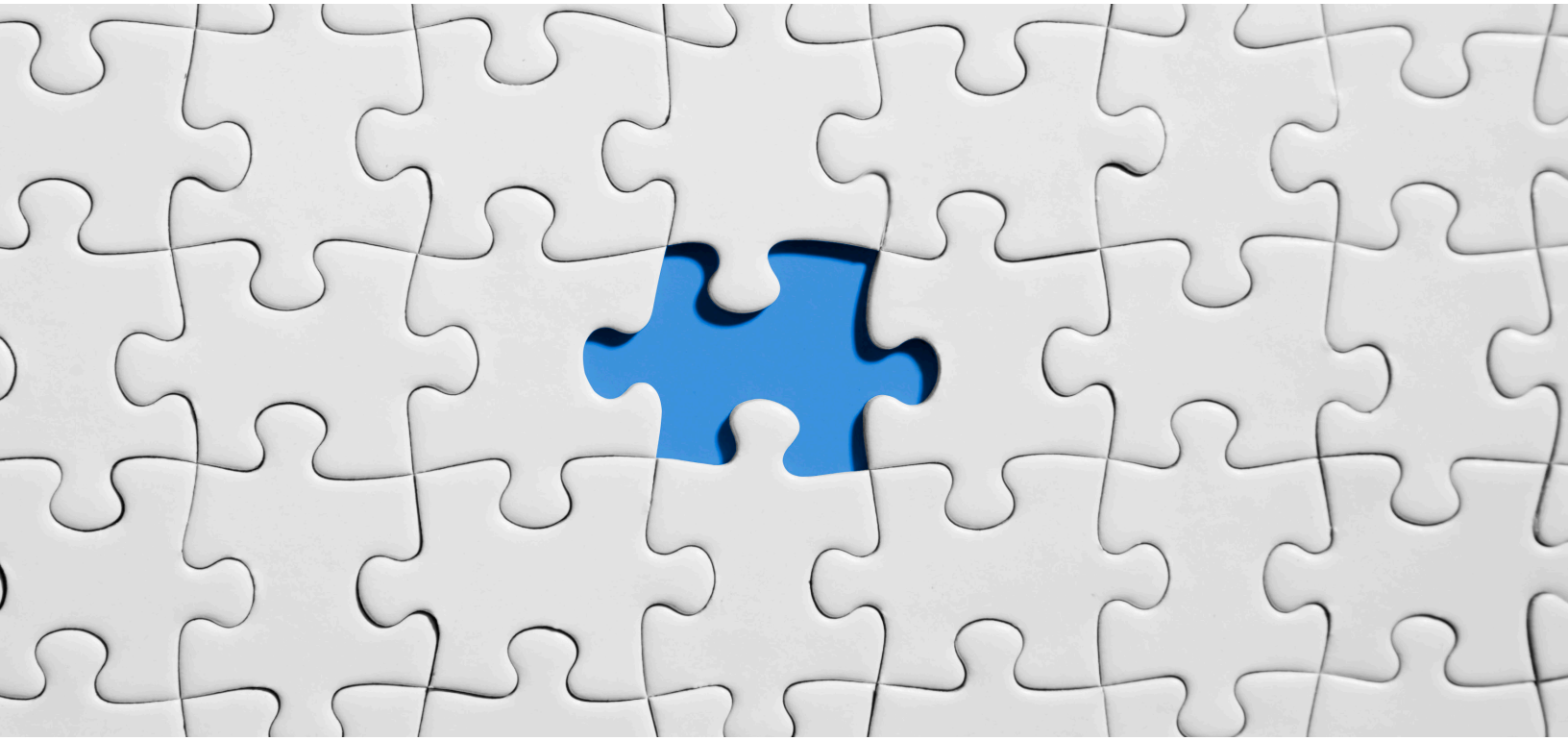
DELL Technologies

SECURING MEMORY SHARING FOR CLOUD TENANTS



Sameh Talaat

Senior Advisor, Customer/Technical Training

Dell Technologies

Sameh.talaat@dell.com

Hussein Baligh

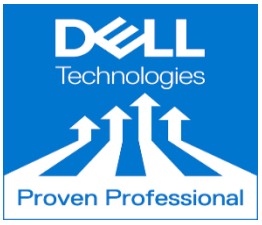
Senior Advisor, Sales Engineer Analyst

Dell Technologies

hussein.baligh@dell.com

DELL Technologies Proven Professional

Knowledge Sharing Article



The Dell Technologies Proven Professional Certification program validates a wide range of skills and competencies across multiple technologies and products.

From Associate, entry-level courses to Expert-level, experience-based exams, all professionals in or looking to begin a career in IT benefit from industry-leading training and certification paths from one of the world's most trusted technology partners.

Proven Professional certifications include:

- Cloud
- Converged/Hyperconverged Infrastructure
- Data Protection
- Data Science
- Networking
- Security
- Servers
- Storage
- Enterprise Architect

Courses are offered to meet different learning styles and schedules, including self-paced On Demand, remote-based Virtual Instructor-Led and in-person Classrooms.

Whether you are an experienced IT professional or just getting started, Dell Technologies Proven Professional certifications are designed to clearly signal proficiency to colleagues and employers.

Learn more at www.dell.com/certification

Table of Contents

1. Introduction

Cloud Computing has become the backbone of the IT business, lowering CapEx and providing a top-notch, yet affordable class of service to all, from companies to individuals. Cloud depends heavily on virtualization technologies that allow resource abstraction and pooling to efficiently optimize resource utilization. However, it is a common practice to share memory resources between Virtual Machines (VMs), raising concerns of shared technology vulnerabilities. The technique that acts as a deduplication on memory level proved to be very fruitful, specifically in Virtual Desktop Infrastructure (VDI) environments, taking advantage of having many identical memory pages between VMs with similar Operating Systems and Applications. Although memory sharing optimizes resource utilization provided for Desktop as a Service (DaaS), it also is susceptible to Side Channel Attacks (SCA) between tenants. A VM that belongs to one of the tenants can have a software installed that scans the memory pages assigned to it and measure slight latencies introduced by page sharing. Thus, it can define which memory pages are shared with other VMs which can lead to sensitive information – such as Operating Systems and applications used by other tenants – to obtain insight to possible entry points and, in a worst-case scenario, identify common encryption keys.

Industry-leader VMware acknowledged the problem and introduced ***Salting*** as the solution. This technique can limit memory sharing to be applied only within trustworthy virtual machines of the same tenant, thus limiting SCA possibilities. Salting can be criticized for its lack of security, because salting is based on giving the VMs of the same tenant a common ID (the salt value) to guide the Cloud Services Provider (CSP) about which VMs are allowed to share memory pages, which raises crucial questions about where the salt values are stored at the CSP's end. Is it secured? Does it add an additional entry point for attackers by which the attack surface is increased? We have reasons to believe that in a VMware environment the salt value is vulnerable which may lead to a confidentiality domino effect as it exposes information, i.e. how many tenants the CSP has, how many VMs each tenant has and how fast is each tenant expanding or shrinking its business. These reasons, which the article aims to explain, show that Salting as implemented today is not a rigid security door. It can be penetrated and can lead to Transparent Page Sharing (TPS) security concerns being re-exploited once again.

This article aims to strengthen tenant confidentiality by finding smart alternatives for storing the salt values in the database of vCenter Server or any similar administration tools of other hypervisors. A vendor-agnostic concept of integrating the salt values within the identity authentication is needed, which sticks the salt value of one tenant to its identity with the possibility of involving Open Identity Providers. This concept introduces three main benefits:

1. **De-centralization of the salt values information**, limiting the risk of confidentiality attacks to a tenant-by-tenant basis. This dissolves the likelihood of a global CSP scale attack.

2. **Resolving security concerns of the salting technique**, which makes it more convenient to use memory sharing in the cloud.

3. **Linking the salt value with the identity authentication, thus minimizing the attack surface**. While the salt value of each tenant is a critically sensitive piece of information for the tenant's confidentiality, the tenant's identity is by far the most critically sensitive information. Sticking a less critical piece of information to the most sensitive one minimizes the attack surface. In other words, if an attacker obtains the tenant's credentials by eavesdropping or any other method,

identity spoofing would give the attacker access to the tenants' VMs themselves with full control, which makes knowing the salt value meaningless. By analogy, there is no advantage in managing to hide the information of what household appliances you have at home if a burglar has the key to your house anyway and is already in there stealing it.

To summarize, the purpose of this article is to:

- **Shed light** on a previously unexamined security vulnerability within a currently implemented solution that hasn't been highlighted in any previous work.

- **Define** an alternative for a more effective application of memory sharing.

- **Convince** CSPs and virtualization industry players to adopt safer VDI environments in the cloud.

2. Transparent Page Sharing – How does it work?

When TPS feature is enabled, the VMware ESXi hypervisor tracks identical memory pages shared among the VMs. A hash value is calculated by the ESXi hypervisor for all memory pages for comparison. Once identical pages are recognized, either on multiple VMs on a host or within a single VM, it deduplicates them by erasing all versions but one. The hypervisor creates pointers to the deduplicated physical memory pages for multiple references.

Assuming a VDI environment with 50 VMs all with the same OS, then 50 memory pages may exist with identical content. ESXi would keep just one and delete the other 49 pages that are substituted by pointers to the only remaining page.

The vmkernel handles the explained algorithm automatically, which entails identification, consolidation and referencing of identical pages. As a result, total host memory consumption is minimized, which helps in adopting more efficient memory overcommitment with fewer chances of memory swapping.

Figure 1 demonstrates how TPS works

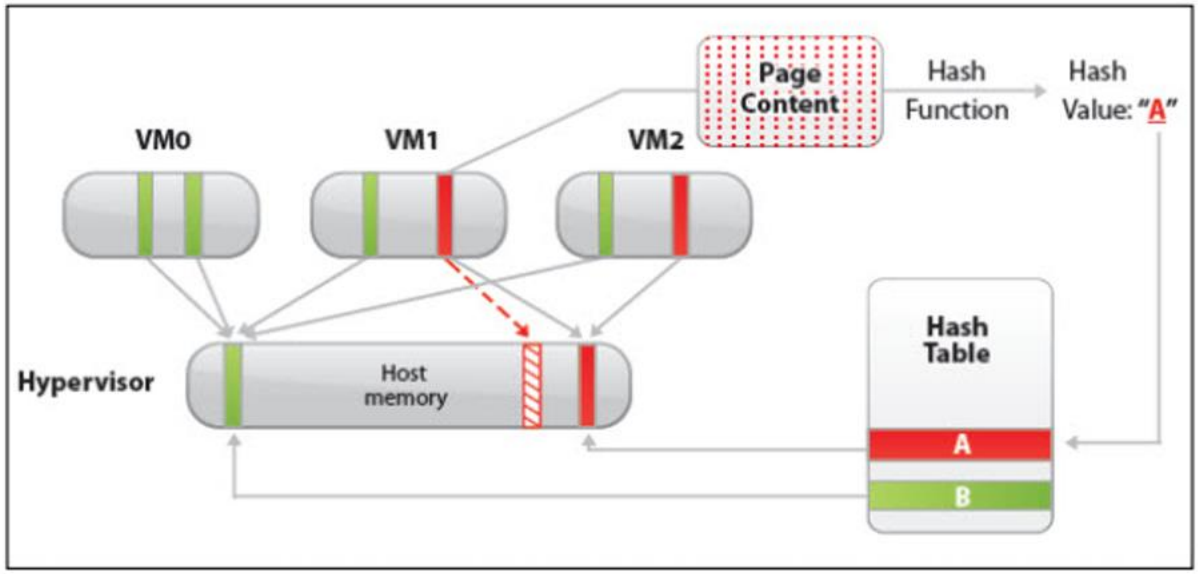


Figure 1: Transparent Page Sharing (TPS) operations

A standard copy-on-write (CoW) technique is used to handle writes to the shared host physical pages. Any attempt to write to the shared pages will generate a minor page fault. In the page fault handler, the hypervisor will transparently create a private copy of the page for the virtual machine and remap the affected guest physical page to this private copy. In this way, virtual machines can safely modify the shared pages without disrupting other virtual machines sharing that memory. Note that writing to a shared page does incur overhead compared to writing to non-shared pages due to the extra work performed in the page fault handler.[1]

3. The problem - TPS security issues

According to VMware:

> *"Published academic papers have demonstrated that by forcing a flush and reload of cache memory, it is possible to measure memory timings to try and determine an AES encryption key in use on another virtual machine running on the same physical processor of the host server if Transparent Page Sharing is enabled between the two virtual machines."* [2]

This was in reference to a major academic paper entitled, **"Wait a minute! A fast, Cross-VM attack on AES"[3]**, written by a group of individuals from Worcester Polytechnic Institute in 2014 that highlighted this issue.

VMware acknowledged the studies in 2018 as below:

> *"This article acknowledges the recent academic research that leverages Transparent Page Sharing (TPS) to gain unauthorized access to data under certain highly controlled conditions and documents VMware's precautionary measure of restricting TPS to individual virtual machines by default in upcoming ESXi releases"* [2]

It is important to mention that memory page sharing vulnerabilities are not specific to VMware's technology. They can generally apply to all hypervisors implementing memory page sharing regardless of vendor.

A study[4] published in 2011 exploited an OS-optimization, namely Kernel Samepage Merging (KSM), and managed to recover user data and identify a user from a co-located VM in a Linux Kernel-based VM. VMware acknowledgment applies to the below as well:

> *"In our experience on KSM (kernel samepage merging) with the KVM virtual machine, the attack could detect the existence of **"sshd"** and **"apache2"** on Linux, and **"Firefox"** on WindowsXP. The attack also could detect a downloaded file on the Firefox browser when the caching is enabled. Even if the network is encrypted by TLS/SSL, a downloaded file is detected."* [4]

> *"The significance of this study is that not only it is possible to exploit the memory deduplication to detect the existence of a VM, but one can also detect the processes running on the target VM. This leads to cipher specific attacks and information thefts, as demonstrated by Suzaki et al. In this study, the authors were able to detect security precautions such as anti-virus software running on the co-resident target VM."* [3]

> *"A serious security problem that threatens VM isolation, stems from the fact that people are using software libraries that are designed to run on single-user servers and not on shared cloud hardwares and VM stacks. For privacy critical data, especially cryptographic data, this gives rise to a blind spot where things may go wrong. Even though classical implementation attacks targeting crypto systems featuring RSA and AES have been studied extensively, so far there has been little discussion about safe implementation of cryptosystems on cloud systems."* [3]

An example for implementing attacks on AES, was proposed by Bernstein[5]. It was using a Cache Timing attack, where the slight time differences of cache access attempts are used to recover the secret key.

Another study by Gullasch[6] uses **Flush+Reload** attack among AES memory access. The attack implemented in the study recovers the key with as few as 100 encryptions.

4. The currently implemented solution – Salting

Aware of the security challenges of TPS, VMware introduced additional TPS management capabilities that were added to the ESXi hypervisor. The new implementation of TPS included granular control over the scope of memory deduplication.

- **Intra-VM TPS:** TPS deduplicates identical memory pages within a virtual machine, but no page sharing with any other virtual machines.
- **Inter-VM TPS:** TPS deduplicates identical memory pages within a virtual machine as well as with other virtual machines with identical content.

Salting was used to allow such granular control over which virtual machines are participating in TPS. With the new salting settings, the virtual machines can share identical pages only if their salt values are identical. This is achieved through setting 2 values as below:

1. A host configuration option called **Mem.ShareForceSalting** that enables or disables salting.
2. An attribute called **sched.mem.pshare.salt** to be set in the VMX file of the VM acts as a manually enforced salt value.

There are 3 possible values of the **Mem.ShareForceSalting :**

- **0** – TPS operates as in the pre-Salting era. Both intra-VM and inter-VM TPS occurs uncontrolled. The value of the **sched.mem.pshare.salt** in the VMX file is ignored even if present.

- **1** – If specified, the salt value is taken from **sched*.mem.pshare.salt***. If not specified, TPS operates as in the pre-Salting era by considering salt values for the virtual machine as 0.

- **2** – Default value. If the ***sched.mem.pshare.salt*** is not specified, salt value is set to a unique value, which is the **vc.uuid**. Therefore, no Inter-VM TPS occurs. If specified, then Inter-VM TPS occurs only among the VMs with identical salt values. Intra-VM TPS operates in either case.

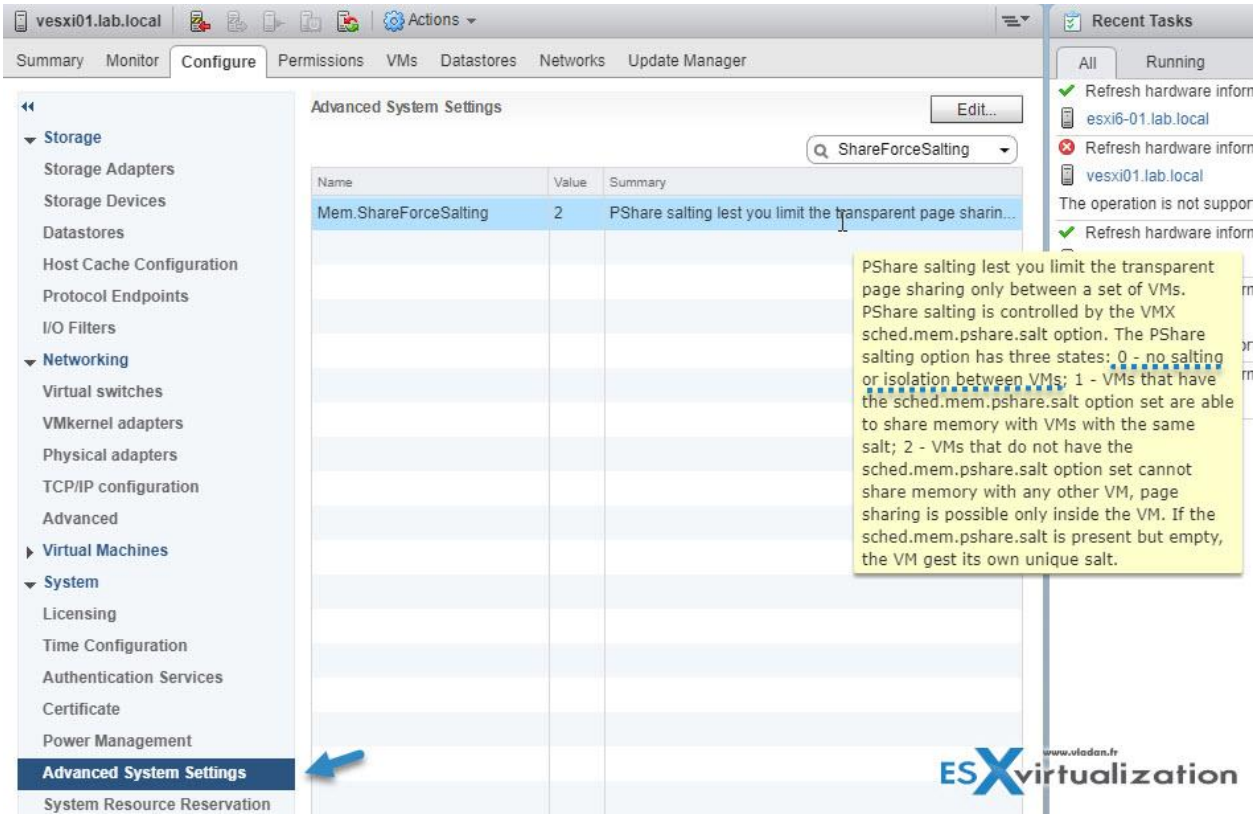


Figure 2: Salting - Advanced System Settings

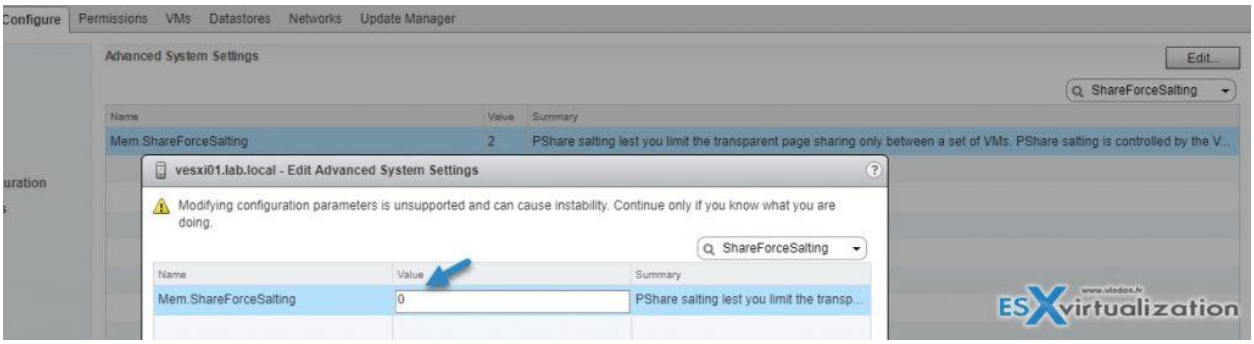


Figure 3: Edit - Advanced System Settings

Salting is enabled by default, (**Mem.ShareForceSalting=2**) and each virtual machine has a unique salt value. This means that the default setting enables intra-VM TPS and disables inter-VM TPS. The salt value is written in the **.vmx** configuration file of the virtual machine. The uniqueness of the default salt value is achieved by setting it to be equivalent to the **vc.uuid.** The **vc.uuid** is a unique identifier given to the virtual machine by vCenter Server. In other words, it can never be repeated within vCenter Server instance and consequently, it can never be repeated on any of the ESXi hosts sharing resources among the virtual machines.

Table 1 shows different TPS settings to control TPS operation for individual virtual machines.

Mem. ShareForceSalting (host setting)	**sched.mem.pshare.salt (per VM setting)**	**vc.uuid (per VM setting)**	**Salt value of VM**	**TPS between VMs (Inter-VM)**	**TPS within a VM (Intra-VM)**
0	Ignored	Ignored	0	Yes, among all VMs on host.	yes
1	Present	Ignored	sched.mem.pshare.salt	Only among VMs with same salt	yes
1	Not Present	Ignored	0	Yes, among all VMs	yes
2	**Present**	**Ignored**	**sched.mem.pshare.salt**	**Only among VMs with same salt**	**yes**
2 (default)	**Not Present (default)**	**Present (default)**	**vc.uuid**	**No inter-VM TPS**	**yes**
2	Not Present	Not Present	random number	No inter-VM TPS	yes

Table 1: TPS Settings and Operations

5. The solution's deficiency - Salting vulnerabilities

According to VMware:

> *"By default, salting is enabled after the ESXi update releases mentioned above are deployed, (Mem.ShareForceSalting=2) and each virtual machine has a different salt. This means page sharing does not occur across the virtual machines (inter-VM TPS) and only happens inside a virtual machine (intra VM). When salting is enabled (**Mem.ShareForceSalting**=1 or 2) in order to share a page between two virtual machines both salt and the content of the page must be same. A salt value is a configurable vmx option for each virtual machine. You can manually specify the salt values in the virtual machine's vmx file with the new vmx option **sched.mem.pshare.salt**. If this option is not present in the virtual machine's vmx file, then the value of **vc.uuid** vmx option is taken as the default value. Since the vc.uuid is unique to each virtual machine, by default TPS happens only among the pages that belong to a particular virtual machine (Intra-VM). If a group of virtual machines are considered trustworthy, it is possible to share pages among them by setting a common salt value for all those virtual machines (inter-VM)."* [7]

Let's take a moment and analyze the information above in 2 points:

1. Salting is enabled by default.

2. VMs by default will have a salt value equivalent to their **vc.uuid**s to ensure that the salt values are unique and no inter-VM TPS will take place unless specified.

Considering these two points, raises crucial questions about how vCenter Server manages the uniqueness of the **vc.uuid**s it provides to the VMs in the case of a hybrid cloud model.

When a VM is moved from a tenant-owned facility to a public cloud, there might be a **vc.uuid** conflict. This is because the **vc.uuid** might be already used in the destination vCenter running in the cloud.

Question 1: How does vCenter handle this conflict?

It assigns the received VM a new **vc.uuid** that does not conflict with any of the VMs that it manages. [8]

Question 2: What will the vc.uuid of the VM be if the VM is migrated back to the tenant-owned facility?

The VM will remain with the **vc.uuid** that was given to it in the cloud unless it conflicts with the local vCenter.

Now, based on that understanding let's simulate a security attack scenario that can shed light on an unnoticed vulnerability in the Salting mechanism.

> **Step 1:** An attacker creates many VMs on his local environment with **vc.uuid**s that are generated by a random generator code (which is available, for example, through APIs from Cloud Orchestrators)
>
> **Step 2:** The attacker migrates his VMs to the Cloud.
>
> **Step 3:** The attacker migrates them back to his local facility.

Step 4: The attacker checks if any of the VMs had its **vc.uuid** changed.

Step 5: The attacker recognizes one or more modified **vc.uuid**s and now understands that the old **vc.uuid**s are actually assigned to other VMs that belong to other tenants in the cloud! The attacker also finds the default salting values of those VMs in the cloud as they are to match their **vc.uuid**s!

Step 6: The attacker modifies the **sched.mem.pshare.salt** value in the VMX files of his VMs to match the old **vc.uuid**s which he now knows are default salt values for some cloud-resident VMs.

Step 7: The attacker migrates his VMs once again to the cloud.

As seen in **Table 1**, a VM resident on the cloud with default settings will have its salt value as its **vc.uuid (5th row in Table 1, blue font)**, while the VM that the attacker re-migrated to the cloud in **Step 7** has a manually set salt value to be considered by the host **(4th row in Table 1, red font)** which is now equivalent to the salt value in another VM with default settings.

Result of the simulation attack

- Salting mechanism has been cracked.
- The attacker's VMs can have inter-VM TPS with VMs from other tenants. The more VMs used in the attack, the greater likelihood to coexist with victim VMs on the same host considering vSphere features like **DRS** and **DPM** that move VMs around the hosts.
- Considering the security vulnerabilities of inter-VM TPS between untrusted VMs, now the attacker can have sensitive data of other tenants exposed. That can include the running OS, the used Antivirus SW, the used internet browser and the AES encryption Key used.

6. The proposed modification

Based on the attack simulation above, we propose a slight re-engineering of the salting mechanism.

The action proposed is to stop the dependency on **vc.uuid** as a default salt value. Instead, another unique identifier is to be used that cannot be a retrievable parameter by migrating the VMs back and forth in a hybrid cloud environment.

The concept

Minimizing the attack surface by sticking less sensitive information to the most sensitive information. That consolidation of potential attack entry points makes it pointless to expose the less sensitive information in the worst-case scenario. Recalling the analogy, there is no advantage of managing to hide the information of items you have at home if a burglar already has the key to your house and is actually in there stealing it.

The alternative to vc.uuid

The proposal is to use a cloud's tenant ID as the default salting value instead of the **vc.uuid**. A tenant's ID managed by the authentication provider is the most sensitive information about the tenant's environment. Spoofing the identity of a tenant in the cloud imposes threats on all aspects of security (**C**onfidentiality, **I**ntegrity, and **A**vailability). It gives near full control on the tenant's VMs and other related assets in cloud.

Use of the tenant's ID instead of the **vc.uuid** has 2 advantages that minimize the attack surface:

1. It makes the default salt values part of the most Trusted Computing Base (TCB). The most protected information behind the last line of defense. In case of leakage of tenant's authentication credentials, exposing the salt value does not add extra capability to jeopardize the VMs security that has collapsed already. Thus, it is a consolidation of entry points to minimize the attack surface.

2. It removes another high-level attack entry point. The **vc.uuid**s are centrally stored in the vCenter Server database. An attack targeting the database or targeting the API used in communication between the Orchestrator Software and vCenter can expose all the **vc.uuid**s, and consequently, all default salt values. Depending on an alternate unique identifier that is external to vCenter database makes the salt values more immune to a CSP-scale attack which also minimizes the attack surface.

There are several alternatives for the tenant ID to be used. It could be a private hashed value of a combination of domain/user/authentication provider data, which is something we find integrating with Open-Identity Providers today (example: a private version of the **Tenant ID** used for Office 360 with Microsoft Azure).

7. Conclusion

In this article, we:

1. Designed a new kind of a security attack that can shed light on an overseen security vulnerability with the salting mechanism used in TPS. We believe that the attack concept is new and that the highlighted vulnerability was neither addressed nor even mentioned in previous work.

2. Proposed a general approach that can be further developed into a more rigid solution for the discovered vulnerability.

3. Rang a bell to encourage additional engineering work on one mechanism that plays a key factor in implementing one of the best resource utilization optimization techniques used in Cloud Service Provider environments.

8. Glossary

CAPEX Capital Expenditures

VDI Virtual Desktop Infrastructure

DaaS Desktop as a Service

SCA Side Channel Attack

CSP Cloud Services Provider

TPS Transparent Page Sharing

CoA Copy on Write

KSM Kernel Samepage Merging

DRS Dynamic Resource Scheduler

DPM Dynamic Power Management

TCB Trusted Computing Base

9. References

[1] Understanding Memory Resource Management in VMware vSphere® 5.0

[2] https://kb.VMware.com/s/article/2080735

[3] Wait a minute! A fast, Cross-VM attack on AES, Gorka Irazoqui, Mehmet Sinan Inci, Thomas Eisenbarth, and Berk Sunar, Worcester Polytechnic Institute, Worcester, MA, USA, fgirazoki, msinci, teisenbarth, sunarg@wpi.edu

[4] Suzaki, K., Iijima, K., Yagi, T., and Artho, C. Software side channel attack on memory deduplication. SOSP POSTER (2011).

[5] Bernstein, D. J. Cache-timing attacks on AES, 2004. URL: http://cr.yp.to/papers.html#cachetiming

[6] Gullasch, D., Bangerter, E., and Krenn, S. Cache Games { Bringing Access-Based Cache Attacks on AES to Practice. IEEE Symposium on Security and Privacy 0 (2011), 490{505}

[7] https://kb.VMware.com/s/article/2097593

[8] https://www.virtuallyghetto.com/2014/07/what-happens-when-virtual-machines-have-duplicate-instanceuuids-on-esxi.html