Smart Cheaters Do Prosper: Defeating Trust and Reputation Systems

Draft: To appear, AAMAS 2009

Reid Kerr
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
rckerr@cs.uwaterloo.com

Robin Cohen
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
rcohen@ai.uwaterloo.com

ABSTRACT

Traders in electronic marketplaces may behave dishonestly, cheating other agents. A multitude of trust and reputation systems have been proposed to try to cope with the problem of cheating. These systems are often evaluated by measuring their performance against simple agents that cheat randomly. Unfortunately, these systems are not often evaluated from the perspective of security—can a motivated attacker defeat the protection? Previously, it was argued that existing systems may suffer from vulnerabilities that permit effective, profitable cheating despite the use of the system. In this work, we experimentally substantiate the presence of these vulnerabilities by successfully implementing and testing a number of such 'attacks', which consist only of sequences of sales (honest and dishonest) that can be executed in the system. This investigation also reveals two new, previously-unnoted cheating techniques. Our success in executing these attacks compellingly makes a key point: security must be a central design goal for developers of trust and reputation systems.

1. INTRODUCTION

In the field of multiagent systems, the success of an agent may depend on its ability to choose reliable partners; for this reason, trust and reputation systems have received significant attention from researchers. A particular focus has been on the electronic marketplace scenario, a well-established and important example of a multiagent system. In this setting, agents act as traders, buying and selling amongst one another. The ability to find trustworthy partners is critical to an agent's success, because an untrustworthy agent may deliver an inferior good (or fail to deliver at all), or may not pay for goods purchased. The nature of electronic marketplaces complicates the evaluation of trustworthiness: identity is difficult to establish (because new accounts can be created easily), agents might not engage in repeated transactions together (because of the size of the market and the diversity of products), and one agent might have an advantage over another during a transaction (for example, when a buyer must pay in full before a seller ships the good (or not)). A variety of approaches have been proposed to cope with these difficulties; these approaches are outlined in the next section.

The fundamental motivation for work on trust and reputation systems (TRSes)[1] is the understanding that some individuals may be dishonest. Typical proposals seek to provide some measure of protection for market participants against such dishonest traders—most frequently, the proposals attempt to predict to what degree an agent will be honest in the future. Work in this area often adopts a limited perspective, however. While it is assumed that agents may attempt to exploit *each other*, little consideration is given to the possibility that the agents may attempt to exploit the *system itself*. In fact, existing systems commonly suffer from vulnerabilities—weaknesses that may allow an unscrupulous trader to undermine or bypass the protection offered by the system.[2] Kerr and Cohen [4] present a catalogue of such vulnerabilities. Additionally, they survey a number of TRSes, claiming multiple vulnerabilities in each. Vulnerabilities in a TRS may allow an agent to cheat other users without the system preventing it, or may allow cheating without penalizing the agent after the fact. In either case, such vulnerabilities represent fundamental breaches in the protection offered by the system.

Kerr and Cohen [5] contend that security is a critical issue for designers of trust and reputation systems—if vulnerabilities exist that allow agents to achieve increased profit by cheating, we should expect profit-maximizing agents to take advantage of them. Nevertheless, this issue seems to have received little attention in the trust community. This is borne out, for example, in the simulations typically used by authors to evaluate their proposals. For example, many proposals are validated using simulations (e.g., [7, 8, 10]) populated by random selections of agents that behave consistently, or by agents whose cheating is governed by simple probability distributions, where each time step is independent of previous ones.

In this work, we examine the issue of security of TRSes, by developing agents that purposefully employ cheating tactics. We experimentally substantiate the practicality of attacks on vulnerabilities by successfully using them against a number of existing TRS proposals. This process provides insight into the nature and degree of the danger presented by each. Further, this process results in the identification of two new, closely related cheating tactics: *proliferation* and *countermeasures*, where large numbers of offers can be used to unfairly gain sales and interfere with the operation of trust

Cite as: Title, Author(s), *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

[1] For convenience, we use the abbreviation TRS, for 'Trust/Reputation System', in reference to both trust systems and reputation systems.

[2] We wish to be clear about our notion of 'attacks'. In this work, we do not refer to conventional attacks on the system implementation itself (for example, breaching the computer running a TRS, and modifying the software.) Rather, we refer only to attacks composed of actions within the system itself (for example, carefully-chosen combinations of honest and dishonest transactions.)

systems. Importantly, we demonstrate that these agents can cheat successfully without knowledge of the specific TRS in use, or even the general nature of the system. This undermines the notion of 'security by obscurity' for TRSes: ignorance of the system does not prevent an agent from cheating successfully. While central in illuminating the issue of security for TRSes, we also expect this research to be useful in the evaluation of future systems.

2. RELATED WORK

2.1 The Security of TRSes

Kerr and Cohen [4] identified the theoretical possibility of a number of vulnerabilities in TRSes.

Reputation Lag: A common policy in many electronic marketplaces is that the buyer pays before the seller ships the good. In this scenario, a seller is likely to know that he intends to cheat from the moment he receives payment. The buyer, however, will not know for some time afterward, because of processing, shipping time, etc. Under some TRSes, this presents an opportunity for a seller: he can cheat a virtually unlimited number of times before his reputation is updated to warn buyers of the new cheating activity.

Value Imbalance: In some TRSes, all reviews are weighted equally, regardless of the value of the transactions. This presents an opportunity: a seller can honestly execute small sales, then use the reputation gained to cheat on very large ones.

Re-entry: It is broadly accepted that in electronic marketplaces, we cannot assume that the identities of traders can be established. Users can create new accounts freely; in large markets, it is infeasible to verify the identity of every trader. This presents the opportunity for a dishonest trader to shed his bad reputation, starting fresh by opening a new account. This is particularly dangerous in systems that treat unknown sellers as preferable to disreputable ones.

Initial Window: In some TRSes, buyers rely only on their own experience in evaluating sellers. Once a buyer has found trustworthy sellers, this policy works well. Unfortunately, the buyer is vulnerable until he finds those trustworthy sellers—he does not have enough information to avoid cheaters.

Exit: If a seller cheats, it may damage his reputation, and hinder his ability to engage in future sales. If the seller is planning to leave the market, however, he has no further need for his good reputation. Thus, he can cheat freely, to the maximum extent possible, without consequence. This is an extremely difficult problem to combat, and affects most TRSes.

2.2 Trust and Reputation Models

We sought both to validate the practicality of our attacks against a range of systems, and to evaluate the security of noteworthy TRSes. In the interest of fairness, we selected models that self-identified as applicable to marketplaces. We outline our choices here.

Tran and Cohen

The work of Tran and Cohen [8] is representative of a *direct experience* model: agents make use only of their own experience in evaluating the trustworthiness of others. Tran and Cohen employ reinforcement learning. Each agent maintains a set of expected outcomes for each possible action (here, choice of product and partner), and chooses from among the actions in order to maximize the expected value. After an action is taken, the real outcome is used to update the expected outcome for that action, before the next such choice is made. Over time, the buyer will learn which agents can (and cannot) be trusted, and which ones give the best value for any given product. Here, we consider only the evaluation of sellers by buyers under Tran and Cohen's system.

Kerr and Cohen [4] claim that this model is vulnerable to the initial window problem; in a very large marketplace, this is likely to be especially problematic, because repeated transactions are rare between traders. It suffers from the re-entry problem (because unknown sellers are favored over disreputable ones), and the exit problem.

The Beta Reputation System

In contrast to Tran and Cohen, the Beta Reputation System (BRS) [3] represents a *witness information* model: agents employ not only their own experience in evaluating a seller, but also reports made by other agents. BRS uses the well-known beta probability distribution, which models binary events (e.g., either success or failure, or in this case, honesty or dishonesty). The beta distribution takes two parameters, a count of the number of past honest transactions, and a count of dishonest transactions. (Both counts reflect the agent's own experience, as well as reports from others.) Based on these two values, the distribution allows estimation of the probability that the seller will be honest on a future sale.

In witness information models, lying is always a potential issue: how can one know whether to trust another buyer's report or not? In the original paper [3], the authors propose a system where the reports from each buyer are discounted, based on the recipient's faith in the sender, before they are incorporated into the final estimate. Later, in [9], another system is proposed, where agent's reports are discarded if they are statistical outliers (i.e., if they are so far outside the distribution of most agent's experiences, so as to be suspect).

This model appears to be vulnerable to reputation lag (because agents rely on the recommendations of others, which do not immediately reflect cheating) and re-entry (because the beta distribution favours unknown sellers over those with more failures than successes). It is expected to be vulnerable to value imbalance (since transactions are counted equally regardless of value). It also suffers from the exit problem.

TRAVOS

TRAVOS [7] is a recent proposal that is closely related to BRS. It, too, makes use of the beta distribution to estimate the probability of honesty for a potential partner. It differs from BRS primarily in how it approaches handling the reports of others.

Under TRAVOS, when an agent wishes to evaluate a potential partner, it first makes a prediction based only on its own experience. It then estimates its confidence in that prediction: the probability that the real likelihood of cheating falls within some acceptable range from the agent's prediction. If the confidence level is high enough, the agent relies on its own prediction. If not, it solicits reports from other agents. Each report is discounted based on the accuracy of previous information provided by the reporting agent, before being combined into the final prediction.

Given the similarity to BRS, we would expect this system to have similar performance and a similar vulnerability profile. As will be shown below, however, the different handling of direct experience/witness information yields different performance. Moreover, we include the model because it reflects a common thread of recent proposals: offering new methods of coping with inaccurate reports, while using established methods to compute trustworthiness of partners. This trend has important implications, which we discuss later in the paper.

Yu and Singh

The proposal of Yu and Singh is also a predictive model. It makes use of a different probability model than other proposals, however: the Dempster-Shafer theory of evidence. Like many probability models, this model represents the strength of an agent's belief (i.e., probability) that a partner will cheat, and its belief that it will not cheat. Beyond this, however, the model also explicitly represents the agent's 'lack of belief' in those outcomes, i.e., the strength of belief that the partner might do either.

Under this proposal, like TRAVOS, an agent relies on its own experience if it believes it is sufficient. If not, it seeks the opinions of others using a 'TrustNet'. An agent has a set of neighbors; when needed, it solicits information from these neighbors. If the neighbor cannot provide information, it may refer the agent to one of its own neighbors.

Kerr and Cohen [4] claim that this model is subject to the re-entry problem (because unknown agents are intentionally treated differently from dishonest ones), reputation lag (to the degree it relies on witness information), value imbalance (because updates are not weighted to reflect transaction value), and the exit problem.

Basic Trunits

Basic Trunits [4] stands in contrast from these systems, in that it is a transactional system in which the market operator intervenes, controlling an agent's ability to engage in transactions. In this system, trust is represented using numerical units (trust units, or *trunits*), in much the same way that money represents value. In order to engage in a sale, a seller must have a sufficient number of trunits, where the required quantity depends on the value of the sale; if the seller does not have sufficient trunits, the operator prevents the sale. When a sale is made, the required trunits are held in escrow pending feedback from the seller—the same trunits cannot secure two sales simultaneously. If the seller executes the sale honestly, his trunit balance grows; if he is dishonest, he loses the trunits that secured the sale. Thus, honesty enables further sales (and profits) in the future, while dishonesty curtails future sales—an incentive for honesty. Continued dishonesty will render an agent unable to sell at all, effectively removing him from the market.

Kerr and Cohen [4] note that Basic Trunits is resistant to a number of vulnerabilities, but does suffer from the exit problem. It also faces other issues: how does one acquire an initial quantity of trunits? The obvious solution is to give a new seller an initial quantity of trunits. Unfortunately, this opens the system to re-entry. Basic Trunits is also vulnerable to another problem not noted above, called *surplus trust*. Each time a seller executes an honest sale, he gains additional trunits. If his sales are constant, however, he may not need these extra trunits to conduct this honest business. Thus, he can cheat with these extra trunits, without consequence.

3. EXPERIMENTAL METHOD

Our experiments are performed by marketplace simulation. An existing testbed, ART [2], has been developed within the trust and reputation community for both competition and experimentation. While ART has much value, the scenario used makes it unsuitable for these experiments. For example, the role of agents as both buyers and sellers makes it difficult to isolate the effects of individual buyer/seller strategies. Further, specifics of the market (e.g., a single fixed price for all transactions) makes a number of the attacks we wish to study impossible to execute. We outline our experimental scenario below.

Scenario

We model an 'advertised-price' marketplace: sellers offer goods for sale, and buyers choose whether or not to make purchases, and from whom. A fixed set of products (1000) is available for sale. Because we wish to study trust primarily, and not other price-/cost-based forms of competition, the cost to produce/acquire any given good is the same for all sellers. A typical marketplace will have more inexpensive items for sale than expensive ones. To reflect this, the cost of each good is randomly determined using the right half of a Gaussian distribution (i.e., the median occurs at $0, and probability decreases as price increases). Again, to remove focus from price-based competition, all sellers apply a fixed markup (25% of selling price)—for a given good, all vendors charge the same price.

Each seller is assigned a random number of products that she is able to produce, selected from a uniform distribution (maximum of 10). To reflect the greater availability of less expensive products, the products are again randomly assigned using the right half of a Gaussian distribution (i.e., the median occurs at the least expensive product, with declining probability as price increases).

A single TRS is in use in each simulation run. There are four sets of agents in the market during each run: buyers(100), honest sellers (250), randomly cheating sellers (250, cheating with probability 0.5), and agents implementing the cheating strategy we wish to evaluate (250). This mixture ensures a variety of agents for buyers to encounter, provides sellers with competition from sellers using different approaches, and provides comparison groups to evaluate performance.[3]

Events in a round

Each round consists of one day. After entering into a sale, a buyer will not know whether or not he has been cheated until after some number of days (14) has passed, reflecting processing, shipping, etc; we refer to the rendering of feedback after this *lag* (14 days) as the *completion* of the sale. At the beginning of each day, buyers discover whether each completing sale was executed honestly or not. Because we seek to validate attacks on TRSes (and we concern ourselves here only with attacks mounted by sellers), we wish to evaluate their effectiveness in the worst-case scenario (i.e, the best case for the TRS). Thus, in those systems in which buyers report their experiences with sellers, they always do so honestly.

Sellers decide what products to offer, and publicly post those offers. No limits are placed on sellers' capacity or inventory. Each buyer is randomly assigned a set of products (up to 5) that it needs to purchase that day; again, these are selected using the right half of a Gaussian distribution. For each product that it needs to buy, the buyer can evaluate each of the offers (i.e., evaluate the trustworthiness of each seller, using the system in place), before making a selection.

Sellers are informed of accepted offers, and paid. Each seller at this point decides whether or not to be honest. If she is honest, then she incurs the cost of furnishing the product (i.e., it is 'shipped out' that day). If she is dishonest, we assume maximal cheating: no good is shipped, and no cost is incurred. The buyer will learn the results after the lag has lapsed.

[3] It might be argued that cheaters comprise too great a fraction of our marketplace. We note, however, that: a) if cheating is successful, then cheating will be encouraged, resulting in high rates of dishonesty; b) the protection offered by existing proposals should not be fragile in the face of the very cheating against which they are meant to defend. Experimentation with different market compositions is planned future work.

Agent turnover

Marketplaces are usually dynamic—traders join and leave regularly. This is important for TRSes, because new sellers are unknown, and departing sellers result in obsolete knowledge. For efficiency of simulation, agents join/exit the market at specific intervals (100 days). On each day, each agent departs the marketplace with a fixed probability (0.05). That said, we do not want the effectiveness of our systems to be clouded by changes in market size (e.g., profits increasing because the number of buyers increases.) Thus, for every departing agent, one agent joins, keeping the participant count constant. Note that only buyers and honest sellers join and depart by this mechanism; dishonest agents stay to try to continue cheating, opening and closing accounts as their strategies dictate.

Attacker model

Here, we specify additional key points regarding the capabilities of sellers. At the time of making an offer, sellers do not know or control whether an offer will be accepted, or by whom. A seller can only provide products that she is able to produce. She is able to *advertise* (dishonestly) any product, however. While we do not consider collusive attacks, as noted above sellers can freely create new accounts at will. An unavoidable consequence is that the same seller can control and operate multiple accounts at the same time. (Note that even if a seller opens multiple accounts, the set of products she can produce remains unchanged.)

TRS Implementation

All of the systems were implemented essentially as described by the authors, except as noted here. First, while the authors of these systems describe the calculation of reputation scores (essentially the expected probability of honesty), some do not describe the actual usage of this value in selecting a seller. In these cases, we have made the reasonable assumption that buyers choose the sellers with the highest scores (subject to any adjustment for confidence made by the system.) Second, while these proposals specify how to combine ratings from multiple reviewers, which reviewers to solicit (e.g., the neighbors of an agent) may not be specified. To ensure the toughest tests for our attacks, and to investigate the soundness of each system's underlying evaluation of potential trustees, we assume the ideal case of complete connectivity: every agent receives reviews from every other agent (and can discount/disregard them as desired).

Reviewers may lie. Some of these systems make more of an effort to deal with the reliability of ratings than others—indeed, this has been a recent focus of much research effort. As above, we assume the ideal case: buyers are perfectly honest in their reports to one another. If a system is to be resistant to manipulation, it should certainly be so when noise/deception is eliminated from reviews.

Beyond what is specified here, where models require parameters we have used numbers provided by the authors in their own works wherever possible. Where no such numbers are provided, we have used reasonable values. Our TRAVOS implementation makes use of its integrated system for evaluating reviews. By comparison, in our BRS implementation all reviewers are considered to be reputable (i.e., reviews are not discounted), for several reasons: a) buyers and sellers are separate, and it was not clear how reputation scores for buyers might be established; b) all reviewers are honest, in our tests; c) this provides contrast with the TRAVOS system, allowing us to investigate the impact of attempting to cope with dishonest reviews.

3.1 TRS performance in the 'normal' case

Figure 1 depicts the operation of TRAVOS in the situation typically used for evaluation: where simple sellers cheat randomly (here, with probability 0.5). (Lines represent the total sales (in dollars) and profits for each group of agents for each day, smoothed for presentation; profits = sales − costs incurred by the seller. In this figure (and those that follow), 'random' refers to the randomly-cheating agents.) In this situation, the model operates 'as it should'—cheating quickly drops to very low levels, relative to honest sales. It is important to note the lines for profit. Recall that there are equal numbers of agents in each group of sellers. If the total profit for cheaters were higher than that for honest sellers, on average an agent would make more money by cheating than by being honest—dishonesty would be encouraged. Here, however, honesty is more profitable than cheating.

Space prevents the inclusion of such figures for every system considered; throughout this paper, we provide key data in numerical form, using charts where illustration is informative. Tables report key results of such tests for all systems considered. All tables report results over the second half (days 501 – 1000) of each simulation—after convergence to reasonably stable levels of sales for honest/dishonest sellers, reflecting long-term behavior. The first column in each table represents the average sales (in dollars) per cheating agent, relative to those of an honest seller.[4] Dishonest sales (where no good is provided) result in higher margins than honest ones, and thus potentially higher profits. The second column reflects the profit realized by a cheating agent, relative to an honest one. Results greater than 100% would mean that the average cheating agent makes more money than an honest agent. For example, a value of 124% would mean that cheating agents earned 24% more than honest agents per capita. Such a situation would be troubling, meaning that a profit-maximizing agent should choose to cheat rather than be honest.

Table 1 reflects the operation of all models when faced with simple randomly-cheating sellers. Again, this table shows the systems largely working as intended: honesty is more profitable than (random) cheating. Several points should be noted, however. First, most of the models still suffer some degree of cheating. The efforts of the systems to learn who (not) to trust are hindered by the departure of known agents, the entrance of new agents, the inability to find a known trustworthy agent offering the desired product, etc. Basic Trunits is an exception here, virtually eliminating cheating. This is because cheating agents quickly remove themselves from the market by losing their trunits. Second, the Tran and Cohen model fares badly here. This is a particularly difficult test for this model, because relying only on direct experience, the model is not designed for scenarios with turnover of selling agents. (It also does not cope well with a large variety of products.)

Table 1: Sales/profit (per capita) for randomly cheating sellers, compared to honest sellers.

TRS	Cheater sales (% of honest)	Cheater profit (% of honest)
Tran & Cohen	48.8%	121.9%
Beta	25.1%	62.9%
TRAVOS	22.9%	57.6%
Yu & Singh	33.0%	82.4%
Basic Trunits	0.0%	0.0%

These results validate both the operation of the models as expected, and the simulation scenario.

[4] Note that we cannot infer from this data the likelihood that a buyer will be cheated in general, because cheating levels will depend on the composition of the market population.

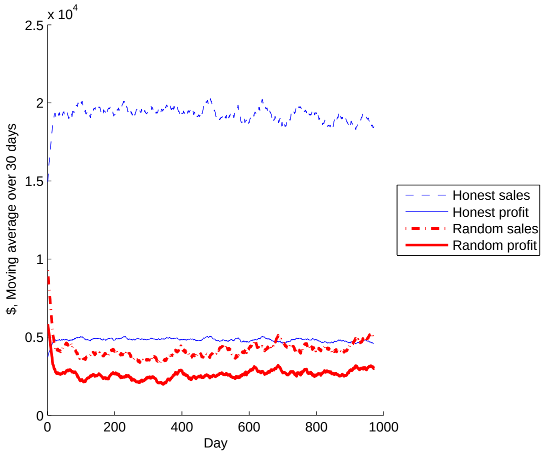

Figure 1: TRAVOS against randomly cheating sellers

4. ATTACKS

While a number of vulnerabilities have been theoretically identified, they do not constitute attacks in themselves. Rather, an attack is an actual method which exploits these vulnerabilities. As noted below, we may think of attacks as plays: sequences of events, with a desired outcome. An attack may take advantage of multiple vulnerabilities. In this section, we outline attacks that we constructed, evaluate their effectiveness, and discuss several issues raised. In each case, we present the results of one example simulation run for each system. We note that, although events, product allocations, etc. are random, the results are very consistent from one run to the next.

It must be noted that our system can provide existential evidence only. Success in employing a tactic implies that a vulnerability exists in the system. Failure, in contrast, does not mean that no vulnerability exists, only that our agents as implemented did not successfully exploit one.

4.1 Playbooks

Our work employs a technique presented previously (e.g., [1, 6]), that of a *playbook*. Our agents seek to employ profitable strategies (cheating or otherwise), strategies consisting of sequences of actions. Unfortunately, there is an enormous number of possible sequences of actions that an agent might execute; it is difficult to learn strategies from amongst the set of all possible arbitrary sequences. is As a solution, two proposals suggest the use of *plays*: pre-defined sequences of actions. Agents would have a 'book' full of known plays; selecting which play to employ at any given moment becomes the problem. This, too, presents an issue: the difficulty in directly specifying policies for which play to employ at which time, due to the enormous state space of the scenario. Each proposal takes a different approach to this issue.

The work of Ros et al. [6] employs Case-Based Reasoning. For the given scenario, a number of important features are defined that express aspects of any given state. Then, a number of example situations are created, consisting of a set of feature values and the correct choice of play for that situation. To select the appropriate play for a real situation, the agent chooses the play whose situation has the highest similarity score to the current state.

In contrast, Bowling et al. [1] suggest a technique which attempts to choose good plays for achieving a desired outcome. For each play, we track the number of times it has been executed, and the reward that has been earned each time it has been used. To select a play, a probability distribution is calculated over each applicable play: the probability of choosing a play is proportional to the reward earned using it in the past.

While our work makes use of plays, we were able to cheat very effectively without needing to use techniques such as these to choose between plays; our approach is detailed in Section 5. Investigating such techniques is planned future work.

Many of the attacks described below take input parameters. For example, when executing the reputation lag attack, for how long should one be honest, and then for how long should one cheat? One might envision using a learning algorithm to optimize these parameter values during execution. This is a tricky optimization problem, however, for a number of reasons. First, we have sparse data. We can gain very few samples of profitability at various parameter settings while the market is running. Second, the data is noisy. As shown in the charts above, sales and profits move in apparently random manner. Third, the function for which we are trying to optimize is constantly changing, as buyers are constantly updating the reputation values. For these reasons, particularly in this first attempt, we instead arbitrarily chose seemingly-reasonable parameter values. It is, perhaps, telling that our attacks were successful with simple behaviors, arbitrary parameters set, and no attempt (even by hand) to optimize parameter values.

4.2 The Proliferation Attack

During experimentation, we identified attacks that (to the best of our knowledge) have not yet been noted in the literature. These attacks are based on two properties of markets and TRSes. First, when faced with multiple sellers, each with the same rating (specifically, the maximum across sellers), TRSes will typically choose randomly/arbitrarily between the sellers. Second, as noted above, agents can freely open new accounts. In these attacks, the seller simply opens a multitude of accounts, and attempts to sell the same products through each of them. Consider the case where a product has only two sellers, both unknown to the buyer. If each makes one offer, then each has a 0.5 probability of winning the sale. However, if a dishonest seller offers the product through nine separate accounts, his probability of winning the sale increases to $9/10 = 0.9$. In a strict sense, this might not be considered a cheating attack: the attacker need never cheat a buyer. Most would consider this tactic to be problematic, however. We call this *proliferation*, after the marketing notion of 'product proliferation': having more products on the shelves results in more sales.

This attack is extremely simple to launch, and extremely effective. Figure 2 depicts the results of this attack against BRS. (In this and following charts, 'smart' refers to agents using the attack in question. Also, the line for random-cheaters' profit has been removed to avoid clutter.) The attack is devastating, with attackers dominating the marketplace.[5] (Remember that the number of honest and attacking agents is the same, with the same product distribution.) As shown in Table 2, this attack is extremely successful against all systems tested.

Table 2: Sales/profit for sellers using proliferation.

[5] One might hypothesize that such an attack might be detected. We know of no system which does so. Further, in attempting to counter this attack, it is unclear how one might differentiate the honest offer from the dishonest ones.

TRS	Cheater sales (% of honest)	Cheater profit (% of honest)
Tran & Cohen	332.2%	332.2%
Beta	307.3%	307.3%
TRAVOS	318.0%	318.0%
Yu & Singh	491.8%	491.8%
Basic Trunits	520.6%	520.6%

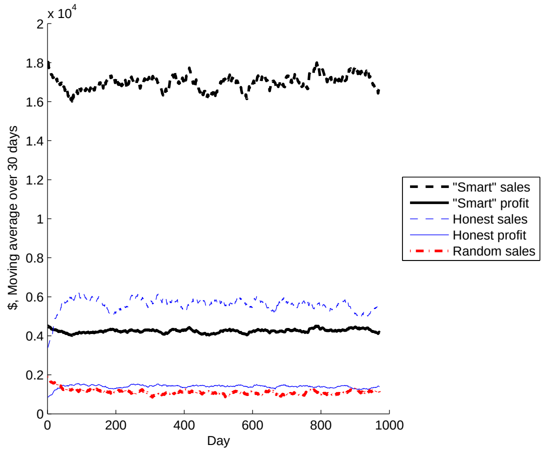

Figure 2: BRS, against proliferation

This attack sheds light on another previously-unnoted technique, one we call *countermeasures*. Many TRSes rely on gaining experience with honest sellers, to identify safe buyers. Even if an agent cheats 'normally', then this process is difficult. However, an agent can employ this same idea to further complicate the process. Consider again the situation where there is one honest seller of a product, and one dishonest one, but this time, the cheating seller plans not to deliver the product. If each offers the product once, then the agent has a 0.5 chance of picking the honest seller. If this happens, not only does he benefit from the honest sale, but he gains information—he now has made progress in identifying a trustworthy seller. This will undermine the cheater's efforts in the future. But now, consider the case where the seller offers the product through nine separate accounts. Now, not only does he have a 0.9 chance of being cheated, but he also has only a 0.1 chance of identifying the honest seller! This makes it much more likely that the seller will be able to continue cheating in the future. Worse still, the seller doesn't even gain any useful information about which sellers to avoid, because the seller will simply open another account, and abandon the old one. The effect occurs in some of our other attacks, noted below.

4.3 The Reputation Lag attack

In this attack, the seller behaves honestly for a period (45 days), and then cheats for a period (15 days—the 'lag' before an act of cheating impacts reputation). After the cheating period, the seller abandons the accounts, and opens new ones. This attack takes advantage of reputation lag and re-entry, in particular.

Figure 3 depicts the use of this attack against Tran and Cohen. The oscillations reflect the periodic honesty, then cheating, of the sellers. While it is difficult to see the total returns in this chart, Table 3 shows that cheating is significantly more profitable.

BRS and TRAVOS fared better than expected against this attack. On the first iteration, cheaters were successful. It appears that on subsequent iterations, both systems had already identified trustworthy sellers, which should occur quickly in a (perfect) witness information model. These known good sellers were preferred over unknown (re-entering) ones. TRAVOS does fare worse than BRS, however, a pattern we will notice throughout our results, despite their similarities. TRAVOS agents only trust each individual reviewer's reports to the degree that the reviewer has shown itself to be reliable. Where reviewers may lie, this would likely prove beneficial. Here, where all agents report honestly, it slows the acceptance of useful information.

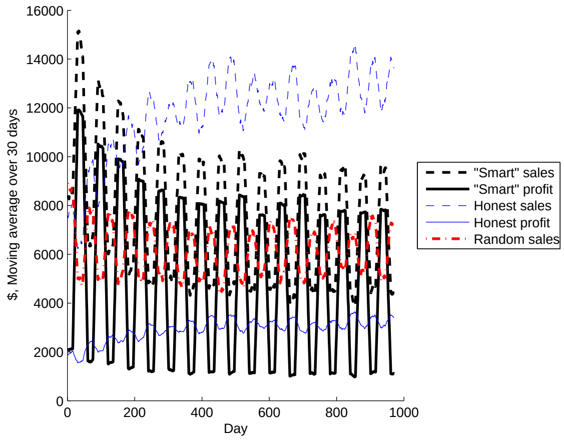

Figure 3: Tran and Cohen, against reputation lag

Basic Trunits performed poorly against this attack. This might be puzzling, as normally Trunits is resistant to reputation lag—trunits for each transaction are placed in escrow pending completion of the sale. Re-entry is a big part of the attack, however, and this implementation of Basic Trunits is quite vulnerable due to the initial sum of trunits provided to new sellers.

Table 3: Sales/profit for sellers using reputation lag.

TRS	Cheater sales (% of honest)	Cheater profit (% of honest)
Tran & Cohen	54.7%	138.7%
Beta	26.2%	59.2%
TRAVOS	28.6%	73.8%
Yu & Singh	100.7%	269.7%
Basic Trunits	49.4%	101.2%

4.4 The Re-entry attack

This attack is similar to the one above, except the agent never attempts to be honest. He simply opens an account, uses it to cheat for a period, then abandons it to open another. This attack is intended to exploit those systems that allow unknown sellers to trade effectively.

The execution of this attack, against Yu and Singh, is depicted in Figure 4. (Note that the profit and revenue lines for cheaters are superimposed in this chart—the cheaters never execute an honest transaction, so they incur no cost.) The results against all systems are shown in Table 4. This attack is very successful against every system, even those that defended against reputation lag. A key reason for this is the countermeasures phenomenon cited above. Since each cheating seller has no intention of delivering the product, he offers every product for sale (even those he cannot produce). This prevents the buyers from identifying honest sellers that they can

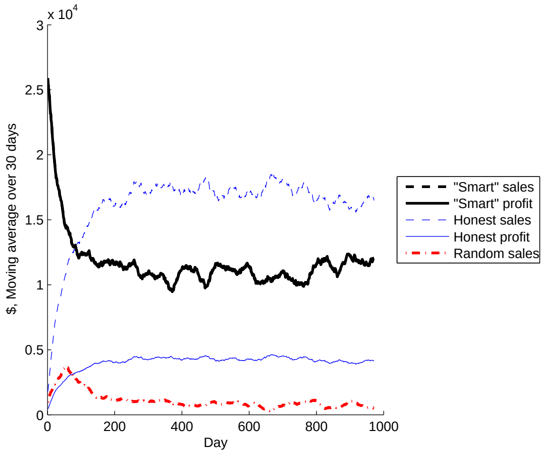

Figure 4: Yu and Singh, against re-entry

rely on in the future.

Table 4: Sales/profit for sellers using re-entry.

TRS	**Cheater sales (% of honest)**	**Cheater profit (% of honest)**
Tran & Cohen	1393.3%	5573.4%
Beta	57.2%	229.0%
TRAVOS	77.5%	310.0%
Yu & Singh	65.9%	263.4%
Basic Trunits	72.9%	291.7%

4.5 The Value Imbalance attack

In this attack, the seller attempts to be honest on small transactions to gain reputation, then cheat on large ones to gain extra profit. Unlike the previous attack, this is not periodic. Instead, the seller attempts to maintain a minimum threshold ratio of honest sales to dishonest sales, with the idea of maintaining a reasonably high level of reputability throughout.

This attack is successful against BRS (Figure 5): cheating is somewhat more profitable than honesty (and with much lower investment/sales volume). Further, as noted above, this attack was based on arbitrary parameter settings. It may fare even better with tuning of the parameters. Table 5 shows this attack to be effective against most systems. Basic Trunits fares well because each update is proportional to the value of the sale. (This technique has broad potential applicability, and has also been employed in other systems.)

Here, the difference between BRS and TRAVOS is even more pronounced. Beyond the effect noted above, it appears that another factor is at play here. Once a TRAVOS agent has enough experience with that seller, it relies only on that direct experience. This appears to slow their response when agents' behavior changes—an agent does not learn from others' warnings when a seller has begun to cheat, and so must learn it directly.

Table 5: Sales/profit for sellers using value imbalance.

TRS	**Cheater sales (% of honest)**	**Cheater profit (% of honest)**
Tran & Cohen	94.8%	146.3%
Beta	79.8%	126.2%
TRAVOS	155.0%	225.1%
Yu & Singh	135.8%	199.9%
Basic Trunits	28.2%	35.3%

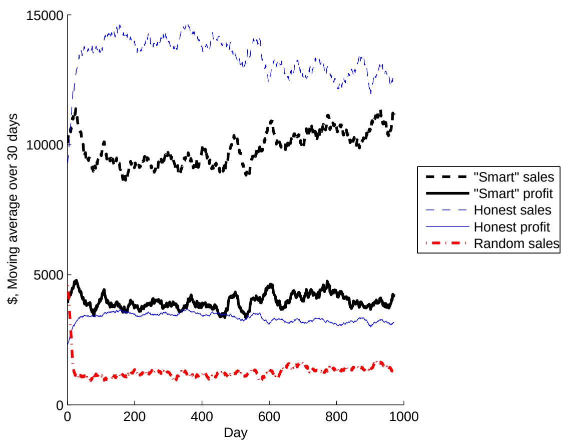

Figure 5: BRS, against value imbalance

We believe this is compelling evidence that these techniques are practical: every system tested was vulnerable to multiple attacks that made cheating more profitable than honesty.

5. SECURITY BY OBSCURITY?

While each attack described above can be launched successfully against certain TRSes, it may be less effective against others. Researchers might be tempted to suggest that, because a seller might not know what system is in use, he might not be able to launch an attack. Here, we hope to dispel this notion. The question is, can an agent successfully manage a portfolio (or playbook) full of attacks without knowledge of which TRS is in use?

We had originally intended to use learning strategies to choose between attacks (as noted in Section 4.1). We found a much simpler approach to be effective, however. We note two important points. First, as noted above, accounts cannot be tied to real identity, so an agent is free to open multiple accounts. There is no reason why an agent cannot open several accounts simultaneously to launch several attacks. Second, in the given scenario, the sellers offer goods for sale, which the buyers may or may not select. If a seller is seen as disreputable, he does not suffer any direct financial penalty—being bypassed for sales is the indirect penalty. Thus, there is no reason the seller cannot keep multiple accounts open, using each one for a different attack in parallel. The successful attacks generate profit, while the unsuccessful ones essentially result in dormant accounts. Thus, we do not need to choose between attacks—the more successful attacks will generate more activity on their own.

In implementing this method, we used all of the attacks except proliferation—it was so successful on its own, it would have rendered the results meaningless. The execution of the suite of attacks together against Basic Trunits is depicted in Figure 6; the results against all systems are shown in Table 6. In every case, the profitability from cheating is dramatically higher than honesty. This is an extremely important result: every system considered could be soundly defeated, employing simple tactics with no special optimization. This is a clear indication that security requires more attention from researchers.

If the use of multiple accounts were not possible, the agent would need to explore each attack sequentially. This is certainly feasible, and will be explored as future work.

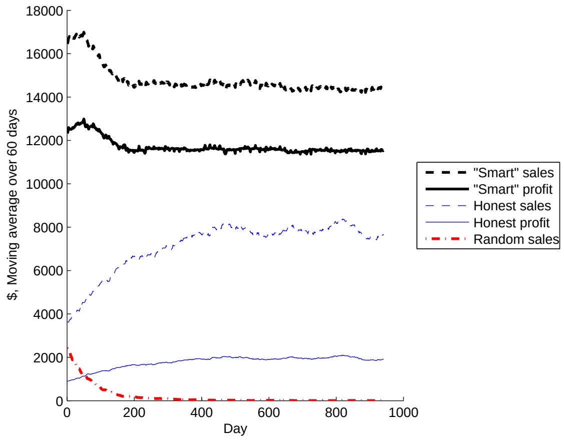

Figure 6: Basic Trunits, vs. Multi-tactic cheating sellers

Table 6: Sales/profit for sellers using multiple attacks.

TRS	**Cheater sales (% of honest)**	**Cheater profit (% of honest)**
Tran & Cohen	1775.3%	6765.1%
Beta	107.1%	288.3%
TRAVOS	274.6%	613.0%
Yu & Singh	274.9%	723.4%
Basic Trunits	181.8%	577.7%

6. CONCLUSIONS AND FUTURE WORK

We implement a number of cheating attacks that prey upon the vulnerabilities in trust and reputation systems, vulnerabilities that generally had only been identified theoretically to date. These attacks are composed only of conventional transactions that are permissible within a marketplace, and that could be executed by virtually any trader. These attacks were executed successfully against multiple TRS proposals. To the best of our knowledge, this is the first demonstration that multiple vulnerabilities do exist, and can be exploited, in practice. One might find surprising the ease with which such attacks are performed against each TRS tested, and the effectiveness of the attacks. Indeed, our results confirm the critical role of security in the design of trust and reputation systems. While we have selected a small number of TRSes for this initial study, we have no reason to believe that other systems will prove impervious to these attacks. If such attacks can be launched so easily, we must expect that in any real application of a TRS, traders can and will take advantage of them.

This process has also revealed two new cheating techniques that (to our knowledge) have not yet been identified in the literature: *proliferation* and *countermeasures*. These closely related tactics make use of an agent's ability to flood the market with product offers. While simple, they are extremely effective, and difficult to combat in marketplaces where identity cannot be established easily.

Not every attack is successful against every TRS. The question must be asked, can an agent manage a playbook of attacks in order to be successful against a system, without advance knowledge of the system in place? For the first time, we have demonstrated that the use of a simple technique allows effective cheating without any knowledge of the system in use; this technique was extremely successful in breaching the defenses of every TRS tested.

Our results make a compelling argument that security must be a central design goal for designers of trust and reputation systems, not a secondary one. Much current research attention is devoted to the problem of how to deal with inaccurate/deceptive reports from other agents. We believe this to be important work. Our investigation shows, however, that even in the absence of such unreliable reports (or presumably, perfect systems to compensate for such unreliable reports), existing TRS proposals can be defeated by simple strategies. Further, the results of the BRS and TRAVOS systems suggest that efforts to cope with dishonest reports can actually be detrimental in some cases (here, when reports are mostly honest). We believe this demonstrates that more work remains to be done in developing robust underlying models.

This paper presents early results of our investigation. While much has been learned, there is also still much to explore. Planned future work includes: a) Exploration of methods for optimizing the execution of individual cheating tactics, likely through the use of learning algorithms; b) Investigation of the ability for an agent to choose between attacks, when multiple attacks cannot be executed simultaneously; c) Testing of a larger set of TRSes, to further validate the broad applicability of these attacks; d) Exploration of collusive attacks launched by groups of agents; e) Introduction of buyer dishonesty in reporting their experiences, and an investigation of its effects on security.

It is our belief that these directions will yield further important insights and tools for the developers of trust and reputation systems.

7. REFERENCES

[1] M. Bowling, B. Browning, and M. Veloso. Plays as effective multiagent plans enabling opponent-adaptive play selection. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS'04)*, 2004.

[2] 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 *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 512–518, New York, NY, USA, 2005. ACM.

[3] A. Jøsang and R. Ismail. The beta reputation system. 15th Bled Electronic Commerce Conference e-Reality: Constructing the e-Economy, June 2002.

[4] R. Kerr and R. Cohen. Modeling trust using transactional, numerical units. In *PST '06: Proceedings of the Conference on Privacy, Security and Trust*, Markham, Canada, 2006.

[5] R. Kerr and R. Cohen. Towards provably secure trust and reputation systems in e-marketplaces. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, Honolulu, Hawaii, USA, 2007.

[6] R. Ros, M. Veloso, R. L. de Màntaras, C. Sierra, and J. L. Arcos. Retrieving and reusing game plays for robot soccer. In *Advances in Case-Based Reasoning. 8th European Conference on Case-Based Reasoning (ECCBR-06), Fethiye, Turkey, September 4-7, 2006*, volume 4106 of *Lecture Notes in Artificial Intelligence*, pages 47–61. Springer, 2006.

[7] W. T. Teacy, J. Patel, N. R. Jennings, and M. Luck. Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2):183–198, 2006.

[8] T. Tran and R. Cohen. Improving user satisfaction in agent-based electronic marketplaces by reputation modelling and adjustable product quality. In *AAMAS '04: Proceedings*

of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pages 828–835, Washington, DC, USA, 2004. IEEE Computer Society.

[9] A. Whitby, A. Josang, and J. Indulska. Filtering out unfair ratings in bayesian reputation systems. In *Proceedings of the 7th Int Workshop on Trust in Agent Societies*, 2004.

[10] B. Yu and M. P. Singh. Distributed reputation management for electronic commerce. *Computational Intelligence*, 18(4):535–549, 2002.