Thwarting market specific attacks in cloud

Tziakouris, Giannis; Bahsoon, Rami; Chothia, Tom; Buuya, Rajkumar

DOI:
10.1109/CLOUD.2016.0015

License:
None: All rights reserved

Document Version
Peer reviewed version

Citation for published version (Harvard):
Tziakouris, G, Bahsoon, R, Chothia, T & Buuya, R 2016, Thwarting market specific attacks in cloud. in *Proceedings of the 9th IEEE 2015 International Conference on Cloud Computing (CLOUD 2016).* IEEE Computer Society Press, pp. 35-42, The 9th IEEE International Conference on Cloud Computing, San Francisco, United States, 27/06/16. https://doi.org/10.1109/CLOUD.2016.0015

Thwarting Market Specific Attacks In Cloud

Giannis Tziakouris, Rami Bahsoon, Tom Chothia
School of Computer Science, University of Birmingham
Birmingham, United Kingdom
{yxt101, r.bahsoon, t.chothia}@cs.bham.ac.uk

Rajkumar Buyya
Cloud Computing and Distributed Systems (CLOUDS) Lab
Department of Computing and Information Systems
The University of Melbourne, Australia
rbuyya@unimelb.edu.au

***Abstract*—Market oriented methodologies have been extensively used for solving dynamic allocation problems in online systems including the Cloud. Despite their extensive use, very little has been known about their security against market specific security threats (e.g. monopoly, shill bidding, etc.). This work follows an experimental driven approach for: (i) promoting the development of threat-aware, market-oriented Clouds, (ii) exposing existing market specific security vulnerabilities and (iii) developing security mechanisms for online markets. We show that the designs of existing market-oriented Clouds are limited when facing market specific attacks and when thwarting malicious bidders and sellers from manipulating auction mechanisms for personal gains. Furthermore, we show that our solutions can effectively resolve market specific attacks and secure bidders, sellers and auctioning mechanisms in the context of Cloud.**

Keywords-Security; Market specific threats; Cloud.

I. INTRODUCTION

Market-oriented methodologies have been widely employed by software engineers for solving dynamic allocation problems [1-4] in online systems such as the Cloud. Despite the growing work in the area, the majority of the existing market-oriented methodologies such as SHARP [5], Tycoon [6], Bellagio [7], Shirako [8], Aneka [9] and Gridbus [10] have not provided treatment for market specific threats. To ensure the secure operation of these systems, it is not sufficient to protect them against generic attacks (e.g. denial of service, etc.), but to also consider possible market specific threats that can disturb the operation of bidders, sellers and auction mechanisms.

Even though advanced strategy-proof (e.g. Vickrey–Clarke–Groves) auction mechanisms exist for warranting that the established auctioning protocol is respected, we are not aware of any systematic or ad-hoc attempt in deploying them in the context of Cloud. This may be attributed to the following: firstly, strategy-proof solutions can be complex and/or of limited scalability when applied in dynamic and fundamentally elastic environments like the Cloud. This can render them ineffective, where the perceived benefits are likely to be overtaken by the cost and overhead of their application. Secondly, these proofs have been to a big extent theoretical in nature and concerned with the fundamentals of auctions. Consequently, their assumptions and solutions may be challenged or can break when applied in scalable and unpredictable environments. Thirdly, proofing bottleneck might impact Service Level Agreement compliance in many ways, for example, (i) the time required to take mitigation decisions will increase exponentially as the number of violation alerts increases, (ii) given the dynamism of the system, mitigation decisions are likely stale by the time they are effected, and (iii) the auction controller may become overloaded and thus fail, making the Cloud system outrightly unavailable. Henceforth, Cloud-specific proofing shall be fundamentally light-weight and scalable; they may need to operate with assumptions suited for the Cloud covering areas related to distribution and/or federation management.

This work identifies security limitations in the engineering of the commonly used market-oriented Clouds. The work attempts to overcome these challenges by proposing mechanisms that will cater for the security of bidders, sellers and auctioning mechanisms in the Cloud. More specifically, there are four main contributions in this paper:

1) To experimentally demonstrate the deficiency of existing market-oriented Clouds in facing market security threats. We motivate the need for engineering threat-aware market-oriented Clouds. We use CloudAuction [11], a market-oriented component of CloudSim simulation framework, as our test-bed to test the market to four market specific threats:
   - *Shill bidding attack:* Bidders forge bids and submit them to auctions in order to escalate the final price for a seller and defraud legitimate bidders.
   - *Reputation attack:* False bad feedback is submitted to sellers to harm their status, profit and clientele. False feedback damages the reputation of sellers and the information available to bidders, so meaning that markets can no longer function correctly.
   - *Monopoly attack*: A seller buys up resources from other sellers in the market in order to acquire the biggest percentage of resources (corner the market) and escalate their prices. Monopolies can distort investment incentives and damage market profit.
   - *Denial of payment attack:* Attackers create fake bidding accounts to bid and win resources from sellers and then intentionally deny payment to harm their profit.
2) Use the observations of the experiments to develop candidate, lightweight and scalable defensive mechanisms for securing market-oriented Clouds against the four aforementioned market-specific attacks.
3) Compare and analyze how the market is affected in the absence and presence of the selected attacks and the proposed solutions.

Through our simulation tests we were able to illustrate that market inspired methodologies such as [5-10] have not been designed with market-specific attacks in mind. We have demonstrated that market-specific attacks can significantly affect the operations, costs, profit and utility of both bidders and sellers in a market. Lastly, we have shown that by deploying our candidate solutions it was possible to efficiently counter the selected attacks and cater for a secure market.

The paper is organized as follows. Section II reviews related work. Section III describes the experimental setup and design for testing the resilience of the market in the absence and presence of the selected attacks and their candidate solutions. Section IV concludes with future work.

II. Related Work

The existing work on market-oriented Cloud security has primarily focused on improving the anonymity [12-14], confidentiality [15,16] and integrity [17] characteristics of these systems; however it has less considered market-specific threats, which can affect the operation of bidders, sellers and auction mechanisms. Scarce work exists on the analysis and counteraction of shill bidding and reputation attacks in online markets; where no work associated to monopoly and denial of payment attacks were encountered in the context of Cloud.

A. Shill Bidding

The work of Bhargava et al. [18] proposes a shill bidding counteracting algorithm for risk neutral online English auctions. The proposed algorithm uses an equilibrium strategy that maximizes the bidder's utility, holding the bidding strategies of all other bidders fixed. The authors assume that agents bid according to a symmetric strategy that maximizes their utility as it results into Bayes Nash equilibrium.

Kauffman et al. [19] analyze bidding data from Ebay coin auctions to gain insight and counter reverse price shill bidding. To identify shill bidding the authors analyze the following attributes: ratio of number of auctions with questionable bids compared to the total number of auctions held by a seller; the experience level of the seller; the seller's reputation; the starting bid in an auction; duration of an auction; and the coin value.

The work of Threvathan et al. [20] presents an algorithm that detects shill bidding in English auctions. The algorithm observes bidding patterns over a series of auctions, providing each bidder a score indicating the likelihood of their potential involvement in shill behavior. The assigned score is determined by analyzing the following bidding characteristics: seller received bids; bidding frequency; number of won auctions; time of bid submission; and bidding price.

The existing work on shill bidding restricts its application to simplistic e-commerce markets, that trade single items and utilize the English auction model. Contrary, market-oriented Clouds often necessitate the utilization of combinatorial auctions to allocate bundles of services and resources to bidders, which is not addressed by existing literature. Furthermore, the majority of the existing work assumes that the operating environment is semi-trusted with bidders that operate in a symmetric fashion, which does not hold for heterogeneous, shared and dynamic environments such as the Cloud.

B. Reputation Attack

FIRE reputation system [21] computes a trust metric for each user in a market by classifying trust information into: direct experience, witness information, role based rules and third party referrals. It then filters out and penalizes inaccurate opinions. FIRE uses an inaccuracy tolerance threshold to specify the maximal permitted differences between the actual performance and witness rating. In order to operate, FIRE requires data from multiple sources, which in Cloud can be proven complex due to absence of third party referrals and user refusal to provide data.

Sharma et al. [22] demonstrate a reputation system with a built-in attack resilience for markets. The proposed framework reduces the incentive for dishonest behavior and minimizes harm in case of attacks by dishonest bidders. This is achieved by: (i) setting reputation and dis-reputation thresholds for sellers and advisors; (ii) introducing effective reputation value increase/decrease factor; and (iii) reducing the incentive for change-in-identity. An increase in transnational experience leads to an increased weight-age of individual reputation. Fraudulent sellers are penalized with a vast drop of their reputation which results into the quick detection of fraudulent behavior.

TRAVOS [23] is a reputation system that uses Bayesian probability to compute the trust of an agent by analyzing the past experience between two agents. To remove unfair opinions TRAVOS estimates the accuracy of reputation advice based on the number of valid and invalid advice submitted by an agent in the past. TRAVOS assumes that sellers act consistently which may not be the case.

Reputation model Trunits [24] is founded on the accumulation of trust units (trunits). A seller must possess sufficient trunits before performing a transaction. To engage in a transaction, a seller must risk a particular quantity of trunits. After a transaction, if a buyer is satisfied, the seller gets more trunits, otherwise it loses the risked trunits.

Beta Reputation System (BRS) [25] estimates the reputation of sellers by using the beta probability density function. It combines the ratings of a seller being provided with multiple advisors by accumulating the number of good and bad ratings. To handle unfair opinions, BRS filters out ratings that are not in the majority. BRS can be proven effective only when the majority of the ratings are fair.

III. Experimental Setup and Design

We testify our hypothesis that market-oriented Clouds are unsecured against the four selected market specific attacks. More specifically, our experiments allow us to answer the following questions: (i) how is the utility of bidders and sellers affected by the deployed attacks? (ii) how is CloudAuction's combinatorial double auction mechanism affected by these attacks? (iii) can online markets that are vulnerable to market specific attacks pose as dependable solutions? and (iv) can our candidate solutions resolve the selected attacks and secure the market?

For the purposes of our experiments we make two assumptions concerning the market environment: we operate in a regulated market-oriented Cloud that restricts users and vendors from acting maliciously; and the market auctioneer is able to monitor the behavior of vendors and buyers within the market in order to identify and eliminate malicious behavior. To assert our hypothesis we utilize CloudAuction as our test-bed. Observations of the experiments were used to develop defensive mechanisms for securing market-oriented Clouds against the selected attacks. We report on the added value of introducing our candidate defensive mechanisms for securing the market. The source code of the developed attacks and solutions can be found in: *github.com/GiannisT/MarketAttacks*.

A. Experimental Setup

1) CloudSim Framework and CloudAuction Component: CloudSim [11] is a framework for modeling and simulating Cloud computing infrastructures and services. It enables the simulation of: (i) large-scale Cloud environments on a single physical computing node, (ii) service brokers, (iii) service provisioning, (iv) allocation policies, (v) network connections among the simulated system elements and (vi) a federated Cloud environment that inter-networks resources from private and public domains. CloudSim facilitates a visualization engine that aids in the creation and management of multiple, co-hosted virtualized services. CloudAuction component [11] is an extension of the Cloudsim framework which enables the system to handle auction-based services. CloudAuction simulates a market-oriented Cloud in which a varied number of Datacenters (sellers) and Databrokers (bidders) trade their resources (RAM, bandwidth and CPU MIPS) with the assistance of a combinatorial double auction mechanism.

Combinatorial auctions [26], refer to auctions of multiple goods, as opposed to single auctions. In combinatorial double auctions there are X items $x_1, ..., x_n$, m bidders and k sellers. Bidder i has (true) reservation value P_i per unit for a bundle of items $S_i \subseteq \{x_1, \cdots, x_n\}$, and submits a bid B_i that demands up to D$_i$ units of the bundle S_i, such as $B_i\{P_i, D_i, S_i\}$. On the other hand, each seller j forms and submits an ask Aj that has Cj as the unit cost and offers to sell up to S_j units of x_j at a unit price of F_j, such as $A_j\{S_j, x_j, F_j\}$. In each auction round, all bids and asks are simultaneously submitted to the auctioneer for auctioning. The auctioneer sorts bids in a descending price order $B_1 \geq B_2 \geq ... \geq B_n$ and asks in an ascending price order $A_1 \geq A_2 \geq ... \geq A_n$, where a subset h of them are used for auctioning. h is the largest index such that $B_m \geq A_k$. The subset of the selected bids ($q \subseteq B_n$) and asks ($z \subseteq A_n$) are in the price range of $[\max(A_h, B_h + 1), \min(B_h, A_h + 1)]$ which results in an equilibrium price as both demand and supply is h. To match the selected asks and bids the auctioning mechanism first ensures that bids are matched with asks up to their maximum demand ($D_i < S_j$) and that asks are matched with bids up to their maximum supply ($S_j < D_i$). Finally, a settlement price is calculated by deriving the average of the selected bid and ask prices and then the average of the two (Equ.1). All sellers who asked less than the settlement price sell and all bidders who bid more than that price buy resources at the settlement price.

$$AVG(\frac{\sum_{i=1}^{q} D_i \times P_i}{q} + \frac{\sum_{j=1}^{z} S_j \times F_j}{z}) \tag{1}$$

As Cloud users often require the composition and allocation of diverse resources and services, supplied by different sellers in a market, it is essential to utilize combinatorial auctions. Single item auctions can be proven insufficient, time consuming and expensive as Cloud users need to submit multiple bids to acquire different resources and services.

2) Why Use CloudSim Simulation Framework: The development of real test-beds limit the experiment to the scale of the test-bed and make the reproduction of results difficult. Furthermore, the creation of real test-beds introduce low-level tasks, such as setting up basic hardware and software, which is time and money consuming. Additionally, attributes such as allocation and provisioning algorithms are beyond the control of developers in real life markets, which restricts the experiment and its outcomes. A suitable alternative is the utilization of simulation tools, which allows the evaluation of hypothesis in an environment where one can reproduce tests. Furthermore, the development of simulation frameworks enable users to test their services/resources in a repeatable and controllable environment free of cost. At the provider side, simulation environments allow the evaluation of different resource leasing scenarios under varying loads and pricing distributions [11]. Added, simulation tools allow for the homogeneous quantification of results as they are architecture imperative, where real life market-oriented Clouds have their own composition and deployment requirements. To test our hypothesis we have selected CloudSim tool as our test-bed, due to its plethora recognition from academia and industry.

B. Experimental Design

We describe the use of CloudAuction to test our hypothesis. We report on the effects of each market specific attack and solution on the operation of sellers, bidders and the combinatorial double auction mechanism. The market parameters (e.g. number of Databroker, detection thresholds, etc.) used for our experiments are tailored according to the idiosyncrasies and characteristics of each attack/solution. The selected parameters allowed our defensive mechanisms to attain the highest detection and lowest false positive and false negative rates. We may note that these parameters are flexible thresholds, which can be set by the auctioneer according to the needs of the market. Unfortunately, it is impossible for us to establish a set of optimal thresholds for the parameters of our defensive mechanisms. This is due to the diverse composition of markets and the environments they operate in, which render these parameters case specific. One possible method to overcome this problem is to test the sensitivity of our defensive mechanisms by altering the thresholds, until the required results are witnessed.

1) Shill Bidding:

Shill Bidding Attack On CloudAuction: To test CloudAuction market to shill bidders we have simulated 3 Datacenters and a varying number of Databrokers (between 5-220) and assessed how the increase in the number of shill bidders can affect the average accumulated profit margin of sellers in the market. At the beginning of each auction round, a number of bidders (currently set to 10%) were randomly selected and "forced" to act as shill bidders by submitting forged bids. As we operate in a market that utilizes the combinatorial double auction model, a large number of bidders (i.e. 10%) are required to submit high bidding prices for the settlement price (Equ. 1) to significantly increase. In case where a low number of shill bidders are deployed in the market a shill attack can remain undiscovered or have marginal implication on the expected gain (increase in price) of malicious sellers. The price P_i submitted in the bids of the shill bidders is semi-randomly generated by adding a percentage v (10%-20%) on the highest ask price t in an auction. The selected value v represent the tendency of shill bidders that make unnecessarily large price increments to rapidly drive up selling prices. However, the submission of bids that vastly deviate from the prices submitted by legitimate bidders increases the likelihood of detection as they pose as price anomalies. Hence, our bidding prices aim to only moderately increase resource prices, for legitimate bidders to retain their interest in auctions and evade detection. To compute the bidding price for each shill bidder we use the following equation: $P_i = (t \times v) + t$.

Shill Bidding Defensive Mechanism On CloudAuction: Our shill bidding defensive mechanism is founded on the real time analysis of bidding records. Each bidding record archives the following attributes for each bidder: (i) price difference from the second highest bid, (ii) bidder feedback (e.g. positive, negative), (iii) number of lost auctions, (iv) number of bids submitted per auction, (v) total number of auctions participated, (vi) number of times that a bidder overbid himself/herself while winning an auction, (vii) number of bids submitted before and after the half time of an auction, (viii) the number of bids submitted to each seller and (ix) the overall number of bids submitted to the market. At the end of each auction round all bidding records are analyzed. The record analysis entails the comparison of the 9 archived attributes with given thresholds which result as a shill value for each bidder. In case that a comparison is true, 1 is returned and added to the "shill value", whereas if it is false 0 is returned instead. The shill value illustrates the number of malicious conditions met by a bidder in a single iteration of our algorithm. The higher the score of a shill value, the higher the likelihood of a bidder being malicious.

Our algorithm performs the following assertions to determine if a bidder is malicious (Alg.1): We first examine the feedback of a bidder. If no or negative feedback is found we increase the likelihood of a bidder being malicious (*feedback is bad || neutral*), as shill bidders usually do not receive feedback. Following, we examine the ratio between the lost auctions of a bidder and the total number of auctions that he/she participated in. As the goal of shill bidders is to drive up prices and allow legitimate bidders to win the resources, shill bidders maintain an unusually high number of lost auctions. Therefore, we assume that if a bidder has lost above $a = 60\%$ of the auctions he/she participated in *(LostAuctions > NumOfAuctPartic*$\times a$*),* he/she is probably malicious. Our algorithm, then examines if the number of bids submitted by a bidder to an auction is greater than the $b = 45\%$ of the overall bids submitted in an auction. We have selected 45% as our threshold b as shill bidders tend to submit an abnormally high number of bids in auctions to drive prices up quickly (*NumOfBidsSubmittedToAuction > BidsInAuction*$\times b$*)*. Added, we analyze if a bidder has overbid himself/herself while winning an auction. Shill bidders often overbid themselves even if they are winning, as their goal is only to increase auction prices. Our algorithm tolerates bidders that overbid themselves once ($c = 2$), to avoid penalizing legitimate bidders that accidentally overbid themselves (*OverbidWinSelf* $\geq c$). Next, we speculate concerning the frequency of the bids submitted by a bidder during the first and second half of an auction. Shill bidders are more eager to bid during the first half of an auction in order to give legitimate bidders sufficient time to place their bids and win (*BidNumFirHalf > BidNumSecHalf*). Next, we examine if the bidder has submitted the majority of its bids (i.e. $d = 54\%$) to a single seller (*BidsToSeller > TotalBids*$\times d$). Shill bidders usually submit all their bids to a very small number of sellers, as their goal is to increase the revenue of specific seller(s). Lastly, we analyze the price difference between the bidding price submitted by a bidder and the second highest price in that auction. We then compare this price difference with the average price difference exhibited in similar auctions in the market (*CurBidPrice-OverbidedPrice > AverageMarketPriceDiff*). This enables us to determine if the prices submitted by a bidder are unnecessarily high, which is behavior often encountered by shill bidders.

Once the shill value of a bidder is calculated, it is compared with a threshold (i.e. $t \geq 5$), to determine if the bidder is malicious. The value selected for t is significantly high to avoid the detection of legitimate bidders that exhibit shill-like behavior (meet some of the malicious conditions in our algorithm), but still effective for detecting shill behavior in the market. If a bidder is found to be malicious, it is flagged as a possible shill bidder and is been given a warning. If the exhibited malicious behavior is repeated from the flagged bidder a number of times (current set to $q = 2$), the bidder has to pay a fine f to the market and its account is deleted. The algorithm has been configured to tolerate the first occurrences of shill bidding from each bidder in order to decrease the occurrence of false positives. The imposed fine f aims to discourage shill bidding by exceeding the expected gain of a shill bidder and the seller it represents. The imposed fine is obtained by calculating the average market price of resources in similar, shill-free auctions and the average of resource in the presence of the shill bidder in question and then subtract one from the other and increase the resulting value by $z = 10\%$ (*f = (AVG(ShillResourcePrice) - AVG(NonShillResourcePrice))*$\times z$*)*.

The proposed defensive mechanism was able to successfully detect 83% of the shill bidders (results deducted with the execution of our solution 100 times) in the market. To test how sensitive is the detection rate of our defensive mechanism to threshold t we have lower it from 5 to 3. The obtained results showed a perfect detection rate (100%), however we have caused the occurrence of false positives (9,5%). Our results were ineffective as the benefits of lowering threshold t were overtaken by the occurrence of false positives.

Algorithm 1 Shill bidding defensive mechanism pseudocode

```
for i=1 to Nth Bidder do
    shillValue:=0
    if (feedback = bad || neutral) then
        shillValue:=shillValue+1
    end if
    if (LostAuct > NumOfAuctPartic $* a$) then
        shillValue:=shillValue+1
    end if
    if (NumOfBidsSubmittedToAuction > BidsInAuction $* b$) then
        shillValue:=shillValue+1
    end if
    if (OverbidWinSelf $\geq c$) then
        shillValue:=shillValue+1
    end if
    if (BidNumFirHalf > BidNumSecHalf) then
        shillValue:=shillValue+1
    end if
    if (BidsToSeller > TotalBids $* d$) then
        shillValue:=shillValue+1
    end if
    if (CurBidPrice-OverbidedPrice > AverageMarketPriceDiff) then
        shillValue:=shillValue+1
    end if
    if (ShillValue $\geq t$) then
        FlagAsPossibleShillBidder()
        WarnBidder()
    end if
    if (TimesFlagged $\geq q$) then
        PayFine($f$)
        DeleteAccount()
    end if
end for
```

Comparative Results: Our comparative results illustrate the variations observed in the average accumulated profit of Datacenters in the presence and absence of shill bidders and our solution in CloudAuction market. Our initial tests were performed in the absence of shill bidders, witnessing a profit margin that proportionally increased to the numbers of bidders in the market (Fig. 1, round dotted blue line).

Following, we have tested CloudAuction to shill bidders, where a higher average market price with a disproportional, continuous growing trend exhibited. The average accumulated market price in the absence of shill attackers was $242.36, where in their presence it was increased to $313.91 (Fig. 1, long dashed red line). The upsurge in the price demonstrates that even if shill bidders submit bids with only moderate increments in their bidding prices, legitimate bidders will still not be able to acquire resources at optimal prices.

Lastly, we have tested CloudAuction to the existence of shill bidders and our defensive mechanism. We have witnessed a significant drop in the average profit margin of Datacenters and similar price fluctuations to the experiment conducted in the absence of shill bidders (Fig. 1, square dotted green line).

2) Reputation Attack:

Reputation Attack On CloudAuction: To perform reputation attacks on CloudAuction we have introduced a reputation status for each seller and allowed bidders to: (i) submit feedback to sellers according to their end service satisfaction and (ii) specify in their bids the reputation of the sellers they would like to obtain services from. Depending on the nature of the market and the services/resources that it trades, different types of feedback (comprising different information) can be used to enable users to express their opinions. For the purposes of our experiment we assume that all feedback are grounded on five causes: (i) seller failure to meet resource demands (e.g. allocated fewer resources, etc.), (ii) seller failure to allocate resources, (iii) seller failure to satisfy the QoS requirements of a bidder, (iv) payment issues and (v) hardware or software errors that restricted the bidder from acquiring the won resources. To test the CloudAuction market to reputation attacks we have simulated 3 Datacenters and 25 Databrokers. At the end of each auction round we randomly select one bidder that won, paid and received resources from a seller to submit a false bad feedback to that seller.

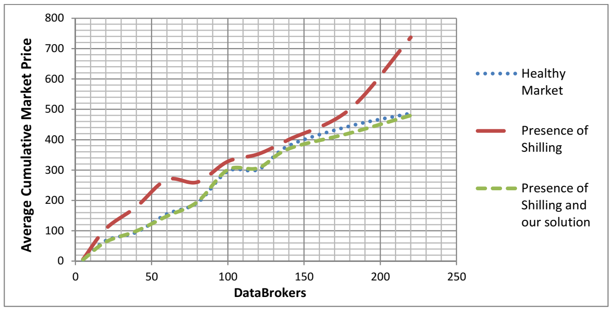


Figure 1. Illustrates the average market price in the absence and presence of shill bidding attackers and our solution

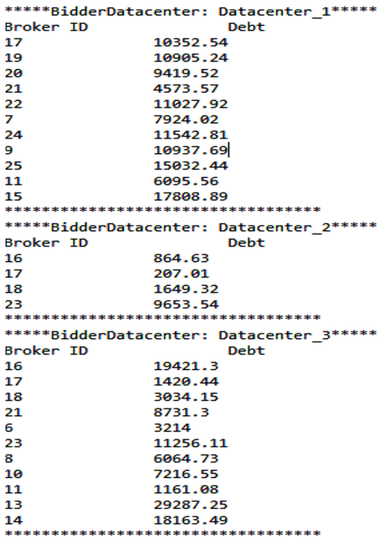
Reputation Attack Defensive Mechanism On CloudAuction: To defend against reputation attacks, we automatically intercept bad feedback for analysis prior to their publication. During analysis the following aspects (Alg. 2) are examined: (i) has the bid successfully received by the seller in question? (ii) has the bid been served? (iii) has the bidder been allocated the requested amount of bandwidth, CPUs and RAM? (iv) is the agreed price paid? and (v) have any hardware or software errors been reported during the utilization of the resources by the bidder? If the analysis illustrates that the bidder has valid reasons for submitting bad feedback, the feedback is released and submitted to the seller, whereas if the analysis shows that the bidder had invalid reasons, its feedback is discarded and a bad feedback is submitted to the bidder.

Algorithm 2 Reputation defensive mechanism pseudocode

```
if (BadFeedback isReceived) then
    Intercept(Feedback)
end if
if (Bid isReceived) & (Bid isServed) & (ReqBW $\geq$ AllocBW) &
(ReqMIPS $\geq$ AllocMIPS) & (ReqRAM $\geq$ AllocRAM) & (PaidPrice $\leq$
BidPrice) & (Payment is Performed) & (SoftErrors || HardErrors not found)
then
    Delete(Feedback)
    SubmitBadFeedback(bidder)
else
    ReleaseFeedback
    SubmitBadFeedback(seller)
end if
```

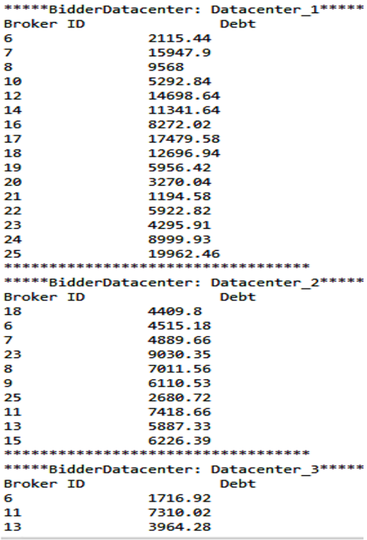
Our solution was able to detect all cases of false feedback. The defensive mechanism illustrated that in some occasions it is feasible to efficiently (low overhead) and proactively determine the validity of feedback in electronic markets, without the need for human intervention and manual revision.

Comparative Results: To determine how sellers are affected in the absence and presence of false bad feedback in the



```
*****BidderDatacenter: Datacenter_1*****
Broker ID               Debt
17              10352.54
19              10905.24
20              9419.52
21              4573.57
22              11027.92
7               7924.02
24              11542.81
9               10937.69
25              15032.44
11              6095.56
15              17808.89
***********************************
*****BidderDatacenter: Datacenter_2*****
Broker ID               Debt
16              864.63
17              207.01
18              1649.32
23              9653.54
***********************************
*****BidderDatacenter: Datacenter_3*****
Broker ID               Debt
16              19421.3
17              1420.44
18              3034.15
21              8731.3
6               3214
23              11256.11
8               6064.73
10              7216.55
11              1161.08
13              29287.25
14              18163.49
***********************************
```

Figure 2. Illustrates the profit of each seller in the absence of false bad feedback



```
*****BidderDatacenter: Datacenter_1*****
Broker ID               Debt
6               2115.44
7               15947.9
8               9568
10              5292.84
12              14698.64
14              11341.64
16              8272.02
17              17479.58
18              12696.94
19              5956.42
20              3270.04
21              1194.58
22              5922.82
23              4295.91
24              8999.93
25              19962.46
***********************************
*****BidderDatacenter: Datacenter_2*****
Broker ID               Debt
18              4409.8
6               4515.18
7               4889.66
23              9030.35
8               7011.56
9               6110.53
25              2680.72
11              7418.66
13              5887.33
15              6226.39
***********************************
*****BidderDatacenter: Datacenter_3*****
Broker ID               Debt
6               1716.92
11              7310.02
13              3964.28
```

Figure 3. Illustrates the profit of each seller in the existence of false bad feedback

market, we have initially configured CloudAuction to operate with no false bad feedback and then introduced bad feedback to seller Datacenter3. Bidders were programmed to avoid sellers with bad reputation to simulate the reluctance of real world bidders to be associated with bad reputation sellers. Our test illustrated a vast variation in the profit margin of the seller with bad reputation. In the absence of false bad feedback Datacenter3 was able to obtain 11 bids (Fig. 2), where in the presence of false bad feedback the number of bids (3 bids) and its profit dramatically decreased (Fig. 3). The debt column in Fig. 2 and Fig. 3 presents the money received by Datacenters for their resources in the absence and presence of false bad feedback respectively.

3) Monopoly Attack:

Monopoly Attack On CloudAuction: To perform our monopoly experiments we have simulated 3 Datacenters and 25 Databrokers. At the beginning of each simulation round we randomly select one seller that forwards requests for leasing resources from rival sellers until it exceeds the ownership of the 50% of the resources in the market. This aims to simulate the malicious behavior of a seller that buys up resources and corners the market.

Monopoly Defensive Mechanism On CloudAuction: According to the work of Oswald [27], if a vendor owns more than 40% of the market shares then it is likely that it is a case of monopoly. Based on [27], our monopoly solution flags sellers that own more than $m = 40\%$ of the overall market resources as suspicious. Suspicious sellers are further examined to determine if they were intentionally buying up resources to corner the market. The analysis entails the examination of a seller's: idle resources; used resources; and the number of requests made for leasing resources. If the vast majority of a sellers resources are in idle state (current threshold: $i = 70\%$), and has repeatedly submitted requests for leasing resources (current threshold: average number of leasing requests per hour) from rival sellers, it is classified as malicious. If a seller is found malicious, its leasing capability is revoked for a period of time t and its selling price is restricted to the average market selling price (Alg. 3). Thus, sellers have no incentive for cornering the market as if they are discovered they will lose their purchasing capabilities and will not be able to increase the price of their resources.

Algorithm 3 Monopoly defensive mechanism pseudocode

```
if (sellerRAM > marketRAM $* m$) || (sellerStorage > marketStorage $* m$) || (sellerBW > marketBW $* m$) || (SellerMIPS > marketMIPS $* m$) then
    flag(suspicious Seller)
end if
if (seller is suspicious) & ((idleSellerRAM > totalSellerRAM $* i$) || (idleSellerBW > totalSellerBW $* i$) || (idleSellerStorage > totalSellersStorage $* i$) || (idleSellerMIPS > totalSellersMIPS $* i$)) & (NumOfReq > MarketReqPerHour/NumOfSellers) then
    Flag (Malicious seller)
    PausePurchaseCapabilities($t$)
    RestrictHighSellPriceForSeller (AverageMarketSellingPrice)
end if
```

The proposed defensive mechanism was able to successfully detect all cases of monopoly attack. Regardless of the successful detection and mitigation of monopoly attacks, our algorithm is not able to solve the situation where an attacker distributes the overall amount of owned resources to multiple fake seller accounts, to evade detection and still maintain the majority of resources in the market. However, even in this occasion the attacker is penalized as it is both harder and more expensive to maintain multiple accounts.

Comparative Results: Our results illustrate how the average price of each traded resource is affected in the presence and absence of a monopoly attack in CloudAuction. We have observed that monopoly can significantly increase the average price of resources. This can lead into damaging the profit of other sellers (due to lack of available resources) and harming the financial interests of bidders as they will only be able to obtain resources at higher costs. More specifically, we observed that in the absence of monopoly attacks the average price for leasing RAM (per 1 MByte), storage (per 1 GByte) and bandwidth (per 1 Mbit/s) was $4.8, $4.8 and $5.2

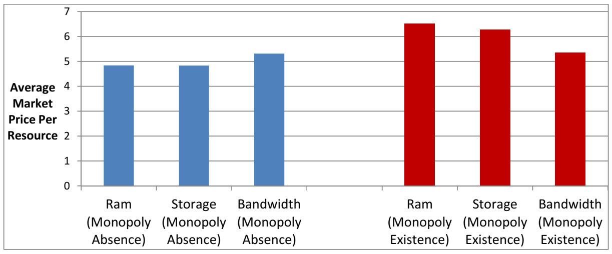


Figure 4. Illustrates the average price of resources in the existence/absence of monopoly

respectively, where when we introduced a monopoly attack the prices escalated to $6.2, $6.1 and $5.2 (Fig. 4).

4) Denial of Payment Attack:

Denial of Payment Attack On CloudAuction: To perform denial of payment attacks in CloudAuction we have simulated 3 Datacenters and 25 Databrokers. At the beginning of each simulation run we randomly select a number (currently set to the 15%) of ongoing auctions and force them to remain at matching stage and not to proceed to payment. This simulates the behavior of bidders that refuse payment to sellers. The big number of bidders selected for performing this attack aims to exemplify the effects of the certain attack on the sellers' profit and the overall revenue of online markets.

Denial of Payment Defensive Mechanism On CloudAuction: The proposed solution (Alg. 4) is founded on the real time analysis of bidding records. Each bidder in the market maintains a bidding record that archives the following attributes: (i) price difference from the second highest bid, (ii) bidder feedback, (iii) number of bids submitted to each seller, (iv) number of bids submitted to the market, (v) number of times that a user denied payment for won resources and (vi) number of won auctions. At the end of each auction round all bidding records are analyzed. The outcome of the analysis is expressed as a value (unpaid-status value). The unpaid-status value is calculated by comparing the six archived bidding attributes with given thresholds. In case that a comparison is true, 1 is returned and added to the unpaid-status value, where if it is false 0 is returned. The unpaid-status value illustrates the total number of malicious conditions met by a bidder in a single iteration of our algorithm. The higher the score of the unpaid-status, the higher the likelihood of a bidder to be malicious. Our defensive algorithm first examines the number of bids send to each seller by a bidder. Bidders that intentionally refuse payment to sellers, have specific targets that they submit all their bids to. Hence, we check if a bidder has submitted more than a number of its overall bids (i.e. $d = 50\%$) to a single seller (*BidsToSeller* > *TotalBids*$\times d$). Following, we examine the ratio between the total number of won auctions and the times denied payment. We consider bidders that have denied payment to more than $w = 50\%$ of their won auctions, as probably malicious, as legitimate bidders bid to win and purchase resources excluding few, exceptional circumstances. Lastly, we analyze the price difference between the bidding prices submitted by a bidder and the second highest price in that auction. We then compare this price difference with the average price difference exhibited in similar auctions (*CurBidPrice-OverbidedPrice* > *AverageMarketPriceDiff*). This enables us to determine if the prices submitted by a bidder are unnecessarily high, which is behavior often witnessed by malicious bidders that want to guarantee that they win auctions and deny payment to sellers.

Once the unpaid-status value is calculated, it is compared with a given threshold (currently set to $u = 3$), which determines if a bidder is malicious. If a bidder is found malicious, he/she is warned and the maximum number of bids that can submit in the market is restricted to the $s = 30\%$ of the average number of bids submitted in the market per hour (*(TotalMarketBidsPerHour/NumOfBidders)*$\times s$). If the bidder repeats the same malicious behavior a number of times (i.e. $k = 3$), then he/she has to pay a fine and its account is discarded. The imposed fine is the $f = 8\%$ of the total cost of the unpaid won auctions. Though we consider f to be reasonably low for attackers to afford and high enough to discourage them, however it can be adjusted according to how strict the penalty should be.

Algorithm 4 Denial of payment defensive mechanism pseudocode

```
for (1st to Nth Bidder) do
    UnpaidStatus:=0
    if (feedback = bad || neutral) then
        UnpaidStatus:= UnpaidStatus+1
    end if
    if (BidsToSeller > TotalBids $* d$) then
        UnpaidStatus:= UnpaidStatus+1
    end if
    if (TimesDeniedPayment > WonAuctions $* w$) then
        UnpaidStatus:= UnpaidStatus+1
    end if
    if (CurBidPrice-OverbidedPrice > AverageMarketPriceDiff) then
        UnpaidStatus:= UnpaidStatus+1
    end if
    if (UnpaidStatus $\geq u$) then
        FlagAsMalicious()
        WarnBidder()
        MaxBidsPerHour(AverageMarketBidsPerHour $* s$)
    end if
    if (TimesFlagged $\geq k$) then
        PayFine(PriceUnpaidResources $* f$)
        DeleteAccount()
    end if
end for
```

We have executed our defensive mechanism 100 times to assert its effectiveness and accuracy. Our results illustrated a high detection rate reaching the 92% of the malicious bidders in the market. We have attempted to improve the detection rate of our mechanism by lowering threshold k from 3 to 2. The results were contradictory, as the detection rate increased to 97.3%, but it also miss-classified 8% of the legitimate bidders as malicious. The increase in false positive can be attributed to the lack of sufficient bidding data for some of the bidders due to the short time they were registered in the market. Regardless of the effectiveness of our solution, attackers can evade detection by discarding their fake bidding account after each auction and then create a new one for the next auction of interest. To mitigate the certain shortcoming, we impose a registration fee (currently set to $80) on newly registered bidders to discourage the creation of multiple fake bidding accounts. The registration fee is reserved by the auctioneer and is used to pay the first won resources of the bidder.

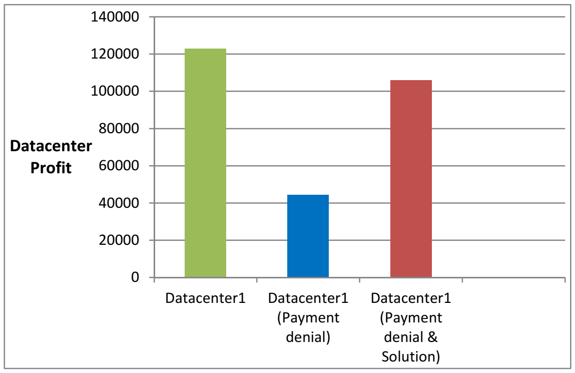


Figure 5. Shows the profit of sellers in the existence and absence of payment denial attacks

Comparative Results: Our results illustrate how the cumulative profit of sellers is affected in the presence and absence of denial of payment attacks. The obtained results (Fig. 5) show that the profit of sellers can be severely affected by bidders that intentionally refuse to pay for won resources as their resources remain bounded, where no money is received. More specifically, in the absence of intentional denial of payments the cumulative profit of sellers reached $120.100, where in their existence the market revenue dropped to $40.100. Once our defensive mechanism was deployed in the market the profit of sellers escalated to $100.150, witnessing a profit margin similar to the one exhibited in a healthy market.

IV. Conclusion and Future Work

We have hypothesized that market-oriented Cloud systems are threat-unaware and fail to deal with market-specific attacks such as shill bidding and monopoly attacks. We have used CloudAuction, a market-oriented extension of CloudSim simulation framework, as our test-bed to verify our hypothesis. Our results have confirmed that existing markets are vulnerable towards market specific attacks and ascertain on how such attacks can affect sellers, bidders and underlying auctioning mechanisms in these environments. Observations of the experiments were used to develop defensive mechanisms for securing Cloud markets against these attacks. We have experimentally illustrated and reported on the added value of introducing these mechanisms to secure the market. Future work entails the classification and analysis of a wide range of market specific threats and their solutions. We also hope to introduce intelligence to proactively detect threats and optimally configure the parameters of our solutions at runtime according to the changing market-oriented Cloud context.

References

[1] L. Jiayin, M. Qiu, J. W. Niu, Y. Chen, and Z. Ming, "Adaptive resource allocation for preemptable jobs in cloud systems." In Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on, pp. 31-36. IEEE, 2010.

[2] D. Shin and H. Akkan, "Domain- based virtualized resource management in Cloud computing", unpublished.

[3] P. Ruth, J. Rhee, D. Xu, R. Kennell and S. Goasguen, "Autonomic Adaptation of virtual computational environments in a multi-domain infrastructure", IEEE International conference on Autonomic Computing, pp.5-14, 2006.

[4] G. Tziakouris, C. J. Mera Gomez, R. Bahsoon. Securing Cloud Users at Runtime via a Market Mechanism: A Case for Federated Identity (Accepted). The 16th IEEE International Conference on High Performance Computing and Communications, Paris, France, IEEE Press.

[5] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat, "SHARP: an architecture for secure resource peering", ACM SIGOPS Operating Sys., pp.133–148, 2003.

[6] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. A. Huberman, "Tycoon: An implementation of a distributed, market-based resource allocation system", Multiagent and Grid Systems, pp.169–182, 2005.

[7] A. AuYoung, B. Chun, A. Snoeren, and A. Vahdat, "Resource allocation in federated distributed computing infrastructures", In Procedings of the 1st Workshop on Operating System and Architectural Support for the On demand IT Infrastructure ,Boston, USA, Oct. 2004.

[8] D. E. Irwin, J. S. Chase, L. E. Grit, A. R. Yumerefendi, D. Becker, and K. Yocum, "Sharing networked resources with brokered leases", In Proceedings of the 2006 USENIX Annual Technical Conference, Boston, USA, June 2006.

[9] C. Vecchiola, X. Chu, and R. Buyya, "Aneka: a software platform for .NET-based Cloud computing". High Speed and Large Scale Scientific Computing, 18, pp. 267-295, 2009.

[10] R. Buyya and S. Venugopal, "The gridbus toolkit for service oriented grid and utility computing: An overview and status report". In Grid Economics and Business Models, GECON 2004. 1st IEEE International Workshop on, pp. 19-66, Apr. 2004.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", Software: Practice and Experience, Volume 41, Number 1, pp: 23-50, Wiley Press, New York, USA, January 2011.

[12] C. Wang and H. F. Leung, "Anonymity and security in continuous double auctions for Internet retails market", In System Sciences, Proceedings of the 37th Annual Hawaii International Conference on, pp. 10-17, 2004.

[13] H. Lipmaa, N. Asokan and V. Niemi, "Secure Vickrey auctions without threshold trust", In Financial Cryptography, Springer Berlin Heidelberg, pp. 87-101, 2003.

[14] Y. Xun, and C. K. Siew, "Secure agent-mediated online auction framework", International Journal of Information Technology, pp. 1-14, 2001.

[15] F. Brandt, "Secure and private auctions without auctioneers", In Technical Report FKI-245-02. Institut fur Informatick, Technishce Universitat Munchen, 2002.

[16] D. Rolli, M. Conrad, D. Neumann and C. Sorge, "An asynchronous and secure ascending peer-to-peer auction", In Proceedings of the ACM SIGCOMM workshop on Economics of peer-to-peer systems. pp. 105-110, 2005.

[17] M. K Franklin and M. K. Reiter, "The design and implementation of a secure auction service", Software Engineering, IEEE Transactions on, 22(5), pp. 302-312, 1996.

[18] B. Bhargava, M. Jenamani and Y. Zhong, "Counteracting shill bidding in online English auction", International Journal of Cooperative Information Systems, 14(2.3), pp. 245-263, 2005.

[19] R. J. Kauffman and C. A. Wood, "Running up the bid: detecting, predicting, and preventing reserve price shilling in online auctions", In Proceedings of the 5th international conference on Electronic commerce pp. 259-265, Sept. 2003.

[20] J. Trevathan and W. Read, "Detecting shill bidding in online English auctions", Handbook of research on social and organizational liabilities in information security, pp. 446-470, 2009.

[21] T.D. Huynh, N.R. Jennings and N.R. Shabolt, "An integrated trust and reputation model for open multi-agent systems", Journal of Autonomous Agents and Multi-Agent Systems, vol. 13, no. 2, pp. 119-154, 2006.

[22] N. K. Sharma, V. Gaur and S. K. Muttoo, "A dynamic reputation system with built-in attack resilience to safeguard buyers in e-market", ACM SIGSOFT Software Engineering Notes, 37(4), pp. 1-19, 2012.

[23] J. Patel, W.T.L. Teacy, N.R. Jennings and M. Luck, "A Probabilistic Trust Model for Handling Inaccurate Reputation Sources", Springer-Verlag, LNCS 377, pp. 193-209,2005.

[24] R. Kerr and R. Cohen, "Modeling trust using transactional, numerical Units", Proceedings of the conference on Privacy, Security and Trust, Markham, Canada, 2006.

[25] A. Josang, R. Ismail and C. Boyd, "A survey of Trust and Reputation Systems for Online service provision", Decision Support Systems, vol. 43, no. 2, pp. 618-644, 2007.

[26] R. Jain and P. Varaiya, "The combinatorial seller's bid double auction: an asymptotically efficient market mechanism", J. Econom. Theory, 2006

[27] L. Oswald. "The law of marketing", Cengage Learning, 2010.