Why Are Repeated Auctions In RaaS Clouds Risky?

Danielle Movsowitz[1], Liran Funaro[2], Shunit Agmon[2], Orna Agmon Ben-Yehuda[1,2,3], and Orr Dunkelman[1]

[1] Computer Science Dept., University of Haifa
dmovsowi@campus.haifa.ac.il, orrd@cs.haifa.ac.il
[2] Computer Science Dept., Technion—Israel Institute of Technology
{funaro,shunita,ladypine}@cs.technion.ac.il
[3] Caesarea Rothschild Institute for Interdisciplinary Applications of Computer Science, University of Haifa

Abstract. The world of cloud computing is progressing from the concept of securing resources by predefined units to dynamically allocating resources using economic mechanisms. New mechanisms offer better utilization of the hardware by sharing it among multiple users. However, they allow new types of economic attacks. We introduce two new economic attacks performed by malicious users. These attacks harm the aggregate utility of Resource-as-a-Service (RaaS) clouds. Our first attack aims at raising bills in the system, and causing victims to pay more for the same amount of resources. Over time the attack may cause victims to exhaust their budget, thus lowering their demand for resource allocation, and allowing the attacker to acquire the freed resources at a negligible cost. Our second attack is designed to hinder the victim's performance at specific points in time by outbidding them for a single round. For resources of high regaining costs or that their full utilization takes time (e.g., RAM), even a single round without the resource may significantly hinder the performance. In this work we demonstrate on a simple representative example how the first attack reduces the victim's profit sevenfold and the second attack causes damage of $\$290 - \$630$ for every dollar spent on the attack.

Keywords: VCG · Resource Allocation · RaaS · Economic Attacks.

1 Introduction

The Resource-as-a-Service (RaaS) cloud [3] is an economic model of cloud computing that allows providers to sell adjustable quantities of individual resources (such as CPU, RAM, and I/O resources) for short intervals—even at a sub-second granularity. In the RaaS cloud, clients can purchase exactly the resources they need when they need them. As price wars drive cloud providers towards this model [5], they start offering plans for dealing with resource requirement bursts: CloudSigma offered time-varying burst prices in 2010 [11], Amazon EC2 offered

burstable performance instances in 2014 [14], Google Cloud offered Pay-as-you-go in 2016 [17], and Microsoft introduced the burstable Azure cloud Instance in 2017 [8]. When resources are dynamically rented, e-commerce requires calculating online economic decisions. Such decisions can only be made in real time by automated agents. E-commerce also requires efficient and computationally simple allocation mechanisms. These mechanisms may be centralized (as in an auction) or decentralized (as in a marketplace [37] or by negotiations [2]).

We see that horizontal scaling (adding more machines) has already matured to the point of incorporating advanced economic mechanisms such as auctions (e.g., AWS Spot Instances [4], Packet [31], and Alibaba Cloud Spot Instances [7]). Nevertheless, in the case of vertical scaling (increasing an existing machine's resources) we are only now seeing signs of early adoption of such mechanisms (e.g., Amazon EC2 T2 Instances, Google Cloud Platform, CloudSigma).

In the past few years, numerous studies have been published regarding different attack methods relevant to clouds, e.g., side channel [30], Resource Freeing Attack (RFA) [33], co-location attacks [34], and Economic Denial of Sustainability (EDoS) [20]. Most of the studied attacks are aimed at penetrating the security of the system and not at the economic mechanism that drives the resource allocation in the system. EDoS attacks are an exception: they cause victims to scale their resources beyond their economic means. In this work we take this line of vulnerabilities further, presenting combined economic-computer-science attacks.

Our contribution is the design of two low-cost economic attacks aimed at auction based clouds. The implementation and evaluation of the attacks were done on a simple representative example using Ginseng [6], a market driven cloud system for efficient RAM allocation. The first attack is the Price Raising attack. This attack raises prices in the system, thus reducing the victim's profit and forcing it to free resources. This enables the attacker to rent the freed resources at a negligible cost. The second attack is the Elbowing attack. This attack hinders the victim's performance by outbidding it for a single round at specific points in time. Due to the nature of RAM usage, the victim suffers from reduced performance even after the attack round ends, and it re-acquires the RAM. We demonstrate how the *Price Raising attack* reduces the victim's profit sevenfold and the *Elbowing attack* causes damage of $\$290 - \$630$ for every dollar spent on the attack.

In Section 2 we describe the auction protocol we attack. In Section 3 we discuss the vulnerabilities in repeated auctions, and the motivations behind attacking such auctions. In Section 4 we describe the experimental setup. In Section 5 we present and analyze our first attack—the Price Raising attack, and in Section 6 we present and analyze our second attack—the Elbowing attack. We review related work in Section 7, and conclude in Section 8.

2 Background: An Auction Mechanism for Vertical Scaling in the Cloud

In resource auctions, guests have private valuations for each resource (e.g., RAM or bandwidth) that reflect how much additional resources are worth to them. In a full Vickrey–Clarke–Groves (VCG) auction [12, 18, 35], guests bid with a full valuation function. The auctioneer chooses the allocation which maximizes the aggregate valuation of the allocated resources (the social welfare). The social welfare is defined as the sum of all of the guests' valuation for resources in a specific allocation. The guests pay according to the exclusion compensation principle: they pay the difference to the other guests' social welfare, incurred by their participation in the auction. We note that in the full VCG auction, guests are truthful—it is rational for them to bid with their true valuations. The Facebook's ad auction [28] and the Ginseng cache auction [16] are implementations of a full VCG resource auction.

VCG is computationally intensive, limiting its use in repeated auctions. Therefore, companies offering Spot Instances (e.g., Amazon EC2, Alibaba Cloud, and Packet), which require a repeated auction, approximate VCG using a uniform price auction. Likewise, auctioning resources in a fine granularity requires VCG approximations, which are only approximately truthful. Lazar and Semret allocate bandwidth using the Progressive-Second-Price (PSP) auction [25]. Ginseng RAM auctions RAM to guests repeatedly and frequently using the Memory-Progressive-Second-Price (MPSP) auction [6]. In this work we analyze the Simplified Memory-Progressive-Second-Price (SMPSP) auction, which resembles both these auctions. The SMPSP auction is identical to repeating the auction proposed by Maillé and Tuffin [26] when the valuation function is approximated using a single point. The SMPSP auction is also used by Agmon et al. [2]. We present the auction in detail in Section 2.1.

2.1 The Simplified Memory-Progressive-Second-Price (SMPSP) auction

In a RAM auctioning system the host auctions RAM to guests. Each guest rents a permanent amount of base RAM on a constant hourly fee. In addition, the guest may participate in the re-occurring auctions to rent additional RAM.[1] The price paid in the auction for additional RAM does not affect the cost of the base RAM. The host represents the provider, and is in charge of running the auction. In order to attract more guests, the host allocates the additional RAM between the guests, using a repeated auction mechanisms, in a manner that optimizes their social welfare. Each auction round is composed of several stages:

1. **Auction Announcement**—The host announces Q, the amount of spare RAM that is auctioned in that round.

[1] Most of the provider's revenue comes from the constant hourly fee the guests pay for their base RAM. This allows the provider to use an auction to optimize the social welfare of the guests without worrying about its own revenue.

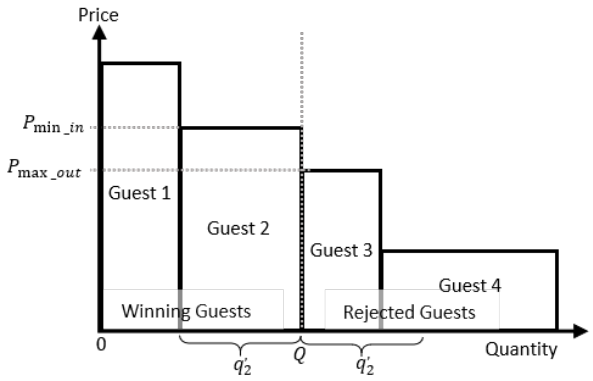


Fig. 1: Allocation Plot

2. **Bidding**—Interested guests bid for a desired amount of RAM. Guest i's bid is a tuple (p_i,q_i). p_i is the maximal unit price that guest i is willing to pay for RAM (in terms of cents per Mb per hour), and q_i is the maximal quantity it is willing to accept.
3. **Allocation and Payments**—The host computes the allocation and payment according to VCG's exclusion compensation principle.
4. **Informing Guests**—The host informs each guest i of its personal results (p_i',q_i'). The host also announces the unit price of borderline bids: the lowest accepted unit price (denoted P_{min_in}) and the highest rejected unit-price (denoted P_{max_out}). If any guest received a partial allocation, P_{min_in} is equal to P_{max_out}.

The host informs the guests about the borderline bids for three reasons. *First*, benign guests use this information to plan their next bids. They can use it to learn the minimal price they can bid with and still win some RAM. *Second*, guests can trivially learn this information over time through the rejection or acceptance of their bids, so it is futile to try and hide it. *Third*, without this information guests may bid iteratively for a few rounds to find the lowest price they can offer, thus disturbing the system's stabilization.

The results of an auction can be visually represented by a plot, as shown in Fig. 1. Each guest, in the order they were allocated RAM and sorted by the allocation algorithm, is represented by a rectangle: the width is the amount of RAM won by the guest, the height is the unit price of the guest's bid, and the rectangle's area reflects the guest's valuation for the given allocation. The guest's bill is reflected by the sum of the rectangle areas calculated from Q to q_i'. Therefore, the social welfare is given by the sum of areas that belong to guests who were allocated RAM (those in the interval $[0, Q]$).

3 Vulnerabilities in Repeated Auctions

VCG is truthful as a single round game. The best strategy for a participant is to declare its true type (e.g., its true valuation of the resources). Even a VCG-like auction (e.g., PSP or SMPSP) is usually truthful when participants consider only a single round [6]. In the case of repeated auctions, a guest can choose a more beneficial strategy on the basis of information learned from previous bid results [9].

In the SMPSP auction protocol presented in Section 2.1, if a guest wants a more accurate estimation of its next bill, it can try to create a model containing the information regarding the bids of other guests in the system. This model can be created by keeping track of the data released by the host—P_{min_in} and P_{max_out}, and fusing it with the results from previous rounds—the bill paid by the guest, and the allocation it received. Given P_{min_in}, the guest knows the minimal bid that was allocated a positive quantity of RAM. At the same time, given P_{max_out} and its bill, the guest knows the maximum rejected bid price, and the average unit price in the interval $[Q, Q + q']$. The guest does not know how many bids were rejected nor their values. By changing its bid price, and following the results of each round, the guest can learn information about the rejected bids.

In Section 5 and Section 6 we will show how a guest can use the data collected in order to attack the system and gain resources for a negligible price.

3.1 Attacking a Repeated Auction Mechanism

A system based on a repeated auction is vulnerable to attacks by both adversarial guests and selfish guests wishing to improve their resource allocation. In a repeated auction, data about the system can constantly be collected. Hence, the attacker can afford sporadic non-beneficial attack rounds, as the following rounds balance out its utility.

When attacking the system, the attacker can have different direct goals driving the attack:

1. Hindering performance—The attacker prevents the victim from utilizing physical resources.
   (a) Resource deprivation—The attacker causes the victim to rent less resources.
   (b) Inefficient resource rental—The attacker harms the victim by making it suffer the overhead of re-acquiring the resource. This attack is specific for resources that have a high acquisition overhead (e.g., RAM, disk space). Even if the victim obtains access to the resource again, it would need to recover or reproduce the data. Until then, its performance is likely to be hindered.
2. Reducing profits—The attacker raises the price of resources, thus causing the victim to spend more money than planned. The immediate result is increasing the victim's expense rate, thus lowering its profit rate.

3. Reducing resource pressure (freeing resources)—If the victim's budget is drained (e.g., due to a long term cost increase), it is forced to free resources. The freed resources can then be obtained by the attacker at a lower price because the demand for the resources is lower. Even if the victim's budget is not completely depleted, it may request less resources in an effort to avoid a complete budget depletion. This reaction immediately frees resources.

An important aspect of an attack is the monetary resources required for its deployment. A high budget attack is less likely to be deployed than an attack that can be performed using a strict budget. Our proposed attacks, the *Price Raising attack* and the *Elbowing attack*, are indeed low-cost attacks. The Price Raising attack costs nothing, and the Elbowing attack causes damage of $\$290 - \$630$ for every dollar spent by the attacker.

4 Experimental Setup

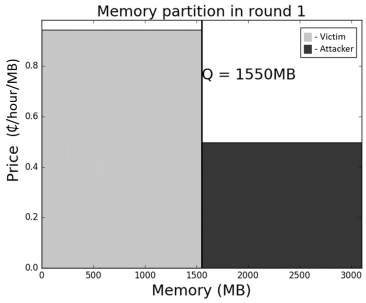
Our experiments, which verified the attacks we performed, were run on a 24-core server with 16GB of RAM. The RAM was auctioned and allocated using Ginseng [6] over Ubunto Linux 4.4.0-72-generic. Ginseng auctions RAM to guests and then changes their physical RAM allocation accordingly. It is implemented on the KVM hypervisor [23] with Litke's memory overcommit manager MOM [1]. It controls the exact amount of allocated RAM to each guest via libvirt using ballon drivers [36]. Ginseng has a host component and a guest component. The host component includes the Auctioneer that runs the MPSP protocol that is an extension of the SMPSP protocol described in Section 2.1. The guest component is suitable for virtual machines (VM's).

We note that although this Ginseng implementation is based on VMs, the principals behind the attacks are also relevant to other implementation (e.g., containers).

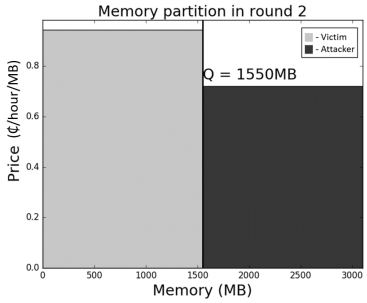
In our experiments the victim guest machine ran an elastic version of memcached [27], a key value storage application commonly used in clouds. Elastic memcached can release the least recently used RAM, thus allowing memory usage elasticity. The performance of elastic memcached is a concave monotonically rising function of its RAM [6]. In this case, MPSP guests bid using the reduced bidding language supported by the SMPSP. Under these circumstances, the protocols are equivalent. The attacker guest machine ran MemoryConsumer [6], a synthetic application which performs linearly with its RAM consumption.

5 Price Raising Attack

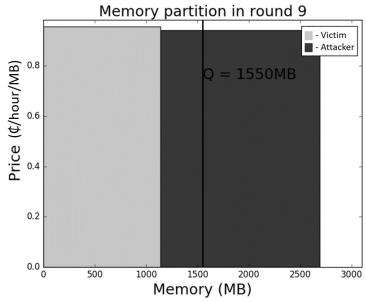
In the **Price Raising** attack, the attacker uses the first rounds of the auction to collect data about the borderline bids. In analogy to Dolgikh et al. [13], it learns information for free by bidding a price that is likely to be rejected. After analyzing the collected data, the attacker uses it to situate itself as the highest rejected bidder, thus setting the unit price P_{max_out}. In every auction round, the



(a) Initial bids. The attacker situates itself as the guest setting P_{max_out}. Bill rate of \$7.7/*hour* for $1.5GB$ that the victim rents.



(b) The attacker raises P_{max_out}, increasing the victim's bill rate to \$11.16/*hour* for the same $1.5GB$.

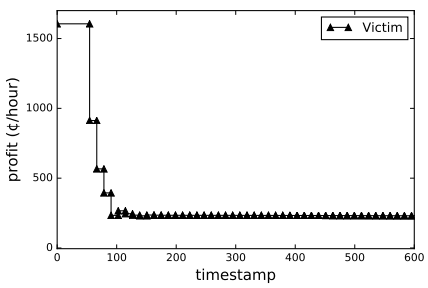


(c) The victim responds by reducing its bid quantity, and raising its bid price. The attacker gains free access to resources. The victim's bill rate is now \$10.73/*hour* for only $1.14GB$ it rents.

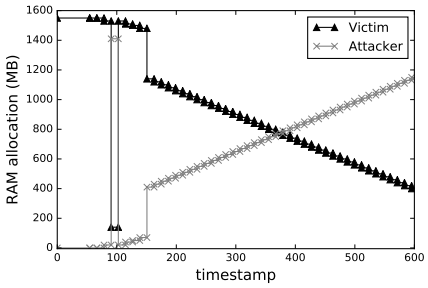
Fig. 2: Price raising attack. RAM allocation plot

attacker requests Q—the full amount of RAM. By bidding P_{max_out} for Q, the attacker directly affects the bill paid by the winning guests (see Fig. 2). Each winner i pays $P_{max\_out} \times q'_i$. In every attack round the attacker raises P_{max_out}. This causes the winning guests to pay more for the RAM resources they receive, thus reducing their profits. The profit reduction may cause the victim to reduce its requested quantity and raise its bid price, in an effort to distance its bid from the borderline unit price, or to remain within its budget limits. Either way, the victim may free resources.

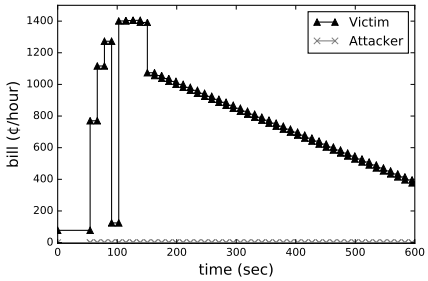
To keep the attack at a low budget, the attacker compares rejected guests' bids with its own valuation. If the attacker's true valuation for RAM is lower than that of any other rejected guest, then the attacker should decrease its bid price to avoid winning resources that will create additional costs that it cannot afford. Otherwise, the attacker can rent the freed RAM.



(a) Victim's profit



(b) Guests RAM allocation



(c) Guests bill. The attacker's bill is 0.

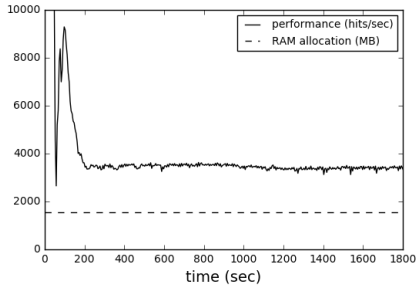
Fig. 3: Price raising attack.

This attack achieved three goals: First, by slowly raising P_{max_out} and causing the victim to pay more for the same quantity of RAM, the attacker reduced the victim's profit sevenfold (see Fig. 3a). Second, by reducing the victim's profits and draining its budget, the attacker forced the victim to free RAM (see Fig. 3b), thus enabling the attacker to rent it at a lower cost (see Fig. 3c). Finally, by forcing the victim to rent less RAM than it ideally wanted, the attack can hinder the victim's performance.

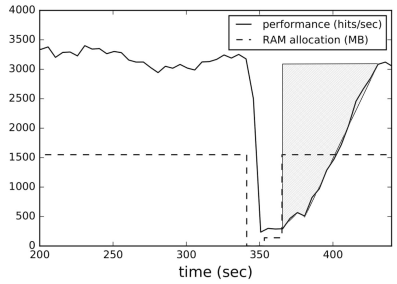
6 Elbowing Attack

The **Elbowing** attack causes the degradation of the victim's performance and profit over time. This attack is designed to harm victims that fill their RAM slowly. For such victims, losing RAM even for a very short period of time can be badly damaging: while the victim re-acquires and re-fills the RAM, it loses the benefits it gained from previously owning the RAM. The attacker benefits from this attack since outbidding the victim every now and then does not cost nearly as much as it would cost to constantly win the allocated RAM. This means that at a relevantly low cost, the attacker can inflict severe damage to the victim.

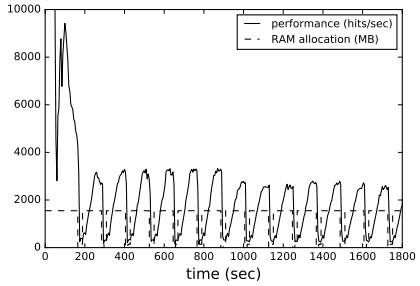
To test this attack type we performed a parametric sweep of attacks in a system that contained two guests—the attacker and the victim. In each attack,



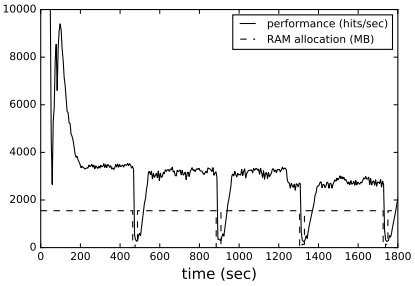
(a) No attacks on the system.



(b) The victim's performance and RAM allocation.



(c) Attack every 10 auction rounds. The victim's damage is \$48/*hour* and the attack cost is \$0.11/*hour*.



(d) Attack every 35 auction rounds. The victim's damage is \$20/*hour* and the attack cost is \$0.03/*hour*.

Fig. 4: Elbowing attack. The performance and RAM allocation of the victim. The victim's damage is computed using its valuation function, which attaches a value of 3 cents to every hit.

the following process was repeated every N rounds. First, the attacker waited for the system to stabilize. During the stabilization period the victim filled its allocated RAM. Then the attacker attacked by outbidding the victim: it bid for a single round for Q with a unit price slightly higher than P_{min_in}. This forced the elastic memcached victim to lose its data (a standard, non-elastic memcached would have swapped its data to a slower storage, suffering a higher penalty). After the attack round, the attacker went back to bidding with a negligible bid price, and the victim had to re-acquire the RAM, suffering a high overhead while doing so.

We present the results of such an attack in Fig. 4. The baseline performance of the victim (without any attack) is presented in Fig. 4a. The damage from a single attack round is shown in Fig. 4b. In this case, it would be wasteful to attack again before 440s, because although the attack only lasted between 340s and 352s, and the RAM was re-rented 12 second later, it was not fully filled for another 70 seconds after that. The shaded area in Fig. 4b represents

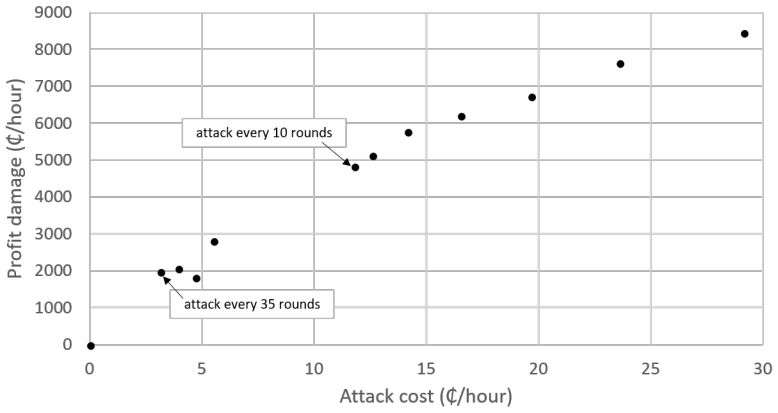


Fig. 5: Profit damage as a function of the attack cost.

the lingering damage of the Elbowing attack. The lingering damage is larger as the recuperation rate of the victim is slower. The longer it takes the victim to recuperate, the more cost-effective the Elbowing attack is, since the attacker does not pay extra for those rounds.

The results of examples of repeating attacks are presented in Fig. 4c and 4d. The attack in Fig. 4c is performed frequently (every 10 rounds), and the victim does not fully recuperate. The attack in Fig. 4d is less frequent (every 35 rounds), and thus causes more damage per single attack round. However, since the attack rate is slower, it causes smaller overall damage to the victim's profit.

The results of a parametric sweep on the attack frequency are shown in Fig. 5. Each point in the graph represents one attack, and shows the average damage caused to the victim's profit as a function of the attack average cost. The cost of the attack is determined by the frequency of the attack rounds. As shown Fig. 5 the Elbowing attack causes a damage of $\$290-\$630$ for every dollar spent on the attack. The Elbowing attacks are not necessarily low budget attacks, but they can be performed on a strict budget. To do this, the attacker needs to decide in advance how much money it is willing to spend on each attack round.

7 Related Work

Vulnerabilities and attacks in clouds have been extensively researched. Subashini et al. [32], Hashizume et al. [19], Fernandes et al. [15], and Movsowitz et al. [29] provide extensive surveys of such attacks.

Fraudulent Resource Consumption (FRC) attacks, in which budgets are exhausted, have also been researched. Idziorek et al. [21] present an FRC attack and discuss two detection methodologies for such attacks. Kumar et al. [24] suggest an in-cloud EDoS mitigation web service (called Scrubber Service) that can

be used on-demand. This service is used to generate and verify a crypto puzzle needed to prove the legitimacy for acquiring services.

Attacks on ad auctions were analyzed by Zhou et al. [38] and Cary et al. [10]. Jellinek et al. [22] study existing cloud billing systems, uncovering difficulties in predicting charges and bugs that lead to free CPU time and over-charging for storage.

8 Conclusions and Future Work

In this work, we demonstrated two low-cost economic attacks on an auction based mechanism for vertical resource allocation in the cloud. The Price Raising attack is a low cost attack that causes the victim to deplete its economic resources, thus freeing resources for the attacker to obtain at a negligible cost. We demonstrate how this attack reduces the victim's profit sevenfold. The Elbowing attack hinders the victim's performance by outbidding it every several rounds. We showed that an attacker can cause damage of $\$290 - \$630$ for every dollar it spends on the attack. This attack can be applied to various economic mechanisms. Future work will try to amplify the effects of the Elbowing attack, e.g., by coupling it with an additional attack which will inform the attacker of optimal attack times. An optimal time for an attack like this depends on the quality and quantity of the evicted data. The RAM utilization is of high quality when the victim values its RAM usage the most. This valuation might be deduced from its bid price, or from side channels such as the victim's traffic volume or destination.

9 Acknowledgments

This work was partially funded by the Amnon Pazi memorial research foundation. We thank A. Schuster and E. Tromer for fruitful discussions. We thank Y. Lev, A. Ohayon, and S. Levenzon for their contribution in creating the allocation plots presented in Section 5. We also thank the Caesarea Rothschild Institute for Interdisciplinary Applications of Computer Science in the University of Haifa for their support.

References

1. A. G. Litke. Memory overcommitment manager, `https://github.com/aglitke/mom`, Accesses on 19.07.18
2. Agmon, S., Agmon Ben-Yehuda, O., Schuster, A.: Preventing Collusion in Cloud Computing Auctions. In: Economics of Grids, Clouds, Systems, and Services - 15th International Conference, GECON 2018, Proceedings. Springer (2018)
3. Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., Tsafrir, D.: The Resource-as-a-Service (RaaS) cloud. In: USENIX Conference on Hot Topics in Cloud Computing (HotCloud) (2012)

4. Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., Tsafrir, D.: Deconstructing Amazon EC2 Spot Instance Pricing. ACM Trans. Econ. Comput. **1**(3), 16:1–16:20 (Sep 2013). https://doi.org/10.1145/2509413.2509416, http://doi.acm.org/10.1145/2509413.2509416
5. Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., Tsafrir, D.: The Rise of RaaS: The Resource-as-a-Service Cloud. Commun. ACM **57**(7), 76–84 (Jul 2014). https://doi.org/10.1145/2627422, http://doi.acm.org/10.1145/2627422
6. Agmon Ben-Yehuda, O., Posener, E., Ben-Yehuda, M., Schuster, A., Mu'alem, A.: Ginseng: Market-Driven Memory Allocation. ACM SIGPLAN Notices **49**(7), 41–52 (2014)
7. Alibaba Cloud Spot Instances, https://www.alibabacloud.com/help/doc-detail/52088.htm, Accessed on 11.03.2018
8. Azure, https://tinyurl.com/burstable-azure-cloud-instance, Accessed on 03.06.2018
9. Brandt, F., Weiß, G.: Antisocial Agents and Vickrey Auctions. In: Meyer, J.C., Tambe, M. (eds.) Intelligent Agents VIII, 8th International Workshop, ATAL 2001 Seattle, WA, USA, August 1-3, 2001, Revised Papers. Lecture Notes in Computer Science, vol. 2333, pp. 335–347. Springer (2001). https://doi.org/10.1007/3-540-45448-9_25, https://doi.org/10.1007/3-540-45448-9_25
10. Cary, M., Das, A., Edelman, B., Giotis, I., Heimerl, K., Karlin, A.R., Mathieu, C., Schwarz, M.: Greedy bidding strategies for keyword auctions. In: Proceedings of the 8th ACM conference on Electronic commerce. pp. 262–271. ACM (2007)
11. Charting CloudSigma Burst Prices, https://kkovacs.eu/cloudsigma-burst-price-chart, Accessed on 21.04.2018
12. Clarke, E.H.: Multipart pricing of public goods. Public choice **11**(1), 17–33 (1971)
13. Dolgikh, A., Birnbaum, Z., Chen, Y., Skormin, V.: Behavioral Modeling for Suspicious Process Detection in Cloud Computing Environments. In: Mobile Data Management (MDM), 2013 IEEE 14th International Conference on. vol. 2, pp. 177–181. IEEE (2013)
14. EC2 instances with burstable performance, https://aws.amazon.com/blogs/aws/low-cost-burstable-ec2-instances/, Accessed on 11.03.2018
15. Fernandes, D.A., Soares, L.F., Gomes, J.V., Freire, M.M., Inácio, P.R.: Security issues in cloud environments: a survey. International Journal of Information Security **13**(2), 113–170 (2014)
16. Funaro, L., Agmon Ben-Yehuda, O., Schuster, A.: Ginseng: Market-Driven LLC Allocation. In: 2016 USENIX Annual Technical Conference. p. 295 (2016)
17. Google Cloud Platform, https://cloud.googleblog.com/2016/09/introducing-Google-Cloud.html, Accessed on 11.03.2018
18. Groves, T.: Incentives in teams. Econometrica: Journal of the Econometric Society pp. 617–631 (1973)
19. Hashizume, K., Rosado, D.G., Fernández-Medina, E., Fernandez, E.B.: An analysis of security issues for cloud computing. Journal of internet services and applications **4**(1), 5 (2013)
20. Hoff, C.: Cloud computing security: From DDoS (distributed denial of service) to EDoS (economic denial of sustainability), https://tinyurl.com/from-ddos-to-edos, Accesses on 27.05.18
21. Idziorek, J., Tannian, M.: Exploiting cloud utility models for profit and ruin. In: Cloud Computing (CLOUD), 2011 IEEE International Conference on. pp. 33–40. IEEE (2011)
22. Jellinek, R., Zhai, Y., Ristenpart, T., Swift, M.M.: A Day Late and a Dollar Short: The Case for Research on Cloud Billing Systems. In: HOTCLOUD (2014)

23. Kivity, A., Kamay, Y., Laor, D., Lublin, U., Liguori, A.: kvm: the linux virtual machine monitor
24. Kumar, M.N., Sujatha, P., Kalva, V., Nagori, R., Katukojwala, A.K., Kumar, M.: Mitigating economic denial of sustainability (edos) in cloud computing using in-cloud scrubber service. In: Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on. pp. 535–539. IEEE (2012)
25. Lazar, A.A., Semret, N.: Design, analysis and simulation of the progressive second price auction for network bandwidth sharing. Columbia University 1998
26. Maillé, P., Tuffin, B.: Multi-bid auctions for bandwidth allocation in communication networks. In: Proceedings IEEE INFOCOM 2004, The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, March 7-11, 2004. IEEE (2004). https://doi.org/10.1109/INFCOM.2004.1354481, https://doi.org/10.1109/INFCOM.2004.1354481
27. memcached, https://github.com/ladypine/memcached, Accessed on 12.03.2018
28. Metz, C.: Facebook Doesn't Make As Much Money As It Could—On Purpose, https://tinyurl.com/facesbook-ads, Accessed on 12.03.2018
29. Movsowitz, D., Agmon Ben-Yehuda, O., Schuster, A.: Attacks in the Resource-as-a-Service (RaaS) Cloud Context. In: International Conference on Distributed Computing and Internet Technology. pp. 10–18. Springer (2016)
30. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM conference on Computer and communications security. pp. 199–212. ACM (2009)
31. Spot marketing pricing—discount packet bare metal servers., https://www.packet.net/bare-metal/deploy/spot/, Accessed on 02.06.2018
32. Subashini, S., Kavitha, V.: A survey on security issues in service delivery models of cloud computing. Journal of network and computer applications **34**(1), 1–11 (2011)
33. Varadarajan, V., Kooburat, T., Farley, B., Ristenpart, T., Swift, M.M.: Resource-freeing attacks: improve your cloud performance (at your neighbor's expense). In: Proceedings of the 2012 ACM conference on Computer and communications security. pp. 281–292. ACM (2012)
34. Varadarajan, V., Zhang, Y., Ristenpart, T., Swift, M.M.: A Placement Vulnerability Study in Multi-Tenant Public Clouds.
35. Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders. The Journal of finance **16**(1), 8–37 (1961)
36. Waldspurger, C.A.: Memory Resource Management in VMware ESX Server. In: Culler, D.E., Druschel, P. (eds.) 5th Symposium on Operating System Design and Implementation (OSDI 2002), Boston, Massachusetts, USA, December 9-11, 2002. USENIX Association (2002), http://www.usenix.org/events/osdi02/tech/waldspurger.html
37. Yu, D., Mai, L., Arianfar, S., Fonseca, R., Krieger, O., Oran, D.: Towards a Network Marketplace in a Cloud. In: HotCloud (2016)
38. Zhou, Y., Lukose, R.: Vindictive bidding in keyword auctions. In: Proceedings of the ninth international conference on Electronic commerce. pp. 141–146. ACM (2007)