Vulnerability Assessment by Learning Attack Specifications in Graphs

Virginia N. L. Franqueira*
University of Twente
Dept. of Computer Science, IS Group
Enschede, The Netherlands
franqueirav@ewi.utwente.nl

Raul H. C. Lopes
Brunel University
School of Engineering and Design
London, England
raul.lopes@brunel.ac.uk

Abstract

This paper presents an evolutionary approach for learning attack specifications that describe attack scenarios. The objective is to find vulnerabilities in computer networks which minimise the cost of an attack with maximum impact. Although we focus on Insider Threat, the proposed approach applies to networks in general, including social networks and computer grid.

Keywords: Network Security, DoS, Intrusion Prevention

1. Introduction

Assuring security in organisations involves both detecting intrusions and preventing attacks. Attacks, which become intrusions if their goals are reached, can be initiated either by insiders, i.e. entities with legitimate access and privileges over the organisational LAN and resources, or outsiders, the opposite situation. Surveys (e.g. [11]) and studies (e.g. [8]) have shown that Insider Threat, i.e. activities caused by abuse of access and privileges, is becoming comparable in attack frequency to outsider threat. However, insiders are particularly dangerous because they are aware of security measures in place, organisation-specific vulnerabilities and location of valuable information. We consider these elements in our approach for network vulnerability assessment. However, although our focus relies on Insider Threat assessment, the approach is also applicable to outsiders, social networks and computer grid.

Current approaches for vulnerability assessment, model attacks without considering simultaneity and synchronisation aspects of attacks. However, in practice these aspects are relevant. Let's examine two motivating examples: one with computer network and one with social network.

The 9/11 terrorist attack [12] in the US involved four different targets, i.e the World Trade Center Twin Towers, in New York City, the Pentagon, near Washington, and a fourth unknown target. The activities were organised in 6 core groups, all 19 participants affiliated to al-Qaeda. The targets were hit in a coordinated fashion, approximately within 1 hour time-frame. Thus, the 9-11 attack involved synchronisation, parallelism but no target sharing.

A Denial of Service (DoS) attack happens when resources of a computer server are consumed by the arrival of a big number of simultaneous requisitions, which involves parallelism, synchronisation and target sharing.

1.1. Paper contribution and organisation

Our contribution is twofold. First, we present an algorithm that learns attack specifications with minimum cost and maximum value, from a LAN graph. **Attack specifications** represent attack scenarios which are composed of attack steps, i.e. arcs of the graph. We assume: (i) attackers share the goal of maximum return with less effort; (ii) a network node has a value (representing the benefit obtained by attacking it), and that several nodes may have a greater value together (called **added value**) than the sum of their individual values; (iii) different communication channels provide different levels of protection, thus costs are assigned to arcs. Awareness of attack specifications allows organisations to take preventive measures.

Second, the algorithm itself is a contribution since it is an evolutionary algorithm that follows the typical paradigm of local search but is different from traditional approaches. It works with a system of three pools: one where attacks can mature, another where attacks can improve in complexity, and finally one where non-attacks die, if they have shown hard to improve. It also uses a system of credits that avoids search space explosion while maintaining a big number of candidate attacks being matured.

Section 2, reviews related work. Section 3 describes our approach. Section 4 presents design details and the algorithm. Finally, Section 5 provides conclusions and pointers to future work.

*Supported by the research program Sentinels (www.sentinels.nl). Grateful for comments from dr. Pascal van Eck.

0-7695-2876-7/07 $25.00 © 2007 IEEE
DOI 10.1109/IAS.2007.42

2. Related work

Vulnerability assessment and attack modelling has been extensively researched using techniques such as trees and graphs. However, to our knowledge, none of these approaches permit the modelling of synchronism between attack steps, required for representing distributed attacks, such as DoS.

Fault Tree, Attack Tree and Event Tree (e.g. [6]) refine a tree root representing an attack goal or known vulnerability. These trees cannot represent cycles, cannot represent order between nodes and cannot model parallelism.

Attack graphs represent attack steps as nodes, and step transitions as arcs. Phillips and Swiler [9] uses near-optimal shortest path applied to attack graphs to assess the vulnerability of targets. Although their approach permits the modelling of attack step cycles, it does not permit the modelling of simultaneous steps performed by an attacker, as the authors pointed out themselves.

Sheyner et al. [10] generate attack graph using a symbolic model checker. This process requires as input a model of the network on a finite state machine representation and a safety property to be satisfied by the model. Attack graphs are generated from all possible counter-examples, if the property is not satisfied by the model. The approach does not consider any form of parallel attacks.

Dacier et al. [3] propose privilege graphs where nodes are possible attack initiators and possible targets, and arcs are vulnerabilities which allow the acquisition of privileges for a node-initiator towards a node-target. They transform a privilege graph to Petri-net and then derive a state graph, which is used for vulnerability assessment in terms of MTTF (Mean Time To Failure), i.e. mean time for an attacker to reach a target. The assessment is performed in terms of attack step and does not permit the composition of attack scenarios.

Chinchani et al. [2, 1] present challenge graphs. Nodes are entities which provide information or capabilities, represented by keys. Arcs are channels of interaction, which have key-challenges and costs attached. The set of vertices which reaches the set of target with minimum cost indicates vulnerable attack scenarios. Their graph can model colluding attackers, i.e. more than one attacker aiming the same target. However, it cannot model synchronised attack steps.

Gorman et al. [4] uses a graph approach to represent internet autonomous systems, as nodes, and their connections, as arcs. They analyse security in terms of statistical parameters and infection propagation, using different attack and defence strategies. Their assessment of vulnerability is based on investment with defence. Gregg et al [5] measure attack effectiveness using Probability of DoS computed as a function of timeout settings, number of connections allowed, and rate of attack requests. Like Gorman et al., they draw conclusions in a quantitative way only, without interest on attack scenarios.

3. Our approach

Our approach aims to learn attack specifications from a graph representing a computer network. Thus, the goal consists of constructing attack specifications that minimise the cost to perform an attack and maximise the values of the targets. We assume that we have been given: (i) a **graph** whose nodes are computers and arcs are communication channels between a pair of computers; (ii) a **cost function** that maps arcs to a number representing the level of protection provided by the communication channel; (iii) a **value function** that maps sets of nodes to a number representing its **added value**; (iv) a set of **initial nodes** the organisation has under suspicion or simply wants to investigate.

Attack specifications are represented in a fragment of Hoare's [7] language, CSP (Communicating Sequential Processes). The advantage of such language is its expressiveness. It permits to represent parallelism, concurrency and synchronism in a more structured (and frequently compacted) way than what can be achieved with Petri Nets, for example.

An attack specification is represented by an algebraic data type that we will call **CSP**. A CSP is defined in Haskell notation next.

```
data CSP =  Arc (a,b)
          | Seq [CSP]
          | Par [CSP]
```

Definition 1 *A **CSP** is either: (i) an arc from the input graph, or (ii) a sequential composition of a list of CSPs, or (iii) a parallel composition of a list of CSPs.*

A sequence composed of an arc a and a CSP c represents that a occurs and then c follows. It permits the representation of synchronism. The parallel composition of a set of CSPs denotes their simultaneous, concurrent execution.

Definition 2 *The head of an $Arc(a, b)$ is a and its tail is b. The **head set** of a CSP c is the subset of nodes h in c such that no node in h is tail of any arc in c. The **tail set** of a CSP c is the subset of nodes t in c such that no node in t is head of any arc in c.*

Definition 3 *An **attack** is a CSP c that starts on a node from the given* initial set *and ends on a target, where a **target** is a set of nodes from c's tail set that has added value greater than a given threshold.*

Definition 4 *The **value** of a CSP is the added value of its tail set and its **fitness** is the difference between its value and the sum of costs of its arcs.*

Figure 1 shows an example of a potential attack with node 0, as a start, and node 5, as target. It can be shown as a sequence $Seq[s_0, s_1, s_2]$, stating that s_0 comes first, followed by s_1, and then s_2. Sequences s_0, s_1, s_2 are themselves CSPs: s_0, s_1 and s_1, s_2 share nodes and s_1 is a parallel CSP. The CSP is represented in Haskell notation next.

```
CSP = Seq[
        Arc(0,1),
        Par[Seq[Arc(1,2),Arc(2,4),Arc(4,8)],
            Seq[Arc(1,6),Arc(6,8)]],
        Arc(8,5)]
```

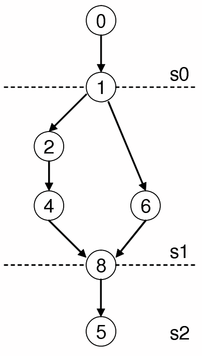

Figure 1. Attack scenario with concurrency

Figure 2 is an example where synchronism is important. Its corresponding CSP follows.

```
CSP = Seq[
        Arc(20,21),
        Par[Seq[Arc(21,26),Arc(26,27)],
            Arc(21,22),
            Seq[Arc(21,24),Arc(24,25)]],
        Par[Arc(27,21),
            Arc(22,21),
            Arc(25,21)]]
```

This is an example where parallelism (concurrency with more than one path leading to the same target) and synchronism are required to express a typical DoS scenario.

4. Algorithm

We propose an evolutionary-based algorithm to learn attack specifications, i.e. CSPs. The rationale behind the algorithm differs from other evolutionary approaches, like Simulated Annealing and Genetic Algorithm because individuals are allowed some time to mature (i.e. to improve their fitness), instead of being eliminated when born unfit.

Edition operations to produce new CSPs from existing ones are randomly chosen from the following set.

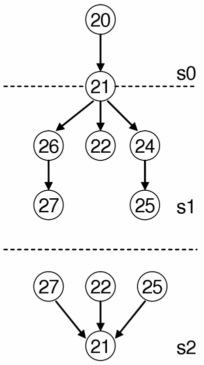

Figure 2. Attack scenario with synchronism

1. With low probability, addition of a new CSP with just one arc.
2. Extension to tail by adding a least cost arc to it.
3. Parallel combination of two CSPs.
4. Appending CSP b to CSP a if a's tail set is equal b's head set.
5. Synchronising two CSPs into a parallel one when they share a head or a tail.

The main characteristics of the algorithm are described next. We assume that the input graph has n nodes.

- A newly created CSP is immediately discarded if it has more than n^2 arcs. Otherwise, the CSP is attributed a fixed initial credit, representing its time for improvement.
- When a CSP is edited its credit is reduced by an amount proportional to the complexity of the edition algorithm. For example, adding an arc to a tail may demand reconstructing the whole CSP, consuming credit equal to the number of arcs involved.
- A program iteration, called cycle, also has credit, which is reduced by the amount of credit consumed in each of its editions.
- The algorithm uses three CSP pools, named Speculation Pool (SP), Attack Pool (AP) and Dying Pool (DP).
- The SP contains non-attack CSPs and attack CSPs with positive credit; all edition operations apply to the CSPs belonging to this pool.
- The AP contains attack CSPs removed from the SP; editions to attacks in this pool are limited to parallel compositions that involve at least one attack and tail extensions that increase the csp's value.

- The DP contains non-attack CSPs removed from the SP; a final check to see if the csp tail is within a predefined distance of a target is applied; in affirmative case, the CSP is transfered back to the SP, otherwise it is deleted.

Definition 5 *A* ***reproduction phase*** *consists of n_{SP} SP editions followed by n_{AP} AP editions followed by n_{DP} DP editions (n_{SP}, n_{AP} and n_{DP} are parameters).*

Definition 6 *A* ***retirement phase*** *consists in (i) moving any CSPs with negative credit from SP to AP (attacks) or from SP to DP (non-attacks); and (ii) attacks from AP to the output; and (iii) CSPs from DP to actual death or back to SP.*

The basic algorithm follows; cycle_credits sums credits consumed in the reproduction phase.

```
SP = {}, AP = {}, DP = {}
for n cycles with MAX_credits each
   reproduction_phase
   if cycle_credits > MAX_credits
      retirement_phase
```

Proposition 1 *The complexity of the algorithm is $O(n_C * n^2)$, where n is the number of nodes in the graph and n_C is the number of cycles.*

Proof. Each cycle has a $(n_{SP} + n_{AP} + n_{DP})$ editions demanding time $O(n^2)$, given limit on the number of arcs of a CSP, and the fact that n_{SP}, n_{AP}, n_{DP} are small constants compared to the size of the graph. The number of cycles in the main loop is in general equal or greater than n, which makes the algorithm run with a cubic upper bound. However, it must be observed that in general CSPs have size much smaller than n because their size is limited by the credit they receive.

An initial prototype has been implemented in Haskell. It identified examples characterising DoS and parallel attacks, like the 9/11, in a random graph of 300 nodes.

5. Conclusions and Future Work

We presented in this paper an evolutionary algorithm for learning attack specifications in a LAN graph. The approach permits the representation of sequence, parallelism and atomic arc compositions in a way that allows the modelling of concurrency, synchronism and simultaneity. The idea of attack specifications comes from the specification of parallel programming. Thus, by analogy, we use the term attack CSP, as a reference to Hoare's language.

We have a new development in Python, that adds algorithms for cleaning attacks and a new concept of fitness which constrains the number of non-attack CSPs preserved.

References

[1] R. Chinchani, A. Iyer, H. Q. Ngo, and S. Upadhyaya. Towards a Theory of Insider Threat Assessment. In *DSN 2005: Int. Conference on Dependable Systems and Networks*, pages 108–117. IEEE Publishing, July 2005. `http://ieeexplore.ieee.org/iel5/9904/31476/01467785.pdf`.

[2] R. Chinchani, A. Iyer, H. N. Q., and S. Upadhyaya. A Target-Centric Formal Model For Insider Threat and More. Technical Report 2004-16, University of Buffalo, US, October 2004.

[3] M. Dacier, Y. Deswarte, and M. Kaaniche. Models and Tools for Quantitative Assessment of Operational Security. In *IFIP SEC'96*, pages 177–186, May 1996.

[4] S. P. Gorman, R. G. Kulkarni, L. A. Schintler, and R. R. Stough. A Network Based Simulation Approach to Cybersecurity Policy. `http://policy.gmu.edu/imp/research.html`. George Mason University, School of Public Policy.

[5] D. M. Gregg, W. J. Blackert, D. V. Heinbuch, and D. Furnanage. Assessing and Quantifying Denial of Service Attacks. *MILCOM'01: Military Communications Conference*, 1:76–80, 2001.

[6] G. Helmer, J. Wong, mark Slagell, V. Honavar, L. Miller, and R. Lutz. A software fault approach to requirements analysis of an intrusion detection system. *Requirements Engineering*, 2002. `http://dx.doi.org/10.1007/s007660200016`.

[7] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978. `http://doi.acm.org/10.1145/359576.359585`.

[8] M. Keeney, E. Kowalski, D. Cappelli, A. Moore, T. Shimeall, and S. Rogers. Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors, May 2005. U.S. Secret Service and CERT Coordination Center.

[9] C. Phillips and L. P. Swiler. A Graph-Based System for Network-Vulnerability Analysis. In *NSPW '98: Proc. 1998 workshop on New Security Paradigms*, pages 71–79, New York, NY, USA, 1998. ACM Press.

[10] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated Generation and Analysis of Attack Graphs. In *SP'02: Proc. 2002 IEEE Symposium on Security and Privacy*, pages 273–284, Washington, DC, USA, 2002. IEEE Computer Society. `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1004377`.

[11] Survey. E-Crime Watch 2006, CSO Magazine and U.S. Secret Service and CERT Coordination Center and Microsoft Corporation, 2006. `http://www.cert.org/archive/pdf/ecrimesurvey06.pdf`.

[12] Wikipedia. September 11, 2001 attacks. `http://en.wikipedia.org/wiki/September_11_2001_attacks`.