Towards Modelling Adaptive Attacker's Behaviour*

Leanid Krautsevich[1], Fabio Martinelli[2], and Artsiom Yautsiukhin[2]

[1] Department of Computer Science, University of Pisa,
Largo B. Pontecorvo 3, Pisa 56127, Italy
krautsev@di.unipi.it
[2] Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche,
Via G. Moruzzi 1, Pisa 56124, Italy
{fabio.martinelli,artsiom.yautsiukhin}@iit.cnr.it

Abstract. We describe our model for the behaviour of an attacker. In the model, the attacker has uncertain knowledge about a computer system. Moreover, the attacker tries different attack paths if initially selected ones cannot be completed. The model allows finer-grain analysis of the security of computer systems. The model is based on Markov Decision Processes theory for predicting possible attacker's decisions.

Keywords: Attacker Model, Attack Graphs, Markov Decision Process

1 Introduction

Most methods for the analysis of security of computer systems (e.g., networks, Cloud, etc.) consider attackers as omniscient entities which know all weaknesses of a computer system [4, 12]. In addition, attackers are frequently assumed to make only right decisions during an attack and to exploit only the best possible way for the attack.

In contrast, descriptions of real complex attacks (e.g., [7]) show, that attackers have limited knowledge of a target system and explore the system step by step during the attack. Attackers make mistakes in their reasoning about the system, and search for alternative ways to compromise the system when the initially selected attack fails. This means that the model of powerful attacker does not provide a real description of a situation, but prepares for a worst case scenario. In reality, security teams have a limited budget and would like to concentrate on the most important security issues that can be solved within a budget. Taking properly into account attacker's behaviour is important because some attacks may

* This work was partly supported by EU-FP7-ICT NESSoS and 295354 SESAMO projects.

be even not considered by the attacker because of her uncertain knowledge about the system or lack of the resources. Wasting the budget on preventing such attacks is not the most cost-effective decision.

In this paper, we strive for a more refined attacker model introducing the attacker's view of a system, which is sometimes different from the real system. This view drives the actions of the attacker depending on the knowledge and resources the attacker possesses. Moreover, in our model an attacker may give up on her current attack and follow an alternative attack path. We use Markov Decision Process (MDP) to model the behaviour of attacker as the method for the selection of attack steps.

The rest of the paper is organised as follows. Section 2 explains our concerns on uncertain knowledge of an attacker about a system. Section 3 focuses on models of attacker's behaviour. Related work is presented in Section 4. Section 5 concludes the paper.

2 A System and an Attacker

We consider a computer system as an attack graph G that represents the ways to compromise the system [4, 12]. A node $s_i \in S$ of the attack graph denotes a successfully exploited vulnerability and an edge $a_{ij} \in A$ denotes further possible exploitation of vulnerability s_j after previously exploited vulnerability s_i. Thus, successful exploitation of vulnerabilities leads an attacker to new states with new privileges. There are several methods for automated construction of attack graphs [12, 13].

Similarly to our previous work [3] we group attackers into attacker profiles. Within a profile, attackers have the same goal and similar parameters such as money, skills, etc. Formally, the attacker profile is the tuple $\mathcal{X} = \{\Gamma, goal, intang, tang, skill\}$ where Γ is the set of attacks $\gamma \in \Gamma$ known by the attacker, *goal* is the goal of the attacker, *intang* is an amount of intangible resources possessed by the attacker, e.g., time, *tang* is the amount of tangible resources possessed by the attacker, *skill* defines how trained is the attacker. We modify the attack graph to capture properties of the attacker. First, we add to the graph an initial node corresponding to initial privileges of the attacker. Second, we define the goal nodes in the attack graph G that correspond to vulnerabilities that complete the attack (the ultimate step of each attack).

We assume that the attacker has certain amount of time units to perform the attacks. She spends a unit of time for executing a single attack step. The attacker stays in a goal state if she reaches it before

spending all units of time. This situation is modelled by adding edges that start and end in the same goal state.

We separate the *real system* and the *attacker belief* about the system. When the attacker is omniscient, her view of the system coincides with the real system. We consider a more realistic case, when the view does not coincide with the real systems. The attacker's knowledge about the system determines the set of vulnerabilities that the attacker believes present in the system. These believed vulnerabilities define a new graph G_B. This graph is similar to the attack graph for the real system while has believed vulnerabilities as nodes:

$$G_B = (S_B, A_B) : S_B = S_{true} \cup S_{false}, A_B = A_{true} \cup A_{false} \tag{1}$$

where $S_{true} \subseteq S$ and $A_{true} \subseteq A$ are the subset of vulnerabilities and the subset of attack steps really existing in the system and also believed by the attacker to exist, S_{false} and A_{false} are the set of vulnerabilities and the set of action that are believed to exist but are absent in reality.

The set of vulnerabilities that are believed by the attacker is further reduced according to attacker's skills and tangible resources. Finally, the attacker has her own *view* (a graph $G_{\mathcal{X}}$) of the system:

$$G_{\mathcal{X}} = (S_{\mathcal{X}}, A_{\mathcal{X}}) : S_{\mathcal{X}} \subseteq S_B, A_{\mathcal{X}} \subseteq A_B \tag{2}$$

We assume that the system behaves probabilistically. We introduce probability $\mathbf{Pr}_{ij}$ of system transition from state i to state j in response to an attacker's action. For the attacker this probability is:

$$\mathbf{Pr}_{ij} = \mathbf{Pr}^{p}_{ij} \cdot \mathbf{Pr}^{exp}_{ij} \tag{3}$$

where $\mathbf{Pr}^{p}_{ij}$ is the probability that the vulnerability j presents in the systems and $\mathbf{Pr}^{exp}_{ij}$ is the conditional probability that the vulnerability may be successfully exploited in case it exists in the system. The probability $\mathbf{Pr}_{ij}$ depends only on the successive state j while we use both indexes i and j for the uniformity with usual definition of transition probabilities.

We measure $\mathbf{Pr}^{p}_{ij}$ assuming that the attacker knows which software is installed in the system but may not know whether the software is patched or it is not. The probability of presence of the vulnerability in the system depends on the period passed after the vulnerability was discovered: the more time passed since the discovery the lower the probability of presence of the vulnerability [8]. We assume $\mathbf{Pr}^{p}_{ij}$ decreases linearly in time:

$$\begin{aligned} \mathbf{Pr}^{p}_{ij} &= -\tfrac{1}{T_{patch}} \cdot t + 1 \quad \text{if } T_{patch} \geq t \\ \mathbf{Pr}^{p}_{ij} &= 0 \qquad\qquad\quad \text{if } T_{patch} < t \end{aligned} \tag{4}$$

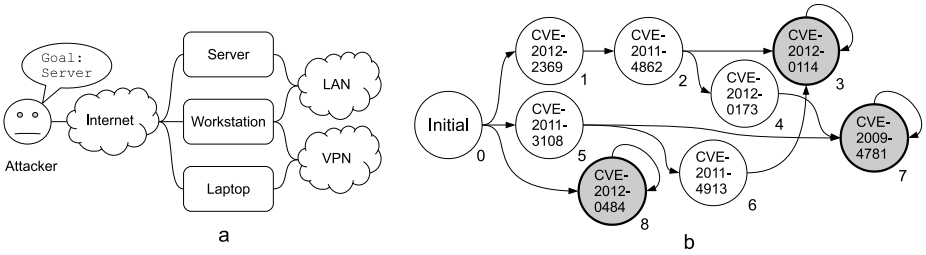

Fig. 1. a) the network system, b) the attack graph of the network system

where T_{patch} is the time required for patching all systems, t is time passed since the release of a patch and for $t > T_{patch}$ we assume that all systems are patched. The probability $\mathbf{Pr}_{ij}^{exp}$ may be computed on the basis of score from vulnerability databases similarly to [2] or by security experts. Our approach does not depend on the method of computation of $\mathbf{Pr}_{ij}^{p}$ and $\mathbf{Pr}_{ij}^{exp}$, thus, any other methods can be used.

Example 1. We consider a company which saves information in an on-line database service. A competitor company would like to steal the information by attacking the server where the database is installed. The server operates FreeBSD 7 and MySQL 5. The database is managed by an administrator that uses a local workstation operated by Linux Mint 12 with Pidgin Messenger installed. Moreover, the administrator manages the database from her home laptop using a VPN connection to the workstation. The laptop runs Windows 7, Chrome browser, and TUKEVA Password Reminder. The whole system is depicted in Figure 1a.

The attacker composes the following attacks to the system[3]:

- The shortest possible attack requires registration in the on-line database service and execution of vulnerability CVE-2012-0484 in MySQL.
- Another possible attack is based on vulnerability CVE-2011-3108 in Chrome browser and CVE-2009-4781 in TUKEVA where the administrator saves passwords to a database management tool.
- The attacker exploits CVE-2012-2369 in Pidgin gaining the access to the workstation. Then she causes a buffer overflow on the server using CVE-2011-486 and exploits CVE-2012-0114 against MySQL.
- Since the laptop is connected by VPN to the workstation, the attacker gains the access to the laptop executing CVE-2012-0173 in Windows 7. Then she exploits CVE-2009-4781 in TUKEVA.

[3] Please, follow `http://nvd.nist.gov/home.cfm` for details of vulnerabilities.

– The attacker may gain the access to the workstation after successful attack to the laptop by executing CVE-2011-4913 in the Linux kernel. Then she exploits CVE-2011-486 on the FreeBSD server.

The resulted attack graph is displayed in Figure 1b. We enumerate the nodes for the sake of convenience. The node s_0 is the initial node. The nodes n_3, n_7 and n_8 (coloured in grey) are goal nodes.

3 Model of Attacker's Behaviour

We use Markov Decision Process (MDP) [9] to model decision making process of attackers. An attacker observes a system and can influence the behaviour of the system by making actions at moments of time (decision epochs). The system responds to an action probabilistically. The attacker does not make the decisions about actions blindly but takes into account past, current, and possible future states of the system and also possible rewards that are connected with the actions. The goal of the attacker is to maximise the expected total reward according to a some criterion.

Formally, MDP is a set $\mathcal{P} = \{S, A, P, R, T\}$ where S is a set of system states s_i, A is a set of sets A_i of actions $a_{ij} \in A_i$ available for the attacker in the state s_i, P is a set probabilities $\mathbf{Pr}_{ij}$ that the system transits from state s_i to s_j in response to attacker's action a_{ij}, R is a set of rewards functions r_{ij} dependent on the state s_i and the action a_{ij}, T is a set of decision epochs (moments of time) t. Regarding transition probabilities, in general, the system may transit to any state available from s_i in response to the action a_{ij}. We assume that the system only transits to the state s_j with probability Pr_{ij} or stays in the state i with probability $1 - Pr_{ij}$.

We model attacker's behaviour as an MDP policy π which determines how an attacker selects actions. The policy is composed of decision rules. A decision rule is a procedure for the selection of an action for each moment of time. The attacker always selects a certain action in a state if the rules are deterministic. She may select at random any action available at the state if the rules are probabilistic.

A total reward u_π obtained by the attacker as a result of the execution of policy π is computed on the basis of instant and terminal rewards. The attacker obtains an instant reward after execution of an action. The instant reward depends on s^t, a^t and s^{t+1}. The terminal reward $r^N(s^N)$ depends on the state of the process at the last decision epoch N. Thus, the total reward:

$$u_\pi = \sum_{t=1}^{N-1} r^t(s^t, a^t) + r^N(s^N) \tag{5}$$

Algorithm 1 Computation of a deterministic policy

$t = N$
for all $s^N \in S$ **do**
  $u^N(s^N) = r^N(s^N)$
end for
while $t > 1$ **do**
  $t = t - 1$
  for all $s^t \in S$ **do**

$$u^t = \max_{a \in A_{s^t}} \left\{ r^t(s^t, a^t) + \sum_{a_{ij} \in A_i} \mathbf{Pr}_{ij}^t \cdot u^{t+1}(s_j) \right\}$$

$$A^*_{s^t,t} = \arg\max_{a \in A_{s^t}} \left\{ r^t(s^t, a^t) + \sum_{a_{ij} \in A_i} \mathbf{Pr}_{ij}^t \cdot u^{t+1}(s_j) \right\}$$

  end for
end while

Note, that we use upper index (e.g., s^t for a state) to denote the current value of a variable at a moment of time.

We evaluate every attacker's action as an amount of money. E.g., for an attacker that tries to cause maximal damage to a system, the reward are losses faced by the system owner in case of a successful attack.

Deterministic Attacker The simplest model of attackers behaviour [4, 12, 13] may be defined by an optimal deterministic policy of MDP. In this case, an attacker always prefers the best possible action in a state which belongs to the optimal attack path in the attack graph. The algorithm for the computation of optimal deterministic policies is the backward induction [9] (see Algorithm 1). Variables u^t and u^n describe intermediate values of total reward. The algorithm finds sets $A^*_{s_t,t}$ of actions that maximise the expected total reward of the attacker.

Adaptive Attacker We modify the behaviour of the deterministic attacker so that she may reconsider her course of action when she cannot complete her current attack path. We assume that the attacker sets $\mathbf{Pr}_{ij}^p = 0$ (and $\mathbf{Pr}_{ij} = 0$) when she cannot complete an attack step a_{ij} and understands that the vulnerability is absent in the system. In addition, the attacker sets $\mathbf{Pr}_{xj} = 0$ for all other edges entering s_j from all states s_x. Then the attacker uses Algorithm 1 to compute a new strategy using the updated attack graph and the amount of decision epochs left after the initial part of the attack.

The attacker sets $\mathbf{Pr}_{ij}^p = 1$ and $\mathbf{Pr}_{ij} = \mathbf{Pr}_{ij}^{exp}$ if she understands that the vulnerability exists in the system as a result of the unsuccess-

ful attempt of the attack step. Then the attack strategy is recomputed according to Algorithm 1 with the rest of the decision epochs. If the attacker successfully exploits the vulnerability s_i she adds edges a_{0j} and sets $\mathbf{Pr}_{0j} = \mathbf{Pr}_{ij}$ for all states s_j reachable from s_i in one step. This modification is required to remember the privileges gained by the attacker for future adjustments in her strategy.

Example 2. In our example, the attacker has N decision epochs and gets terminal rewards ($10K) only if she reaches states 3, 7, 8 i.e. $r^N(s_3) = r^N(s_7) = r^N(s_8) = 10$ other terminal rewards equal 0. Instant rewards also equal to 0. Due to space limitations we skip the computation of deterministic policies. For the attack graph presented in Figure 1, the policy is $\pi = (a^1 = a_{08})$ at the initial state during the first decision epoch. Suppose, the action is unsuccessful because the vulnerability was timely patched by the administrator. The attacker sets the probability $\mathbf{Pr}_{08} = 0$, reconsiders her initial policies using $N-1$ decision epochs, and obtains new policy $\pi = (a^1 = a_{05})$.

4 Related Work

There are several works which use attack graphs for the analysis of a system [12, 13]. Sheyner et al. [12] determine possible attacks to the system on the basis of attack graph assuming deterministic attackers behaviour. LeMay et at. [4] and Sarraute et al. [11] proposed works that consider attack planning where successful execution of an exploit is uncertain. In contrast to our work, the authors assume that the attacker has complete knowledge about the system and always selects the same path to his goal during the attack. Several attacker models, which assume that attacker may select alternative ways for compromising a system, were proposed for the analysis of cryptographic protocols [1, 5, 6]. These models also assume that attackers know the system, i.e., the protocol, while have bounded resources. Those models indeed consider different limitations from ours, e.g., computational power and message manipulation capabilities.

The very recent paper of Sarraute et al. [10] proposes to use Partially Observable Markov Decision Processes (POMDPs) for attack planning during penetration tests. The authors analyse the system considering network configuration graph, while we consider an attack graph. In terms of knowledge collecting, authors introduces special actions that allow scanning network hosts. While we provide a way to update the graph as a result of successful and unsuccessful attack steps, and adjust the reasoning during the attack.

5 Conclusion

This work presented our initial ideas on modelling the behaviour of an attacker. We think, such approach is important if we would like to get versatile analysis of our system and protect it in the most efficient way. In particular, we considered an attacker which does not know every detail about the system, but gains the knowledge step by step. In addition, we made the attacker more flexible, i.e., the attacker is able to re-consider her plans when the initial ones fail.

As for future work we would like to incorporate the notion of decreasing attackers resources in our model as penalties of MDP. Moreover, within our approach we cannot take into account zero-day vulnerabilities, but we believe that some statistical methods could be used to tackle zero-day vulnerabilities at least approximately. Finally, we aim at creating a software prototype to evaluate system security on the basis of different metrics and versus various attacker types.

References

1. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE TIT*, 29:198–208, 1983.
2. L. Gallon and J.-J. Bascou. Cvss attack graphs. In *SITIS*, 2011.
3. L. Krautsevich, F. Martinelli, and A. Yautsiukhin. Formal analysis of security metrics and risk. In *WISTP*, 2011.
4. E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke. Model-based security metrics using adversary view security evaluation (advise). In *QEST*, 2011.
5. D. Marchignoli and F. Martinelli. Automatic verification of cryptographic protocols through compositional analysis techniques. In *TACAS*, 1999.
6. J. C. Mitchell, A. Ramanathan, A. Scedrovb, and V. Teaguea. A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. *TCS*, 353:118–164, 2006.
7. K. D. Mitnik and W. L. Simon. *The Art of Intrusion: The Real Stories Behind the Exploits of Hackers, Intruders and Deceivers.* Wiley, 2005.
8. Y. N. Pettersen. Renego patched servers: A long-term interoperability time bomb brewing. `http://my.opera.com/yngve/blog/2010/06/02/renego-patched-servers-a-long-term-interoperability-time-bomb-brewing` was available online on 20/07/2012.
9. M. L. Puterman. *Markov Decision Processes Discrete Stochastic Dynamic Programming.* Wiley-Interscience, 2005.
10. C. Sarraute, O. Buffet, and J. Hoffmann. Pomdps make better hackers: Accounting for uncertainty in penetration testing. In *AAAI*, 2012.
11. C. Sarraute, G. Richarte, and J. L. Obes. An algorithm to find optimal attack paths in nondeterministic scenarios. In *AISEC*, 2011.
12. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated generation and analysis of attack graphs. In *IEEE SSP*, pages 273–284, 2002.
13. L. Wang, A. Liu, and S. Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *CC*, 29:2917–2933, 2006.