Research Statement

Venkatanthan Varadarajan

Resource management across diverse workloads is a non-trivial and challenging problem. The complexity of the problem skyrockets when the system needs to arbitrate resources across arbitrary and potentially malicious users. Such is the problem faced by multi-tenant public cloud systems. These systems, apart from achieving profitable resource utilization for the cloud provider, should also guarantee reasonable performance and, more importantly, make the users feel as secure as when running tasks in their own private infrastructure. Current state-of-the-art public clouds are far from achieving the necessary level of isolation between its users to provide this guarantee. **My research focuses on systematic evaluation of isolation in the presence of malicious users in public clouds and building operating systems with improved isolation without compromising on efficiency.**

My research has helped: (i) find new, previously unknown security issues with resource managers that provides monetary advantages for a malicious or greedy user [1,2], (ii) provide detailed, systematic evaluation of the co-location or co-residency attacks on major Infrastructure and Platform as a Service public clouds (IaaS, PaaS), e.g., Amazon EC2 and Heroku, respectively [4], and (iii) develop new design paradigm that tackle the lack of performance isolation through simple CPU scheduler primitives that improve security and privacy of public cloud tenants without compromising on performance [3]. All of these projects have been published in top-tier security and systems conferences (ACM CCS, USENIX Security, ACM SoCC) and one paper was recently recognized as one of the top ten best applied security research at CSAW NYU-Poly in 2014. Throughout these projects, I have demonstrated and honed a skill set that is essential for system security research: thorough evaluation of complex systems across two important dimensions of security and performance. It will also be evident from these works (detailed below) that I meticulously follow an important system design principle: “a security solution without practical deployability is no solution”. The importance of this principle is often overlooked in system security research.

My research vision is to build systems that improve the security of public clouds that is equivalent or better than what is achievable in a restrictive private infrastructure without compromising on the economic and efficiency benefits of public clouds. The completed projects detailed below help take the first step towards this goal by understanding the isolation guarantees provided by the state-of-the-art public clouds. I believe, my accrued experience and skill set will aid me in moving towards this vision.

**New resource-gaming attacks in public clouds.** The term isolation in virtualized environments is often overloaded and refers to two different but related properties of a multi-user system. First, *logical isolation* guarantees that one user cannot directly name and access another user’s allocation of the shared resources. Second, *performance isolation* guarantees that one user cannot observe any performance variations that are dependent on another user’s usage of a shared resource.

Although logical isolation across arbitrary users is essential, it is not sufficient for public cloud systems. This is corroborated by prior works showing that performance interference enable several security attacks in public cloud starting from performance degradation and denial of service attacks to complex side-channel attacks. Apart from these natural side-effects of performance interference, the pay-as-you-go pricing model of public clouds uniquely motivate a new class of attacks we call resource-gaming attacks, where a malicious user can manipulate his own resource allocation by cleverly controlling one’s own resource consumption and inducing performance interference on other users. Previous research has demonstrated a resource-gaming attack on CPU schedulers used in public clouds, where a malicious VM can surreptitiously steal extra CPU cycles...
from a VM sharing the same CPU (called a cycle-stealing attack).

My research discovered a more general class of attacks called Resource-Freeing Attacks (RFAs), where a malicious user can manipulate a resource scheduler to shift resource allocation away from a neighboring VM and towards a target VM of interest that is sharing the same host [2]. For this shifting of resource allocation to work, the attacker creates a bottleneck on one or more of the neighboring VMs that contend for the target resource of interest. One way to create bottleneck on neighboring VMs is by using the public interface exposed by them (e.g., webservers listening on standard ports). To demonstrate how a malicious user can utilize a RFA, I showed that, on Amazon EC2, a RFA can improve performance of a cache-bound workload when running along side a highly loaded webserver by about 66% by freeing up last-level cache bandwidth. RFAs are not limited to this particular instance and generalize to schedulers of any other shared resources (e.g., disk, network). A key contribution of this work is the implications of scheduler designs on RFA-like vulnerabilities. For instance, RFAs are not profitable on resources allocated using a non-work conserving scheduler as there is no additional resource allocation available for a greedy customer to game. On the other hand, a fully non-work conserving scheduler hurt resource utilization as it wastes unused idle resources. Hence, striking the right balance between non-work conserving and work-conserving nature of schedulers is essential to balance the trade-off between security (performance isolation) and efficiency (resource utilization).

The previous work demonstrated the problem of performance variability due to poorly isolated per-machine resources. In a second work [1], my collaborators and I showed that physical resource heterogeneity in public clouds is also a significant contributor to performance variability. Customers experience measurably different performance for the same amount of money and type of VM instance. For instance, VMs are often hosted on a public cloud infrastructure that include a mix of old and new generation machines. Running a set of microbenchmarks that exercise various resource like cache, memory and disk, we found that new generation machines performed significantly better than others. This performance variation incentivize a greedy customer to game the cloud interface to launch and terminate VM instances until one or more VMs are hosted on a high-performing machine. We call this placement gaming and in this project we explored various black- and gray-box strategies that improve performance to cost ratio for the user. Using a placement gaming strategy on Amazon EC2, we boosted the performance of a real-world network-bandwidth intensive application by 34%. My role on this project involved building a VM placement model and simulator for fast evaluation of placement strategies in order to find the best strategy for Amazon EC2. The research contributions of this publication is under a patenting process.

Defenses against side-channel attacks. The previous projects explore various adversarial behaviors that exploit the lack of performance isolation for monetary benefits. However, the lack of performance isolation may also violate privacy guarantees by inadvertently leaking cryptographic secrets across VM’s isolation boundary. These attacks are called access-driven side-channel attacks and many recent works demonstrate these complex attacks in a realistic public cloud setting. The success of these attacks rely on three essential requirements: shared per-core state, access to a high-precision system clock and the ability to frequently preempt a victim VM at short intervals (cross-VM preemptions). All prior defenses tried to either: 1. avoid sharing by strictly partitioning or carefully allocating physical resources, or 2. add noise to the timing channel or remove it altogether. Note that none of the mechanisms are deployed in practice as they either incur significant overhead or result in the loss of important features like fine-grained timers that benefit general-purpose cloud applications.

My research addresses this problem with a focus on ease of deployability by limiting potentially dangerous cross-VM preemptions at the CPU scheduler [3]. Recall that the quick cross-VM
preemptions is one of the three requirement for a successful access-driven side-channel attack. Every context switch between VM contexts may leak secret information via shared resources. Rate-limiting these leaky cross-VM preemptions provides a form of isolation that is less stringent than strict partitioning of resources, which we call soft-isolation. My research showed that such a soft-isolation mechanism is sufficient to render existing access-driven side-channels useless. Further, the proposed soft-isolation mechanism is a simple scheduler primitive that ensures a VM is guaranteed to run for a minimum runtime before allowing a cross-VM context switch. Such a mechanism already exists in all widely used CPU schedulers and thus, ease deployability. In the evaluation, I showed that the mechanism incurred no overhead in the average case and in the worst case incurred at most 7% increase in the tail latency of memcached-like interactive workloads. In contrast, prior mechanisms add overheads as high as 2.8x, require new hardware or remove the ability to make fine-grained measurements. The promising results of this simple defense invited dialogues with one of the popular cloud providers.

**Co-location attacks in public clouds.** Resource-gaming attacks like cycle-stealing and RFAs that violate performance guarantees or side-channels and covert-channels attacks that violate privacy or others exploit multi-tenancy in public clouds. All these attacks consists of two steps: placement and breach. Placement involves placing an adversarial VM on the same host as a target victim VM followed by executing any desired attack (i.e., breach). Numerous works demonstrated various security and privacy violations that are possible in the second step (e.g., side-channels), but there is only one work, published in 2009, that demonstrated the placement or co-location vulnerability (the first step) in public clouds. Particularly, this prior work showed that poor logical isolation in cloud management infrastructure enabled mapping of hosted VMs on the cloud’s physical infrastructure with IP address of the VMs (called cloud cartography). Since then, there have been many performance and security enhancing technologies deployed in public clouds (e.g., Virtual Private Clouds or VPCs) that raise the bar for an adversary. In fact, I found that cloud cartography is no longer possible in current clouds. Hence, it is not clear if VM placement or co-location attacks are possible in these security-hardened public clouds.

In spite of this state of public clouds, using systematic evaluation of placement policies of public clouds I found that three popular public clouds (Amazon EC2, Google Compute Engine and Microsoft Azure) are still vulnerable to co-location attacks on chosen victim VMs [4]. Here chosen refers to simulated victim VMs that are under our control. Though the security enhancing technologies raise the bar for an adversary, it is still possible to arrange two VMs from different accounts on the same host by clever interaction with the cloud interface. We call the interactions with the cloud interface as placement strategies. For example, I found that on some clouds, launching adversary instances one hour after the launch of victim instances is one of many placement strategies that significantly increases the chance of co-location. As part of this work, I also defined a framework to quantify the cost of such vulnerable placement strategies (or placement vulnerabilities) and evaluate placement policies used in public clouds against a secure reference placement policy.

**Other works.** Prior to my focus on cloud computing security, I collaborated on a several of systems project where I particularly exercised my system building skills. I implemented a highly optimized fast RPC subsystem as part of Aerie, which is a memory-mapped filesystem for non-volatile memory system [5]. Crossing system boundary for inter-process communication has a high cost and this cost is particularly unacceptable for applications that are highly sensitive to performance. The worst-case null RPC latency on an Intel Core i7 took 0.8µs. The fast user-level RPC design was inspired from the earlier work on User-level Remote Procedure Call (URPC) and synchronization primitives based on Anderson’s array-based queue lock.

As part of my internship at VMware, I productized a research prototype called vScale, a tool
that automatically scales a multi-tier applications horizontally with the aim of guaranteeing a 
user-specified service-level agreement on end-to-end performance under changing load conditions. 
vScale is currently in production at VMware Inc.

**Future research directions.** Public cloud computing is continuing to entice small- and medium-
sized businesses with its simple pricing model and cheap compute resources. Unfortunately, these 
opportunities also create an equally cheap and comfortable battleground for adversaries. Al-
though the security in the state-of-the-art public clouds has immensely improved in the recent 
past, prior works have shown that they are still far from achieving the levels of security possible 
in highly-restrictive private infrastructure. Hence, the demand for designing practical, deployable 
secure systems with better security guarantees for public clouds has never been higher. I plan to 
work on several specific problems to raise the security bar for public clouds.

*Secure scheduling of stateful resources.* Stateful resources like processor caches, translation looka-
side buffers (TLBs), etc. are often the target of security attacks like side-channels because of their 
lack of performance isolation. These resources are stateful as they have different performance pro-
files based on how competing tasks use them. Better performance isolation will not only make 
designing timing channels harder but also improve the quality of service for cloud applications. 
This is even more important in PaaS environment as opposed to IaaS, where the ratio of tenants 
to physical machine resources is higher than in IaaS. I plan to improve performance isolation for 
the stateful resources through better accounting and interference-aware task scheduling. This is 
a viable approach for two reasons. First, the ubiquitous presence of hardware performance counters 
in modern systems for stateful resources enable cheap fine-grained accounting. Second, with 
this accounting information, fine-grained timeslicing of a physical CPU helps control allocation 
of all these diverse per-core stateful resources via a single, performance-aware CPU scheduler. In 
addition, the accounting information can also enable a better multi-core scheduler to decide task 
to core allocation.

*Anomaly detection in Linux containers.* Linux containers are gaining widespread popularity not 
only in PaaS public clouds but also in managing private enterprise datacenters for supporting 
hybrid clouds. These PaaS clouds gives easy access to compute resources that are cheaper than 
their IaaS counterpart. This may lead to attackers using these cheap compute resources for 
running malware tasks like botnets, distributed denial of service attacks, side-channels, password-
crackers, etc. Traditionally network middleware (e.g., Intrusion Detection Systems, wirefall) are 
used to detect and inhibit some of these attacks that rely on network. But, a task's resource usage 
pattern may also reveal the nature of the task (i.e., potentially malign or benign), which could 
be used to thwart adversarial tasks. Anomaly detection mechanism often relies on access to task 
state. In hypervisors-based virtualization environment accesses to task state relies on complex 
VM introspection techniques for each type of guest operating systems, which often deliberately 
brake the strong VM isolation boundaries. In contrast, container-based virtualization (e.g., Linux 
containers) have weaker isolation than their hypervisor-based counterparts. This weakness enable 
easy access to task states and hence can be cleverly used to strengthen the container-based 
ecosystem with better anomaly detection mechanisms.

*Key management for cloud applications.* Many cloud applications follow a multi-tier architec-
ture (e.g., webserver, database server, map-reduce cluster), where there are several individual 
VMs constituting a distributed application that often spans multiple datacenters for scalability, 
reliability and availability. Securing network communications between these individual VMs is 
paramount as some communications paths may use insecure communication links, e.g., openly 
accessible cross-datacenter links or backdoored network routers. To address these issues develop-
ers often turn to secure-channel protocols like TLS that were originally designed for the Internet.
Unfortunately, there are no easy way for developers to generate and manage keys/certificates and configure TLS for a distributed application. The process often involves tedious and error-prone manual administration by an admin who may have minimal security expertise. As a result, secure channel configurations deployed in the wild are often weak and vulnerable. Example of poor security configurations include: using self-signed certificates for each VM and sharing the same credentials across all nodes. I plan to understand the challenges of an average administrator and design a tool that plugs into any distributed application with very simple interface that can immensely improve the overall standard of security in cloud applications.

References


