Vulnerability Evaluation of Distributed Reputation Management Systems

Vincenzo Agate
vincenzo.agate@unipa.it

Alessandra De Paola
alessandra.depaola@unipa.it

Giuseppe Lo Re
giuseppe.lore@unipa.it

Marco Morana
marco.morana@unipa.it

Università degli Studi di Palermo
Viale delle Scienze, Ed.6, Palermo, Italy

ABSTRACT

In distributed environments, Reputation Management Systems (RMSs) aim to estimate agents' trustworthiness by exploiting different sources of information. The distributed nature of these systems makes them vulnerable to several types of security attacks, and the response provided by a specific RMS depends on various factors, such as the algorithms adopted for estimating the reputation values and the communication protocols used to enable the cooperation among agents. This work examines the most important security attacks against RMSs and proposes a set of metrics for a quantitative evaluation of the RMS vulnerabilities. A parallel simulation framework is used to automatically give a vulnerability score to a RMS according to the computed metrics. Experiments performed on a case-study RMS show the effectiveness of the metrics we defined, and the convenience of using a simulation environment to support the design of a secure RMS.

Keywords

Distributed Reputation Management; Security Attacks; Evaluation Metrics

1. INTRODUCTION

Several distributed services, such as peer-to-peer applications [19], Service Oriented Architectures [16, 5], and e-Commerce frameworks [9], rely on Reputation Management Systems (RMSs) in order to estimate the behavior of unknown agents.

The role of RMSs is particularly relevant if the benefit perceived by the agents that constitute the distributed system depends on the cooperativeness of other agents, so that a non-cooperative behavior may cause a detriment for the whole community.

In a distributed RMS, a central authority is missing and a common value of reputation is produced by all the agents according to several sources of information, such as their direct experience or information provided by other agents. Such systems allow to avoid the presence of a single point of failure, that can represent also a performance bottleneck, and provide a well scalable solution.

Nevertheless, the distributed nature of such systems makes them vulnerable to several types of security attacks, performed by isolated or colluding malicious agents that aim to obtain an advantage over other agents, or to abuse system resources. The response provided by a specific RMS to an attack depends on various factors, such as the algorithms adopted to estimate the reputation values and the communication protocols used to enable the cooperation among agents.

In this work we propose a set of metrics for evaluating the vulnerability of a RMS to different attacks, so as to understand the effect of different design choices on the RMS's performance. Moreover, a parallel simulation framework has been designed to allow researchers to easily specify the behavior of their RMS, to simply test its responses in different scenarios, and to obtain the corresponding vulnerability indices.

The remainder of the paper is organized as follows: related work is reported in Section 2. The RMS features and components influencing its response under attack are presented in Section 3. Section 4 describes the four security attacks considered here, whilst Section 5 illustrates the proposed vulnerability metrics. The simulation framework is presented in Section 6, and the experimental results are shown and discusses in Section 7. Finally, conclusions follow in Section 8.

2. RELATED WORK

Several reputation management models for distributed systems have been proposed in the literature. The authors of [25] analyzed the main components of trust systems in the context of multi-agent systems, identifying a *trust evaluation* phase, which assesses the reliability of the agents involved in the interactions, and a *trust-aware decision making* phase that uses reputation values to select the agents to interact with. Moreover, the authors classified trust evaluation methods in four classes, depending on the fact that they are based on: (i) past direct interactions, (ii) opinions and testimonies from other agents, (iii) knowledge about social relationships among agents, and (iv) certificates provided by third-party authorities. RMSs for distributed systems,

where a central authority is missing, belong to the second class, according to which each agent relies on a distributed protocol to obtain opinions from other agents, and merges them with its past experience in order to compute the reputation of a given agent.

In recent years, a great attention has been devoted to identify and analyze the security attacks against RMSs, which exhibit many vulnerabilities due to their intrinsic distributed nature. To the best of our knowledge, works addressing this topic miss the definition of quantitative vulnerability metric for RMSs, and a tangible experimental evaluation that can drive researchers to select the best design approach according to the desired behavior.

Authors of [13] describe five attacks against RMSs, partially referenced in Section 4, and outline the defense techniques adopted by some RMSs presented in literature. A similar analysis, focusing only on *feedback-based reputation systems*, is proposed in [21].

Some testbeds and simulators have been proposed for assessing the performances of a RMS. ART (*Agent Reputation and trust*) [10], a popular simulation testbed in the field of multi-agent systems, allows to apply several evaluation metrics, and to define competitions in which different strategies can be combined and compared with respect to the utility obtained by each agent at the end of the simulation. TREET [15] limits the evaluation of RMSs in a marketplace scenario. This framework allows to measure the resistance of the RMS to some attacks, but does not consider other typical attacks, which are instead addressed in our work. In [4] the RMSs are modeled as a sequence of transformations on a graph that represents transactions and trust among agents. Even thought this tool allows to evaluate the effects of some relevant attacks, it does not simulate agent interactions making not possible to perform large-scale simulations in which agents may modify their behavior. The solutions discussed so far do not allow for large-scale simulations of the behavior of a RMS under security attacks.

In this work we extend the parallel simulation framework described in [2] and [1], in order to support researchers to evaluate the behavior of a RMS through large-scale simulations of different security attacks, and to obtain the corresponding vulnerability indices. Our framework, presented in Section 6, allows to specify the behavior of a RMS and to neglect the low-level details necessary to run the simulation, making it possible to focus only on the high-level response of the RMS under attack.

3. RMS ARCHITECTURE

As highlighted in [13], RMSs have a common structure and their differences depend on the algorithmic specification of their constituting components. Two steps can be identified: (i) a formulation and calculation phase during which agents collect and manage information in order to obtain a value for the adopted reputation metric, and (ii) a dissemination phase, when agents cooperate to compute the complete reputation values.

We move step forward by considering four main components, as shown in Figure 1. Such higher level of detail allows to deeply analyze how different attacks exploit system vulnerabilities. The first component is the *local trust evaluation* which uses information from past direct interactions with other agents in order to provide an initial and partial estimation of their reputations. By considering only this component, each agent would have a very limited view of other agents behavior, and would not be able to predict the actions of unknown agents. For such reason, distributed RMSs comprise other two components that allow agents to cooperate obtaining a more precise estimation of the reputation of their neighbors, namely a *gossip protocol* which propagates information among different agents, and an *information fusion* mechanism which allows each agent to merge the direct experience with information obtained through the gossip protocol. RMSs which aim not only to detect antisocial behaviors, but also to discourage them, comprise also an *incentive mechanism* which uses reputation values to reward honest agents and to limit malicious ones.

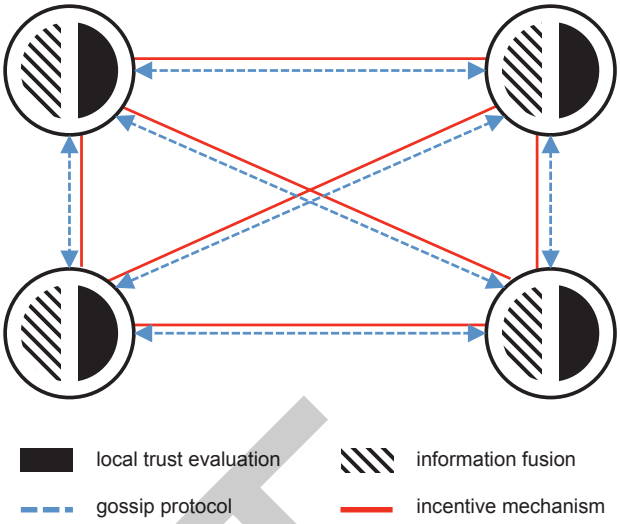

Figure 1: The components of a distributed RMS. Each agent privately performs the local trust evaluation and the information fusion algorithms; the gossip protocol and the incentive mechanism regulate the interactions with other agents.

4. SECURITY ATTACKS ON RMSS

The distributed nature of RMSs makes them vulnerable to several types of security attacks, as described in [13, 21].

We consider the worst case of attackers that are insiders, i.e., authorized users of the system that can take part to all phases of the reputation evaluation. Moreover, we assume that an attacker can get multiple identities and that it can also cooperate with other malicious agents. A classification of security attacks can be made on the basis of their goals: *self-promoting*, *slandering*, *whitewashing*, and *traitor* attacks. The classification presented in [13] includes also the *denial of service* attack, that is not considered here since it is more frequent in a centralized system, where malicious agents may overwhelm the central server, thus hindering the reputation evaluation tasks.

4.1 Self-promoting

In a *self-promoting* attack [17], a malicious agent aims to increment its own reputation value, generally in order to hide an antisocial behavior, or to achieve an unjustified advantage over its competitors, e.g., in an e-commerce scenario. This type of attack requires an orchestrated plan in-

volving several malicious agents, since, generally, reputation management systems do not allow an agent to disseminate reputation information about himself. In order to analyze the resistance of a RMS to a self-promoting attack, it is irrelevant whether it is conducted by distinct malicious agents or by multiple identities of a single malicious agent.

The classical way to perform the self-promoting attack is to introduce false positive information into the system through the gossip protocol. The RMS component capable of resisting to such attack is the *information fusion* mechanism, which determines the balance between gossiped information and direct experiences, and that can also assign different weights to information obtained, e.g., on the basis of the reputation of gossiper agents.

4.2 Slandering

The *slandering* attack [3] aims to decrease the reputation of some "victim" agents. Such attack may be performed by a single malicious agent, but in a large-scale system, an isolated intervention would have a limited effect. For this reason, slandering attacks are generally performed by a group of colluding agents. Also in this case, it is not relevant whether such malicious group is composed by distinct agents or by duplicated identities. Such an attack is typical of e-commerce systems, where the attacker aims to sabotage a competitor in order to obtain an indirect economic profit. The slandering attack is performed by introducing false negative information through the gossip protocol, and, similarly as the self-promoting attack, the *information fusion* mechanism is the main component that can let the system resist through an opportune balancing of gossiped information.

4.3 Whitewashing

The *whitewashing* attack [8] is performed by a malicious agent that wants to avoid the consequences of its bad past behavior. The malicious agent leaves the system and rejoins it with a new identity obtaining the default reputation value assigned to new users. The main vulnerability exploited by this type of attack is that a new agent receives a default reputation value comparable with the long-term reputation of a honest agent. RMSs that adopt such initial value have an optimistic approach and rely on negative feedbacks in order to discover a malicious behavior. A greater resistance is expected by RMSs that impose a low initial reputation value and use positive feedbacks to rise the reputation value. Moreover, a whitewashing attack can be reinforced by a combined self-promoting attack to extend its effect.

4.4 Traitor

In such type of attack, a *traitor* agent [19], with a bad reputation, acts honestly for a limited portion of time in order to increase its reputation. Once such goal is achieved, the traitor begins to abuse system resources again, and maintains an antisocial behavior until its reputation became too low; from this point the loop repeats, by alternating good and bad behaviors. A traitor attack is convenient in RMSs where the default reputation value is low, and thus a whitewashing attack does not produce an immediate benefit for the attacker, that has to behave honestly for a given time period before being able to obtain system resources. RMSs more vulnerable to such type of attack are those which weight the past history more than the recent experience.

5. SECURITY METRICS

The evaluation of a RMS requires new metrics to be defined while respecting some overall principles, partially inspired by the guidelines proposed in [20] and adopted in [22] to define some security metrics for distributed systems:

P1. the metrics should be assigned by measuring specific, unambiguous facts rather than abstract rules;

P2. the security of the RMS should be evaluated by considering all the observed vulnerabilities;

P3. the metrics should be intuitive;

P4. the metrics should be computed in an efficient way.

According to these principles, we propose to use a set of intuitive metrics (P3) that allow to measure the effort required to perform a specific attack (P1), and to evaluate and how long an attacker is able to exploit system resources:

m1) Time-to-compromise: the time necessary to reach the goal of an attack aimed to alter the reputation of some agents, i.e., the number of time steps required for the attack to succeed;

m2) Exploitation-time: the time a malicious agent is able to exploit the system resources before its behavior is detected;

m3) Collusion-complexity: the number of agents involved in cooperative attacks.

Following P2, we adopted distinct metrics for each attack so as to separately evaluate the robustness of RMSs against different threats. Each of the selected metrics is then mapped to a set of qualitative ratings that provide a textual representation of the numeric scores S, where $0 \leq S \leq 10$. The qualitative severity rating scale we used is that defined by the Common Vulnerability Scoring System (CVSS) [6], a well-established standard for classifying the severity of security vulnerabilities. In particular, the simulation framework rates the significance of each vulnerability as *none* (if $S < 0.1$), *low* ($0.1 \leq S < 4.0$), *medium* ($4.0 \leq S < 7.0$), *high* ($7.0 \leq S < 9.0$), *critical* ($9.0 \leq S \leq 10.0$). At the end of this analysis, the final user is notified of the vulnerabilities marked as *high* or *critical*.

The last principle, P4, is satisfied by guaranteeing that all the metrics are computed by the simulator quickly and automatically.

In order to determine whether a specific attack has achieved his goal, we also need to define the following success conditions (defined for a RMS in which reputation values are included in $[0, 1]$):

- **Self-promoting (SP)** is successfully completed if the long-term reputation of the non-cooperative, malicious agent, computed by one of its trustworthy neighbors, is greater than 0.5.
- **Slandering (S)** is successfully completed if the long-term reputation of the victim, computed by one of its trustworthy neighbors, is lower than 0.5.
- **Whitewashing (W)** is achieved within a temporal window T, if, in such window, the reputation of the non-cooperative, malicious agent, computed by one of its trustworthy neighbors, is greater than 0.5.

- **Traitor (T)** is achieved if the reputation of the malicious agent, computed by one of its trustworthy neighbors, is always greater than 0.5.

The long-term reputation is the reputation value observed after T_{max} time steps, given that at each time step an interaction between at least two agents occurs. The value of T_{max} has been set to 500 time steps by empirically analyzing the point when many state-of-the-art RMSs converge to a stable reputation values in a network composed by 100 agents and where each agent has 6 neighbors on average. The time T_{max} is also used both to scale *time-to-compromise* and *exploitation-time*, and to determine an intermediate threshold, $T_{th} = T_{max}/2$, needed to compute the *collusion-complexity*.

According to these definitions, being N the number of simulation time steps required to complete an attack, the time-to-compromise TTC is computed as:

$$TTC = N/T_{max}, \tag{1}$$

where $TTC = 1$ if the attack has failed.

When considering slandering or self-promoting attacks, the time-to-compromise also depends on the percentage of malicious agents that interact with the observer. For this reason different TTC values are computed while varying this percentage from 10-100 in 10% increments, and the average TTC is actually considered.

The *collusion-complexity*, CC, is defined as the percentage of corrupted agents needed for completing the attack within a time T_{th}.

Thus, the vulnerabilities of a RMS to slandering and self-promoting attacks can be expressed as:

$$\begin{aligned} V_S &= [1 - TTC(slandering)] \times [1 - CC(slandering)]. \\ V_{SP} &= [1 - TTC(self\text{-}prom)] \times [1 - CC(self\text{-}prom)]. \end{aligned} \tag{2}$$

The temporal metric that allows to evaluate whitewashing and traitor attacks is quite different from TTC.

In whitewashing attacks we can state that a malicious agent is maintaining his fraudulent behavior as long as his reputation is greater than a threshold $R_{th} = 0.5$. Thus, assuming that N time steps are required for the reputation to go under R_{th}, the *exploitation-time*, and consequently the vulnerability index, can be defined as:

$$V_W = ET(whitewashing) = N/T_{max}, \tag{3}$$

For traitor attacks, the *exploitation-time* is the time a malicious agent can act in a non-cooperative way before its reputation goes under a threshold $R_{th} = 0.5$ This condition is usually reached by alternating right and bad behavior for a certain number of time steps, N_{good} and N_{bad} respectively. Thus, the *exploitation-time* for traitor attacks, and consequently the vulnerability index, can be defined as:

$$V_T = ET(traitor) = N_{good}/(N_{good} + N_{bad}). \tag{4}$$

In order to obtain vulnerability scores in [0, 10] while having significant values in the whole range, a gamma correction, with $\gamma = 0.5$, and a scale factor $K = 10$, is applied:

$$V^* = K \times V^{\gamma}. \tag{5}$$

According to the previous definitions, the overall vulnerability of a RMS is given by a vector containing the list of the detected vulnerabilities V^* and the number of those marked as *high* or *critical*:

$$\begin{aligned} &RMS\_vulnerability = \\ &= \{V^*, num\_of\_V^*_{high\&critical}\} = \\ &= \{[V^*_{SP}, V^*_S, V^*_W, V^*_T], num\_of\_V^*_{high\&critical}\}. \end{aligned}$$

The extended simulation framework we describe in next Section not only allows to analyze the effects of different security attacks on the target RMS, but also makes a clear assessment of the RMS robustness by providing the user with an overall score computed on the basis of different vulnerability metrics.

6. PARALLEL SIMULATOR

To support the evaluation of the above described RMS, we propose here an extension of the simulation framework preliminarily described in [2] and [1], which aims to make easier the assessment of new reputation management strategies in different scenarios. The simulation is based on the synchronous time-discrete model proposed in [18] to describe synchronous distributed algorithms.

In order to separate low-level functionalities from the routines that describe the RMS's behavior, the software architecture of the simulator consists of two different logic layers.

The *reputation layer* is made of nodes connected to each other according to a customizable network topology provided by the user. This is the topmost layer, thus at this level of abstraction all the interactions between the agents occur in a totally distributed way, i.e., without the coordination of any central authority. The simulation environment is totally customizable by means of a set of of high-level configuration utilities that allow the user to specify the number of agents involved in the simulation, set the behavior and the cooperativeness degree of each agent, create ad-hoc network topologies, and specify the parameters of the reputation management strategy.

The level below is named *simulation layer*. Here, each agent is mapped to a distinct software process and the simulation proceeds in rounds according to a synchronous model, i.e., at each round all processes act simultaneously. At each iteration a process performs the operations requested to the corresponding agent: generate and send resource requests to the neighbors, evaluate incoming requests, determine a proper response, send resource responses, use local trust evaluation to update the current view of the reputation network, send updated reputation values by means of the gossip protocol, and update the information fusion mechanism. The set of processes running at the simulation layer also includes a leading process, that is responsible for coordinating the simulation by creating new processes or changing the behavior of those that already exist.

Since each process can run on a distinct remote host, the framework utilizes the Message Passing Interface (MPI) to enable communication among different distributed processes.

In order to evaluate the TTC index for slandering and self-promoting attacks, the framework performs different simulations by varying the percentage of malicious agents that interact with the neutral node, i.e., the observer, used for measuring the reputation of the victim (for slandering) or of the promoted agent (for self-promoting). For each run, the framework stores the reputation trend of the involved

agents, and the time step when the goal of the attack is achieved. The final value of the TTC metric is obtained by averaging all these time steps.

The evaluation of the CC index for slandering and self-promoting attacks requires to identify the smallest percentage of colluding agents needed to reach the goal of the attack goal within T_{th} time steps. For this purpose, it is possible to exploit the same simulation runs performed to evaluate the TTC index, so as to identify the percentage interval within which the goal is achieved. A further run is performed by selecting an intermediate percentage value inside such interval, in order to obtain the CC index with a 5% granularity.

For the whitewashing attack, the ET is simply obtained by analyzing the reputation trend of the malicious agent, observed by a neutral node, and by identifying the time step in which its reputation goes under the R_{th} value. This value is the normalized with respect to T_{max}.

In order to evaluate the ET metric for the traitor attack, different simulations are performed while varying with step of 5% the fraction of time during which the traitor acts non-cooperatively. The ET index is the greatest fraction which guarantees that the reputation of the attacker never goes under the R_{th} during the whole simulation.

7. EXPERIMENTAL EVALUATION

7.1 Case Study: the RMS

In order to evaluate the effect of the security attacks on RMSs designed according to different approaches, we adopted as case-study a RMS that explicitly manages, through tunable parameters, the relationship between past and recent experiences, and between direct experience and gossiped information.

This RMS includes the four components described in Section 3 and is inspired by [14] and [7]. The *local trust evaluation mechanism*, as in [14], requires that each agent i stores the number of satisfactory, $sat(i,j)$, and unsatisfactory, $unsat(i,j)$, transactions occurred with other agents j in the network, during the last time interval. The percentage of satisfied requests in such time interval is considered the *local trust*, lt_{ij}, which is defined according to the following equation:

$$lt_{ij} = \frac{sat(i,j)}{sat(i,j) + unsat(i,j)}. \quad (6)$$

The local trust value influences the reputation that i holds about j, weighted by a factor $\alpha \in [0,1]$:

$$lr_{ij}(t) = \alpha * lt_{ij} + (1-\alpha) * r_{ij}(t-1), \quad (7)$$

where lr_{ij} indicates the *local reputation*, that is the past reputation r_{ij} updated according to the last local experience.

The *gossip protocol* simply states that each agent sends its reputation values, r_{ij}, to all its neighbors, thus, each agent knows the opinion of all its one-hop neighbors about the reputation of its two-hop neighborhood.

The *information fusion* phase is inspired to [7], where each agent merges only information coming from reliable agents, i.e., those whose reputation is beyond a given threshold τ. Merged information is weighted with the reputation of the gossiper agents, and the resulting reputation value r_{ij} is obtained by linearly combining this weighted mean with the local reputation:

$$r_{ij}(t) = (1-\beta) * lr_{ij}(t) + \beta * \frac{\sum_{k \in K} r_{ik}(t-1) * r_{kj}(t-1)}{\sum_{k \in K} r_{ik}(t-1)}, \quad (8)$$

where β is a coefficient in $[0,1]$ and K is the set of reliable agents:

$$K = \{k : r_{ik}(t-1) \geq \tau\}. \quad (9)$$

The adoption of a weighted sum between the local reputation and the average values reported by other agents is a common solution in the literature [23, 24].

Finally, our RMS implements a simple *incentive mechanism* that allows an agent to obtain resources with a probability proportional to its reputation [5].

7.2 Experimental Results

In this section we provide a set of experiments aimed to understand which factors mostly influence the performance of a RMS under attack. Moreover, we show as such results may be concisely represented through the proposed vulnerability metrics.

The RMS which represents the base line for our evaluations, is characterized by the following parameters: the α factor, which weights the local trust to produce the local reputation, is $\alpha = 0.1$; the β factor, which weights the gossiped information to produce the final reputation, is $\beta = 0.1$; the default reputation value assigned to new users is $rep_{ij}(t_0) = 0.9$; the reputation threshold used to select reliable gossiper agents during the *information fusion* mechanism is $\tau = 0.4$; finally, the time interval considered for collecting data about the direct experience, needed to evaluate the local trust, consists of 30 time steps.

Such baseline parameters have been selected so as to obtain a RMS with a fair behavior and in which each attack (i) can be performed successfully and (ii) can be detected in a reasonable time. It is worth to specify that our goal is not to propose a specific RMS, but rather to show the potential of the proposed simulator for analyzing the security response of any RMS, and providing a score which depends on different vulnerability metrics.

The following experimental results have been obtained by simulating a network composed of 100 nodes, where each node has 6 neighbors on average, randomly selected ensuring that the network is not partitioned in isolated clusters.

As described in Section 4, slandering attacks are performed by a set of malicious agents that disseminate false negative feedbacks about a victim. In order to understand how the consequences of slandering attacks can be dependent on the gossip protocol used to propagate the reputation values through the reputation network, different simulations were run while varying the value of β (see equation 8) that weights direct and gossiped information. For example, $\beta = 0.2$ means that during the information fusion phase, the information from the gossiper agents is scaled by a factor 0.2, whilst the local reputation is weighted with a factor 0.8. Fig. 2 shows that the more is the importance of the gossiped reputation, the more is the effectiveness of the slandering attack.

Obviously, we expected that also the number of attacking agents affects the reputation of the victim. Such hypothesis was confirmed by a set of experiments performed while varying the percentage of malicious agents, from 0 to 100. Fig. 3

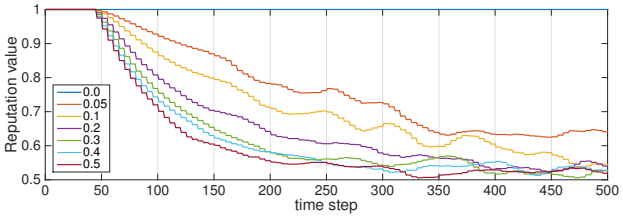

Figure 2: Slandering attack: reputation values of the victim agent as observed by neutral agents while varying the weight β of direct and gossiped information, when 50% of the agents interacting with the neutral agents are malicious.

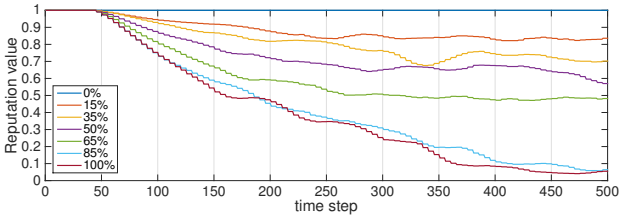

Figure 3: Slandering attack: reputation values of the victim agent as observed by neutral agents while varying the percentage of attacking agents.

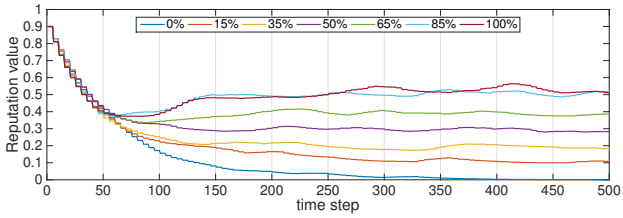

Figure 4: Self-promoting attack: reputation values of the selfish agent as observed by a trustworthy agent while varying the percentage of malicious agents supporting the attack.

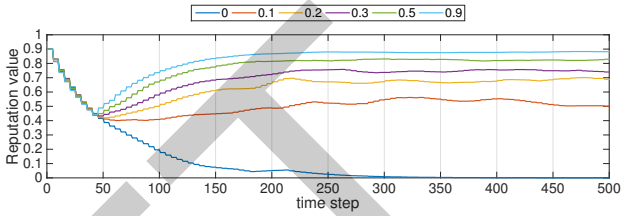

Figure 5: Self-promoting attack: reputation values of the selfish agent as observed by a trustworthy agent while varying the weight β of gossiped information.

shows the reputation of the victim as observed by neutral agents. As expected, the greater is the number of attacking agents, the more effective is the attack. However, even though the 65% of the agents in the network are malicious, the reputation of the victim decrease of only 0.5. As more agents are involved, the reputation of the victim continues to lower, approaching zero when the 85% of the agents are malicious.

Such a behavior is summarized by the slandering vulnerability index. As already said, the $TTC(slandering)$ is the average number of time steps required to put the reputation of the victim agent under $R_{th} = 0.5$, normalized with respect to $T_{max} = 500$, whilst $CC(slandering)$ is the percentage of colluding agents needed to complete the attack within $T_{th} = 250$ time steps.

Figure 3 shows that these conditions are satisfied when at least the 65% of the agents interact with the neutral observer. The resulting value for the slandering vulnerability metrics are the following:

$$TTC(slandering) = 0.71$$
$$CC(slandering) = 0.65$$
$$V_S^* = 3.18$$

The self-promoting attack exploits the same vulnerabilities exploited by the slandering attack, but with the goal of increasing the reputation of a non-coperative agent by means of false information provided by a set of colluding agents. Some experiments were conducted to understand how the reputation of this non-cooperative agent is perceived by a trustworthy agent that interacts directly with it. Fig. 4 shows that even though the behavior of the selfish agent remains unchanged during the simulation, its reputation increases the more malicious agent are involved in the attack. It is worth noting that, since the direct experience is negative, even if the malicious agents provide positive information, the low value of the local trust prevents the reputation value from rising over 0.5. Obviously, if gossiped information weight more than direct experience, such protection misses. Fig. 5 addresses this circumstance and describes a set of experiments where a trustworthy observer is surrounded only by dishonest agents which cooperate in order to promote the reputation of a malicious agent.

The experimental results show that our baseline RMS has a greater resistance to self-promoting attacks than to slandering attacks. Such feature is concisely represented by the self-promoting vulnerability metrics:

$$TTC(self-promoting) = 0.79$$
$$CC(self-promoting) = 0.85$$
$$V_{SP}^* = 1.78$$

The evaluation of the impact of the whitewashing attack, in a system that does not pose restriction on the creation of new accounts, corresponds to analyze how the reputation of a new user varies according to its behavior. The benefit of performing such attack depends on the default reputation value assigned to new users. Fig. 6 shows how the reputation of a new non-cooperative, malicious, agent that joins the network after 70 time steps varies according to the default reputation value. With the baseline settings the dishonest behavior is always discovered, but varying the default reputation value changes the time interval during which the malicious agent is able to abuse the community resources. For such reason, the main defense technique is to assign a low reputation value to new agents that join the system; the weakness of this solution is that a low initial reputation is also assigned to new trustworthy agents, that, consequently, experience a slow start phase, as shown by Fig. 7.

Moreover, a low default reputation value may be very dis-

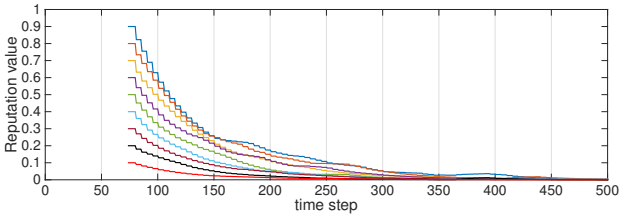

Figure 6: Whitewashing attack: reputation values of a non-cooperative agent that joins the network after 70 time steps, while varying the initial reputation value assigned by the RMS.

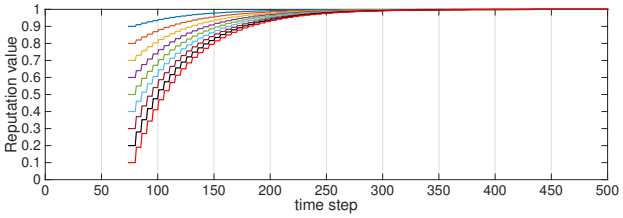

Figure 7: Whitewashing attack: reputation values of a trustworthy agent that joins the network after 70 time steps, while varying the initial reputation value assigned by the RMS.

advantageous if an *incentive mechanism* is automatically executed in order to provide each agent with a percentage of resources proportional to its reputation. Indeed, if a new user is evaluated with a low reputation, the other agents will give him few resources. As a consequence, its opinion of other agents will decrease, and it will give them few resources, thus confirming its bad reputation. In other words, if the *incentive mechanism* is blindly performed, new honest users are not able to improve their initial reputation, as shown in Fig. 8. It is worth noting that during the experiments described by Fig. 7, honest agents always satisfy the received requests, i.e., they do not apply the *incentive mechanism*.

With the baseline settings, which adopts 0.9 as initial reputation value, the corresponding vulnerability metrics have the following values:

$$ET(whitewashing) = 0.054$$
$$V_W^* = 2.32$$

In order to evaluate the impact of the traitor attack, we performed some experiments where a malicious agent alternates cooperative and partially cooperative behavior (i.e., only 50% of received requests are satisfied). The honest behavior is maintained for a longer time interval in order to guarantee that the reputation value increases enough. As expected, Fig. 9 shows that RMSs with higher values of α, i.e., the weight of recent experience, detect earlier a change in agent behavior.

Moreover, experimental results show that the percentage of time during which the traitor can behave non-cooperatively without being detected is 20%, against a percentage of 35% for a partially cooperative behavior. The resulting vulnerability index is then:

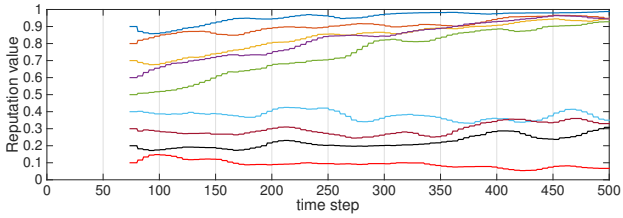

Figure 8: Whitewashing attack: reputation values of a trustworthy agent that joins the network after 70 time steps, while varying the initial reputation value assigned by the RMS, with an automatic *incentive mechanism* enabled.

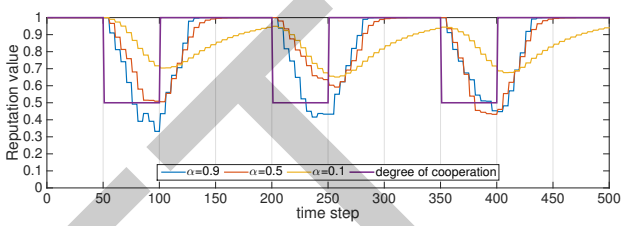

Figure 9: Traitor attack: reputation values of a traitor agent which alternates honest and non-cooperative behaviors, while varying the α factor that weights the recent experience to build the local reputation.

$$ET(traitor) = 0.25$$
$$V_T^* = 5.00$$

In conclusion, the baseline RMS does not exhibit high or critical vulnerabilities, whilst a medium vulnerability to traitor attack is detected. These characteristics are summarized by the following vulnerability indexes:

$$RMS\_vulnerability =$$
$$= \{V^*, num\_of\_V^*_{high\&critical}\} =$$
$$= \{[V^*_{SP}, V^*_S, V^*_W, V^*_T], num\_of\_V^*_{high\&critical}\} =$$
$$= \{[1.78, 3.18, 2.32, 5.00], 0\}.$$

8. CONCLUSIONS AND FUTURE WORK

In this work we presented a set of vulnerability metrics to quantitatively evaluate the behavior of a Reputation Management System under attack. These metrics allow to compare the effects of different design choices on the overall security of the RMS. We used a parallel simulation framework for automatically compute the vulnerability indexes associated to four state-of-the-art attacks. This tool allowed us to test the security of a case study RMS while varying the behavior of its inner components, namely local trust, gossiped information and information fusion, and some general characteristics, such as the default reputation values assigned to new agents.

Results showed that the impact the different attacks have on the RMS deeply depends on the choices made since its design. For example, in order to overcome a slandering attack a RMS should rely more on the direct experience than

gossiped information, whilst is recommended to give more importance to the direct and recent experience when dealing with traitor attacks.

As future work, we plan to experimentally analyze the security of some of the most popular distributed RMSs, and to consider possible attacks to the system architecture. In particular, masquerading attacks could allow malicious agents to pretend to be loyal agents, so as to diffuse false feedbacks through the gossip protocol. Moreover, we are considering to extend our analysis to RMSs operating within large-scale social networks [11, 12].

9. REFERENCES

[1] V. Agate, A. De Paola, S. Gaglio, G. Lo Re, and M. Morana. A framework for parallel assessment of reputation management systems. In *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech)*, june 2016.

[2] V. Agate, A. De Paola, G. Lo Re, and M. Morana. A simulation framework for evaluating distributed reputation management systems. In *Proc. of the 13th Int. Conf. on Distributed Computing and Artificial Intelligence (DCAI'16)*, pages 1–8, 2016.

[3] S. Ba and P. A. Pavlou. Evidence of the effect of trust building technology in electronic markets: Price premiums and buyer behavior. *MIS quarterly*, pages 243–268, 2002.

[4] P. Chandrasekaran and B. Esfandiari. Toward a testbed for evaluating computational trust models: experiments and analysis. *J. of Trust Management*, 2(1):1–27, 2015.

[5] C. Crapanzano, F. Milazzo, A. De Paola, and G. Lo Re. Reputation management for distributed service-oriented architectures. In *Proc. of the 2010 Fourth IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshop (SASOW)*, pages 160–165, 2010.

[6] CVSS. Common vulnerability scoring system v3.0. `https://www.first.org/cvss`, 2015.

[7] A. De Paola and A. Tamburo. Reputation Management in Distributed Systems. In *Proc. of the 3rd Int. Symp. on Communications, Control and Signal Processing (ISCCSP)*, pages 666–670, 2008.

[8] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. In *Proc. of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 228–236. ACM, 2004.

[9] P. Fouliras. A novel reputation-based model for e-commerce. *Operational research*, 13(1):113–138, 2013.

[10] K. K. Fullam, T. B. Klos, G. Muller, J. Sabater, A. Schlosser, Z. Topol, K. S. Barber, J. S. Rosenschein, L. Vercouter, and M. Voss. A specification of the agent reputation and trust (ART) testbed: experimentation and competition for trust in agent societies. In *Proc. of the 4th Int. joint Conf. on Autonomous agents and multiagent systems*, pages 512–518, 2005.

[11] S. Gaglio, G. Lo Re, and M. Morana. Real-time detection of Twitter social events from the user's perspective. In *Proc. of the 2015 IEEE Int. Conf. on Communications (ICC)*, pages 1207–1212, 2015.

[12] S. Gaglio, G. Lo Re, and M. Morana. A framework for real-time Twitter data analysis. *Computer Communications*, 73, Part B:236 – 242, 2016.

[13] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Computing Surveys (CSUR)*, 42(1):1, 2009.

[14] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proc. of the 12th Int. Conf. on World Wide Web*, pages 640–651, 2003.

[15] R. Kerr and R. Cohen. Smart cheaters do prosper: defeating trust and reputation systems. In *Proc. of the 8th Int. Conf. on Autonomous Agents and Multiagent Systems-Volume 2*, pages 993–1000, 2009.

[16] J. Lee, J. Zhang, Z. Huang, and K.-J. Lin. Context-based reputation management for service composition and reconfiguration. In *Proc. of the IEEE 14th Int. Conf. on Commerce and Enterprise Computing (CEC)*, pages 57–61, 2012.

[17] Q. Lian, Z. Zhang, M. Yang, B. Y. Zhao, Y. Dai, and X. Li. An empirical study of collusion behavior in the maze p2p file-sharing system. In *Proc. of the 27th Int. Conf. on Distributed Computing Systems (ICDCS'07)*, pages 56–56. IEEE, 2007.

[18] N. A. Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[19] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[20] M. K. Reiter and S. G. Stubblebine. Authentication metric analysis and design. *ACM Trans. Inf. Syst. Secur.*, 2(2):138–158, May 1999.

[21] Y. Sun and Y. Liu. Security of online reputation systems: The evolution of attacks and defenses. *IEEE Signal Processing Mag.*, 29(2):87–97, 2012.

[22] L. Wang, A. Singhal, and S. Jajodia. Toward measuring network security using attack graphs. In *Proceedings of the 2007 ACM Workshop on Quality of Protection*, QoP '07, pages 49–54, New York, NY, USA, 2007. ACM.

[23] J. Weng, Z. Shen, C. Miao, A. Goh, and C. Leung. Credibility: How agents can handle unfair third-party testimonies in computational trust models. *IEEE Trans. on Knowledge and Data Engineering*, 22(9):1286–1298, 2010.

[24] H. Yu, S. Liu, A. Kot, C. Miao, and C. Leung. Dynamic witness selection for trustworthy distributed cooperative sensing in cognitive radio networks. In *Proc. of the 2011 IEEE 13th Int. Conf. on Communication Technology (ICCT)*, pages 1–6, 2011.

[25] H. Yu, Z. Shen, C. Leung, C. Miao, and V. R. Lesser. A survey of multi-agent trust management systems. *Access, IEEE*, 1:35–50, 2013.