Evaluation of Adaptive Attacker Models

Leanid Krautsevich and Artsiom Yautsiukhin

Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche
Via G. Moruzzi 1, Pisa 56124, Italy
{firstname.lastname}@iit.cnr.it

Abstract. In this paper, we model attacker behaviour. We have implemented a usual, i.e., omniscient, attacker and our adaptive attacker. We are going to compare these two models to find whether there is a significant difference between them. We consider the comparison from the attacker point of view and from the point of view of the defender (which wants to minimise the damage from the attacks).

1 Motivation

Existing models of attacker grant the penetrator the complete knowledge of a system, i.e., the knowledge about unpatched vulnerabilities, installed countermeasures, running services, etc [2, 5]. With such knowledge the attacker is able to select the best possible way to attack the system and reach her goal. Such model is similar to the Dolev-Yao model for security analysis of protocols. It is useful to find the weakest place in security system.

In the real world the attacker gains the knowledge about the system during the attack [3]. The attacker collects all available information about the target system, but such knowledge is never complete. Using vulnerability scanners also provide only the information about a part of the system, without looking deep into it. Moreover, aggressive scanning may be detected and the attack may be prevented before its beginning. This means, that the model of the attacker with complete knowledge does not provide the exact risk for the system.

In the previous paper [1], we proposed an adaptive attacker (AA) model. The main peculiarity of this model is that the attacker does not have a complete knowledge about the system but only his view about it. The attacker tries to execute an attack and adjusts its course of actions if his view was wrong.

We continue the evaluation of the system with such kind of attacker. We would like to compare the results of the analysis of the system with the AA and the OA and check whether there is a difference in assessment of the system when the two models are considered. We want to find the conditions, when the models return the same results, and when the results are different.

2 Scientific Contribution

2.1 The Adaptive Attacker Model

Here we briefly outline the main features of the model proposed in [1].

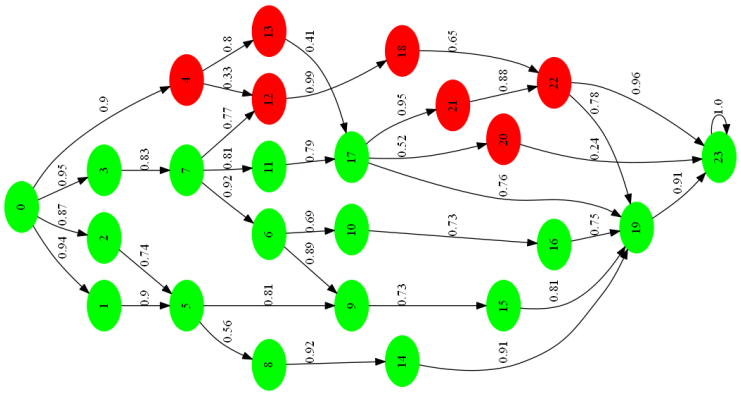

Fig. 1. Test Graph

Let $G = (S, A)$ be a real attack graph, such that a node $s \in S$ represents a successfully exploited vulnerability and an edge $a_{ij} \in A$ denotes further possible exploitation of another vulnerability. Let $G_X = (S_X, A_X)$ be a belief attack graph, where nodes exist according to the belief of the attacker. The real graph is a subgraph of the belief graph. The example of such graph is shown in Figure 1. Green nodes (light grey) indicate the vulnerabilities really existing in the system, when red (dark grey) nodes denote the nodes, which were patched by the administrator, but the attacker has no knowledge about this fact.

Assume that we know the probability of successful exploitation of vulnerability, i.e., $\mathbf{Pr}_{ij}$ for every edge in G_X, and monetary reward for the states, the attacker wants to achieve. The attacker wants to maximise his profit. Thus, the attacker has to solve a Markov Decision Problem (MDP) to find the optimal path to one of his goals within a limited amount of time. We use backward induction algorithm for computation of a policy, i.e., the sequence of steps the attacker should follow. Such attack may include states, which do not exist in reality (e.g., the vulnerability was patched). An example of existing path in the graph shown in Figure 1 is $\{0, 2, 5, 9, 15, 19, 23\}$, when the path $\{3, 7, 11, 17, 21, 22, 23\}$ includes the nodes which do not exist in reality (e.g., nodes 21 and 22).

We modify the behaviour of the omniscient attacker (OA) so that he may reconsider his course of actions when she cannot complete his current attack path. If the attacker fails to complete his attack because some vulnerability does not exist, he assigns $\mathbf{Pr}_{ij} = 0$ for all edges leading to this vulnerability. Then, the attacker uses the backward induction algorithm to compute a new strategy using the updated attack graph and the rest of time

2.2 Evaluation Methodology

Now we would like to compare the AA and the OA. We perform some statistical tests for the comparison. Our goals are to check:

– How paths of the AA and OA differs.
– How this difference depends on the shape of the system (i.e., the attack graph).

First we analyse behaviour of the two attackers for a specific graph (shown in Figure 1) to consider the difference between the strategies of the AA and the OA. We use the following metrics to measure how close the two solutions are:

– *Attack length (al)* is a number of steps an attacker must make before she reaches her goal. This parameter could be seen as an attacker cost, if we assume that an attacker pays one unit for making a step.
– *Similarity (sim)* is the percentage of states in the path of the AA, which are contained in the path of the OA.
– *Same final decision (sfd)* is the percentage of cases, when the path of the OA is entirely included in the path of the AA. In other words, we check the percentage of cases when the AA finally decided to use the same path that the OA does.

We start with the entire graph as a real attack graph taking into account all nodes. Then, we randomly choose some nodes and consider them as non-existing ones (about these nodes the attacker has no precise knowledge). We repeat the experiment 1000 times for every percentage of missing real nodes. We consider only such belief graphs when the source and top nodes are connected by existing vulnerabilities. The results are shown in Figure 2.

In Figures 2a and 2b we see that in average the number of steps for AA is higher (by $20 - 30\%$). Note, that this does not mean that we always have the same path at the end. Figure 2a shows, that sometimes path for AA differs significantly from the path of the OA (if the percentage of real vulnerabilities is 90%, then only 85% of steps of AA are the steps from the paths of OA).

We see that similarity increases when the percentage of real states increases between 60% and 100%. This tendency is evident: the less real states exist in the system, the more wrong paths the attacker will follow first, before he finds the existing way to the goal. Note that the similarity slightly decreases with the increase of the percentage of real states in between 40% and 60%. This can be explained by the fact that the AA gets to a non-existing state faster (than with lower percentage of real nodes) and finds the optimal path with less wrong steps. Figure 2b shows similar results.

Figure 2c indicates, that the number of paths, that use the path of OA is the lowest for 90%. Naturally, when all states exist (100%), then the paths for AA and OA are the same. Also, the less vulnerabilities really exist in the system the less paths to the goal states are left. Thus, the AA finally comes to the (sometimes, the only existing) path selected by the OA. We would like to note here that in our example 40% of states is approximately, 10 states, while the shortest path in the graph requires 5 states.

Now, we would like to consider different graphs, rather than a specific one and check how number of nodes in and connectivity (i.e., average number of edges leading from a node) of the graph affects the considered metrics. We generate a

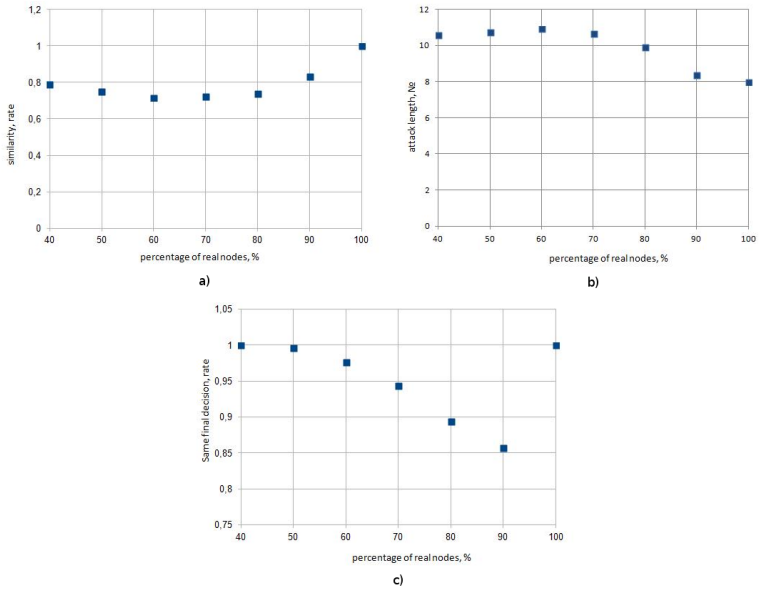

Fig. 2. Results for a) similarity, b) attack length and c) percentage of same final decision for a specific graph

graph randomly, and use exponential distribution to determine how many edges a node should have.

In particular, we distributed nodes in a predefined number of layers. Layers are required to model a depth of the graph and order the edges leading from a source node to the top nodes. The number of nodes per layer is determined using a lognormal distribution and adjusted by coefficient, which is required to force the tree to have a tree-like form (smaller number of vulnerabilities for higher layers and larger number for lower ones). A node is allowed to have an edge to a node to any higher layer, but because of parameters of another exponential distribution, most of them lead to the next layer.

We generated 100 graphs for every parameter we would like to evaluate and took the average value of the received metric. For computation of the metric we still do 1000 random test for every considered percentage of real nodes.

First we check the dependency of the metrics and the connectivity of edges. We used exponential distribution to model the number of edges a node is connected to. Thus, the higher the parameter (lambda) is the more connected the graph is. We fix the number of nodes as 50 and vary lambda. The results are shown in Figure 3.

We see that although similarity rate is higher for higher lambda it is rarer that the whole attack path for OA is included into AA. This could be explained by the fact, that although AA often uses some parts of the path of OA, it has

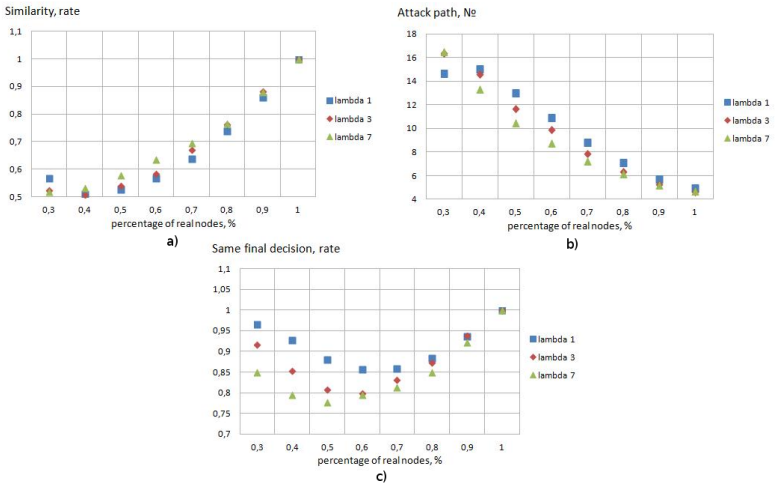

Fig. 3. Dependency of a) similarity, b) attack length and c) percentage of same final decision on parameter *lambda*

more opportunity to try a shorter path which differs from the optimal path. This explanation is partially supported by the Figure 3b for attack path, where the path becomes shorter with higher connectivity.

The next set of experiments was done with a fixed $lambda = 3$, and three different numbers of nodes: 50, 75, and 100. The results are shown in Figure 4.

We see that number of nodes (N) has stronger impact on the metrics. The main trend is evident: the more nodes are in the graph, the more opportunity the AA has to deviate from the path of OA. Note, that if the percentage of real nodes is between 0.3 and 0.7 then only 60% of time AA uses the path of OA for $N = 100$. We see, that the length of the attack path for $N = 100$ decreases with the decrease of percentage of real nodes. This could be explained by the fact, that the graph generating algorithm used the same amount of nodes between the source and top nodes in average. Thus, in this case a higher number of nodes lead to higher number of paths, and attacker has more choices, even if many paths appear to be non-existing.

3 Conclusions

We see that using the omniscient attacker model for getting the real picture of the security strength is prone to errors. Moreover, the more nodes the graph has and the higher the connectivity is the higher the error is. According to the literature on attack graphs [4, 6] these parameters often have large values. As a result, the overpowering an attacker may lead to unnecessary investments in security needs.

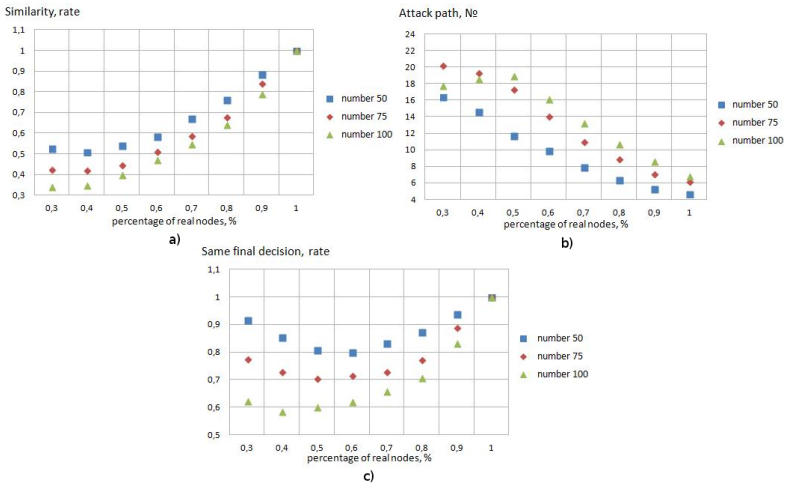

Fig. 4. Dependency of a) similarity, b) attack length and c) percentage of same final decision on *number of nodes*

There are a number of ways the work may be improved. First of all, its integration with real attack graph generating and analysis tool is the most interesting one. Furthermore, we may add some uncertainty to the attacker selecting the next step, using some probability distribution. This uncertainty should allow us to consider not only the most probable path, but also near-most probable ones, modelling uncertainty of attacker decisions.

References

1. L. Krautsevich, F. Martinelli, and A. Yautsiukhin. Towards modelling adaptive attacker's behaviour. In *Proceedings of 5th International Symposium on Foundations and Practice of Security*, 2012.
2. E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke. Model-based security metrics using adversary view security evaluation (ADVISE). In *Proceedings of the 8th International Conference on Quantitative Evaluation of SysTems*, 2011.
3. K. D. Mitnik and W. L. Simon. *The Art of Intrusion: The Real Stories Behind the Exploits of Hackers, Intruders and Deceivers.* Wiley, 2005.
4. S. Noel and S. Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, 2004.
5. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. In *In Proceedings of 2005 IEEE Symposium on Security and Privacy*, 2002.
6. O. Sheyner and J. M. Wing. Tools for generating and analysing attack graphs. In *Proceedings of Formal Methods for Components and Objects*, 2005.