CMU-PT/RNQ/0015/2009

# Trustworthy and Resilient Operations in a Network Environment

TRONE

## Deliverable D16

## Final specification of the enhanced components

Information and Communication Technologies Institute
Carnegie Mellon | PORTUGAL
AN INTERNATIONAL PARTNERSHIP

|  |  |
|---|---|
| Executed by: | **CISUC** |
| Direction/Department: | **DEI/FCT/UC** |
| Date: | **31-03-2014** |

| **Version Number** | **Owner** | **Change Control** | **Date** |
|---|---|---|---|
| 1.0 | FCTUC | Document creation | 31-03-2014 |

## Additional information:

|  |  |
|---|---|
| Author/s: | Filipe Araujo, Serhiy Boychenko, Raul Barbosa and António Casimiro |
| Beneficiaries contributing on the deliverable: | FCUL, FCTUC |
| WP contributing to the deliverable: | WP3 |
| Estimation of pm spent on the deliverable: | - |
| Nature of the deliverable: | Report |
| Total number of pages: | 30 |

# Contents

# List of Figures

# Executive Summary

This deliverable reports the work in Task 3.2, "Plug-in Components for Trustworthy Network Management". This task aims to investigate, enhance and prototype components providing security for the systems studied in the project. These components should work as (possibly optional) plug-ins providing improved resilience to critical but legacy parts of network OAM.

In this text we present and evaluate, both theoretically and via simulation, a number of defensive techniques for the cloud, to make it trustworthy.

Execution of critical services traditionally requires multiple distinct replicas, supported by independent networks and hardware. To tolerate faults, these services depend on the correctness of a fraction of replicas, usually over 2/3 or 1/2. In practical terms, these bounds impose a minimum number of replicas to tolerate failures. For example, to tolerate a single Byzantine failure, 3 or 4 replicas are necessary, depending on the specific solution. Hosting critical services on the cloud is, therefore, a challenge, because dispersing replicas into multiple servers or availability zones entails an economic cost to cloud providers, who wish to consolidate their services as much as possible using multi-tenant virtualization solutions. A possible approach is to fix the number of different physical machines hosting the service according to the required fault-tolerance level, and enable more replicas to run on these multi-tenant resources. This ensures a good consolidation, but raises the challenge of determining the best assignment of replicas by physical machines.

In fact, consolidation creates risks for two kinds of majorities: a conventional one, related to the number of replicas, regardless of the machines where they run; and a second one, related to the physical machines where the replicas run. To assess these risks, under crash and Byzantine failures of virtual and physical machines, we resort to theoretical and experimental evaluation. Contrary to what one might expect, we conclude that it is not always favorable to distribute replicas evenly over a fixed number of physical machines. On the contrary, we found cases where they should be as unbalanced as possible. We systematically identify the best defense for each kind of failure and majority to preserve. We then review the most common real-life attacks on clouds and discuss the *a priori* placement of service replicas that minimizes the effects of these attacks.

# 1 Introduction

The work of Castro and Liskov [19], 15 years ago, considerable increased the interest on Byzantine Fault-Tolerance (BFT). Today we can find a large body of work on this topic, including research projects [17, 28, 16][1]. For example, the Archistar project [29] considers a storage system replicated across multiple cloud providers to overcome Byzantine failures. The most well-known system requiring BFT is perhaps Bitcoin[2], although Bitcoin resorts to brute computational force for correctness. In the case of the cloud, experience with critical BFT services is not so extensive. However, the the elasticity and high availability solutions available in cloud offers, like Rackspace's or Microsoft's [26], among many others, offer excellent conditions for replicated applications. Elasticity enables applications to run with more or less replicas as needed, instead of using a fixed number, whereas fault-containment in multi-server, multi-AZ and multi-region solutions provide the means to ensure high availability.

If HA solutions require a strict separation of resources, at least to a certain number of replicas, a dispersion of replicas by different servers may prevent a rational utilization of resource past a given point. In fact, providers usually take advantage of virtualization techniques [13, 11, 8, 14, 18] to concentrate some of (or all) the replicas in the same server or physical machine (PM). Recent research has explored affinities among virtual machines to consolidate according to traffic [27, 30] or memory pages [33, 23], for example. In other words, we may need more replicas to respond to an increasing load, but we may want to fix the number of PMs hosting some service, because this number already achieves the level of replication desired, and concentrating replicas may be beneficial in economic terms. To provide a concrete example, the service might be designed to tolerate $f = 1$ Byzantine failure, thus needing $3f + 1 = 4$ replicas. When more than 4 replicas exist, these may share the same PMs to reduce network traffic or memory consumption.

What we do in this paper is to analyze the best distribution of Virtual Machines (VMs, i.e., replicas) by a fixed number of PMs. We qualitatively try to mitigate the disadvantage of not using more PMs, by giving a single defense to the cloud provider: the distribution of the VMs by the available PMs. Depending on the service, the defender may care for the number of PMs *or* the number of VMs that stand after each failure. In particular, motivated by many consensus-based algorithms, including Byzantine fault-tolerant ones, e.g. [20] or [19], we count the number of attacks that are necessary before the service loses the majority of machines, physical *or* virtual. We also assume that the attacker is limited (otherwise, the

[1] And also `http://www.secfunet.eu`, and `http://cloudfit.di.fc.ul.pt/index.php?title=Public:Main_Page`, both accessed on May 6th 2014.

[2] `http://bitcoin.org/` accessed on May 6th 2014.

defense would be helpless). Nevertheless, defining the best defensive strategy for the cloud provider is a complex task, because PMs and VMs may fail in many different ways, from crashes to arbitrary Byzantine failures caused by fragilities in hardware or software.

To determine the best placement of VMs from the provider point of view, we consider tow fundamental goals, of ensuring a majority of VMs *or* PMs and a large spectrum of attacks to the infrastructure. These might start and be confined to a single VM, or attacks may spread to other VMs, to the hypervisor, or to the PM hosting the service. Consequences range from a perceived reduction in the quality of service to erroneous responses, due to interferences or complete crashes of machines. Attackers may be able to select specific targets, or they might be limited to perform malicious actions over random targets handed by some scheduler of the cloud service. Based on actual evidence of past attacks, we try to model a large spectrum of realistic settings. Based on these settings, we resort to theoretical and experimental analysis to determine the best distribution of VMs against possible attacks. We use a balls and urns and other probabilistic approaches when possible, and experimentally evaluate the remaining cases.

While intuition suggests that we should balance the VMs among the PMs, a deeper analysis shows cases where other options are better. Depending on the kind of attack and majority to keep, the distribution of VMs might be irrelevant or should even be as unbalanced as possible, given the *same* number of PMs. For example, to ensure a majority of of PMs, the defender should concentrate the VMs as much as possible: one VM for each but the last PM, and all the remaining VMs in the last PM. This is the best way of restricting the attacker to the smallest possible number of PMs.

Hence, the main contribution of this paper is to determine the appropriate strategy to defend a majority from different types of failures, including Byzantine. We believe that knowing this is important, especially in a setting where migrating VMs can help the cloud provider to consolidate resources in favorable ways. If the provider is not bound to a balanced or extremely unbalanced solution, he may take advantage of traffic correlation to consolidate machines and reduce network traffic, for example. We take our conclusions to real-life attacks and threats hanging over cloud infrastructures. Knowledge of the most common attacks and their consequences lets us break the tie between opposing strategies to place service replicas. This conclusion goes beyond the theoretical and experimental analysis we did in [12]. Our study can be useful both for the cloud provider and for the client, as it tells them what to do or expect from the service. E.g., this may help the provider to create more precise and safer SLAs.

The rest of the paper is organized as follows. Section 2 presents the assumptions of our work. In Section 3 we perform a theoretical analysis regarding the distribution of VMs. Since there is a long and established field using the terms "balls" (VMs) and "urns" (PMs), we often keep these terms. In Section 4 we extend the analysis of the previous section and run Monte Carlo simulations, whenever we are not aware of closed formulas that may help us to find the appropriate distributions. In Section 5 we discuss the best distribution strategies. In Section 6 we go through well-known attacks on clouds and evaluate the effect of this knowledge on the theoretical findings of the previous sections. In Section 7 we conclude the paper.

# 2 Model

## 2.1 Failure Models

In our analysis we deal with crash failures and Byzantine failures caused by attackers. Crash failures may stop a VM or the entire PM. Byzantine failures may produce these and other arbitrary deviations from the correct service. In this paper we try to consider a representative number of entry points and behaviors for attackers. Attackers may target a specific PM, if they can, or they might depend on some fragility or assignment from the cloud provider, thus being unable to pick the concrete target. We assume the same for VM attacks (limited or unlimited targets). A faulty PM may compromise all its VMs. Conversely, once the attacker controls a VM, he might be able to escalate the attack to control the hypervisor, disturb somehow the PM or co-located VMs ("interference"), or, in the most benign case, he might be limited to shut down the VM.

We consider a service with $m$ PMs and $v$ VMs. Each of the $m$ PMs may run one or more of the $v$ VMs. When appropriate, we consider a balls and urns problem: the urn is a PM; the VMs are the balls. VM failures correspond to drawing a ball from an urn. In some cases, the attacker will put the ball again in the same urn, sometimes he or she will simply remove the ball. For example, consider a cluster of application servers responding to HTTP requests, or a remote storage service [6, 9, 4]. Subsequent requests to the service may end up in the same application server, making this a case without removal. Contrary to this, if the attacker may not find the same replica, because he disrupted the previous one, we have removal. This model does not cover all the cases, for instance, if the attacker manages to shut down the entire PM (this would remove all balls from the urn).

## 2.2 Goal

Depending on the service, we may want to ensure a majority of correct VM *or* PM replicas. Some consensus algorithms, as the one in [20], use special hardware devices, such as a Trusted

Platform Module (TPM), to improve the tolerance to Byzantine nodes. To ensure multi-tenancy, virtualizing the TPM itself to share it by multiple VMs, as in vTPM [15], emerges as a natural step. Therefore, it becomes important to ensure a majority of correct PMs. When we consider crash failures, we assume that the PM is in the group until the last VM crashes.

We assume that, as soon as one VM in the service is under control of the attacker, all co-located VMs and the respective PM might be compromised and out of the majority of correct (physical or virtual) machines. What we do not consider is the case where the attacker manages to create more machines to participate and corrupt the BFT protocol. We consider this kind of attack to be outside the scope of this paper, as this problem is orthogonal to the cloud and exists in BFT protocols in general.

The set of attacks we consider falls into a moderate number of cases. We may have crash and other arbitrary effects, which we refer to as interference and Byzantine, although crashes may have intentional causes. The attacker may or may not be able to pick his or her targets; VMs or PMs may fail; and we may want to preserve VM or PM majority. Overall we have a total of 14 meaningful cases that we discuss throughout the paper and wrap up in Section 5.

# 3 Theoretical Analysis

## 3.1 Independent VM failures

We start our theoretical analysis by considering that faults are independent: a fault causing the failure of a VM does not affect any other co-located VM. This may be the case of a crash failure. We also assume that to participate in some replicated protocol, the PM needs to have at least one operational VM. The PM becomes unusable as soon as the last VM fails. This could happen if at least one VM is necessary to operate some shared resource, like a TPM, e.g., to sign the messages coming from that specific machine. I.e., given the failure of $n \leq v$ VMs, we can calculate the probability that a PM with $v_i$ VMs fails. We compute this value in Equation 1, for $1 \leq v_i \leq n \leq v$, using the Hypergeometric distribution. $P_i(v_i, n, v)$ is the probability that PM $i$ fails due to the failure of all its $v_i$ VMs. $P_i(v_i, n, v) = 0$, for $n < v_i \leq v$. That is, at least one VM of PM $i$ will survive the $n$ failures.

$$\begin{aligned} P_i(v_i, n, v) &= \frac{\binom{v_i}{v_i}\binom{v-v_i}{n-v_i}}{\binom{v}{n}} \\ &= \frac{n \times (n-1) \times \ldots (n - v_i + 1)}{v \times (v-1) \times \ldots (v - v_i + 1)} \end{aligned} \tag{1}$$

Since we assume a fixed number of $n$ crash failures, and a total of $v$ virtual machines, we analyze the impact of varying $v_i$ and use the notation $P_i(v_i)$. We first show in Equation 2 that for $v_i \in \{1, \ldots, n-2\}$, $P_i(v_i) - P_i(v_i+1) > P_i(v_i+1) - P_i(v_i+2)$. That is, the marginal gain (in terms of reducing the probability of failure of a PM) achieved by an increment of the number of VMs allocated to one PM decreases as the number of VMs in that PM approaches the number of possible failures, n.

$$\begin{aligned}
&P_i(v_i) - P_i(v_i+1) - P_i(v_i+1) - P_i(v_i+2) = \\
&P_i(v_i)\left(1 - \frac{n-v_i}{v-v_i} - \frac{n-v_i}{v-v_i} - \frac{(n-v_i)(n-v_i-1)}{(v-v_i)(v-v_i-1)}\right) = \\
&P_i(v_i)\cdot \\
&\quad\cdot\left(1 - \frac{2(n-v_i)(v-v_i-1)-(n-v_i)(n-v_i-1)}{(v-v_i)(v-v_i-1)}\right) \\
&\geq P_i(v_i)\left(1 - \frac{(n-v_i)(n-v_i-1)}{(v-v_i)(v-v_i-1)}\right)
\end{aligned} \tag{2}$$

Since $v \geq n$, the denominator of the subtrahend is greater or equal than the numerator, which makes the overall expression greater or equal than 0. In Equation 3, we define a new function $Q_i(x)$ that “extends” $P_i$ for the domain $[1, n]$. $P_i$ and $Q_i$ are equal in the domain $\{1, \ldots, n\}$, but $Q_i$ has line segments connecting consecutive points of $P_i$ in $[1, n] \setminus \{1, \ldots, n\}$.

$$Q_i(x) = \begin{cases} P_i(1) + (x-1)(P_i(2) - P_i(1)), 1 \leq x < 2 \\ \ldots \\ P_i(j) + (x-j)(P_i(j+1) - P_i(j)), \\ \qquad\qquad 1 < j \leq x < j+1 < n \\ \ldots \\ P_i(n-1) + (x-n+1)(P_i(n) - P_i(n-1)), \\ \qquad\qquad n-1 \leq x \leq n \end{cases} \tag{3}$$

From Equation 2, it follows that $Q_i(x)$ is convex in the domain. Now, the number of failed machines can be computed as an expectation $E = \sum_i P_i(v_i) = \sum_i Q_i(v_i)$. From Jensen's inequality:

$$Q\left(\frac{\sum_i v_i}{m}\right) \leq \frac{\sum_i Q_i(v_i)}{m} \tag{4}$$

where equality occurs when all the $v_i$'s are equal. Since we want to minimize the right side of the equation we should evenly balance the virtual machines by the physical machines (assuming that the division is integer).

## 3.2 VM failure contaminates other VMs

We now consider Byzantine attacks, where a malicious user may take over all the co-located VMs, once he or she successfully attacks the first one (malware injection, side-channel or protected environment escape attacks). In mathematical terms, we can treat this problem as an urns and balls problem, and use known results, such as [25]. Urns represent a PM, while balls represent the VMs. To distinguish between correct and compromised VMs we may assign colors to balls: white balls represent correct machines, whereas black balls represent compromised machines. The objective of our analysis in this section is to show that there is a proper way of initially distributing white balls, which minimizes the number of urns that end up having black balls as a result of malicious actions successively changing the color of balls from white to black. Our assumption in this particular case is that the previously white ball returns to the same urn as black and the attacker has no option concerning the urn from where it picks a ball. This represents a case where the service is apparently running and the attacker might find the same (possibly compromised) VM.

The option left to the cloud provider is to select a distribution of VMs by the PMs and, in this particular case, we care about the number of urns that do not have black balls, which we denote by the random variable $X$. $P(X \geq k)$ is the probability that $k$ or more urns have no black balls.

**Theorem 3.1** *Assume that we have $v$ white balls distributed by $1 < m < v$ urns, such that each urn has at least one ball. Assume that the attacker works in successive turns, picking one ball at a time from an urn, and always putting back a black ball in the same urn. The attacker does not select the ball or the urn. $\forall\ k \in \{1, \ldots, m\}$, $P(X \geq k)$ is maximized when $m-1$ urns have $1$ ball, and the remaining urn has the remaining $v-m+1$ balls.*

**Proof** Refer to Figure 1. Note that white urns have only 1 ball, gray urns have at least 1, but less than $v-m+1$ balls, and black urns have $v-m+1$ balls. Consider setting $A$, where only 1 urn has more than 1 ball. I.e., $m-1$ urns have 1 ball, whereas the last urn in the figure has $v-m+1$ balls. Setting $B$ represents any other case, with the same number of urns, $m$, but with a different distribution of balls. Let us match pair-wise the urns in setting $A$ with the urns in setting $B$ (original), starting by the 1-ball urns (on the left side of the figure). After this first set of urns, we define another set $O$, with the urns that have more than 1 ball in $B$, but only one ball in $A$. This definition excludes the first urns that have 1 ball in $A$ and $B$, as well as the last urn of $A$, which has more than 1 ball. Note that $1 \leq |O| \leq m-1$.

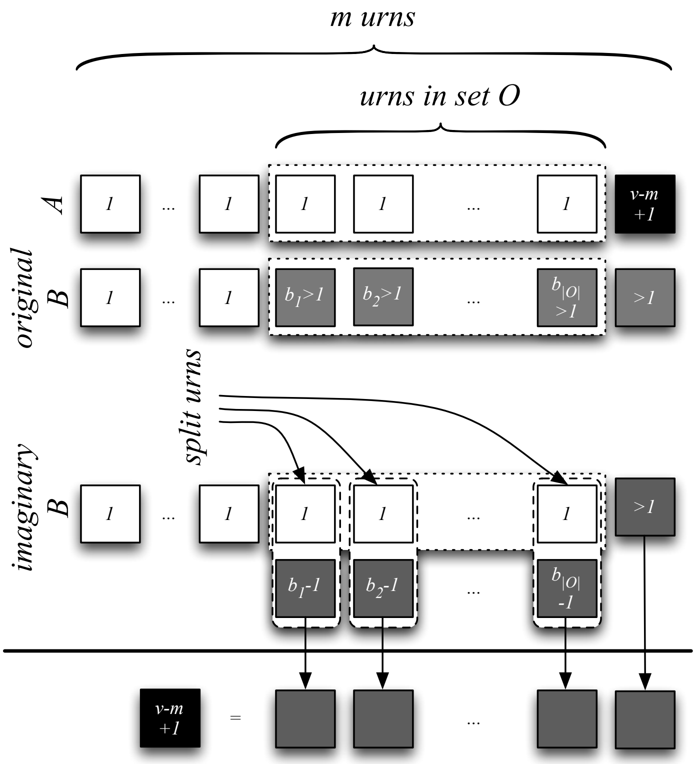

Figure 1: Distributions of balls by urns in settings $A$ and $B$

We now resort to an artificial division of set $O$ in $B$. We split each one of the $|O|$ urns in $B$ into two other urns: one with 1 ball, and the other with the remaining balls. This makes for a total of $m-1$ urns with only 1 ball, just like in setting $A$. The remaining $v-m+1$ balls are spread over $|O|+1$ urns (in gray in the “imaginary $B$”), each having one or more balls. In the rest of our reasoning, we should refer to settings $A$ and “imaginary $B$”, with $m$ and $m+|O|$ urns, respectively. In both cases there are $m-1$ urns with only 1 ball. We first observe that these urns have exactly the same probability of having a black ball. What about the single black and $|O|+1$ gray urns in $A$ and “imaginary $B$”, respectively? Since the number of balls is the same, $v-m+1$, the probability of having one or more black balls is exactly the same in both settings. However, these black balls only “contaminate” (or exist) in a single urn in setting $A$, whereas in “imaginary $B$” they may spread over multiple urns. As a consequence, $P(X \geq k)$ in setting $A$ must be the same or greater than in setting $B$.

To calculate $P(X \geq k)$, we can use a result from [25, Equation (3.5)], which we restate here:

$$
\begin{aligned}
P(X \geq k) = &\sum_{\mathbf{a}}^{(k)} \left(1 - \sum_{j=1}^{k} p_{a_j}\right)^n - \\
&- \binom{k}{k-1} \sum_{\mathbf{a}}^{(k+1)} \left(1 - \sum_{j=1}^{k+1} p_{a_j}\right)^n + \\
&+ \binom{k+1}{k-1} \sum_{\mathbf{a}}^{(k+2)} \left(1 - \sum_{j=1}^{k+2} p_{a_j}\right)^n - \\
&\ldots \\
&+ (-1)^{m-k} \binom{m-1}{k-1} \sum_{j=1}^{m} {p_j}^n
\end{aligned} \tag{5}
$$

The $m$ urns define a set of integers $\{1, 2, \ldots, m\}$. The variable $p_j$ is the probability that we assign a black ball to urn $j$. From this set, we define subsets with $k$ elements $\{a_1, a_2, \ldots, a_k\}$. $\sum_{\mathbf{a}}^{(k)}$ denotes a summation over all these subsets. Thus, there are $\binom{m}{k}$ terms in this sum. For a better understanding of this formula, we should realize that summation $\sum_{\mathbf{a}}^{(k)}$ concerns the probability of having $k$ urns without black balls when $n$ black balls have been dropped in the urns. The summation that follows in the formula considers the probability of having $k+1$ empty for the same $n$ balls and so on.

## 3.3 Keeping a Majority of Virtual Machines

The next question we consider is the impact of machine failures on the number of VMs that stay alive in a correct state. In many cases, this number might be more important than the number of different PMs where the replicas run. One may ask whether concentrating many VMs in the same (or few) machine(s) could reduce the average number of VMs that survive (other) PM crashes. Interestingly, the answer is *no*. Assume $Z(t)$ to be a random variable that represents the number of VMs that are running at time $t$. Variable $v_i$ is the number of VMs running on PM $i$; $Y_i(t)$ is a random variable that assumes the value 0 if PM $i$ is off at time $t$ or 1 if it is on. If we assume that all PMs have the same characteristics, at any given time $t$, $Y_i(t)$ is the same for all values of $i$, and we can simply remove the subscript and write $P[Y(t) = 1]$. Equation 6 shows that the distribution of VMs by the PMs is not relevant from the point of view of the average:

$$E[Z(t)] = \sum_{i=1}^{m} v_i \cdot P[Y_i(t) = 1] = P[Y(t) = 1]v \tag{6}$$

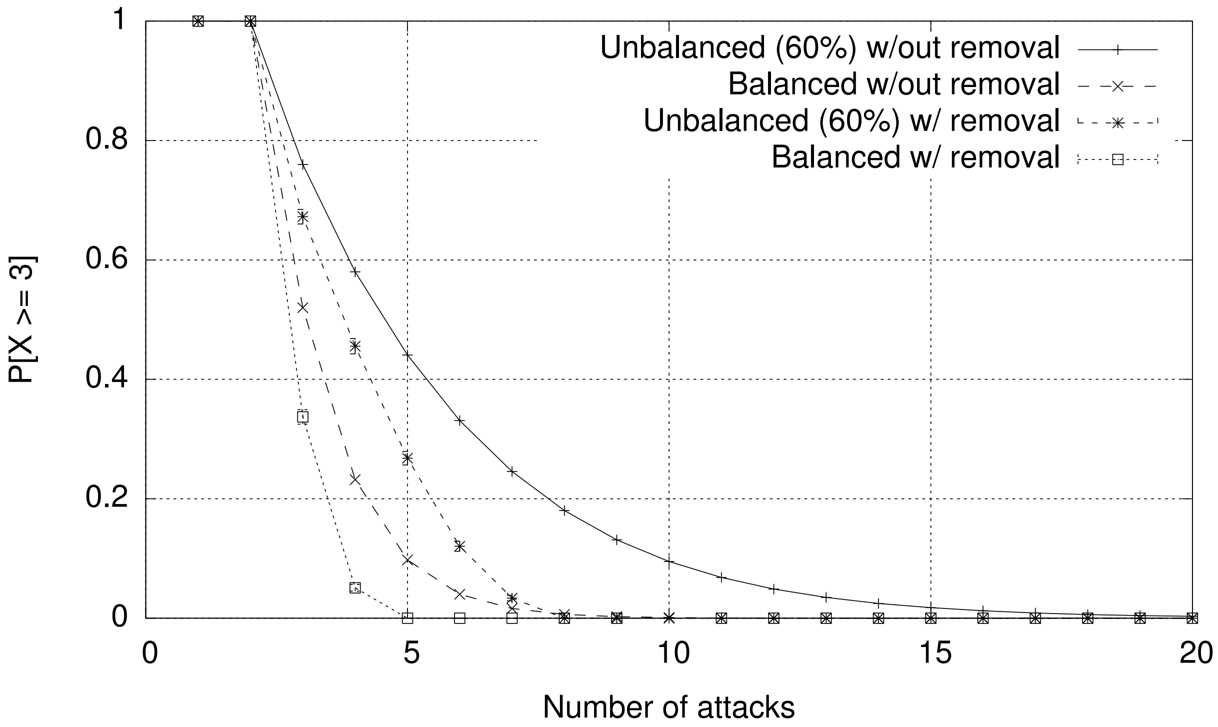

Figure 2: Probability of conserving majority for 5 physical machines

However, one should notice that other metrics may be relevant as well. On the few occasions when the most loaded PM fails, much fewer replicas will be available. I.e., the unbalanced setting will most of the time keep a few more VMs running, but sometimes, it will have much less.

# 4 Experimental Evaluation

Since we cannot evaluate all the interesting scenarios using analytical expressions, in this section we partially resort to simulation. To evaluate the likelihood of keeping a majority of correct PMs, we start with $m = 5$ PMs and $v = 10$ VMs in Figure 2. We considered four cases: in one of them, all the machines (urns) have the same probability of receiving an attack (black ball) and this probability remains constant. We call this case "balanced without removal". We also consider the case where a single machine has a probability of 60% of receiving an attack, while the remaining 40% probability is equally distributed among the remaining machines. This corresponds to running 6 VMs in a single $PM$, while the remaining 4 VMs run in the other 4 PMs, in a one-to-one correspondence. To this case we call "unbalanced (60%) without removal". We also consider removals of the VMs once the attacker dominates them. This corresponds to a scenario where the attacker is successively

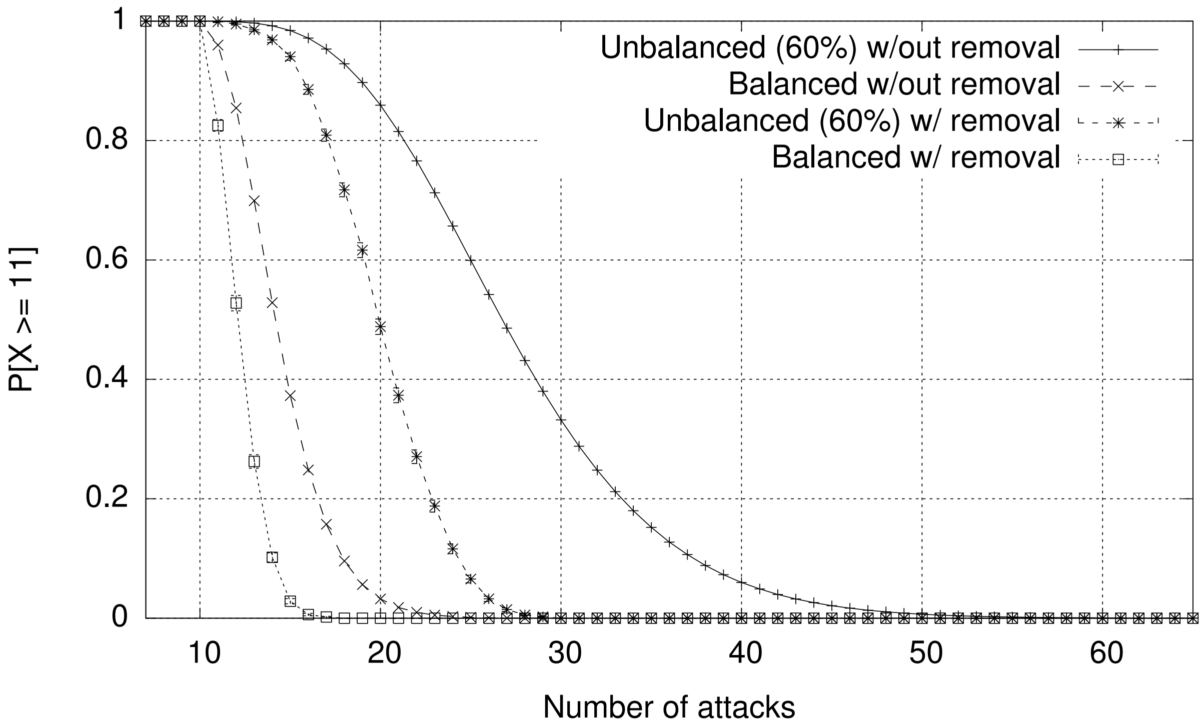

Figure 3: Probability of conserving majority for 21 physical machines (PM or all VM contamination)

requesting some instance of a service from a limited set. Each time it gets a new instance, the attacker will not get the same, but a new one (e.g., a new VM). These are the two cases with removal.

In the plots without removal, we use Equation 5 to compute the probability that at most 2 out of 5 PMs have black balls, as the number of attacks grows. We can see a clear difference between the balanced and unbalanced cases. The chances of keeping a majority significantly improve for the latter case. For instance, after 5 attacks, there is still more than 40% chances of conserving the majority in the unbalanced case, while in the balanced case, this probability is below 10%. To plot the lines with removal we resorted to Monte Carlo simulation. We show error bars that correspond to a 99% confidence interval of the average, assuming a normal distribution. Since we used 10,000 trials to get these plots, the intervals are very small and the error bars are barely visible. As we expected, removal makes it easier for the attacker to reach a larger number of different PMs, thus negatively affecting the probability of keeping a majority of correct PMs. The four plots of the figure actually depict two extreme pairs of cases: the one without removal approximates a scenario where we have a very large number of VMs, or where the same VM can be handed to the attacker. The pair of lines with removal corresponds to the other extreme case, where we have a small

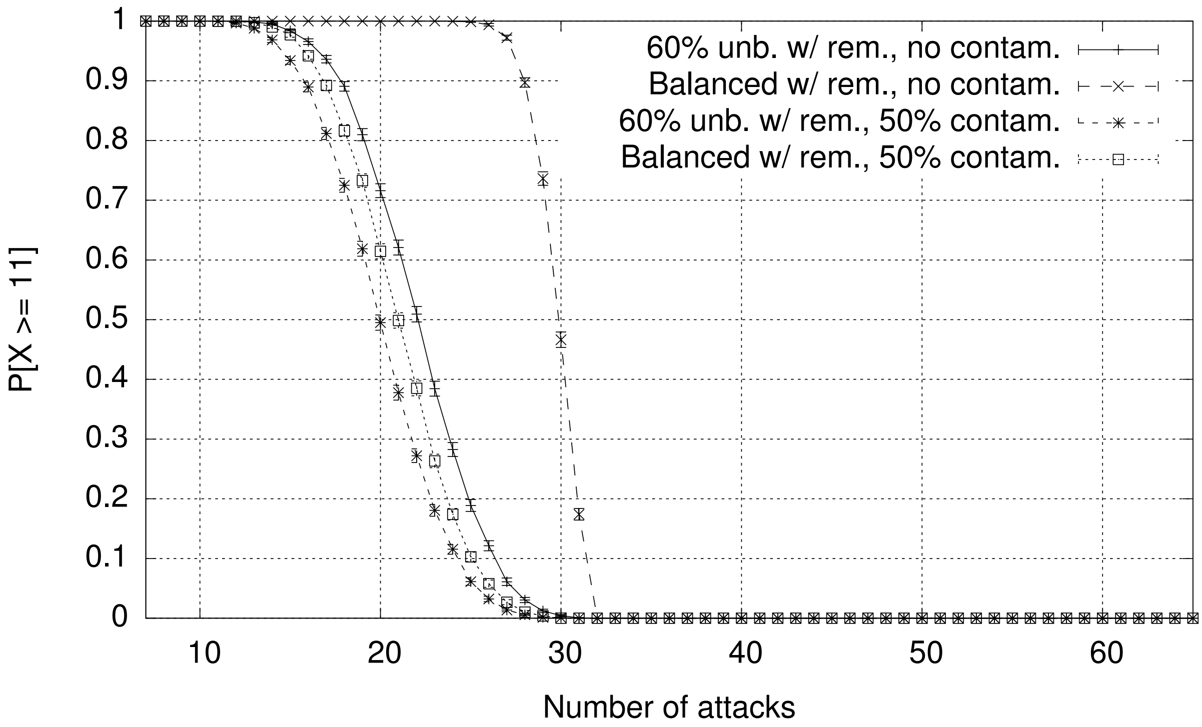

Figure 4: Probability of conserving majority for 21 physical machines (no contamination and 50% contamination)

number of VMs (10) for the available PMs (5).

Next, we try a larger number of PMs, $m = 21$, to compare against the smaller set of 5. For this larger set, the advantage of unbalancing the distribution of VMs is even more visible, as we depict in Figure 3. For the case without removal, given by Equation 5, after a little more than 20 attacks, there is nearly no chance that the majority of the PMs is still correct for the balanced option. This takes more than 60 attacks in the case of the unbalanced scenario. The difference in the removal case is also important. Note that in this setting we assumed that the entire PM is compromised once the attacker penetrates the first VM, either because he or she managed to control the hypervisor or because he or she gained control of all the VMs in the same physical machine. Unlike this, in Figure 4 we consider other cases: a 50% and a 0% chance (no contamination) of controlling the service via hypervisor or other machines. We can see that as the chance of dominating the entire service in the PM decreases, balancing becomes increasingly better. This is not surprising, if we think that this case is more similar to crash cases, where the attacker cannot engage more co-located resources in arbitrary behaviors. Hence, by unbalancing the VMs, it becomes more likely that PMs with only one VM are left empty when the single VM fails.

These plots raise the question of determining the number of attacks that it takes until

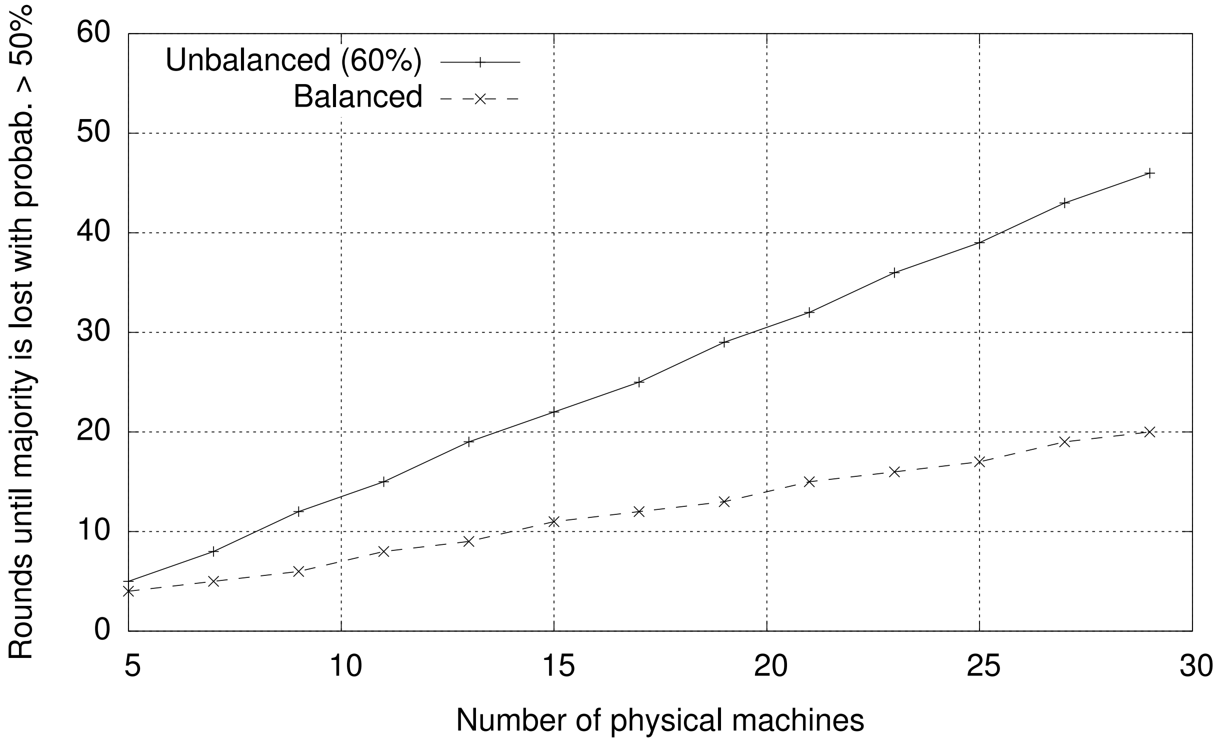

Figure 5: Evaluation of the rounds the attacker needs until it probabilistically gets the majority as a function of the number of physical machines

the attacker succeeds in (probabilistically) holding the majority of the machines. We do this evaluation in Figure 5, for the balanced and unbalanced (60%) cases, without removal, for a varying number of PMs. In both cases, the growth seems to be approximately linear, but the slope is much higher in the unbalanced case, thus making it much more difficult to break.

Finally, in Figure 6 we evaluate the unbalance factor. We use 11 machines and make the balance change from 1/11 (balanced) to $1 - 1/11 \approxeq 91\%$ (most unbalanced) in steps of 1/11. For all these unbalanced factors, we plot the number of attacks until the attacker gets the majority of machines with probability greater than 50%. There is no removal.

# 5 Placement Strategies of the Defense

We now identify the best strategies to distribute the VMs by the PMs. Should the defender use a balanced approach or an unbalanced one? We assume that failures render the target unusable either because it crashed or because it ceased to follow the protocol. Then, we consider multiple contamination models: an attack to one VM may or may not become an attack to co-located VMs and PM. We also consider that attacks might be "limited" or "unlimited", to VMs, or directly to the PMs. "Limited" means that the attacker selects

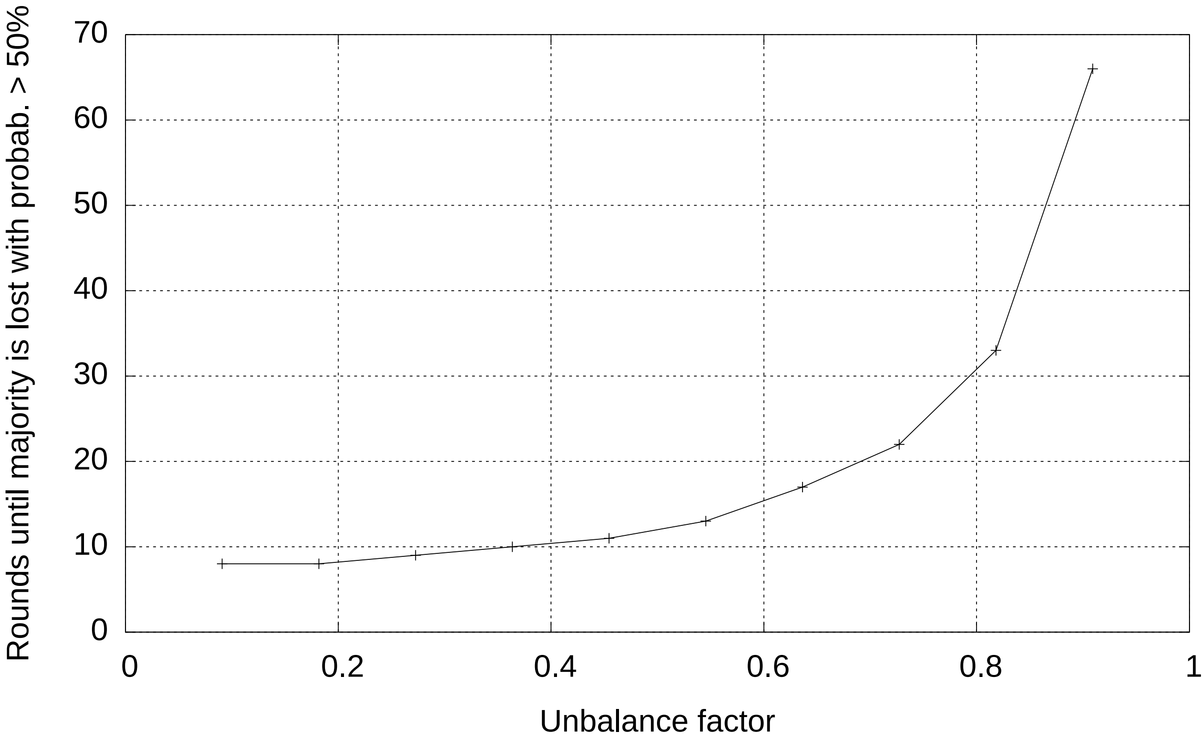

Figure 6: Evaluation of the rounds the attacker needs until it probabilistically gets the majority as a function of the unbalance factor

a target at random with uniform probability. “Unlimited” is the extreme opposite case, where the attacker may select the target he wants. In the next sections we consider different scenarios of attacks and contamination models. Table 1 shows the best distribution of VMs by PMs for each attack and contamination model.

## 5.1 Attacks to PMs

In this section we consider attacks to PMs regardless of their source. They could well begin in a VM and escalate to affect the entire PM. If we assume that the targeted PM disappears from the service or acts incorrectly, the distribution of VMs by the PMs is not relevant for a PM majority. On the other hand, if we consider VM majorities, the distribution of VMs might be relevant. If the attacker can pick his targets (“unlimited” case), it is better to evenly distribute the machines, otherwise the attacker will select PMs with more VMs. If not (“limited” case), the distribution of VMs is irrelevant, as we saw in Section 3.3.

Table 1: Distribution of VMs by PMs as a function of the failure and the objective to achieve by the defender

| Failure/Objective | PM Majority | VM Majority |
|---|---|---|
| Attacks to PMs | | |
| Limited PMs | Irrelevant | Irrelevant |
| Unlimited PMs | Irrelevant | Balanced |
| Attacks contained within VMs | | |
| Lim. Crash VMs | Balanced | Irrelevant |
| Unlim. Crash VMs | Balanced | Irrelevant |
| Attacks to a VM interfere with the PM service | | |
| Lim. Interf. VMs | Unbalanced | — |
| Unlim. Interf. VMs | Irrelevant | — |
| Cross-VM and VM-to-Hypervisor escalation | | |
| Lim. Byzantine VMs | Unbalanced | Balanced |
| Unlim. Byzantine VMs | Irrelevant | Balanced |

## 5.2 Attacks Contained within VMs

Now, consider VM targets, but assume that the attacker cannot leave the borders of the VM, i.e., the VM might misbehave somehow, but it cannot tamper with other VMs, the PM or any other device related to the service, like a TPM. Crashes are a typical cause for this behavior. In this case, the distribution is irrelevant to keep a majority of VMs, as only one VM is faulty after each successful attack. Keeping a PM majority might be different. If the PM is out of the service when the last VM it holds stops, balancing is better to avoid PMs without VMs.

## 5.3 Attacks Affect the Replicated Service, but not Other VMs or the PM

Now, we assume that attacks to VMs do not cross the frontier of the VM, except to disrupt the service provided by that PM. This corresponds to a case where the corrupted VM manages to control the service, at least sometimes, e.g., by responding first to requests entering the machine. However, this model assumes that the attacker cannot invade other VMs or the PM from the VM (e.g., to shut down other VMs). From the perspective of keeping a majority of VMs, we omit this case from our analysis, because it is similar to the contained VM attacks. For a PM majority, in the “Limited” case, the best thing to do is to unbalance the VMs, as we saw in Sections 3.2 and 4. In the “Unlimited” case, where an attacker can disturb the service in any PM he/she wants, the distribution is irrelevant. It is interesting to note the striking difference to the case where attacks do not cross VM borders, where we balance the VMs, to ensure that, say, crashes do not leave a PM without VMs. We reviewed a similarly interesting case, where balancing may be a better option in Section 4, Figure 4: as the probability of affecting the service decreases to zero, balancing becomes better.

### 5.4 Cross-VM or VM-to-Hypervisor Attacks

We now consider attacks that can cross VM boundaries to take over other VMs (cross-VM) or the PM (VM-to-Hypervisor attack). This case ends up being the same as the "Unlimited PMs", and thus gets the same distribution. However, it includes some subtleties worth discussing, as it may happen that an attacker is unable to reach arbitrary VMs, but he might end up shutting down the PM or many co-located VMs to reach other VMs and PMs in future interactions. In other words, an attacker with limited access to VMs might end up affecting other co-located VMs, to reach a different set of victims. Although this would not be purely "Unlimited Byzantine", the same distributions (Irrelevant/Balanced) would hold. However, if the attacker cannot improve his chances of reaching arbitrary VMs and is constrained by the provider ("Limited Byzantine VMs"), we have the same case as the "Limited Interference" for the PM majority. For a VM majority we need a Balanced distribution of VMs, or, otherwise, the attacker would have a higher probability of reaching VMs with more co-located VMs.

## 6 Classification of Real Attacks

In this section we identify and describe attacks found in the literature that may affect virtualized cloud infrastructures. We categorize real attacks and the resulting failures that we found in the literature, namely described by Jensen *et al.* [24], Zhang *et al.* [34], Gruschka *et al.* [22], and Verizon [32], onto the classes of Table 1. Our analysis intends to support selection and prioritization of the placement strategies, according to the main security concerns and impact of attacks.

### 6.1 Denial of Service

One of the most attractive features of cloud systems is their on-demand provision of computing power for legitimate users. However, this feature may raise serious security problems if it is exploited with the wrong intentions. Attackers may flood a service with requests, inducing the infrastructure, to create countless VMs in response to those requests, thereby exhausting the available hardware resources. This leads to a Denial of Service (DoS), not only on the specific service under attack, but also on other services that share the infrastructure.

There are numerous examples of such attacks on Cloud services and providers. The most famous cases are not necessarily connected to models we propose in this paper, but can affect operation of an entire provider. The Spamhaus anti-spam organization and the cloud provider CloudFlare, recently suffered a Distributed Denial of Service (DDoS) attack [1]. By exploiting core Internet infrastructures and open DNS resources, attackers managed to create an attack of great dimension. Although the attack was mitigated and the adversaries were unable to achieve the goal of shutting down the `spamhaus.org` site, they managed to cause delays and outages in some other Internet resources. The work of Domingues *et al.* [21] shows that VMs can actually affect each other, especially when they perform I/O operations, thus opening the door to DoS problems, although cloud providers are evidently reluctant to disclose these cases.

## 6.2 Malware Injection

The malware injection attack consists of installing some malicious service implementation or virtual machine into the cloud. The adversary deceives the cloud system to accept malicious software as a valid instance of the attacked service. Successful invasion allows the infected instance to execute user requests. The effects of malware injection may vary from eavesdropping to completely disabling or functionality changing the attacked service. Companies like Intel are actively developing solutions against this kind of threat. This is the case of the "Trusted Execution Technology" (TXT).

One of the most notorious injections of malicious software occurred against the Google password system [10]. Vulnerabilities of Internet Explorer 6, which was still used by some workers, allowed hackers to inject malicious software that enabled them to access the company's internal network. Although Google claimed that no passwords were stolen, it is difficult to evaluate all the consequences of the attack.

## 6.3 Metadata Spoofing

In metadata spoofing, attackers modify the Web Services' metadata descriptions. Attackers may modify the syntax and operation of some functions described in the WSDL. Unaware of the fraudulent modification, an unknowing user will invoke functions that execute unexpected operations on the server. A successful attack could allow the adversary to manipulate service operations or even to obtain user credentials, to perform a broader range of attacks.

Although we were not able to find reported cloud-related incidents involving metadata spoofing, we found descriptions of such attacks on web-service based resources. For example, hackers were able to configure administrative settings (without having privileged user access) in an attack to D-Link routers [5], using vulnerabilities of the SOAP-based device control protocol. In some models of the D-Link routers, the attackers were able to execute the SOAP action `GetDeviceSettings` without authentication. This action alone did not allow the adversary to obtain any sensitive data, but it could be used to bypass the authentication requirements for other SOAP actions.

## 6.4 XML Signature Wrapping

The XML signature wrapping attack modifies authenticated SOAP messages. Several different variants of this attack exist. In one of the variants, after obtaining the original signed SOAP packet, the attacker moves the message body to a wrapping element inside the SOAP header and creates a new body with a different operation. This may deceive some servers to accept the signature, because the original signed contents still exist inside that wrapping element. However, the new function in the SOAP body of the message may harm the server. As a result, the attacker could execute arbitrary operations on the cloud system as a legitimate user, influencing the availability of the services.

Some studies revealed potential vulnerabilities of cloud providers to XML Signature Wrapping attacks. A group of researchers from the Ruhr-University Bochum demonstrated this fragility in the Amazon Web Services (AWS) [2]. Using this approach, the researchers managed to delete and create new images on the customer's EC2 instance and hijack AWS sessions to get sensitive data, including plaintext passwords. Despite being restricted to AWS, researchers believe that other cloud providers might be affected by the same type of vulnerabilities.

## 6.5 VM-to-Hypervisor Attack

The VM-to-hypervisor attack is based on exploiting security vulnerabilities of the hypervisor. The complexity of the virtualization software may leave some open possibilities for adversaries to escape from the protected environment and gain full access to the hosting physical machine [31]. Consequences of the VM-to-Hypervisor attacks may vary from creation of rogue VMs to complete interruption of the attacked hardware, both of which may compromise multiple services in many different ways.

The media continuously reports privilege escalation vulnerabilities of different virtualization technologies. In a recent case [7], a major vulnerability in the 64-bit Xen hypervisor running 64-bit para-virtualized guests on Intel CPUs was disclosed and patched before any hacker managed to exploit it [7]. Successful attack would allow the adversary to escape from guest status and gain administrative access on the host machine. After breaking the protected environment, the adversary could run arbitrary code in privileged mode, install and run new programs, and create new accounts with administrative rights.

## 6.6 Cross-VM Side-Channel Attack

Cross-VM side-channel attack consists of taking advantage of information leaks from the system's shared components, such as cache or memory. Timing information in the access to some memory addresses may let the attacker know whether or not data from the address was stored in cache. Several techniques of cache pattern classification, noise reduction and error correction are applied to minimize the search space for cryptographic information. The effect of such attacks varies from eavesdropping to service interruption, depending on the goals of the attacker.

We are not aware of any cloud providers that have been victims of cross-VM side-channel attacks. Nonetheless, a research team managed to collect and reconstruct cryptographic keys from information leaked by the CPU cache. This allowed them to take control over a victim VM [3].

## 6.7 Hacking

Hacking consists of obtaining illegitimate access to a system by circumventing security mechanisms, either by exploiting security vulnerabilities or by obtaining access credentials. Some of the most common forms of attack include stealing login credentials, brute force attacks, SQL injection and backdoor exploitation. The main goals of hacking generally are data stealing or damaging.

These types of attack have become increasingly common in recent years, as reported by Verizon [32], regarding network infrastructures and data centers. One may expect hacking to be one of the main forms of attack on cloud infrastructures as well, given that the applications running in the cloud make use of common technologies and security mechanisms, and are therefore vulnerable to the same strategies. According to the Verizon study, hacking along with malware injection are the most widespread forms of attack.

## 6.8 Defending Against Real Attacks

In Table 2 we go through the list of attacks and identify the potential consequences. Since a given attack type can have a range of different effects on a victim, our classification

Table 2: Classification of the attacks according to the worst consequences documented

| Class of Attack | Real Case |
|---|---|
| Limited PMs | Denial of Service<br>VM-to-Hypervisor Attack |
| Limited Crash VMs | VM Software Faults |
| Lim. Byzantine VMs | Cross-VM Side-Channel<br>XML Signature Wrapping<br>Metadata Spoofing |
| Unlim. PMs/Byzant. VMs | Malware Injection<br>Hacking |

Table 3: Placement strategies according to attack type and majority goal

| PM Maj. | VM Maj. | Real Case |
|---|---|---|
| Irrelevant | Irrelevant | Denial of Service<br>VM-to-Hypervisor Attack |
| Balanced | Irrelevant | VM Software Faults |
| Unbalanced | Balanced | Cross-VM Side-Channel<br>XML Signature Wrapping<br>Metadata Spoofing |
| Irrelevant | Balanced | Malware Injection<br>Hacking |

considers the worst documented consequences of each type of attack. In some cases, the worst documented consequences of an attack are not necessarily the most frequent consequences. As we shall see, this does not have an impact on the choice of placement strategy regarding PM majorities nor VM majorities.

Based on information about each attack's outcome we assign it to one of the classes listed in Table 1. For example, if the adversary may take full control of one particular PM, he could make it stop, which would be a crash failure. However, since the attacker may also disrupt the service in some other more complex way, we consider this failure as Byzantine instead. After classifying attacks we determine the appropriate defense strategy, taking into account different factors explained below.

Let us start with the situations in which resource distribution is irrelevant. Two kinds of attack share this result: Denial of Service and VM-to-Hypervisor Attacks. The main reason why the distribution is irrelevant in case of a DoS attack has to do with how resources are used with virtualization and cloud computing. Since the physical resources are shared, an attacker, even by targeting one specific VM, may prevent the other co-located VMs from operating properly. In an even more difficult scenario, on-demand resource provisioning may lead to VM instance flooding, congesting the infrastructure of the cloud provider.

While the strategy for DoS attacks is straightforward, the placement replicas to defend against VM-to-Hypervisor attacks requires some additional discussion. Although the adversary starts malicious actions from one VM (randomly assigned by the cloud provider), the attack to the target service occurs only when the hypervisor gives in. In fact, intrusion into the target service occurs from a successfully hacked PM (hypervisor). This intrusion method and its worst possible effects lead us to associate VM-to-Hypervisor attacks with the Limited Byzantine PM category, and in this case the placement of VMs for majority preservation is irrelevant. Hence, the first line of Table 3 shows the distribution of replicas as being irrelevant regardless of the kind of majority that is to be assured.

Despite not being exactly attacks, we include software faults that cause VM crashes, because accidental crashes are so common. In this case, if we assume that any VM might be affected with uniform probability, the distribution of VMs is only relevant to ensure a PM majority. We do not show PM crashes in the tables, as they would fall in the "Limited PMs" attacks, which we covered already.

We proceed with the discussion of Limited Byzantine VM attacks, where the placement of replicas has great impact on defender's success. There are several characteristics of this attack class corresponding to failures described in Section 6. The analysis of adversary limitations, targets and intrusion consequences lead us to classify XML Signature Wrapping, Metadata Spoofing and Cross-VM Side-Channel as Limited Byzantine VM attacks. Usually the target of these attacks are the VMs hosting the victim's service. The adversary, despite having some information about the victim, does not seem capable of picking the attack starting point and target at will. Nonetheless, a broken service instance may enable the hacker to compromise other service replicas running on the shared resource. The placement strategy in this case greatly depends on the desired majority. When the goal is keeping the majority of VMs, a balanced placement strategy should be adopted, otherwise unbalancing is better.

The classification of Malware Injection and Hacking as Unlimited PMs or Unlimited Byzantine VMs, which ends up being the same, is based on the worst possible consequences achieved by an adversary (partial or total access to the infrastructure). Possible scenarios include exploitation of the same vulnerability in many nodes to start a full scale attack. In the real case we described in Section 6.2, hackers seemed to have accessed the entire intranet of the Cloud provider. Analysis performed in Section 5 suggests a balanced placement strategy for Unlimited Byzantine PM attacks when VM majority is the goal, whereas the distribution is irrelevant otherwise.

By analyzing Malware Injection and Hacking attacks, along with information provided by the Verizon report [32], the most harmful consequences of these attacks are likely to be uncommon. Although such an attack may provide unlimited access to a Cloud infrastructure (e.g., leaked or stolen administration passwords) one may consider that an attacker will in general be more successful in obtaining illegitimate access into a single application and its VMs rather than the whole infrastructure. This would, in fact, suggest that most malware and hacking attacks should be classified as Limited Byzantine VMs.

A careful analysis of the relation between attack types and placement strategies on lines two and three of Table 3 helped us to solve this ambiguous situation. We noted that the replica distribution methods are not contradictory for any of the majorities. When the defender wants to keep a VM majority, both lines suggest a balanced strategy. When the defender's goal is to keep a PM majority under intentional attacks, the irrelevant indications suggest that the unbalanced placement strategy should be used, allowing the defender to protect against a larger fraction of attacks.

# 7 Conclusion and Future Perspectives

Clouds are changing the surface of information technologies. As a consequence, the concentration of resources in the same region, network, or even machine poses an evident challenge to designers of dependable systems. In this paper we evaluate the problem of distributing resources over physical machines. We adopt the perspective of the cloud provider that needs to distribute VMs by a given fixed set of PMs. While intuition could perhaps suggest that a balanced distribution of VMs would make a more dependable system for most scenarios, this is not the case.

Based on real evidence of security incidents, the behavior of a defender should consider whether he or she needs to keep a majority of PMs or VMs. Whereas balancing is indeed better for the latter case, unbalancing is the best option for the former. In any case, it is wise to adopt a strategy for placing replicas, rather than leaving the distribution uncontrolled, given that there are relatively few cases in which replica placement may be considered irrelevant.

An interesting perspective for the future is to include probabilities in our analysis. For example, given the probability of having a compromised VM, what is the probability that other co-located VMs might become compromised as well? The same for the hypervisor: given a compromised VM what is the probability that the attack escalates to affect the hypervisor? Standing on figures owned by cloud providers, this may give an idea of the probability that an attack manages to control a majority of VMs or PMs, thus helping providers to select the best defensive strategy.

# References

[1] CloudFlare blog - The DDoS That Almost Broke the Internet. http://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet. Retrieved on May 6, 2014.

[2] Computerworld - Researchers Demo Cloud Security Issue with Amazon AWS Attack. http://www.computerworld.com/s/article/9221208/Researchers_demo_cloud_security_issue_with_Amazon_AWS_attack/. Retrieved on May 6, 2014.

[3] Dark Reading - Researchers Develop Cross-VM Side-Channel Attack. http://www.darkreading.com/attacks-breaches/researchers-develop-cross-vm-side-channel-attack/d/d-id/1138623? Retrieved on May 6, 2014.

[4] Google App Engine — Google Developers. https://developers.google.com/appengine/. Retrieved on May 6, 2014.

[5] Hacking D-Link Routers With HNAP. http://www.sourcesec.com/Lab/dlink_hnap_captcha.pdf. Retrieved on May 6, 2014.

[6] Heroku — Cloud Application Platform. http://www.heroku.com. Retrieved on May 6, 2014.

[7] InformationWeek - New Virtualization Vulnerability Allows Escape To Hypervisor Attacks. http://www.informationweek.com/security/application-security/new-virtualization-vulnerability-allows/240001996/. Retrieved on May 6, 2014.

[8] Papers — Oracle VM VirtualBox. https://www.virtualbox.org/wiki/Papers. Retrieved on May 6, 2014.

[9] Ruby On Rails and PHP Cloud Hosting PaaS — Managed Rails Development — Engine Yard Platform as a Service. http://www.engineyard.com. Retrieved on May 6, 2014.

[10] SFGate.com blog - The Google Attack Scenario Offense and Defense. http://blog.sfgate.com/ybenjamin/2010/04/20/the-google-attack-scenario-offense-and-defense/. Retrieved on May 6, 2014.

[11] Technical White Papers — VMWare. http://www.vmware.com/resources/techresources/. Retrieved on May 6, 2014.

[12] Filipe Araujo, Raul Barbosa, and António Casimiro. Replication for dependability on virtualized cloud environments. In *Proceedings of the 10th International Workshop on Middleware for Grids, Clouds and e-Science*, MGC '12, pages 2:1–2:6, New York, NY, USA, 2012. ACM.

[13] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM.

[14] Fabrice Bellard. QEMU, a fast and portable dynamic translator. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, pages 41–41, Berkeley, CA, USA, 2005. USENIX Association.

[15] Stefan Berger, Ramón Cáceres, Kenneth A. Goldman, Ronald Perez, Reiner Sailer, and Leendert van Doorn. vTPM: virtualizing the trusted platform module. In *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*, USENIX-SS'06, Berkeley, CA, USA, 2006. USENIX Association.

[16] Alysson Bessani, Leucio A. Cutillo, Gianluca Ramunno, Norbert Schirmer, and Paolo Smiraglia. The TClouds Platform: Concept, Architecture and Instantiations. In *Proceedings of the 2Nd International Workshop on Dependability Issues in Cloud Computing*, DISCCO '13, pages 1:1–1:6, New York, NY, USA, 2013. ACM.

[17] Alysson Neves Bessani, Eduardo Pelison Alchieri, Miguel Correia, and Joni Silva Fraga. DepSpace: A Byzantine Fault-tolerant Coordination Service. *SIGOPS Oper. Syst. Rev.*, 42(4):163–176, April 2008.

[18] F.L. Camargos, G. Girard, and B. des Ligneris. Virtualization of Linux servers. In *Proceedings of the Linux Symposium*, July 2008.

[19] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, November 2002.

[20] Miguel Correia, Giuliana S. Veronese, and Lau Cheuk Lung. Asynchronous Byzantine consensus with $2f+1$ processes. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 475–480, New York, NY, USA, 2010. ACM.

[21] P. Domingues, F. Araujo, and L. Silva. Evaluating the performance and intrusiveness of virtual machines for desktop grid computing. In *IEEE International Symposium on Parallel Distributed Processing (IPDPS) Workshops 2009.*, pages 1–8, 2009.

[22] Nils Gruschka and Luigi Lo Iacono. Vulnerable cloud: SOAP message security validation revisited. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 625–631. IEEE, 2009.

[23] Diwaker Gupta, Sangmin Lee, Michael Vrable, Stefan Savage, Alex C. Snoeren, George Varghese, Geoffrey M. Voelker, and Amin Vahdat. Difference engine: harnessing memory redundancy in virtual machines. *Communications of the ACM*, 53(10):85–93, 2010.

[24] Meiko Jensen, Jorg Schwenk, Nils Gruschka, and Luigi Lo Iacono. On technical security issues in cloud computing. In *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*, pages 109–116. IEEE, 2009.

[25] Norman L. Johnson and Samuel Kotz. *Urn Models and Their Application — An Approach to Modern Discrete Probability Theory.* John Wiley & Sons, Inc., 1977.

[26] Michael McKeown, Hanu Kommalapati, and Jason Roth. Disaster Recovery and High Availability for Windows Azure Applications. Web page `http://msdn.microsoft.com/en-us/library/windowsazure/dn251004.aspx` visited on December 5th 2013.

[27] Xiaoqiao Meng, Canturk Isci, Jeffrey Kephart, Li Zhang, Eric Bouillet, and Dimitrios Pendarakis. Efficient resource provisioning in compute clouds via vm multiplexing. In *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, pages 11–20, New York, NY, USA, 2010. ACM.

[28] Elsa Prieto, Rodrigo Diaz, Luigi Romano, Roland Rieke, and Mohammed Achemlal. MASSIF: A Promising Solution to Enhance Olympic Games IT Security. In Christos K. Georgiadis, Hamid Jahankhani, Elias Pimenidis, Rabih Bashroush, and Ameer Al-Nemrat, editors, *ICGS3/e-Democracy*, volume 99 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 139–147. Springer, 2011.

[29] D. Slamanig and C. Hanser. On cloud storage and the cloud of clouds approach. In *Internet Technology And Secured Transactions, 2012 International Conferece For*, pages 649–655, 2012.

[30] Jason D. Sonnek, James B. S. G. Greensky, Robert Reutiman, and Abhishek Chandra. Starling: Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration. In *ICPP*, pages 228–237. IEEE Computer Society, 2010.

[31] Jakub Szefer, Eric Keller, Ruby B Lee, and Jennifer Rexford. Eliminating the hypervisor attack surface for a more secure cloud. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 401–412. ACM, 2011.

[32] Verizon. Data breach investigations report. `http://www.verizonenterprise.com/DBIR/2013/`, 2013. Retrieved on May 6, 2014.

[33] Timothy Wood, Gabriel Tarasuk-Levin, Prashant Shenoy, Peter Desnoyers, Emmanuel Cecchet, and Mark D. Corner. Memory buddies: Exploiting page sharing for smart colocation in virtualized data centers. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '09, pages 31–40, New York, NY, USA, 2009. ACM.

[34] Yinqian Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Cross-VM side channels and their use to extract private keys. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 305–316. ACM, 2012.