Properties of Reputation Lag Attack Strategies

S. Sirur
University of Oxford
United Kingdom
sean.sirur@gmail.com

Tim Muller
University of Nottingham
United Kingdom
t.j.c.muller@gmail.com

ABSTRACT

The trustors in a reputation system share trust-relevant information about trustees; their reputation. The presence of lag in the sharing mechanism can be exploited by malicious trustees to perform otherwise impermissible additional bad actions. This is the reputation lag attack. In this paper, we use simulations to explore properties of the reputation lag attack and strategies which improve the attacker's success. We demonstrate the following key findings: Attackers in lagged systems clearly outperform attackers in lag-free systems. The attacker's success is proportional to the rate at which they can interact with victims, plateauing at a maximum. Attackers who wait for their good reputation to disseminate outperform those who do not. Attackers who perform only good actions, wait for dissemination and then perform only bad actions outperform attackers who do not follow this ordering. This implies reputation-lag attacks are effective exit strategies. In typical social networks, smart attackers may *cheat* users with a low centrality, but in a homogeneous network, this strategy is ineffectual. Our findings help allow developers of reputation systems defend against a class of attacks that has not yet received a great deal of attention.

KEYWORDS

Reputation System; Social Simulation; Reputation Lag; Attacks

ACM Reference Format:
S. Sirur and Tim Muller. 2022. Properties of Reputation Lag Attack Strategies. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

A *trustor* interacting with a *trustee* whilst uncertain of their cooperation can be considered to be *trusting*. Trust raises the risk of interacting with a malicious party. Trustors can reduce their individual uncertainty and risk by sharing information about trustees. We consider this sharing of information to be an instance of a reputation mechanism. The reputation of a trustee from each trustor's perspective is comprised of the information each trustor knows about that trustee.

Reputation systems may possess a weakness in the form of lag. A source of lag is the time delay present in user-to-user communication. As a result of such delays, some users may be unaware of new information when deciding (not) to trust. A malicious trustee may attempt to actively exploit this lag. This is known as a *reputation lag attack*.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

We use a simulated multi-agent network to model a reputation system in which users share information about an attacker. This attacker attempts positive and negative transactions with individual users, aiming to maximise how many negative transactions they successfully perform. Reputation is modelled by users sharing the outcomes of these transactions with each other and then using that knowledge to decide whether or not to trust the attacker.

This paper discusses factors and strategies which benefit the attacker. We investigate the the attacker's choice of action: the attacker can *deal*, *cheat* or wait at each opportunity. We show the superiority of attacker action "ordering" (perform all planned *deals*, wait for the news to spread and then begin cheating). This strategy's success highlights the reputation lag attack a malicious exit strategy. We study the effect of the attacker avoiding users with high centrality values to minimise spreading *cheats* and the impact of network structure on said strategy. We study an attacker who accesses user trust states before attempting transactions, allowing them to avoid distrustful users.

Our simulation models users as continuous-time stochastic processes, that communicate with each other as defined by a network, akin to a social network. The attacker is offered opportunities to act according to a continuous-time stochastic process too, but the attacker's choices are determined by the attack strategy. This allows us to compare impact of: how often the attacker can act, the network topology, and the attacker's strategy.

We contribute novel insights, in addition to quantifying known qualitative relations. For the former: the impact of using centralities to target victims; the relationships between strategies (e.g. the impact of rate on waiting duration and centrality targeting); and the benefit to the attacker of knowing user trust states before committing to an action. For the latter: the sigmoid relationship between the log of the attack rate and the number of cheats, with the system rate being the midpoint of the sigmoid; the linear relationship between successful deals and cheats, with the gradient of the line increasing with ordering and waiting; and the success of the RLA as an exit strategy.

2 RELATED WORK

The reputation lag attack (RLA) was first identified in security literature by Kerr and Cohen [8] who studied it amongst other trust attacks in a simulated marketplace [9]. However, their informal definition of the attack was restrictive, lessening its apparent impact. Almost all the other attacks benefitted somewhat from reputation lag but this was not attributed to the RLA. The attack remained largely unstudied after this, presented only in some trust attack surveys [7, 10].

Sänger & Pernul tested a visualisation-based detector against the RLA [13]. They identified two signs of the attack: an increase

in the number of sales and an increase in the time to rate transactions due to withheld deliveries. They visualise the "not yet rated" transactions to highlight unusual behaviour. Their system assumes the attacker remains in the network whereas we use the RLA more as an exit strategy.

A recent work by Voloch, Gudes & Gal-Oz tested a proposed privacy-defending trust model against a trust-delay based attack [17]. In a simulated online social network (OSN), a subgraph of malicious users try to present a trustworthy front to trick a specific user into breaking their privacy. In this defensive model, the size of the subgraph necessary for a successful attack was infeasibly large.

Economic analyses of reputation lag vulnerabilities exist. Shapiro shows the benefit of buyer knowledge lag for malicious sellers: lagged optimal product quality is lower than under perfect information [14]. He also shows that sellers may seek short-term gains by exploiting buyers' lack of foresight, unless compensated by selling high quality items above cost [15].

A cunning attacker may behave well to users who contribute greatly to spreading and badly to users who do not. Centralities provide a metric for the structural importance of a node [18]. Highly central nodes should contribute more to spreading [3].

Various centrality measures exist, each calculated from different properties of the network [12]. In particular, there are four well-known centrality measures: degree centrality (*DC*) [11], eigenvector centrality (*EC*) [2, 11], closeness centrality (*CC*) [5, 11] and betweenness centrality (*SPBC*) [4, 11]. Borgatti argues that, while each centrality predicts some form of "spreading on networks", none predict a node's contribution to "infection" or "gossip" style spreading [3] of the sort present in our model.

3 MODEL SPECIFICATION

In this paper, we study a simulated model of the reputation lag attack. The foundation of the model is a stochastic, continuous-time Markov chain (CTMC) implemented in Python [1]. The model consists of a multi-agent network of users and a malicious reputation lag attacker. The rationale behind using a continuous-time model is that timing is extremely important for modelling reputation lag, and it allows us to increase the attacker's rate without the issue of discrete turns distorting results. The continuous progression of time results in a smooth increase in the probability that the attacker's reputation spread. Our choice for stochastic processes (over deterministic) is because the agents we simulate are not deterministic.

The model's fundamental behaviour is as follows: Attacker actions are attempted transactions with individual users. The attacker is free to choose any victims, unconstrained by the network. These actions can have a positive or negative polarity depending on if they benefit the user (a *deal*) or harm them (a *cheat*). The attacker's primary objective is to perform as many *cheats* as possible.

Users use known transaction outcomes to determine the attacker's trustworthiness. This knowledge can also be shared with others via pairwise communications. Users that distrust the attacker will no longer accept transactions. Each attacker action and pairwise communication is a CTMC event. The simulation stops when it reaches a state in which the attacker is sufficiently distrusted that they can no longer act.

[1]The code can be found at https://github.com/Se-Si/reputation-lag/

3.1 Users

The system contains a static set of users $\mathcal{U}$. The i^{th} user is denoted as u_i. They are the trustors of the system. Every user has a set of messages M_i which is initially empty and which is a subset of the complete set of messages in the network $\mathcal{M}$. Each message contains information about a single transaction: the outcome, the victim and the time of occurrence. Each user u_i will use only its own set of messages M_i to decide whether the attacker is trustworthy or not. The subset of the messages with *deal* (*cheat*) as outcome is denoted as M_i^{deal} (M_i^{cheat}).

Every user u_i has a judgement function δ_i that takes in their set of messages and returns a binary value to trust or not to trust. This defines the attacker's reputation from the perspective of u_i. For u_i, the decision to accept a transaction depends solely on δ_i. This avoids confounding effects from other potential attacker vectors.

The judgement function δ applied by all users is the same:

$$\delta_i(M_i) = \delta(M_i) = |M_i^{cheat}| - |M_i^{deal}| \leq \tau \tag{1}$$

Where τ is a threshold set to 4 in our simulations.

There are two ways to interpret this threshold. It could denote a preexisting degree of "tolerance" exhibited by each user. It can also be interpreted as the good reputation that the attacker has built prior to events captured in the simulation. Section 5.1.3 provides further discussion on this interpretation.

Users have undirected, pairwise connections called "edges" in the network. Every user has at least one edge connecting it to another user. There are no self-edges. An edge denotes that there is probability that those two users communicate. After a communication, they learn all information about the attacker from each other. In other words, $M_i \leftarrow M_i \cup M_j$ and $M_j \leftarrow M_i \cup M_j$.

The time intervals between pairs of events in a CTMC are exponentially distributed with some constant event rate r. As such, the probability that two users u_i and u_j communicate at a given time t is exponentially distributed with constant rate r_{ij}:

$$P\Big(comm(i,j) \text{ at time } t\Big) = r_{ij}e^{-r_{ij}t} \tag{2}$$

where $r_{ij} = 1$ if they are connected and $r_{ij} = 0$ if not. We also define the *system rate* as the total sum of rates:

$$r_{sys} = \sum_{i,j \in \mathcal{U}} r_{ij} \approx 4000 \tag{3}$$

where the edge rates are 1 and the number of edges is 3900 for BA graphs and 4000 for CWS graphs as described in Section 3.3.

If, after a communication, $\delta_i(M_i)$ is false (*cheat*s minus *deal*s exceeds threshold τ), the u_i will reject any transaction attempt, causing the attacker to waste that opportunity. It is possible for such a user to learn new information from other users such that they re-evaluate the attacker as trustworthy. If all users have gone sour of the attacker and no new information is sufficient to change anyone's mind, we say the network is *saturated*. Upon the network becoming saturated, the simulation stops, as the attacker can no longer interact with any users.

3.2 Attacker

The malicious attacker $\mathcal{A}$ may attempt to transact with any user they wish, when given the opportunity to act. The attacker's opportunities to act happen at random intervals, with some rate in a continuous-time Markov chain. The probability that the attacker has an opportunity to act at a given time t is exponentially distributed with rate $r_{\mathcal{A}}$.

$$P(action \text{ at time } t) = r_{\mathcal{A}} e^{-r_{\mathcal{A}} t} \tag{4}$$

For each of their turns, the attacker can attempt the following actions: *deal*, *cheat* or choose not to act (i.e. to just wait). The RLA has been discussed previously in marketplace contexts. The terms "*deal*" and "*cheat*" reflect this. However, as we present a security-focused analysis, we wish to avoid restricting the scope of the attacker to that of a seller. Hence, "*deal*" and "*cheat*" simply represent some positive or negative interaction with the users respectively. The choices the attacker makes when given the opportunity to act comprise the attacker's *strategy*. The attacker seeks to employ a strategy which best manages their reputation to allow them to perform the maximal *cheats*. This is achieved when each user is aware of the same (relatively) small set of *deals*, but a different equal-sized set of *cheats*. Therefore, the theoretical maximal *cheat* values for both a single user u_i and the entire system depend solely on the number of *deals*, the judgement function threshold τ (as defined in Equation 3.1) and the size of the set of users $\mathcal{U}$:

$$max(|M_i^{cheat}|) = max(|M_i^{deal}|) + \tau + 1 = |\mathcal{M}^{deal}| + \tau + 1 \tag{5}$$

$$max(|\mathcal{M}^{cheat}|) = \sum_{i=1}^{|\mathcal{U}|} max(|M_i^{cheat}|) = |\mathcal{U}| \cdot (|\mathcal{M}^{deal}| + \tau + 1) \tag{6}$$

In all sections where the attacker performed *deals*, it was necessary to cap the number of *attempts* the attacker had to *cheat*. Once the attacker has used all their *cheat* attempts, the simulation is stopped. Note that this value is distinct from the maximal cheats $max(|\mathcal{M}^{cheat}|)$ defined in Equation 3.2. The chosen *cheat* cap was 80000 with smaller scale uncapped sub-tests being used to determine that, in reality, the attacker's last successful *cheat* attempt came far before their 80000^{th} attempt.

The *cheat* cap was a pragmatic decision to keep the run-times of the simulation within a reasonable bound. When the network is almost saturated, a number of communications is required to fully saturate the network but almost no *cheat* attempts will be successful in the meantime. This can lead to disproportionally long run-times with little meaningful behaviour for very high rate attackers: a vast number of attacker actions must be simulated per remaining communication yet most if not all of these will be rejected. The *cheat* cap prevents this.

3.3 Network

The networks in this model represent the communication structure of the users. For example, the network edges could represent "friends" on an OSN, nearby nodes on an ad-hoc network or corresponding enterprises on a business network. The attacker's interactions are not bounded by the network. As such, they could represent a common mutual "friend" on an OSN, a node in a central location of an ad-hoc network or a common news source.

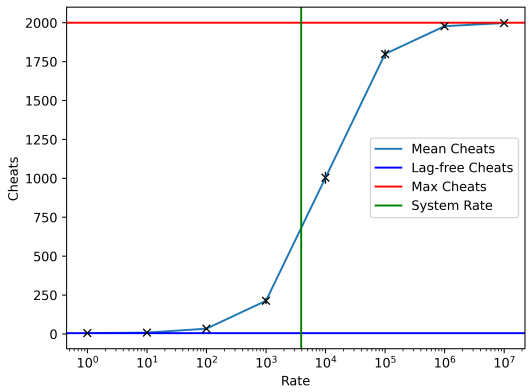

Figure 1: Number of successful *cheats* at various attacker rates.

The user networks in this paper were randomly generated. Every attacker was simulated for multiple runs, with the outcomes of these runs being aggregated. Each run used a different network drawn from a fixed set of networks, the networks are fixed to reduce noise.

The primary network model used to generate the user networks was the Barabási-Albert (BA) model [1]. This model is parameterised by the total number of nodes n and the number of neighbours m that each new node connects to as the network is grown. In this case, $n = 400$ and $m = 10$ resulting in 3900 edges. The networks were generated using the NetworkX [6] Python package.

In Section 5.2.2, centrality metrics were compared on Connected Watts-Strogatz networks [19]. This model is parameterised by the total number of nodes n, the number of neighbours k and rewiring probability rp. Three sets of parameters were used. In all cases, $n = 400$ and $k = 20$ resulting in 4000 edges. However, three rp values were used: 0.001, 0.01 and 0.1. This was to cover the range of usable rp values found by Watts and Strogatz [19].

4 REPUTATION LAG AND ATTACKER RATE

To demonstrate that the reputation lag attack is a vulnerability in a particular system, it is sufficient to demonstrate that an attacker in the lagged version of the system can perform *cheat* actions that an attacker in the lag-free version cannot. In a lag-free system, all users acquire new information instantly. For all users $u_i \in \mathcal{U}$, $M_i = \mathcal{M}$.

The crucial attacker property here is their rate of action. This dictates how frequently the attacker may attempt to transact with the users and, as such, can be seen to dictate the "degree of lag". Figure 1 was generated by an attacker that chooses random targets and only *cheats*. It shows that their average success is proportional to their rate. The red line denotes the maximal *cheats* that can be achieved *when no deals are performed*. As defined in Equation 3.2, $max(|\mathcal{M}^{cheat}|) = 2000$ for our system where $|\mathcal{U}| = 400$ and $\tau = 5$. (Note: an attacker who performs *deals* can exceed this value.)

The plot of the logarithm of the rate resembles a sigmoid function. At low rates, the number of *cheats* is around that of a lag-free system. Then, the attacker's success begins rapidly increasing. The centre of the sigmoid rests where the attack rate matches the system rate

Table 1: Difference in mean *cheats* between randomly dealing attackers and the non-dealing attacker all at rate 10^7.

Deal Cap	4	16	64	256
Cheats Difference	4.05	15.86	64.22	255.95

(defined in Equation 3.1 as $r_{sys} \approx 4000$). Eventually the increase tapers, approaching the maximal value $max(|\mathcal{M}^{cheat}|)$.

5 ATTACKER STRATEGIES

In reality, it would be infeasible for the attacker to increase their action rate arbitrarily. Two other key variables in attacker's behaviour are their choice of action and choice of user (if the chosen action is a *deal* or *cheat*). In this section, we explore how these remaining two factors impact the success of the attacker and how the attacker might use them to their advantage.

5.1 Choice of Attacker Action

At each opportunity, the attacker must choose to perform an action or to wait. The potential actions are to *deal* or *cheat* an arbitrary user. That user may refuse to interact with the attacker, in which case no action is recorded. First, we explore the impact of including a limited number of randomly distributed *deal* actions in the attacker's strategy, showing that it increases the number of *cheats* the attacker successfully performs. Next, we investigate the attacker's success if they randomly wait between *deals* and *cheats*. Finally, we discuss the importance of the order of these behaviours. We show that an attacker which attempts all of their *deals* then waits for some time before performing *cheats* will perform optimally.

5.1.1 Random Deals. Here, we discuss the effect of randomly attempting *deals* in addition to *cheats*. A collection of 32 attackers of different rates which randomly perform *deal* and *cheat* messages were simulated. For each attacker, a cap was placed on the number of *deals* they could attempt. This was varied by factors of 4 from 4 to 256. The rate was varied by factors of 10 from 1 to 10^7. First, we note that Figure 2 shows that the rate has the biggest impact on the success of the attackers. However, it can also be seen that the number of *cheats* performed at each rate increases with the number of *deals* the attacker may attempt.

A vital question is that of how much better the dealing attackers did in comparison to the non-dealing attacker. Table 1 illustrates the difference in mean *cheats* between a dealing attacker and a non-dealing attacker for each *deal* cap (at rate 10^7). From Table 1, it is clear that attacker is only performing a single additional *cheat* for each *deal* that it performs. This is expected if there were no reputation lag, meaning mixing deals amongst cheats is ineffective in combination with the reputation lag attack.

Unsuccessful attempts to *cheat* made by the attacker are shown in Figure 3. The number of rejected *cheats* varies only very slightly with the *deal* cap. This is likely because the attacker only gains one additional *cheat* for each *deal*, allowing them to be just as unsuccessful otherwise. Further evidence for this is discussed in sections 5.1.2 and 5.1.3.

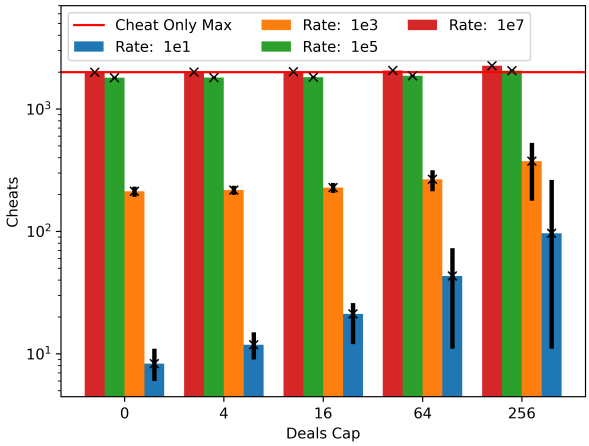

Figure 2: Success of unordered attackers at various rates and *deal* caps.

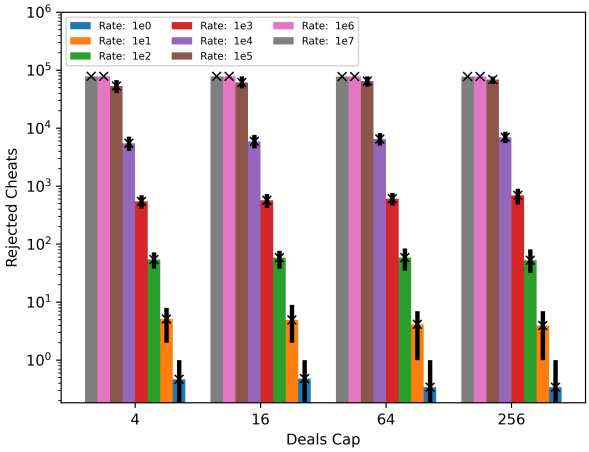

Figure 3: Unsuccessful *cheat* attempts of unordered attackers at various rates and *deal* caps.

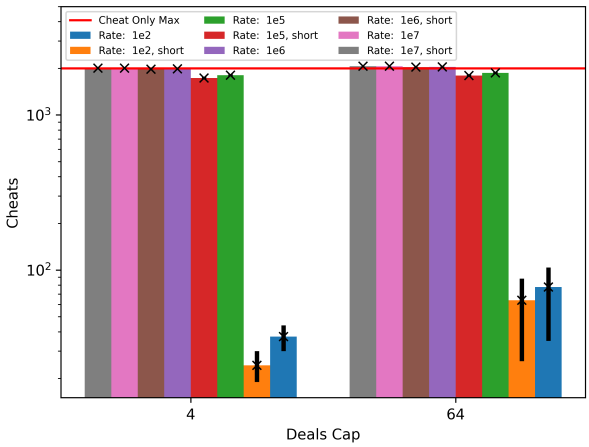

Figure 4: Success of non-waiting versus short waiting unordered attackers at various rates and *deal* caps.

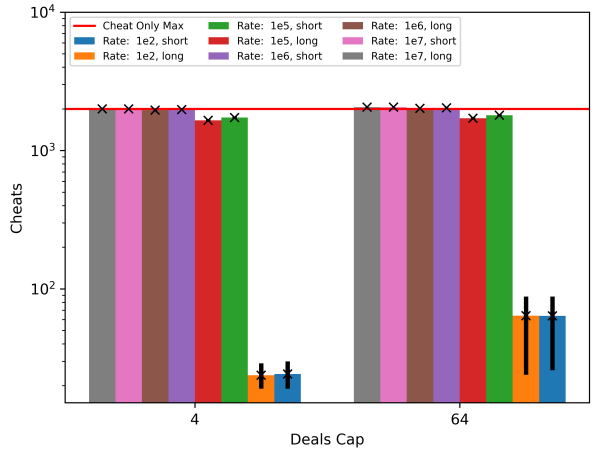

Figure 5: Success of short waiting versus long waiting unordered attackers at various rates and *deal* caps.

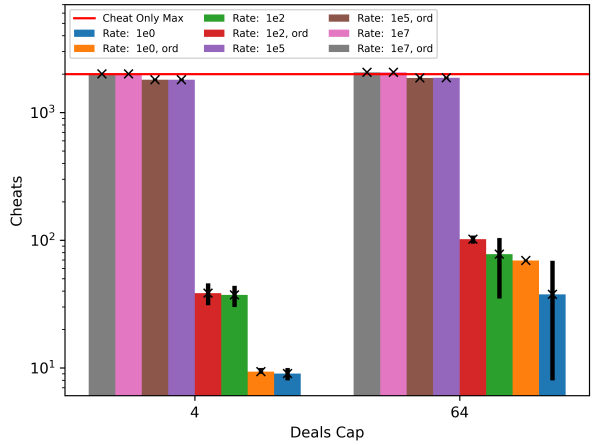

Figure 6: Success of unordered versus ordered non-waiting attackers at various rates and *deal* caps.

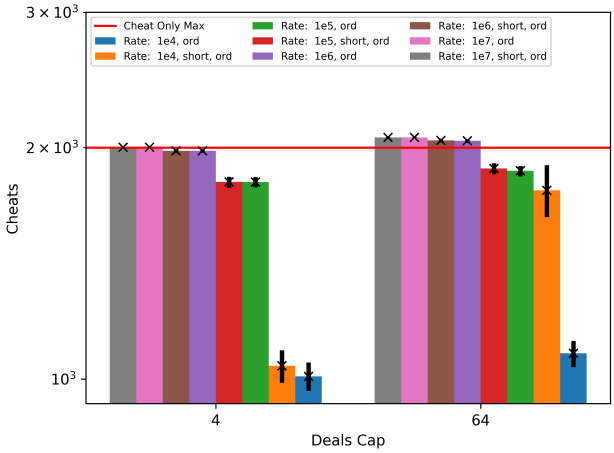

Figure 7: Success of non-waiting versus short waiting ordered attackers at various rates and *deal* caps.

5.1.2 Waiting. Attackers who perform *deals* out-perform attackers who do not but are also rejected often with high rate attackers consistently reaching the *cheat* cap (Figure 3). Table 1 demonstrates that these attackers only gain a single *cheat* per *deal*due to insufficient spreading of the attacker's *deals*.

Reputation spread can work in favor of the attacker, if the reputation is mostly positive. While the attacker wishes to avoid the spread of *cheat* messages, they wish for *deal* messages to spread as much as possible. Well-spread *deals* provide the greatest number of opportunities for subsequent *cheats*.

The attacker may allow their *deals* to spread more effectively by *waiting*. By intentionally not taking opportunities to act, the attacker allows their good reputation to spread. Here, we investigate the impact of randomly waiting whilst continuing to randomly *deal* and *cheat*. We do so by simulating 48 attackers. Half of the attackers are capped at 4 *deals* whilst the other half are capped at 64 *deals*. The attacker rates vary from 100 to 10^7 by a factor of 10. The omission of the rate 1 and rate 10 attackers is due to prohibitive simulation run-times when waiting. Similarly, half the attackers can wait for a total duration of 10^3 time steps and the other can wait for a total duration of 10^4 time steps.

Long-duration waiting attackers appear to have a small but consistent decrease in success compared to short-duration waiting attackers. The above findings combined with the small, directly-proportional constant improvement seen in Section 5.1.1 demonstrate that performing *deal* amongst *cheats* provides little benefit to the attacker. In the event that the attacker does not wait, each *deal* provides only one additional *cheat*. When the attacker does wait, their performance simply worsens.

These are reasonable outcomes. *deals* have the most impact when each *deal* is known by many users and each *cheat* is known by few. By mixing the performance of *deals* with that of *cheats*, the attacker is effectively nullifying their primary benefit. In Section 5.1.3, we explore a strategy that takes advantage of these facts to construct a superior strategy with respect to when the attacker chooses to *deal*, *cheat* or wait.

5.1.3 Ordering. Above, we demonstrated that, on average, it is somewhat beneficial to add some random *deals* to the attacker's behaviour but that it is detrimental to randomly wait. Here, we investigate the impact that the order of these actions have on the attacker's success.

Given a finite, constant number of *deals* present in the system, the attacker can only perform the maximum possible number of *cheats* if each user is aware of every *deal* in the system but only the *cheats* that resulted from their own transactions with the attacker.

For the attacker to guarantee these conditions, the *deals* it performs must spread completely through the system before any *cheats* do. A clear limitation of the randomly attackers introduced in 5.1.2 is that waiting entailed spreading *cheats* in addition to *deals*.

To ensure that waiting only spreads *deals*, the attacker can act as follows: perform all possible *deals*, wait for some time and then begin attempting *cheats*. We refer to this strategy as *ordering*. The superiority of this strategy (under certain basic assumptions) has been shown formally [16]. We attempt to empirically investigate the quantitative impact of ordering in terms of the attacker's *deal* cap and their total waiting duration.

First, we compare unordered and ordered attackers that do *deal* but do not wait in Figures 6. Next, we compare ordered attackers that do not wait with ordered attackers that do wait for a short duration (10^3 time steps) in Figure 7. Finally, we compare ordered attackers that wait for a short duration with those that wait for a long duration (10^4 time steps) in Figure 8.

Figures 6-8 demonstrate the superiority of ordering. With ordering, attackers who wait outperform non-waiting attackers. We note that, for the fastest and slowest attackers, the difference between long waiting times and short waiting times is minimal with the biggest effect being seen at rates 10^4 and 10^5. Also, the markedly increased success of the rate 10^5 attacker. The large divergence in this attacker's performance will be discussed in Section 5.1.4

5.1.4 Optimal Waiting Times. We assert that the reason for this large increase at rates 10^4 and 10^5 in Figure 8 is a result of two phenomena. Firstly, the the attacker must wait a sufficiently long time for *deals* to spread around the network. Secondly, the attacker must be sufficiently fast to *cheat* the users before these *cheats* spread around the network. Attackers with rate less than 10^4 are too slow and their bad reputation outpaces their ability to cheat. On the other hand, the attackers with rate greater than 10^5 do not wait a sufficiently long time due to the waiting period being relative to the attacker's rate rather than the system rate.

In Figure 9, we demonstrate that the success of a sufficiently fast attacker who waits long enough will achieve the maximal number of *cheats* for their *deal* cap of 64. As defined in 3.2, the maximal value here will be 27600 (as indicated by the red line).

We offer the following explanation. As the attacker performs all of their *deals* before waiting, they only need to wait long enough for the last *deal* to spread to guarantee that, on average, all *deals* are spread. In the simulation, the mean time for a single *deal* to spread throughout the system was found to be 0.835, with a 10^{th} percentile of 0.674 and a 90^{th} percentile of 1.000.

Given that the mean is very close to 1 time step, the attacker need only wait around 1 time step for the *deals* to spread. Furthermore, the attacker rate defines the number of attacks, on average, that an attacker can perform in 1 time step. As seen in Figure 9, it is only once each attacker waits at least 1 time step worth of opportunities that they begin to approach the maximal *cheats*.

These results evidence three key points. It demonstrates the attacker's rate to be their most crucial asset. Secondly, it shows that efficient communication in a social network (model) is only a benefit against a sufficiently slow attacker. Finally, it highlights the reputation lag attack as a malicious exit strategy.

5.2 Choice of User/Victim

The attacker attempts each transaction with a specified user. The chosen user impacts the outcome of current and future attempts. Attempting to transact with a mistrustful user effectively wastes a turn. Cheating talkative users risks spreading *cheats* across the network faster than cheating quiet users.

Here, we explore the impact of user choice strategies. The following attackers do not *deal* or wait. The strategies in Section 5.2.1 depend only on centrality as defined in Equation 5.2.1. The strategy in section 5.2.3 depends only on the attacker's knowledge of user trust states.

Table 2: Entropy of different centralities.

Flat	DC	SPBC	CC	EC
5.991	5.795	5.806	5.988	5.793

5.2.1 Network Centrality Metrics. A user's network position affects how it spreads information. Users with many connections may contribute more to spreading. The attacker would avoid cheating such users. If centralities accurately measure a user's impact on spreading, an attacker would *deal* with central nodes and *cheat* less central nodes. We explore four well-known centrality measures: degree centrality (*DC*), eigenvector centrality (*EC*), closeness centrality (*CC*) and betweenness centrality (*SPBC*). We also include $flat$, the Section 4 attacker who *cheats* users at random. The centralities are defined as follows:

(1) Degree Centrality *DC*: The degree centrality of of u_i is the the number of edges connected to u_i: $DC(u_i) = deg(u_i)$.
(2) Eigenvector Centrality *EC*: Similar to *DC* but each edge is weighted by the *EC* of the node that it is connected to. If $\mathbf{v}$ is the leading eigenvector of the network's adjacency matrix A then: $EC(u_i) = \mathbf{v}_i$.
(3) Shortest-Path Betweenness Centrality *SPBC*: This measures the proportion of shortest paths running through a user. If there are $g_{s,t}$ shortest paths between two users u_s and u_t and $g^i_{s,t}$ of those paths run through user u_i then: $SPBC(u_i) = \sum_{s,t \in \mathcal{U}} \frac{g^i_{s,t}}{g_{s,t}}$.
(4) *Closeness Centrality* This is the reciprocal of the mean distance from user u_i to every other user. If $d(u_j, u_i)$ is the distance from user u_i to user u_j then: $CC(u_i) = \frac{1}{\sum_{u_j \in (\mathcal{U} \setminus u_i)} d(u_j, u_i)}$.

The attacker wishes to prioritise less central users for *cheats*. If, for some centrality measure CM, $CM^*(u_i) = \frac{1}{CM(u_i)}$, then the probability of choosing u_i is:

$$P(victim = u_i) = \frac{CM^*(u_i)}{\sum_{u_j \in \mathcal{U}} CM^*(u_i)} \tag{7}$$

This preserves the pairwise relative probabilities of the users (e.g. a user with half the centrality of another will be twice as likely to be chosen).

In Figure 10, it can be seen that *DC* and *EC* have similar performance and slightly but consistently outperform $flat$, *SPBC* and *CC* as predicted by Borgatti [3]. In Figure 11, the previous pattern begins breaking down at 10^3. Eventually, the dominant and inferior metrics swap until the highest rate where all attackers are equal. Table 2 shows the Shannon entropy of each centrality.

The Shannon entropy is a measure of the "flatness" of a distribution with $flat$ having the highest entropy. The entropy values are similar but those of the high rate attackers are in the same order as their success. This implies that "flatness" is a good predictor of success at high rates. At rates 10, 10^2 and 10^3, predicting the spread is beneficial. As the rate increases, users cannot spread the *cheats* fast enough. Yet, attackers with skewed user weightings will blindly keep cheating some users more often, leading to unnecessary rejections.

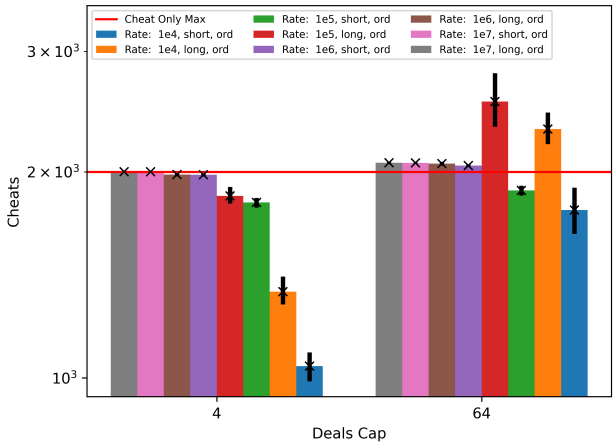

Figure 8: Success of short waiting versus long waiting ordered attackers at various rates and *deal* caps.

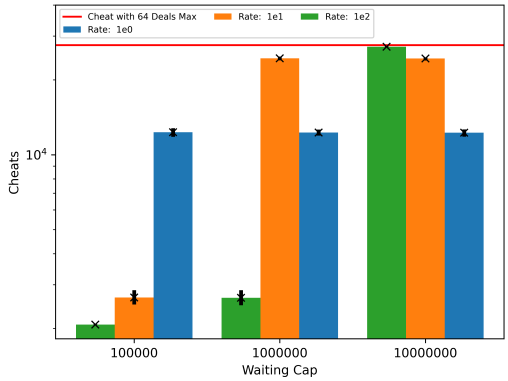

Figure 9: Success of very high rate ordered attackers with extremely long waiting times with *deal* cap of 64.

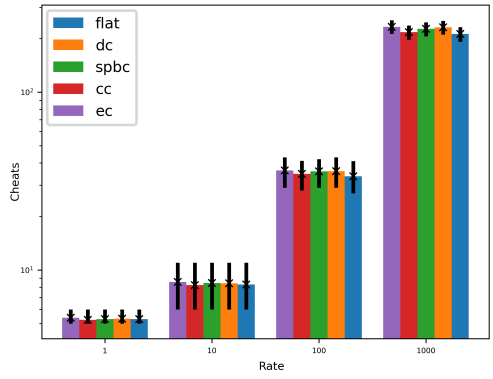

Figure 10: Success of slow attackers with different victim centralities.

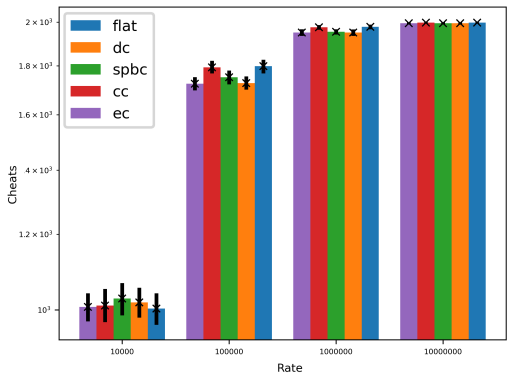

Figure 11: Success of fast attackers with different victim centralities.

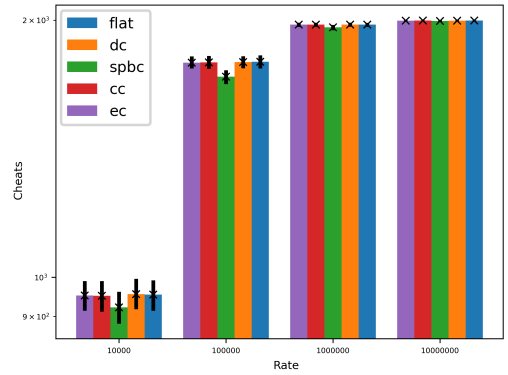

Figure 12: Success of fast attackers with different victim centralities on a CWS network with $rp = 0.01$.

5.2.2 Network Structure. A sample of the results is shown in Figure 12. Effectively no difference was seen between any of the attackers on the Connected-Watts-Strogatz (CWS) networks bar the *SPBC* attacker doing marginally worse. This demonstrates that the advantages gained by the centrality-weighted attackers is lost when attacking a CWS network.

5.2.3 Clairvoyance: Access to User Trust States. The attacker is limited by their unawareness of user trust states. The intractability of predicting the *deals* and *cheats* known by each user means that the attacker can't know which users will accept or reject an attempted transaction. Here, we study an attacker with "clairvoyance": direct access to users' trust states before they act.

For example, clairvoyance is relevant to systems where users openly post their trust state (i.e. their beliefs about the attacker); where the attacker can eavesdrop on network edges; or where they can force users to reveal their trust state before choosing a target. Various mechanisms could enable such cases. The attacker may have (some) control over the network or they may be on a platform which openly displays their current trust rating (e.g. online marketplaces). If the attacker is a news source, they may receive feedback directly from consumers regardless of how consumers

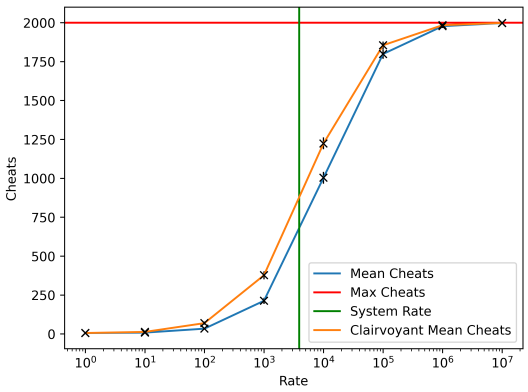

Figure 13: *Cheats* for a normal versus a clairvoyant attacker.

share knowledge amongst themselves. The clairvoyant attacker also provides a theoretical upper-bound on the attacker's success with respect to their ability to estimate or predict user knowledge.

Figure 13 demonstrates that clairvoyance primarily benefits medium-rate attackers. Fast attackers do not benefit because they already act very quickly relative to user sharing. Despite being rejected far more than they are accepted (in the order of 10^5: Figure 3), they eventually perform almost the maximal number of *cheats* as their bad reputation has no time to spread.

6 DISCUSSION

In this paper, we studied an attacker attempting to exploit the lag present in a networked reputation system. Real-world trust systems are conceptually rich and reputation lag effects can become conflated with other exploitable artefacts. The model was constructed to avoid confounding effects (e.g. value imbalance attacks due to non-unitary interactions or playbook attacks due to reputation decay). This ensured that reputation lag was the only angle for an attacker Thus, we focus on timing as the main commonality between the relevant example systems. Furthermore, centralised reputation systems (e.g. marketplaces) are well-studied whilst we aim to capture distributed systems such as P2P gossip protocols, ad-hoc networks (MANETs, VANETs), business networks, news outlets and OSNs.

First, we focused on the attacker's rate. The attacker's rate was the dominant factor in their success. A sufficiently high rate was also necessary for other strategies to succeed. Second, we studied the impact of performing *deals* (i.e. good behaviour) and of waiting. Randomly performing *deals* did not provide any unfair benefit and randomly waiting in addition to this actually worsened the attacker's success, giving users more time to spread known *cheats*. We then showed that a "*deal* then wait then *cheat*" ordering to attacker actions outperforms all previous strategies. Ordering ensures that only positive reputation spreads in the waiting period. Also, if a sufficiently fast attacker waits for the mean duration of just a single *deal* to spread through the network, they approach the maximal number of *cheats*.

Third, we investigated the efficacy of using network centrality measures to rank users' impact on information spreading. This attacker would focus their *cheats* on low ranked users. We found that, while the ranking does improve the attackers success, it is a somewhat marginal improvement, likely due to the lack of centrality measures that correctly capture this model's infection-style spreading. This strategy also relies on the network structure being suitably non-homogeneous. Otherwise, the centrality measures fail to give distinct values for the users.

Finally, we tested an attacker with *clairvoyance*: the ability to access users' trust states before acting. This was of a modest but clear benefit to medium rate attackers as they are the most likely to suffer due to repeated rejections.

Generally, this model shows the attack to be feasible. The model is abstract so we must interpret the findings realistically. In a system with high-speed communications, it may be unrealistic for the attacker to act at 10^5 or 10^7 times the speed of other parties. However, in systems where communications typically occur in the order of hours, days or weeks, it is certainly possible. (Indeed, monitoring for such suspicious activity levels offers a potential mitigation strategy.)

We consider the clairvoyant attacker for theoretical and practical reasons. Theoretically, it demonstrates the impact of such knowledge on the attacker's success and provides an upper bound on the efficacy of reputation lag attacks. Practically, some systems allow the attacker to directly access users' knowledge without the need for prediction. For example, in transparent systems or compromised systems.

We simulated distributed reputation mechanisms (e.g. some forms of social network, ad-hoc networks or MANETs). While reputation lag exists in centralised systems (e.g. eBay or Amazon), our results may not directly apply. There is potential for a comparative study via simulation. This work does not investigate explicit mitigation methods against the attack or any particular strategies. This is a vital area of further work.

Our study aims to focus on the reputation lag attack alone. Other attacks include the value imbalance attack, where attackers do cheap *deals* and expensive *cheats*. A richer system may allow us to study the interplay between different attacks and is a natural next step after we understand the reputation lag attack.

7 CONCLUSION

The reputation lag attack is viable and effective on reputation systems. Its efficacy is most impacted by attacker action rate relative to the lag period. Reducing the lag period or limiting transaction rate can mitigate the attack. The reputation lag attack is most effective as an exit strategy. Centrality-based victim choice strategies have a minor impact on more homogeneous networks. Access to current user trust states is of benefit to medium-rate attackers.

REFERENCES

[1] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics* 74, 1 (2002), 47.
[2] Phillip Bonacich. 1987. Power and centrality: A family of measures. *American journal of sociology* 92, 5 (1987), 1170–1182.
[3] Stephen P Borgatti. 2005. Centrality and network flow. *Social networks* 27, 1 (2005), 55–71.
[4] Linton C Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry* (1977), 35–41.
[5] Linton C. Freeman. 1978. Centrality in social networks conceptual clarification. *Social Networks* 1, 3 (1978), 215–239. https://doi.org/10.1016/0378-8733(78)90021-

7
[6] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11 – 15.
[7] Audun Jøsang and Jennifer Golbeck. 2009. Challenges for robust trust and reputation systems. In *Proceedings of the 5th International Workshop on Security and Trust Management (SMT 2009), Saint Malo, France*, Vol. 5. Citeseer.
[8] Reid Kerr and Robin Cohen. 2006. Modeling Trust Using Transactional, Numerical Units. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services* (Markham, Ontario, Canada) *(PST '06)*. Association for Computing Machinery, New York, NY, USA, Article 21, 11 pages. https://doi.org/10.1145/1501434.1501460
[9] Reid Kerr and Robin Cohen. 2009. Smart cheaters do prosper: defeating trust and reputation systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. 993–1000.
[10] Tim Muller, Yang Liu, Sjouke Mauw, and Jie Zhang. 2014. On robustness of trust systems. In *IFIP International Conference on Trust Management*. Springer, 44–60.
[11] Mark Newman. 2018. *Networks*. Oxford university press.
[12] Francisco Aparecido Rodrigues. 2019. Network centrality: an introduction. In *A mathematical modeling approach from nonlinear dynamics to complex systems*. Springer, 177–196.
[13] Johannes Sänger and Günther Pernul. 2016. TRIVIA: visualizing reputation profiles to detect malicious sellers in electronic marketplaces. *Journal of Trust Management* 3, 1 (2016), 1–22.
[14] Carl Shapiro. 1982. Consumer information, product quality, and seller reputation. *The Bell Journal of Economics* (1982), 20–35.
[15] Carl Shapiro. 1983. Premiums for high quality products as returns to reputations. *The quarterly journal of economics* 98, 4 (1983), 659–679.
[16] Sean Sirur and Tim Muller. 2019. The Reputation Lag Attack. In *Trust Management XIII*, Weizhi Meng, Piotr Cofta, Christian Damsgaard Jensen, and Tyrone Grandison (Eds.). Springer International Publishing, Cham, 39–56.
[17] Nadav Voloch, Ehud Gudes, and Nurit Gal-Oz. 2021. Analyzing the Robustness of a Comprehensive Trust-Based Model for Online Social Networks Against Privacy Attacks. In *Complex Networks & Their Applications IX*, Rosa M. Benito, Chantal Cherifi, Hocine Cherifi, Esteban Moro, Luis Mateus Rocha, and Marta Sales-Pardo (Eds.). Springer International Publishing, Cham, 641–650.
[18] Stanley Wasserman and Katherine Faust. 1994. *Social network analysis : methods and applications [electronic resource]*. Cambridge.
[19] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440–442.