Attacks in the Resource-as-a-Service (RaaS) Cloud Context

Danielle Movsowitz, Orna Agmon Ben-Yehuda, and Assaf Schuster

Technion—Haifa Institute of Technology,
Haifa, Israel
dani.movso@campus.technion.ac.il,
ladypine@cs.technion.ac.il,
assaf@cs.technion.ac.il
http://www.cs.technion.ac.il

Abstract. The Infrastructure-as-a-Service (IaaS) cloud is evolving towards the Resource-as-a-Service (RaaS) cloud: a cloud which requires economic decisions to be taken in real time by automatic agents. Does the economic angle introduce new vulnerabilities? Can old vulnerabilities be exploited on RaaS clouds from different angles? How should RaaS clouds be designed to protect them from attacks? In this survey we analyze relevant literature in view of RaaS cloud mechanisms and propose directions for the design of RaaS clouds.

Keywords: cloud computing, privacy, security, RaaS

1 Introduction

The Resource-as-a-Service (RaaS) cloud [1] is an economic model of cloud computing that allows providers to sell individual resources (such as CPU, memory, and I/O resources) for a few seconds at a time. In the RaaS cloud, clients are able to purchase exactly the resources they need when they need them. In light of global trends and economic incentives driving the providers to a price war [3], we anticipate that the RaaS cloud will gradually replace the IaaS cloud. In the RaaS cloud, e-commerce is quick and frequent. It is impossible for a human to make the economic decisions required to optimize the resource purchases. Hence, in the RaaS cloud, clients will deploy automatic agents to conduct the e-commerce for them. This e-commerce may be centralized (an auction, for example) or decentralized (as in a marketplace or negotiations).

Commercial cloud users are selfish economic entities, with secrets and potentially conflicting preferences. Since some clients may be malicious, most clients expect a certain level of privacy and security within the system. The more private and secure the cloud is, the more motivated the users are to trust the cloud with important tasks. In the past few years, numerous studies have been published on different attack methods (side channel, escape to hypervisor, etc.), levels of isolation in cloud computing systems, and how to detect and limit attacks.

The introduction of economic aspects into the hypervisor, the basic layer of the cloud's operating system, may introduce new vulnerabilities. In addition, known attacks may be launched in different ways against an economically driven machine, or may be combined in different ways with economic attacks. In this paper we survey non-economic attacks in the context of the RaaS cloud, in order to learn how a successful, undetected attack may be launched, and what can be done to defend against it.

We begin this paper with a description of Ginseng, an example of an economic resource allocation mechanism in the hypervisor, in Section 2. In section 3 we survey cloud attacks which may prove relevant in the context of the RaaS cloud. We conclude in Section 4.

2 Allocating RAM Using an Auction

The division of resources according to economic mechanisms is discussed in several academic works [16, 18, 26, 15, 4] and implemented in several commercial clouds. Amazon's spot instances are sold using an auction, in which entire IaaS machines are rented. In CloudSigma's burst price method, clients pay a fast-changing price. [1] Both pricing mechanisms are declared to be affected by supply and demand, but their exact algorithm is kept secret [2]. In this work we use the terminology and mechanism used by Ginseng [4], the first economy-driven cloud system that allocates memory efficiently to selfish cloud clients. It does so by using the Memory Progressive Second Price (MPSP) auction, which is based on the Progressive Second Price (PSP) auction [16].

In a RaaS cloud, each guest has a different, changing, private (secret) *valuation* for memory: how much benefit it expects to gain from different quantities of memory. This is what guides the agent's actions in any economic transaction it performs (i.e., negotiations or auction bidding). We define the aggregate benefit of a memory allocation to all guests—their satisfaction from auction results—using the game-theoretic measure of *social welfare*. The social welfare of an allocation is defined as the sum of all the guests' valuations of the memory they receive in this allocation. An efficient memory auction allocates the memory to the guests such that the social welfare is maximized.

VCG [25, 6, 13] auctions optimize social welfare by incentivizing even selfish participants with conflicting economic interests to inform the auctioneer of their true valuation of the auctioned items. They do so by the *exclusion compensation* principle, which means that each participant is charged for the damage it inflicts on other participants' social welfare, rather than directly for the items it wins. VCG auctions are used in various settings, including Facebook's repeated auctions [17, 14].

The Memory Progressive Second Price (MPSP) auction, which Ginseng uses, resembles a VCG auction. It is a repeated auction in which each auction round takes 12 seconds. In each auction round the participants bid in order to rent

[1] CloudSigma's Pricing `https://www.cloudsigma.com/pricing/https://www.cloudsigma.com/pricing/`, accessed October 2015.

memory for the following 12 seconds. The MPSP protocol is illustrated in Figure 1. To work at this rate, the participants are not human clients who own the guest virtual machines, but rather software agents which work on their behalf, according to the valuation functions and business-logic algorithms embedded in them by their respective owners. Accordingly, the auction is orchestrated by the host's auctioneer, which is a software agent working on behalf of the cloud provider. Ginseng's structure in illustrated in Figure 2.

An MPSP auction round begins with the host's auctioneer announcing the quantity of memory which is up for rent in this round. Then, during the following 3 seconds, each participant may bid by stating a maximal unit price it is willing to pay for the memory (in terms of dollars per MB per second), and desired ranges of quantities it is willing to accept. The limitation of ranges allows the guest to refuse to get and pay for quantities from which it cannot benefit: for example, if the guest requires 1GB to avoid thrashing, and can enjoy up to 1.5 GB, it can refuse to get any memory quantity in the range 0-1GB, but be willing to rent any quantity from 1–1.5GB.

During the fourth second, the host's auctioneer determines the allocation and the bills each guest will have to pay. The host's auctioneer chooses the allocation which optimizes the social welfare according to the bids. The bills are computed according to the exclusion compensation principle: each guest pays according to the damage it causes other guests, as per their own reported valuation. For each guest i, the host's auctioneer computes the social welfare of all guests except guest i. Then it computes what the optimal allocation would be, had guest i not participated in the auction at all, and what the social welfare of all the other guests would be in that case. Guest i's bill is determined as the difference between these two computations. This method of computing payments and choosing an optimal allocation makes truthful bidding the best strategy for the guests: to state the real value they attach to getting a certain quantity of RAM.

Then the host announces the result of the auction to the guests, and gives them 8 seconds to prepare for a change in the memory allocation (e.g., release memory from the main application), before the change actually takes place. Finally, at the end of the 12 seconds, the host actually changes the memory allocation (if necessary).

When the host announces the results, each guest hears in private how much memory it won, and for what price. In addition, the host informs all guests of the lowest bid price among those whose bidders won any memory (denoted by P_{min_in}), and the highest bid price among those whose bidders did not win any memory (denoted by P_{max_out}). This information is broadcast for three reasons. First, the guest agents use this information to plan their next bids: they use it to approximate the borderline unit price bid, below which they are not likely to win any memory in the next round. Second, guest agents can acquire this information over time through the rejection or acceptance of their bids, so it is futile to try to hide it. Third, helping the guest agents learn the borderline unit price bid quickly can help the system stabilize, and thus reach the maximal social welfare quickly.

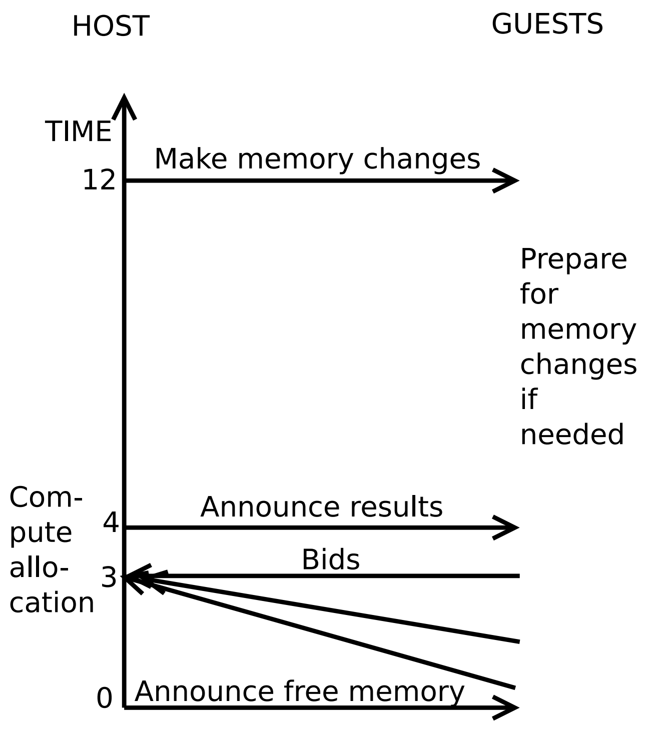

Fig. 1. Ginseng's MPSP protocol.

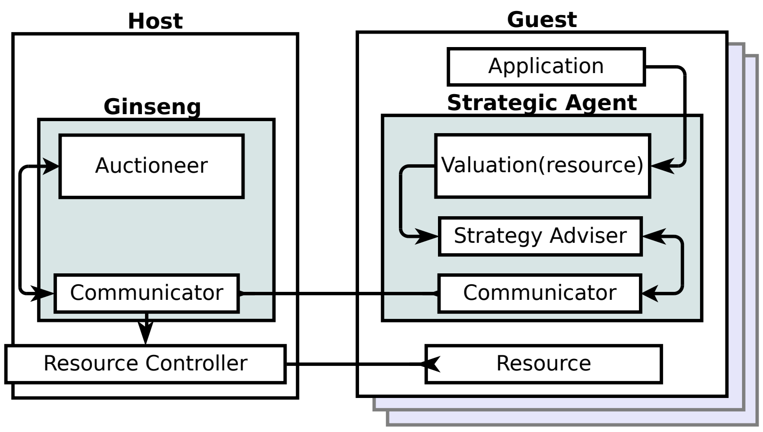

Fig. 2. Ginseng's structure. The auctioneer is a smart agent working for the host. It interacts with the strategic agent within the guest. Once the auction's results are determined, the host actually changes the resource allocation for the guest. The guest uses the resource to operate its applications (presumably one or more main applications, whose performance matters).

Although we refer in this work to MPSP terminology used in Ginseng, many of the observations we make here are also relevant to other mechanisms which mimic market pressure. In Ginseng, resource pressure is felt by participants in the bill they pay, which reflects the damage they caused to the social welfare. In mechanisms which rely on computing a clearing price (the highest price for which the demand is equal to the supply of exceeds it), resource pressure is felt through the increase in the clearing price.

3 Attacks on Traditional Clouds

Cloud computing is one of the most dominant paradigms in the information technology industry nowadays. More and more companies are moving to cloud computing solutions, which in turn requires attackers to find new and inventive ways to attack cloud computing systems. In this section we will classify attack types and explain how to map the internal system infrastructure, how to determine levels of isolation, and how to detect and limit attacks.

3.1 Classifying Attack Types

Many types of attacks can be launched against cloud computing systems. These include attacks aimed at obtaining information from innocent users or resource-freeing attacks (RFAs) to improve personal cloud performance. Younis et al. [27] survey the different types of cache based side channel attacks and point out weaknesses in currently researched solutions. Varadarajan et al. [23] show how to improve a VM's performance by forcing a competing VM to saturate some bottleneck (a resource used by the VM). This can slow down or shift the competing applications' use of a desired resource.

Our goal is to determine which of the above attacks are most likely to be launched against Ginseng, which are irrelevant, and which are most likely to succeed. Is it possible, for example, that an attack analogous to the RFA attack can be launched against Ginseng in order to obtain a maximum amount of memory at the expense of other system guests? This might be done, for instance, by slowly raising P_{max_out} and forcing the rest of the guests to exhaust their resources up to the point where they need to bid for a smaller amount of memory, thus freeing memory that the attacker can obtain for a lower bid. This type of attack can be carried out either by an attacker who is the highest bidder not allocated memory, or as a part of a grand-scheme collusion with other agents.

3.2 Mapping the Internal System Infrastructure and Determining Levels of Isolation

A successful attack within a cloud computing system usually requires a profound understanding of the internal system infrastructure and the capability to map the system's users and their level of isolation. Ristenpart et al. [20] showed that one can inexpensively probe and explore the system to determine the location of an instance in the cloud infrastructure, determine whether two instances are co-resident on the same physical machine, launch instances that will be co-resident with other users instances, and exploit cross-VM information leakage once co-resident.

Today it would be very hard—though not impossible—to do what Ristenpart et al. did in 2009. One reason is that the spot instances of only one zone can contain tens of thousands of machines. Moreover, machines today are live migrated and their IP may no longer indicate the IP or identity of guest machines co-located with them on the same physical machine. Finally, machine types are mixed on physical machines. All of this makes it harder to get a machine co-resident with a predesignated victim machine. However, on a RaaS cloud machine an attacker can directly gain from attacking guests sharing the same physical machine. We can learn from Ristenpart et al. that **if** the population is small and there are no migrations (as in the case of Ginseg today), **then** it is easier to learn about neighbors.

Zhang et al. [28] introduced a system called HomeAlone that allows guests to use cache-based side channel probing in order to determine their level of isolation. The HomeAlone system allows users to silence their activity in a selected cache region for a period of time, in which they can monitor cache usage and detect unexpected activity. This system can be used to find information on other virtual machines that share the same hardware but do not belong to the same owner.

Caron et al. [5] proposed a placement heuristic that allows guests to determine the level of security they require by stating co-residency requirements (alone, friends, enemies) or by stating the level of privacy/security they need. Ginseng does not currently take into consideration guest preferences regarding security/privacy levels or their co-residency requirements. This opens the door to numerous types of attacks that do not exploit the Ginseng protocol itself. In the future, it might be interesting to explore the option of determining the level of security/privacy and isolation between guests by allowing them to state bidding borders (for example a price/memory range that will define co-residency).

3.3 Detecting and Limiting Attacks

Security measures to detect and limit attacks can range from simple alarm systems (such as alarms triggered when trying to access an unauthorized area) to complex systems that monitor and learn user actions and performance over time.

1. Dolgikh et al. [7] showed that malicious users (attackers) can be detected in two phases: the training phase and the detection phase. In the training phase the system learns and classifies the "normal" behavior of system users. In the detection phase, user activities are monitored and observed; any deviation from the "normal" behavior is detected. This work is relevant to Ginseng in two manners:
   (a) Detection of malicious behavior. The attacker may also use the two phase approach. During the training phase, the attacker gathers information about the system's behavior, the neighboring guests, and their bid needs. Furthermore, the attacker can collect information regarding user schedules that can influence changes in supply and demand. It may even figure out the best time to attack. This information can be used to plan the attack, and in particular, the best cues for timing costly attacks. During the detection phase, the attacker will hunt for those cues, and launch the attack at the perfect time.
   During the training phase, the attacker may be monitored and certain actions may be considered as "out of the ordinary behavior" and thus stopped. However, this approach has its risks, as benign agents who online-learn their best strategy may be misidentified as attackers.
   (b) Automatic prevention of malicious behavior. Several mechanisms were proposed to prevent rapid memory allocation changes. These include an affine-maximizer based method, which taxes the difference between allocations [19], and a reclaim factor method [19], which controls the fraction of the memory that is reclaimed by the system to be sold by the next auction. This method resembles Waldspurger's tax on unused memory [26]. This means that an attack on Ginseng might fail due to the system's sluggishness.

We note that the sluggishness fails actions according to the action and not the intention behind it. Hence, it might also fail benign guest actions, if they are considered harmful to the system.

2. Shi et al. [22] presented Chameleon, a non-intrusive, low-overhead dynamic page coloring mechanism that provides strict cache isolation only during security-critical operations. If an attack on Ginseng is cache based, implementing the Chameleon mechanism may obstruct attempts to attack the system.
3. Varadarajan et al. [24] introduced the concept of soft isolation—reducing the risks of sharing through better scheduling. They show that a minimum run time guarantee for VM virtual CPUs that limits the frequency of preemptions can effectively prevent existing prime+probe cache-based side-channel attacks. This particular work is relevant to RaaS machines that use economic measures to allocate CPU resources, such as CloudSigma, which uses CPU burst prices. It is not directly relevant to the memory allocation method used by Ginseng.
   Note that Varadarajan et al.'s method protects the system at the cost of introducing an inefficiency in resource allocation. In that, it resembles the sluggish mechanisms, which protect the system against quick changes, at the expense of reducing its responsiveness.

4 Conclusion

We have reviewed several kinds of attacks on traditional clouds and on the new RaaS cloud. In addition to its vulnerability to regular attacks, an economically driven hypervisor is also vulnerable to attacks designed specifically for economic systems, using the special features of the system against it. Therefore, economic cloud systems have to be designed while considering both types of attacks, and include built-in defenses. This design might consist of patches to the original designs, protecting against specific vulnerabilities. It may even require a whole new mechanism, which prioritizes privacy and security over other considerations. There is a large volume of work addressing privacy in distributed systems where no trusted entity exists [9, 21, 12, 11]. However, it might be enough to assume that the resource provider, the host and its auctioneer are trusted entities. Data mining which preserves client privacy [8, 10] may be used to reduce the amount of information that leaks by announcing global data about the auction's result, such as P_{min_in} or P_{max_out}.

Acknowledgment

This work was partially funded by the Prof. A. Pazi Joint Research Foundation. We thank Dr. Eran Tromer, Prof. Katrina Ligett, Dr. Arik Friedman and Shunit Agmon for fruitful discussions.

References

1. O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. The resource-as-a-service (RaaS) cloud. In *USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*, 2012.
2. O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing amazon EC2 spot instance pricing. *ACM Trans. Econ. Comput.*, 1(3):16:1–16:20, Sept. 2013.
3. O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. The rise of RaaS: The resource-as-a-service cloud. *Commun. ACM*, 57(7):76–84, July 2014.
4. O. Agmon Ben-Yehuda, E. Posener, M. Ben-Yehuda, A. Schuster, and A. Mu'alem. Ginseng: Market-driven memory allocation. *ACM SIGPLAN Notices*, 49(7):41–52, 2014.
5. E. Caron and J. R. Cornabas. Improving users' isolation in IaaS: Virtual machine placement with security constraints. In *IEEE International Conference on Cloud Computing (CLOUD)*, pages 64–71, 2014.
6. E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, Sep 1971.
7. A. Dolgikh, Z. Birnbaum, Y. Chen, and V. Skormin. Behavioral modeling for suspicious process detection in cloud computing environments. In *IEEE International Conference on Mobile Data Management (MDM)*, volume 2, pages 177–181, 2013.
8. A. Friedman and A. Schuster. Data mining with differential privacy. In *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 493–502, 2010.
9. A. Friedman, I. Sharfman, D. Keren, and A. Schuster. Privacy-preserving distributed stream monitoring. In *Annual Network and Distributed System Security Symposium (NDSS)*, 2014.
10. A. Friedman, R. Wolff, and A. Schuster. Providing k-anonymity in data mining. *VLDB J.*, 17(4):789–804, 2008.
11. B. Gilburd, A. Schuster, and R. Wolff. k-ttp: a new privacy model for large-scale distributed environments. In *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 563–568, 2004.
12. B. Gilburd, A. Schuster, and R. Wolff. Privacy-preserving data mining on data grids in the presence of malicious participants. In *International Symposium on High-Performance Distributed Computing (HPDC)*, pages 225–234, 2004.
13. T. Groves. Incentives in teams. *Econometrica*, 41(4):617–631, Jul 1973.
14. J. Hegeman. Facebook's ad auction. Talk at Ad Auctions Workshop, May 2010.
15. F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.
16. A. Lazar and N. Semret. Design and analysis of the progressive second price auction for network bandwidth sharing. Technical report, Columbia University, 1998. `http://econwpa.repec.org/eps/game/papers/9809/9809001.pdf`.
17. B. Lucier, R. Paes Leme, and E. Tardos. On revenue in the generalized second price auction. In *International Conference on World Wide Web (WWW)*, 2012.
18. P. Maillé and B. Tuffin. Multi-bid auctions for bandwidth allocation in communication networks. In *IEEE INFOCOM*, 2004.
19. E. Posener. Dynamic memory allocation in cloud computers using progressive second price auction. Master's thesis, Technion, 2013.
20. T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security (SIGSAC)*, pages 199–212, 2009.

21. A. Schuster, R. Wolff, and B. Gilburd. Privacy-preserving association rule mining in large-scale distributed systems. In *Cluster, Cloud and Grid Computing (CCGrid)*, pages 411–418, 2004.
22. J. Shi, X. Song, H. Chen, and B. Zang. Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring. In *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 194–199, 2011.
23. V. Varadarajan, T. Kooburat, B. Farley, T. Ristenpart, and M. M. Swift. Resource-freeing attacks: improve your cloud performance (at your neighbor's expense). In *ACM Conference on Computer and Communications Security (SIGSAC)*, pages 281–292, 2012.
24. V. Varadarajan, T. Ristenpart, and M. Swift. Scheduler-based defenses against cross-vm side-channels. In *Usenix Security*, 2014.
25. W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1), 1961.
26. C. A. Waldspurger. Memory resource management in Vmware ESX server. In *USENIX Symposium on Operating Systems Design & Implementation (OSDI)*, volume 36, pages 181–194, 2002.
27. Y. Younis, K. Kifayat, and M. Merabti. Cache side-channel attacks in cloud computing. In *International Conference on Cloud Security Management (ICCSM)*, page 138. Academic Conferences Limited, 2014.
28. Y. Zhang, A. Juels, A. Oprea, and M. K. Reiter. Homealone: Co-residency detection in the cloud via side-channel analysis. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 313–328. IEEE, 2011.