic (eds.), Kluwer Academic Publisher, 2003.
6. R. A. Martin, *Managing vulnerabilities in networked system*, IEEE Computer, November 2001. p. 32 - 38.
7. P. Moore, R. J. Ellison, R. C. Linger,*Attack modelling for information security and survivability*, CMU/SEI- 2001-TN001.
8. P. Ning , P.Cui , D. S. Reeves, *Constructing attack scenarios through correlation of intrusion alerts*, 9th ACM Conference on Computer and Communications security, Nov. 2002, Washington, DC, USA.
9. S. Jha, O. Sheyner , J. Wing, *Two Formal Analysis of Attack Graphs*, 15th IEEE Computer Security Foundations Workshop , p.49, June 2002.
10. C. Phillips, L. Painton Swiler, *A graph-based system for network-vulnerability analysis*,Workshop on New Security Paradigms, p.71-79, Sept.1998.
11. R. Ritchey, B. O'Berry, S. Noel, *Representing TCP/IP Connectivity For Topological Analysis of Network Security*, Proc. of the 18th Annual Computer Security Applications Conference, p.25, Dec. 2002.
12. O. Sheyner, J. Haines, S. Jha , R. Lippmann, J. M. Wing, *Automated Generation and Analysis of Attack Graphs*, Proc. of the 2002 IEEE Symposium on Security and Privacy, May 12-15, 2002 .
13. O. M. Sheyner, *Scenario Graphs and Attack Graphs*, CMU-CS-04-122,2004.
14. L.P. Swiler, C. Phillips, D. Ellis, S. Chakerian, *Computer-Attack Graph Generation Tool.* Proc. of the DARPA Information Survivability Conference, June 2001.
15. V.Swarup, S.Jajodia, J.Pamula, *Rule-Based Topological Vulnerability Analysis*, Proc. of MMM-ACNS 2005, Sept. 2005.