Redundancy Planning for Cost Efficient Resilience to Cyber Attacks

Supplementary material

Jukka Soikkeli, Giuliano Casale, Luis Muñoz-González, Emil C. Lupu

TABLE 1
Workload service demands and parameters

Workload service demands			
Job class	**WLS CPU**	**DBS CPU**	**DBS I/O**
NewOrder	12.98ms	10.64ms	1.12ms
ChangeOrder	13.64ms	10.36ms	1.27ms
OrderStatus	2.64ms	2.48ms	0.58ms
CustStatus	2.54ms	2.08ms	0.3ms
WorkOrder	24.22ms	34.14ms	1.68ms
Parameters: client numbers and think times at different loads			
Parameter	**Low load**	**Moderate**	**Heavy**
NewOrder clients	30	50	100
ChangeOrder clients	10	40	50
OrderStatus clients	50	100	150
CustStatus clients	40	70	50
WorkOrder lines	50	100	200
Cust. think time	2s	2s	3s
Manuf. think time	3s	3s	5s

1 Further details on case study workload

The service demands and parameter values for the Queueing Network (QN), from [1], are provided in Table 1. There are five classes of jobs processed in the system: NewOrder is a new order request; ChangeOrder is a request to change an existing order; OrderStatus requests the status for a given order, while CustStatus lists all orders by a given customer; WorkOrder is an order for widgets from a manufacturer.

The processing times for each job are given for the three processing components: WebLogic server CPU (WLS CPU), Database Server CPU (DBS CPU) and Database Server I/O (DBS I/O). Client numbers refer to the number of customers that are concurrently in the system creating jobs of the different classes, while the counts of manufacturing lines are the number of WorkOrder jobs in the system. Full details of the workload and queueing model in the case study are provided in [1].

2 Why is a QN model required?

For the purposes of redundancy analysis, a performance model is required for investigating the performance impact of using different server multiplicities in the system components, as opposed to a case where components are considered without the detail of server multiplicities. Not considering the multiplicities, we cannot fully consider the benefits of redundancy, as the cases that could be modeled are limited to diversification where the servers are duplicated (e.g. a service requiring two regular servers can only be diversified using a set of two alternative servers, not any fewer).

A confirmation of the usefulness of this approach is found in the results of our study. In Fig. 8 of the main document, the optima where redundancy is not simple duplication of the server numbers required for minimal acceptable performance, such as in the allocation [2,1,2,1], would not be found without a detailed modeling of how the server multiplicity affects the performance. As QNs are a standard way of modeling the impact of system capacity and server multiplicities on performance, QNs are an appropriate modeling method for our question, and required for finding some of the detailed results.

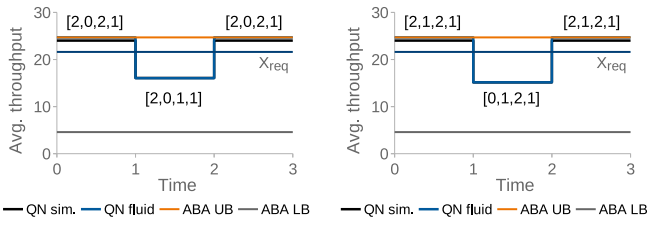
In the rest of this section, we provide a small example demonstrating the point that a closed form approximation is not adequate to approximate the performance in our case study when changes occur due to attacks, so a QN approximation is preferred. For this, we use the Asymptotic Bound Analysis (ABA) bounds, as discussed in e.g. [2]. In the real world N customers would issue a request workflow, but we map this to $R = 5$ job classes, each having population $N_r \cdot N$ such that $\sum_r N_r = 1$ where N_r represents the average fraction of the N customers that issue a request to the service r. The classes are treated as disjoint, representing different service requests. Due to the disjointedness of the classes, the ABA bounds apply.

The ABA bounds used for the throughput of job class r are:

$$\text{ABA UB}_r = \min\left(\frac{1}{D_{max}}, \frac{N_r \cdot N}{D_{sum} + Z_r}\right)$$
$$\text{ABA LB}_r = \frac{N_r \cdot N}{N_{sum} \cdot D_{sum} + Z_r}$$

where r is a job class, i a station in the QN model, $D_{max} = \max_i(D_i)$, $D_{sum} = \sum_i D_i$, and $N_{sum} = \sum_r N_r \cdot N = N$. Where N is the number of concurrent jobs in the system, N_r the fraction of customers issuing a request of class r, Z_r is

The authors are with the Department of Computing, Imperial College London, London, UK.
Email: {j.soikkeli,g.casale,l.munoz,e.c.lupu}@imperial.ac.uk
The support of the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant Reference EP/L016796/1) is gratefully acknowledged.



(a) Temporary allocation [2,0,1,1] (b) Temporary allocation [0,1,2,1]

Fig. 1. Comparison of performance evaluation with QN and two simpler approximations

the think time per job class, and D_r the service demand per class.

Fig. 1 shows two examples of when a performance approximation without a QN model would be misleading. The subfigures show attack outcomes that can occur under certain server allocations, where simple approximations fail to detect a performance drop below the level required by the SLA, X_{req}.

In Fig. 1a, the server allocation is [2,0,2,1], and at $t = 1$ an attack disrupts one database server in S_D, causing the active server allocation to be [2,0,1,1]. When the allocation [2,0,2,1] is in effect, the performance of the system can be approximated well by the ABA upper bound (ABA UB). The queueing model results, both with simulation (QN JMT) and a fluid approximation (QN fluid) are close to the ABA UB approximation. However, the simple approximation does not capture the drop in performance detected by the queueing model when the allocation [2,0,1,1] is in effect. The figure also shows the ABA lower bound (ABA LB), which is far below the QN estimates even in the case of the [2,0,1,1] allocation. A similar situation occurs in Fig. 1b, where the diversified allocation [2,1,2,1] is used, and an attack occurs to the regular application servers (the first element), causing the effective server allocation to be [0,1,2,1]. Using the approximation methods without a QN model would lead to misleading result, as the performance drop below X_{req} would go undetected.

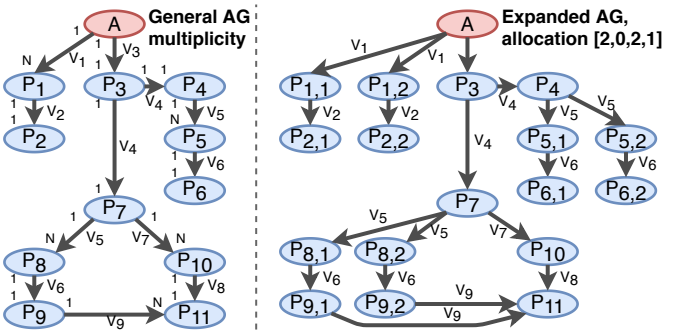
It can be shown that solving the multiclass case is no easier than a convex optimization program, and thus it is reasonable that a closed form formula tends to struggle to approximate something as complex.

3 List of Privileges and Vulnerabilities Used in the Case Study

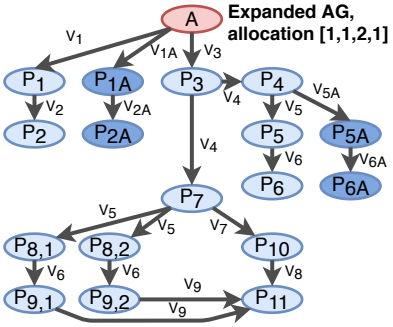
The specific privileges represented by the Attack Graph (AG) nodes of Fig. 4a, and the vulnerabilities in the edges, are listed in Table 2. These represent possible vulnerabilities that could have occurred in the case study system, relating to the appropriate component versions and time period for the hypothetical system. In practice there could be various vulnerabilities that can achieve the same attack outcomes, including attack techniques that do not require application vulnerabilities.

4 Handling of Multiplicity in the AG and DG

The multiplicity of servers, and diversification, affects the structure of the AG for the system, as different server copies



(a) General structure and expanded AG for [2,0,2,1]



(b) Expanded AG for [1,1,2,1]

Fig. 2. AG multiplicity notation, and expanded cases

can have separate copies of privileges required. How this notation relates to the underlying AG, which depends on the server multiplicity, is shown in Fig. 2.

There are four points of note that are visible in Fig. 2:

1) *Multiplicity notation, as in the left panel of Fig. 2a, is used to simplify an AG based on its general structure*, without showing the full complexity arising from duplicates of privileges in different server copies.
2) *Each server allocation will generate a different underlying AG*, due to the server multiplicities. For example, the right-hand side panel of Fig. 2a shows the underlying AG corresponding to the allocation [2,0,2,1].
3) *Server diversification will generate a similar AG structure as a non-diversified allocation with the same number of servers. However, this will differ in the number of distinct vulnerabilities* that an attacker requires to exploit. For example, in Fig. 2b, the server allocation [1,1,2,1] creates an AG that looks similar to the one for the allocation [2,0,2,1] in Fig. 2a. However, as the application servers are diversified, to disrupt two application servers now requires privileges in diverse servers that require the exploit of different vulnerabilities, for example, obtaining administrator privileges on the application servers (P_2 and P_{2A}) requires four different vulnerabilities V_1, V_{1A}, V_2 and V_{2A}. With a non-diversified system the attack could be done with two different vulnerabilities V_1 and V_2.
4) *The introduction of different vulnerabilities due to diversity influences two things:* the *likelihood of detection*, and the *time to complete all the exploits required* for an attack. In our model, the key impact is from the former, as the time to exploit does not affect the time of the disruption,

TABLE 2
Privileges and vulnerabilities in the case study

Privileges				Vulnerabilities			
P_1: U, WebLogic 7	P_5: U, SuSE 8 (AS)	P_9: A, SuSE 8 (DS)	P_{1A}:U, alt. server	V_1: CVE-2003-0151	V_5: CVE-2004-1175	V_9: CVE-2004-0638	V_{1A}:obtain user priv.
P_2: A, WebLogic 7	P_6: A, SuSE 8 (AS)	P_{10}: U, Oracle 9i DB	P_{2A}:A, alt. server	V_2: CVE-2003-0640	V_6: CVE-2004-0495		V_{2A}:priv. escal. (U-A)
P_3: U, LAN 2	P_7: A, DB Admin	P_{11}: A, Oracle 9i DB	P_{5A}:U, alt. OS (AS)	V_3: Phishing attack	V_7: CVE-2002-0965		V_{5A}:obtain user priv.
P_4: A, Server Admin	P_8: U, SuSE 8 (DS)		P_{6A}:A, alt. OS (AS)	V_4: CVE-2006-5051	V_8: CVE-2004-1707		V_{6A}:priv. escal. (U-A)

Abbreviations: U – User; A – Admin; AS – application server; DS – Database Server; DB – Database; alt. server – alternative server; priv. – privilege; escal. – escalation

only the lead time before the disruption occurs. In our model, diversification will lead to an increased likelihood of detection, as more distinct attacks need to be attempted, so attack outcomes with lesser extent of disruption are more likely.

5 Attack detection: example attack outcomes

The detection of attacks creates different outcomes from a given attack scenario. For an attack with a full length of n steps, the probability of the attack succeeding fully is $(1-p_d)^{n-1}$. Any sub-path with length $m > 0$ steps will have the probability of occurrence $(1-p_d)^{m-1} \cdot p_d$. There are two outcomes yielding the full path: going fully undetected, and getting detected during the final move step. The latter case does not limit attack success, as it implies containing the attack to what is already the terminal node of the path. With the outcomes yielding partial paths, the attack gets contained at the privilege where the detection occurred, which happens at probability p_d.

For example, for the attack path $[A \rightarrow P_3, P_3 \rightarrow P_4, P_4 \rightarrow P_5, P_5 \rightarrow P_6]$ we would have the attack outcomes:

a) $[A \rightarrow P_3, P_3 \rightarrow P_4, P_4 \rightarrow P_5, P_5 \rightarrow P_6]$ with probability $(1-p_d)^3$
b) $[A \rightarrow P_3, P_3 \rightarrow P_4, P_4 \rightarrow P_5]$ with probability $(1-p_d)^2 \cdot p_d$
c) $[A \rightarrow P_3, P_3 \rightarrow P_4]$ with probability $(1-p_d) \cdot p_d$
d) $[A \rightarrow P_3]$ with probability p_d

6 Complexity: Attack outcomes to evaluate, and variant cases

For each server allocation (e.g. [2,0,2,1]), the expected impact of cyber attacks is calculated based on the losses due to performance impacts across the different possible outcomes of the attack scenarios. That is, for each candidate server allocation, the performance of the system (and related SLA losses, if any) must be evaluated for each attack outcome in each attack scenario considered. Consequently, the computational complexity of the cost impact evaluation depends largely on the number of different attack outcomes that must be tested. In this section we explain in more detail what this means, and also how the memoization approach we use speeds things up by reducing the number of times a performance calculation is conducted.

6.1 Variant attack cases for attack scenarios

Adding a diverse implementation of a given server leads to added variant cases for the attack scenarios, accounting for the attacker being able to exploit both the existing ("regular") and the alternative server, or only one of them (or neither). The likelihood of each of them is determined based on the assumptions about attacker capabilities. In this section we focus on the number of cases created, as that contributes to computational complexity, while the probabilities assigned to each variant are discussed in Section 16 below.

Thus, each case of splitting a server cluster into two types (regular and alternative) yields four variant cases for an attack scenario. For example, Table 3 shows the variants arising from diversification for attack Scenario 1. This also includes two sub-variants (1a and 1b) for the case where the attacker exploits both the regular and alternative servers, where the order of the exploits is changed. This different ordering may not affect the impact or likelihood of the attacks, but is considered for completeness.

TABLE 3
Variants to attack Scenario 1 due to diversification

Variant	**Non-diversified case**	Variant	**Diversified case**
1	$[A, P_1, P_2]$	1a	$[A, P_1, P_2, P_{1A}, P_{2A}]$
2	$[A]$	1b	$[A, P_{1A}, P_{2A}, P_1, P_2]$
		2	$[A, P_1, P_2]$
		3	$[A, P_{1A}, P_{2A}]$
		4	$[A]$

Where the variants yield the same attack impact matrix M_{attack}, with memoization the performance impact gets evaluated only once for variants with equivalent impacts. Thus the addition of variants only increases computational complexity in terms of computing the appropriate probabilities to be assigned to each case, which is computationally cheap.

6.2 Attack outcomes to test for a given allocation

The number of outcomes that must be evaluated for a given scenario (or its variant) depends on: a) the number of steps in the attack path for the scenario; b) number of copies of privileges the attacker needs to obtain for the full success of the attack; c) assumptions on attack progression and defense (e.g. Case 1 vs Case 2 defense, the order in which the attacker conducts repeated exploits, etc.).

The simplest measure is the number of steps there are in the path that describes the success outcome of a scenario (or its variant arising due to diversification). This is, however, typically not that relevant for the computational complexity of our approach, as many outcomes will only consist of stepping stone exploits and do not have a performance impact (as only steps that affect the servers or the DB have an impact). Therefore only a subset of the outcomes, those that have a performance impact, require a performance evaluation with separate M_{attack} matrices, while "no impact" outcomes get grouped together and assigned a M_{attack} matrix without an attack stage.

TABLE 4
Attack outcomes with different defense assumptions, Scenario 1 attack variant 1a

#	Case 1 defense, allocation [2,2,2,1]
1	$[A, P_1, P_1, P_2, P_2, P_{1A}, P_{1A}, P_{2A}, P_{2A}]$
2	$[A, P_1, P_1, P_2, P_2, P_{1A}, P_{1A}]$
3	$[A, P_1, P_1, P_2, P_2]$
4	$[A, P_1, P_1]$
5	$[A]$

#	Case 2 defense, allocation [2,2,2,1]
1	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$
2	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}]$
3	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}]$
4	$[A, P_1, P_2, P_1, P_2, P_{1A}]$
5	$[A, P_1, P_2, P_1, P_2]$
6	$[A, P_1, P_2, P_1]$
7	$[A, P_1, P_2]$
8	$[A, P_1]$
9	$[A]$

Note that with the Case 1 defense assumption, the number of distinct outcomes does not increase with the number of servers in a given cluster (i.e. [2,2,2,1] and [2,5,2,1] yield the same number of distinct outcomes), because it is assumed that if the exploit of the first copy of a server of that type was not detected, the rest will not be detected either.

Algorithm 1 Evaluation of SLA penalties for each candidate allocation

Precondition: m is a candidate server allocation; $scenarios$ a list of scenario objects; $params$ is a list of other parameters, including $clients$ (the number of clients), p_c (the SLA breach cost per client) and $breach\_limit$ (breach limit, β).

1: **function** SLAPENALTIES($m, scenarios, params$)
2: $SLApenalty := clients * p_c$
3: $outcomes := gen\_outcomes(m, scenarios, params)$
4: $att\_mats := gen\_att\_mats(m, outcomes, params)$
5: $E\_pen := 0$ ▷ Expected penalty
6: **for** each am in att_mats **do**
7: $breach\_metric := perf\_eval(am, m, params)$
8: **if** $breach\_metric > breach\_limit$ **then**
9: $E\_pen := E\_pen + am.weight * SLApenalty$
10: **end if**
11: **end for**
12: **return** E_pen
13: **end function**

The pseudocode Algorithm 1 shows the key parts impacting the complexity of the approach. This algorithm must be run for each candidate allocation tested in the optimization. This shows that the attack outcomes (line 3) must be computed for each allocation m, as they depend on m and the attack scenarios, and the attack matrices (M_{attack}) have to be generated from these (line 4). After this, the system performance must be evaluated for each attack outcome matrix (line 7), in order to calculate the expected penalty from SLA breaches (line 9).

The number of rows in the M_{attack} matrix determines how many performance evaluations are required for a given attack outcome. However, there is a lot of repetition in these stages, so memoization can drastically limit the number of times the performance evaluation has to be conducted.

6.3 Worst case complexity: Optimization bounds and attack outcomes

The number of times the evaluation is conducted during an optimization depends on the size of the solution space and the number of attack outcomes to consider, in addition to the effectiveness of the optimization algorithm. In the worst case the optimization will simply attempt all candidate allocations in the search space. Due to the way server numbers in an allocation affect attack outcomes and their impact, many of the same outcomes and impacts occur across multiple server allocations. Here the memoization of solutions based on M_{attack} matrices helps.

Furthermore, the "effective allocations" resulting from attack outcomes have a lot of repetition across different allocations and attack scenarios. This can occur within a given scenario as well when not all attack steps cause server impacts, as can seen in Table 5. Due to this, we can use another type of memoization, saving the performance values relating to a given effective allocation. While the time to reach these effective allocations depends on the number of steps, and thus the M_{attack} matrix will differ, we use memoization to the performance levels corresponding to the allocations, and then apply them for the duration that is appropriate for each attack outcome.[1] With this memoization approach, **the number of times we evaluate the performance is capped by the number of distinct combinations of servers in the search space, plus the number of distinct effective allocations from attack outcomes.**

TABLE 5
Effective allocations from attack outcomes, Scenario 1 attack variant 1a

#	Case 1 defense, allocation [2,2,2,1]	Functioning servers after outcome
1	$[A, P_1, P_1, P_2, P_2, P_{1A}, P_{1A}, P_{2A}, P_{2A}]$	[0, 0, 2, 1]
2	$[A, P_1, P_1, P_2, P_2, P_{1A}, P_{1A}]$	[0, 2, 2, 1]
3	$[A, P_1, P_1, P_2, P_2]$	[0, 2, 2, 1] (repeat)
4	$[A, P_1, P_1]$	~~[2, 2, 2, 1]~~ (no impact)
5	$[A]$	~~[2, 2, 2, 1]~~ (no impact)
#	**Case 1 defense, allocation [2,3,2,1]**	Functioning servers
1	$[A, P_1, P_1, P_2, P_2, P_{1A}, P_{1A}, P_{1A}, P_{2A}, P_{2A}, P_{2A}]$	[0, 0, 2, 1]
2	$[A, P_1, P_1, P_2, P_2, P_{1A}, P_{1A}, P_{1A}]$	[0, 3, 2, 1]
3	$[A, P_1, P_1, P_2, P_2]$	[0, 3, 2, 1] (repeat)
4	$[A, P_1, P_1]$	~~[2, 3, 2, 1]~~ (no impact)
5	$[A]$	~~[2, 3, 2, 1]~~ (no impact)
#	**Case 2 defense, [2,2,2,1]**	Functioning servers
1	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$	[0, 0, 2, 1]
2	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}]$	[0, 1, 2, 1]
3	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}]$	[0, 1, 2, 1] (repeat)
4	$[A, P_1, P_2, P_1, P_2, P_{1A}]$	[0, 2, 2, 1]
5	$[A, P_1, P_2, P_1, P_2]$	[0, 2, 2, 1] (repeat)
6	$[A, P_1, P_2, P_1]$	[1, 2, 2, 1]
7	$[A, P_1, P_2]$	[1, 2, 2, 1] (repeat)
8	$[A, P_1]$	~~[2, 2, 2, 1]~~ (no impact)
9	$[A]$	~~[2, 2, 2, 1]~~ (no impact)
#	**Case 2 defense, allocation [3,3,2,1]**	Functioning servers
1	$[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$	[0, 0, 2, 1]
2	$[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}, P_{1A}]$	[0, 1, 2, 1]
3	$[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$	[0, 1, 2, 1] (repeat)
4	$[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}]$	[0, 2, 2, 1]
5	$[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}]$	[0, 2, 2, 1] (repeat)
6	$[A, P_1, P_2, P_1, P_2, P_1, P_2, P_{1A}]$	[0, 3, 2, 1]
7	$[A, P_1, P_2, P_1, P_2, P_1, P_2]$	[0, 3, 2, 1] (repeat)
8	$[A, P_1, P_2, P_1, P_2, P_1]$	[1, 3, 2, 1]
9	$[A, P_1, P_2, P_1, P_2]$	[1, 3, 2, 1] (repeat)
10	$[A, P_1, P_2, P_1]$	[2, 3, 2, 1]
11	$[A, P_1, P_2]$	[2, 3, 2, 1] (repeat)
12	$[A, P_1]$	~~[3, 3, 2, 1]~~ (no impact)
13	$[A]$	~~[3, 3, 2, 1]~~ (no impact)
#	**Case 2 defense, allocation [2,3,2,1]**	Functioning servers
1	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$	[0, 0, 2, 1]
2	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}, P_{1A}]$	[0, 1, 2, 1]
3	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}, P_{2A}]$	[0, 1, 2, 1] (repeat)
4	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}, P_{1A}]$	[0, 2, 2, 1]
5	$[A, P_1, P_2, P_1, P_2, P_{1A}, P_{2A}]$	[0, 2, 2, 1] (repeat)
6	$[A, P_1, P_2, P_1, P_2, P_{1A}]$	[0, 3, 2, 1]
7	$[A, P_1, P_2, P_1, P_2]$	[0, 3, 2, 1] (repeat)
8	$[A, P_1, P_2, P_1]$	[1, 3, 2, 1]
9	$[A, P_1, P_2]$	[1, 3, 2, 1] (repeat)
10	$[A, P_1]$	~~[2, 3, 2, 1]~~ (no impact)
11	$[A]$	~~[2, 3, 2, 1]~~ (no impact)

1. Note that the use of this particular memoization relies on the transitions between states being fast relative to recovery times, and could lead to an evaluation error if transition times between states are slow. If transition times are significant enough so they might cause an error, another evaluation stage could be added to counter such error, as explained in Section 8 of this supplement document.

The **defense assumptions** on how many server copies are impacted by an attack step play a large role in the complexity of the model if the number of servers in a cluster is large. We considered two cases of this, representing the extremes. The "**Case 1**" assumption, where the defense cannot stop an attack step that affects one copy of servers in a cluster from affecting all other identical copies in the same cluster, yields only two outcomes for attack steps on a given server cluster: either all server copies in the cluster are impacted, or none of them are. So only one additional "effective allocation" resulting from an attack needs to be evaluated, in addition to the fully functioning candidate allocation that is evaluated in any case. Thus the number of evaluations required is not directly related to the number of servers, only the number of distinct combinations in the search space. For "**Case 2**", where it is assumed that the attack (and its impact) can be stopped between exploits of copies of identical servers, the number of attack outcomes to evaluate increases with the number of servers.

A generalized version capturing the Case 1 and Case 2 defense assumptions and the situations in between is the following: Assume a server cluster with m servers, and that attacks to the cluster can succeed in groups of $G \in [1, m]$ servers. Denoting $K = \lceil m/G \rceil$, there can then be $K + 1$ distinct attack outcomes from attack steps affecting that cluster. In Case 1 $G = m$ and $K = 1$, while in Case 2, $G = 1$ and $K = m$. This extends to a whole server allocation as follows: for an allocation vector with n elements, m_n representing the number of servers in the nth cluster and K_n the distinct outcomes to evaluate for the cluster, the number of distinct attack outcomes to evaluate is $\prod_{i=1}^{n}(K_i + 1)$. With Case 1, this is always 2^n, and in Case 2 the worst case depends on the upper bound of solution space, so $\prod_{i=1}^{n}(u_i + 1)$, where u_i is the upper bound given for the number of servers in cluster i.

For example, if the lower bound for servers in candidate allocations was given by $l = [2, 2, 2, 2]$, upper bound $u = [5, 5, 5, 5]$ and $G_i = m_i \forall i$ (Case 1), at worst performance would be estimated $\prod_{i=1}^{n}(u_i - l_i + 1) + \prod_{i=1}^{n}(K_i + 1) = 4^4 + 2^4 = 272$ times (in fact some of the attack outcomes are the same as some candidate allocations, but this overlap is of little consequence for the overall complexity). This case scales well, as the number of evaluations depends on the depends on the difference between the upper and lower bound, not the number of servers. However, with Case 2 $K_i = m_i \forall i$, in the worst case we would have to evaluate all combinations between the upper bound u and $[0, 0, 0, 0]$, as the attack outcomes include each number of servers between 0 and K_i for each cluster i. In that case, we would estimate performance in the worst case $\prod_{i=1}^{n}(u_i + 1) = 6^4 = 1296$ times. With the Case 2 assumption, increasing the number of servers in the clusters makes the number of performance evaluations required explode quickly, e.g. with $u = [20, 20, 20, 20]$ the number of evaluations would be $160\,000$. Using the fluid solver used in our 2nd optimization stage, this would take over 130 hours to evaluate based on an average of around 3s per performance evaluation.

As apparent from the previous paragraph, different values of K_i make a great difference to the complexity of the algorithm. In the intermediate cases where the success of an attack on a cluster j progresses at the level of subgroups $G_j > 1$, the number of evaluations required can still be reasonable even for high numbers of servers. For example, if we assume that a cluster of 100 servers could be defended in four equal compartments so when detected the defense could contain the attack from affecting the remaining non-infected compartment, we would have $K_j + 1 = 5$ distinct outcomes from attacks on the cluster, with 0, 25, 50, 75 and 100 functioning servers (i.e. $G_j = 25$). In this case the complexity of the algorithm would still be manageable, if the bounds given to the optimization problem were also reasonable. Let's assume $u = [105, 105, 105, 105]$, $l = [100, 100, 100, 100]$ and $K_i = 4 \forall i$. Then the estimation count in the worst case is $\prod_{i=1}^{n}(u_i - l_i + 1) + \prod_{i=1}^{n}(K_i + 1) = 6^4 + 5^4 = 1296 + 625 = 1921$.

Although we have included Case 2 to illustrate the extreme where defense can contain each server copy individually, we believe that in practice, defense is more likely to be of the type in Case 1, for two reasons: 1. If an exploit has gone unnoticed by a detection system, it is likely that the same exploit applied to another copy of an identical server is also not noticed by the detection system; 2. Containing the effects of an attack at the level of individual copies (or sub-groups) of servers seems unrealistic in a cluster of identical servers, as the exploits could continue until the attacker is purged, while containing the attacker by blocking communication can yield the same performance impact (the servers are unable to fulfill their tasks).

Model granularity and scaling: It is possible that, when the number of servers is high, it does not make sense to consider small changes in server numbers, but groups of servers. In such a case using a one-server granularity is not necessary, but would increase the complexity of the solution unnecessarily. For example, in the previous example with $G_i = 25$, considering upper and lower bounds $u = [125, 125, 125, 125]$, $l = [100, 100, 100, 100]$ without scaling would cause a high number of cases to consider ($\prod_{i=1}^{n}(u_i - l_i + 1) = 456\,976$), although only a very few of them are actually relevant if server additions were considered in groups of $G_i = 25$. In such cases we can scale the representation in the model, so that each unit in cluster i represents a group of servers of the size G_i. With the example with $G_i = 25$, we end up with a model with $u = [5, 5, 5, 5]$, $l = [4, 4, 4, 4]$ and $K_i = 4$ to obtain $\prod_{i=1}^{n}(u_i - l_i + 1) + \prod_{i=1}^{n}(K_i + 1) = 2^4 + 5^4 = 16 + 625 = 641$ performance evaluations in the worst case.

Note that the scaling is limited by the defense assumption that determines G_i, as we want to consider all distinct attack outcomes appropriately. That is, if we have a given G_i, we must scale in proportion to it so that no attack outcomes are lost. So with $G_i = 25$, we can use a $1/25$ or $1/5$ scale, but not otherwise. Similarly, in Case 1, where $G_i = m$, we can scale using any divisor $S < m$ as long as m/S is an integer.

Observed evaluation times: In our evaluation runs with the QN fluid solution, the average time to evaluate the performance of the system for a given row of an M_{attack} matrix was 3.08s, while the average time to process a memo hit was 0.00055s. The above apply to the 2nd stage of the optimization with the QN fluid solution method. The bounds approximation in the 1st stage of the optimization,

used to quickly evaluate a large search space to find a closer region for the fluid solution, is considerably faster, having taken 0.0056s on average to evaluate one row of an M_{attack} matrix. This was achieved on a standard desktop computer with an Intel i7-6700 CPU with 4 cores (8 threads) at 3.4GHz clock speed, and 16GB of RAM.

7 Details on parameter assumptions

7.1 Recovery time assumptions

We combine findings from [3], [4] to gauge reasonable estimates for recovery times. A report by Ponemon Institute on DoS attacks [3] states that the average downtime due to DoS was 9h. The average number of DoS attacks among respondents was four, and 18% of attacks lead to no downtime. Combining these, we estimate that the average downtime from an attack was $9\text{h}/(4 \times 0.82) \approx 2\text{h}45\text{min}$. As our attack modeling uses an hourly frequency,[2] we round this to 3h. Importantly, both 3h and the 2h 45min estimate lie between 0.1% and 1% of a month (43.2 minutes and 7h 12min, respectively), which are often used as boundaries for compensation in SLAs.

However, we are mainly interested in the recovery time of service availability caused by attacks penetrating the company's network, not ones where the attack vector is flooding the servers with requests (which are likely the most important fraction in [3]). Thus, we use another data point to represent recovery from more general attacks. Reporting the recovery time from the most disruptive breach faced by companies in the previous year, [4] found a mean time to recover of three days for attacks that had an economic impact. This 72h is the high value in our t_r sensitivity range.

7.2 Cost assumptions

Costs in the case study: We assume a server operating cost of \$185 per month (\$2200 annually) for regular servers. This is derived using a finding by [5] that IT capital costs were about 45% of the total annualised cost of server operating costs, assuming that a new server costs \$3000 and has an operating life of 3 years. For the SLA penalty costs, we assume that the baseline penalty for breaching the SLA in a given month is \$50 per each concurrent client request during normal operation (requests across job classes, excluding work orders as these are not subject to client SLAs). In our running case with 260 concurrent client requests this penalty equates to \$13000.

8 Approach to speed up the optimization

As explained in Sec. 5.7 of the main document, we speed up the optimization by doing it in multiple stages, and by the use of memoization. The optimization is done in two stages, the first of which involves a bound estimation to approximate for the average throughput in the queueing model.

We use memoized performance values for the individual stages of an attack, as some QN configurations reoccur often during the optimization. This means the performance (throughput) with a given configuration of the QN, in terms of numbers of active servers e.g. [2,0,2,1], is calculated once and added to a memo, and this memoized value is applied for every attack stage where that active server allocation occurs.

In our case study the memoization approach works well, as the transition from one steady state to another is very fast, due to the performance of the servers (multiple jobs per second) relative to the time window of interest for the attacks (one month), so the transition has a negligible effect to the overall performance over the time window. Given this, we can apply the steady-state solution the QNs during the optimization, and memoization can be applied extensively. However, a limitation of this memoization approach is if the transient performance during the transition from steady state to another is of interest, and transient QN analysis is applied. The use of memoized performance values for individual stages of an attack could lead to an evaluation error for the performance over time when transitions between two QN states are not fast. If the transition between states is slow and therefore can affect the estimates for time spent below a required performance, memoization could lead to incorrect estimates of transitions being applied to different parts of attacks. This means memoization will add an evaluation error if the time is significant relative to the time window of interest and a transient QN solution is wanted.

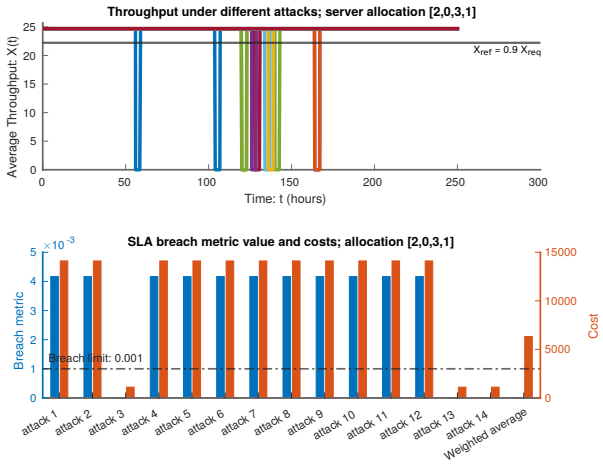
In cases where transition times are significant enough to be of interest, the faster steady-state analysis using memoization can be run first, with a transient analysis of the QNs (using simulation or the LINE fluid solver) applied in a region around the solution obtained from the steady-state analysis, to find if the optimal outcome is affected by the transitions. A neighborhood for the solution could be estimated by applying perturbations to the disruption-time limit β so as to capture the possible errors from omitting the transitions, finding optima for these perturbed versions of the problem, and then evaluating over this neighborhood using the transient solution method.

9 Baseline results

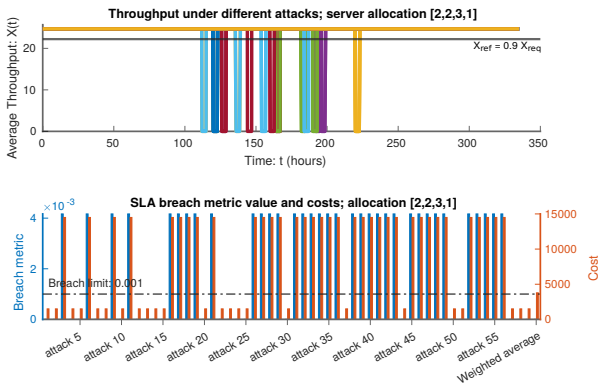
Fig. 3 shows the optimal server allocations in the baseline case, without diversity (non-diversified specification), and with diversification. The parameter values are $t_r = 3$, $p_d = 0.3$, $c_{as} = 1.05 \cdot c_s$, and $\beta = 0.001$. The optimal server allocation is represented by a tuple, e.g. [2,0,2,1]. In the tuple, the first two elements are the number of application servers (in DG node S_A), with standard application servers in the first and alternative application servers (for diversity) in the second element, while the third and fourth elements are the numbers of database servers (in node S_{DB}) and databases (in DB), respectively. In each subfigure, the upper panel shows the average throughput of the system over time under different outcomes of the three attack scenarios. The lower panel depicts the breach metric value (left y-axis) and the cost of the system during the attack (right y-axis).

While under the standard server configuration most of the attack outcomes cause a breach of the SLA and triggering a penalty, diversity mitigates the impact by keeping

2. Although the QN model uses a second frequency for estimating performance, due to the performance of the processing components, we use hours for the time unit of the attacks.

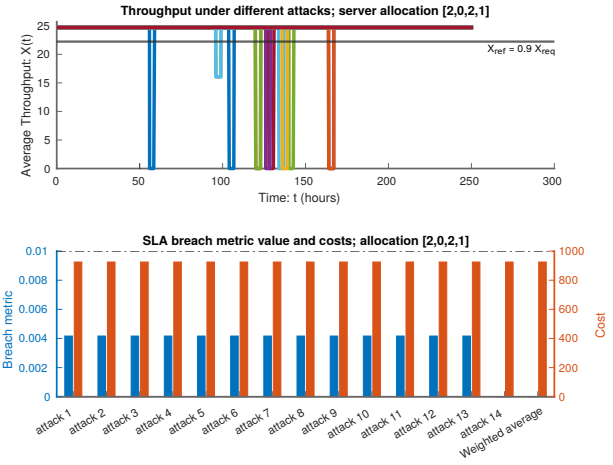


(a) Non-diversified specification

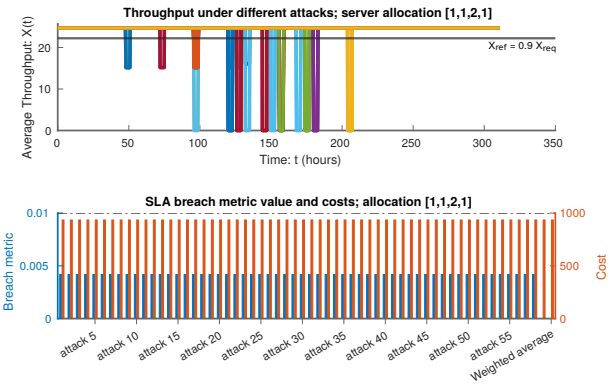


(b) Diversified specification

Fig. 3. Optimal allocations in the case study, baseline, $\beta = 0.001$



(a) Non-diversified specification



(b) Diversified specification

Fig. 4. Optimal allocations in the case study, baseline, $\beta = 0.01$

a large share of the attack outcomes from causing an SLA breach. In this case, diversification helps bring down the expected cost of the attack scenario by 41%, from $6447 to $3795.

Fig. 4 shows another set of baseline results, but with $\beta = 0.01$. While many of the attacks cause some temporary degradation of performance, the optimal allocation avoids a penalty being triggered as the time spent with performance below the contract level is kept below the penalty threshold of 1% of the 30-day time window. With these parameter values, the optimum with diversification is somewhat more expensive, due to the extra cost from investing into the diverse application server configuration instead of the regular choice only.

10 Full sensitivity tables for detection probability

The probability of detection of attack steps p_d affects the relative likelihoods of attack outcomes. As such, it only impacts on optimal allocation if some attack outcome breaches the SLA. If an optimal allocation is found which avoids penalties altogether, varying p_d has no impact on the optimum. We investigated the impact of p_d under the two assumptions over defense effectiveness, Case 1 and Case 2, as discussed in Section 5.2 of the main document.

The results for Case 1 defense model are shown in Table 6, and Case 2 is in Table 7. The tables are structured as follows. The p_d is shown in the first column, and for each level of p_d results for three types of allocation are shown: diversified optimum (Div.), standard non-diversified optimum (N.D), both of which minimize costs under the attack scenarios within their respective type categories, and a reference allocation (Ref.) that has not been optimized. Column two specifies which of these allocation type a given row refers to. The third column shows the optimal server allocation tuple. In the tuples, the first two elements are the number of application servers (in DG node S_A), with regular application servers in the first and alternative application servers (for diversity) in the second element, while the third and fourth elements are the numbers of database servers (in node S_{DB}) and databases (in DB), respectively. The fourth column of the table includes the expected costs under our attack scenarios, the fifth and sixth columns show cost differences relative to N.D. arising from penalty and allocation costs, respectively. Finally, the seventh column shows the percentage difference in the expected cost relative to the optimum without diversification (N.D).

The tables 6 and 7 are split into three parts, highlighting the effects of varying the disruption time limit β and t_r. There are three points that relate to the levels of these parameters which apply to both the Case 1 model (Table 6) and the Case 2 model (Table 7). First, when the recovery time is short enough to be within the "disruption tolerance" of the SLA (e.g. below 7h 12min with $\beta = 0.01$), diversity is

not beneficial as attacks cannot lead to a penalty. Then the optimal approach is to operate with the allocation that is the cheapest during normal times, here [2,0,2,1]. Second, when the recovery time is longer than the disruption tolerance of the SLA (e.g. $t_r >$ 7h 12min with $\beta = 0.01$, or $t_r >$ 45min with $\beta = 0.001$), redundancy can yield cost savings during the attack period relative to operating with the cheapest allocation [2,0,2,1] (labeled as Ref. in the tables). Moreover, diversification (Div.) provides added benefits beyond those from redundancy without diversification (N.D.). Third, the expected costs from attacks leading to an SLA breach are the same regardless of the extent of loss of performance. This is due to the compensation structure in the SLA being fixed if a certain performance level is not met. In our case, where we only use one tolerance level that determines if the SLA is breached, we also find that the results for t_r = 10 with $\beta = 0.01$ are the same as the results for t_r = 3 with $\beta = 0.001$. This is because we apply the same compensation level in both cases, as we conducted the testing by moving the whole compensation structure of the SLA.

The above points apply to both the Case 1 and Case 2 defense models. The analysis for Case 1 of cases when recovery time is such that a breach of the SLA conditions can occur, i.e. excluding $\beta = 0.01$ with $t_r = 3$, was provided in the main document. The remaining analysis in this section relates to Case 2 (Table 7) results in the interesting cases when a breach can occur.

Table 7 shows the sensitivity to detection probability p_d with the Case 2 defense model. When the attacks can lead to a penalty (when β is tight relative to t_r), there is extensive scope for benefiting from redundancy, both with diversity (Div.) and without (N.D). The optima without diversity suggest adding a very large number of redundant services, with the most being 9 redundant application servers (11 in total) when $p_d = 0.1$. This is very different from the situation in Table 6, where redundancy to the application server was never useful if it was done without diversity. The difference arises from the differing assumptions on defensive effectiveness – while in Case 1 the application servers of a given type (regular or alternative) would all become unavailable if an attacker managed to breach one of them (as defense would be unable to stop the further compromises), with Case 2 all redundancy additions increase the likelihood of catching and stopping the attacker before they affect enough servers. It is also notable that with lower detection probabilities, diversification of the application service leads to a lower expected cost than redundancy without diversity, with a much smaller number of servers. With high detection probability, the added allocation cost when using diversification can overcome the other benefits relative to redundancy without diversity. In the table, the 5% cost premium on the alternative server types means redundancy without diversity is marginally more cost efficient than with diversity when p_d = 0.9.

We also find a large impact from the probability of detection. There appear to be two effects – first, redundancy overall becomes less beneficial with a higher probability of detection, although even with at p_d = 0.9 redundancy still yields a cost saving of over 20% relative to the reference allocation. Second, redundancy in services only compromisable via long attack paths (S_{DB} and DB, third and fourth elements of the allocation tuples) disappears completely when p_d increases beyond 0.5, while some redundancy in the application server remains even at p_d = 0.9.

TABLE 6
Sensitivity to detection probability p_d, Case 1 defense

Detection prob. (p_d)	Alloc. type	Optimum allocation	Exp. cost	Cost difference		% diff.
				penalty	alloc.	
β = **0.01,** t_r = **3** (no penalty possible)						
0-0.9	Div.	[1, 1, 2, 1]	934	0	9	+1.0
	N.D.	[2, 0, 2, 1]	925	-	-	-
	Ref.	[2, 0, 2, 1]	925	0	0	0
β = **0.01,** t_r = **10** (penalty possible)						
0	Div.	[2, 2, 2, 1]	11425	-2889	759	-15.7
	N.D.	[1, 0, 1, 1]	13555	-	-	-
	Ref.	[2, 0, 2, 1]	13925	0	370	+2.7
0.1	Div.	[2, 2, 3, 1]	7934	-3309	389	-26.9
	N.D.	[2, 0, 3, 1]	10854	-	-	-
	Ref.	[2, 0, 2, 1]	11143	474	-185	+2.7
0.3	Div.	[2, 2, 3, 1]	3795	-3041	389	-41.1
	N.D.	[2, 0, 3, 1]	6447	-	-	-
	Ref.	[2, 0, 2, 1]	6931	669	-185	+7.5
0.5	Div.	[2, 2, 3, 1]	2119	-2224	389	-46.4
	N.D.	[2, 0, 3, 1]	3954	-	-	-
	Ref.	[2, 0, 2, 1]	4175	406	-185	+5.6
0.7	Div.	[2, 2, 2, 1]	1531	-1317	389	-37.7
	N.D.	[2, 0, 2, 1]	2459	-	-	-
	Ref.	[2, 0, 2, 1]	2459	0	0	0
0.9	Div.	[2, 2, 2, 1]	1322	-434	389	-3.3
	N.D.	[2, 0, 2, 1]	1367	-	-	-
	Ref.	[2, 0, 2, 1]	1367	0	0	0
β = **0.001,** t_r = **3** (penalty possible)						
0	Div.	[2, 2, 2, 1]	11425	-2889	759	-15.7
	N.D.	[1, 0, 1, 1]	13555	-	-	-
	Ref.	[2, 0, 2, 1]	13925	0	370	+2.7
0.1	Div.	[2, 2, 3, 1]	7934	-3309	389	-26.9
	N.D.	[2, 0, 3, 1]	10854	-	-	-
	Ref.	[2, 0, 2, 1]	11143	474	-185	+2.7
0.3	Div.	[2, 2, 3, 1]	3795	-3041	389	-41.1
	N.D.	[2, 0, 3, 1]	6447	-	-	-
	Ref.	[2, 0, 2, 1]	6931	669	-185	+7.5
0.5	Div.	[2, 2, 3, 1]	2119	-2224	389	-46.4
	N.D.	[2, 0, 3, 1]	3954	-	-	-
	Ref.	[2, 0, 2, 1]	4175	406	-185	+5.6
0.7	Div.	[2, 2, 2, 1]	1531	-1317	389	-37.7
	N.D.	[2, 0, 2, 1]	2459	-	-	-
	Ref.	[2, 0, 2, 1]	2459	0	0	0
0.9	Div.	[2, 2, 2, 1]	1322	-434	389	-3.3
	N.D.	[2, 0, 2, 1]	1367	-	-	-
	Ref.	[2, 0, 2, 1]	1367	0	0	0

Notes: c_{as}/c_s=1.05, $P(A|R) = 0.5$, Case 1

To summarize the results from the two tables, *redundancy with diversity is found to be beneficial in all cases when SLA penalties cannot be fully avoided.* However, when the SLA disruption tolerance is loose enough so attacks do not breach the SLA, diversity is of no benefit. This suggests that if the organization is able to insist on an SLA that has sufficiently loose tolerances for performance deviations, it could forgo investment into all types of redundancy, and even attack detection, as it incurs no contractual penalty when attacks occur. Additionally, the probability of detection affects the benefits from redundancy. Higher levels of detection probability reduce the size of the relative benefit from redundancy, and the expected costs from attacks. This suggests that companies could always benefit from increasing their detection capabilities if this is within their budgetary means, while within a given level of detection capability, redundancy can provide added benefits. Of course in practice increasing the level of detection requires investment in both software and human analysts to process alerts, incurring significant costs.

TABLE 7
Sensitivity to detection probability p_d, Case 2 defense

Detection prob. (p_d)	Alloc. type	Optimum allocation	Exp. cost	Cost difference		% diff.
				penalty	alloc.	
β = **0.01,** t_r = **3** (no penalty possible)						
0-0.9	Div.	[1, 1, 2, 1]	934	0	9	+1.0
	N.D.	[2, 0, 2, 1]	925	-	-	-
	Ref.	[2, 0, 2, 1]	925	0	0	0
β = **0.01,** t_r = **10** (penalty possible)						
0	Div.	[2, 2, 2, 1]	11425	-2889	759	-15.7
	N.D.	[1, 0, 1, 1]	13555	-	-	-
	Ref.	[2, 0, 2, 1]	13925	0	370	+2.7
0.1	Div.	[4, 2, 3, 10]	6152	-39	-722	-11.0
	N.D.	[11, 0, 3, 9]	6913	-	-	-
	Ref.	[2, 0, 2, 1]	11143	7560	-3330	+61.2
0.3	Div.	[2, 2, 3, 4]	2911	105	-352	-7.8
	N.D.	[6, 0, 3, 4]	3158	-	-	-
	Ref.	[2, 0, 2, 1]	6931	5253	-1480	+119.5
0.5	Div.	[2, 2, 3, 2]	1939	-101	19	-4.1
	N.D.	[4, 0, 3, 2]	2021	-	-	-
	Ref.	[2, 0, 2, 1]	4175	2894	-740	+106.6
0.7	Div.	[2, 1, 2, 1]	1377	-40	9	-2.2
	N.D.	[3, 0, 2, 1]	1408	-	-	-
	Ref.	[2, 0, 2, 1]	2459	1236	-185	+74.6
0.9	Div.	[2, 1, 2, 1]	1129	-1	9	+0.7
	N.D.	[3, 0, 2, 1]	1121	-	-	-
	Ref.	[2, 0, 2, 1]	1367	431	-185	+21.9
β = **0.001,** t_r = **3** (penalty possible)						
0	Div.	[2, 2, 2, 1]	11425	-2889	759	-15.7
	N.D.	[1, 0, 1, 1]	13555	-	-	-
	Ref.	[2, 0, 2, 1]	13925	0	370	+2.7
0.1	Div.	[4, 2, 3, 10]	6152	-39	-722	-11.0
	N.D.	[11, 0, 3, 9]	6913	-	-	-
	Ref.	[2, 0, 2, 1]	11143	7560	-3330	+61.2
0.3	Div.	[2, 2, 3, 4]	2911	105	-352	-7.8
	N.D.	[6, 0, 3, 4]	3158	-	-	-
	Ref.	[2, 0, 2, 1]	6931	5253	-1480	+119.5
0.5	Div.	[2, 2, 3, 2]	1939	-101	19	-4.1
	N.D.	[4, 0, 3, 2]	2021	-	-	-
	Ref.	[2, 0, 2, 1]	4175	2894	-740	+106.6
0.7	Div.	[2, 1, 2, 1]	1377	-40	9	-2.2
	N.D.	[3, 0, 2, 1]	1408	-	-	-
	Ref.	[2, 0, 2, 1]	2459	1236	-185	+74.6
0.9	Div.	[2, 1, 2, 1]	1129	-1	9	+0.7
	N.D.	[3, 0, 2, 1]	1121	-	-	-
	Ref.	[2, 0, 2, 1]	1367	431	-185	+21.9

Notes: c_{as}/c_s=1.05, $P(A|R) = 0.5$

11 Sensitivity to diversification cost c_{as}

We ran tests with different levels of diversification costs. Tables 8 and 9 show the results of the optimization for different levels of the ratio c_{as}/c_s of per-unit costs for the alternative servers used for redundancy with diversification, in addition to the regular ones. The two tables differ in the SLA disruption-time limit β, i.e. how high a share of disruption is accepted before the SLA is breached and customers are compensated. Table 8 shows the looser SLA disruption-time limit of $\beta = 0.01$ (disruption tolerated for up to 1% of the time in a month), and 9 is with the tighter limit $\beta = 0.001$ (0.1% of time). The tables are structured as follows. The cost ratio c_{as}/c_s is shown in the first column, and the second column shows the optimal server allocations for each cost ratio. In the server allocation tuples, the first two elements of the tuple relate to the number of application servers (in DG node S_A), with regular application servers in the first and alternative application servers (for diversity) in the second element, while the third and fourth elements are the numbers of database servers (in node S_{DB}) and databases (in DB), respectively. The third column of the table includes the expected costs under our attack scenarios, the fourth and fifth columns show cost differences relative to N.D. arising from penalty and allocation costs, respectively. Finally, the sixth column shows the percentage difference in the expected cost relative to the optimum without diversification (N.D.). The optimum without diversification is included below the results for the diversified optimization, specified by the label "N.D." in the first column of the table. The non-diversified optimum does not vary with the cost of the alternative servers, so one comparison point covers all cost-ratios shown.

TABLE 8
Sensitivity to diversification cost levels, $\beta = 0.01$

Cost ratio (c_{as}/c_s)	Optimum allocation	Exp. cost	Cost difference		% diff.
			penalty	alloc.	
Case 1 defense					
1	[1, 1, 2, 1]	925	0	0	0
1.05	[1, 1, 2, 1]	934	0	9	+1.0
1.1	[1, 1, 2, 1]	944	0	19	+2.1
1.2	[1, 1, 2, 1]	962	0	37	+4.0
1.5	[1, 1, 2, 1]	1018	0	93	+10.1
2.0	[1, 1, 2, 1]	1110	0	185	+20.0
N.D.	[2, 0, 2, 1]	925	-	-	-

Notes: p_d = 0.3, t_r = 3, β = 0.01

In Table 8, there is no difference in the optimal allocation chosen in the diversified cases, only the cost for the allocation. The non-diversified allocation has the same amount of servers, but as the servers for the application service are all of the cheaper type, the diversified cases are more expensive. There is no breach of the SLA conditions, so no penalty is applied.

TABLE 9
Sensitivity to diversification cost levels, $\beta = 0.001$

Cost ratio (c_{as}/c_s)	Optimum allocation	Exp. cost	Cost difference		% diff.
			penalty	alloc.	
Case 1 defense					
1.0	[2, 2, 3, 1]	3776	-3041	370	-41.4
1.05	[2, 2, 3, 1]	3795	-3041	389	-41.1
1.1	[2, 2, 3, 1]	3813	-3041	407	-40.9
1.2	[2, 2, 3, 1]	3850	-3041	444	-40.3
1.5	[2, 2, 3, 1]	3961	-3041	555	-38.6
2.0	[2, 2, 3, 1]	4146	-3041	740	-35.7
5.0	[2, 2, 3, 1]	5256	-3041	1850	-18.5
7.0	[2, 2, 3, 1]	5996	-3041	2590	-7.0
8.0	[2, 2, 3, 1]	6366	-3041	2960	-1.3
9.0	[2, 1, 3, 1]	6592	-1520	1665	+2.2
10.0	[2, 1, 3, 1]	6777	-1520	1850	+5.1
N.D.	[2, 0, 3, 1]	6447	-	-	-

Notes: p_d = 0.3, t_r = 3, β = 0.001, $P(A|R) = 0.5$

In Table 9, we again see no differences to the optimal allocation choice due to diversification cost, until the cost of the alternative server type reaches 9 times that of the regular type. Before this level, the cost change is not enough to lead to a different optimal choice. However, the relative benefit of diversification changes. The change in the cost could be enough to impact the choice of whether or not to diversify the system, given the maintenance costs of the redundant servers need to be balanced against the benefits during an attack.

12 Sensitivity to attack scenario weights

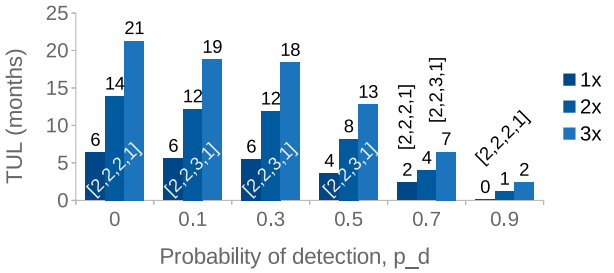
In the baseline situation, we have set the weights of the three different attack scenarios (their relative probabilities of occurrence) as equal, $1/3$ each, as explained in Section 5.3 of the main document. Here we show the impact of varying them.

TABLE 10
Sensitivity to attack scenario weights, attack cost minimization, $\beta = 0.001$

Scenario weights p_{S1}	p_{S2}	p_{S3}	Alloc. type	Optimum allocation	Exp. cost	Cost diff. %	TUL (mths)
0	0.25	0.75	Div.	$[2, 2, 3, 1]$	**3990**	-15.7	2.4
			N.D.	$[2, 0, 3, 1]$	4733	-	**3.5**
			Ref	$[2, 0, 2, 1]$	5384	+13.8	-
0	0.5	0.5	Div.	$[2, 2, 3, 1]$	**4200**	-8.0	2.1
			N.D.	$[2, 0, 3, 1]$	4566	-	**4.4**
			Ref	$[2, 0, 2, 1]$	5384	+17.9	-
0	0.75	0.25	Div.	$[2, 1, 3, 1]$	4404	+0.1	2.6
			N.D.	$[2, 0, 3, 1]$	**4399**	-	**5.3**
			Ref	$[2, 0, 2, 1]$	5384	+22.4	-
0.25	0	0.75	Div.	$[2, 2, 3, 1]$	**3581**	-42.5	**5.2**
			N.D.	$[2, 0, 3, 1]$	6228	-	1.7
			Ref	$[2, 0, 2, 1]$	6544	+5.1	-
0.25	0.25	0.5	Div.	$[2, 2, 3, 1]$	**3791**	-37.4	**4.8**
			N.D.	$[2, 0, 3, 1]$	6060	-	2.6
			Ref	$[2, 0, 2, 1]$	6544	+8.0	-
0.25	0.5	0.25	Div.	$[2, 2, 3, 1]$	**4001**	-32.1	**4.4**
			N.D.	$[2, 0, 3, 1]$	5893	-	3.5
			Ref	$[2, 0, 2, 1]$	6544	+11.0	-
0.25	0.75	0	Div.	$[2, 2, 3, 1]$	**4211**	-26.5	4.1
			N.D.	$[2, 0, 3, 1]$	5726	-	**4.4**
			Ref	$[2, 0, 2, 1]$	6544	+14.3	-
0.5	0	0.5	Div.	$[2, 2, 3, 1]$	**3383**	-55.2	**7.5**
			N.D.	$[2, 0, 3, 1]$	7555	-	0.8
			Ref	$[2, 0, 2, 1]$	7705	+2.0	-
0.5	0.25	0.25	Div.	$[2, 2, 3, 1]$	**3592**	-51.4	**7.2**
			N.D.	$[2, 0, 3, 1]$	7388	-	1.7
			Ref	$[2, 0, 2, 1]$	7705	+4.3	-
0.5	0.5	0	Div.	$[2, 2, 3, 1]$	**3802**	-47.3	**6.8**
			N.D.	$[2, 0, 3, 1]$	7220	-	2.6
			Ref	$[2, 0, 2, 1]$	7705	+6.7	-
0.75	0	0.25	Div.	$[2, 2, 3, 1]$	**3184**	-64.1	**9.9**
			N.D.	$[2, 0, 2, 1]$	8865	-	N/A
			Ref	$[2, 0, 2, 1]$	8865	0	-
0.75	0.25	0	Div.	$[2, 2, 3, 1]$	**3394**	-61.1	**9.5**
			N.D.	$[2, 0, 3, 1]$	8715	-	0.8
			Ref	$[2, 0, 2, 1]$	8865	+1.7	-
1	0	0	Div.	$[2, 2, 2, 1]$	**2800**	-72.1	**18.6**
			N.D.	$[2, 0, 2, 1]$	10025	-	-
			Ref	$[2, 0, 2, 1]$	10025	0	-
0	1	0	Div.	$[1, 1, 3, 1]$	4241	+0.2	5.9
			N.D.	$[2, 0, 2, 1]$	**4231**	-	**6.2**
			Ref	$[2, 0, 2, 1]$	5384	+27.2	-
0	0	1	Div.	$[2, 2, 3, 1]$	**3780**	-22.9	**2.8**
			N.D.	$[2, 0, 3, 1]$	4900	-	2.6
			Ref	$[2, 0, 2, 1]$	5384	+9.9	-
1/3	1/3	1/3	Div.	$[2, 2, 3, 1]$	**3795**	-41.1	**5.5**
			N.D.	$[2, 0, 3, 1]$	6447	-	2.6
			Ref	$[2, 0, 2, 1]$	6931	+7.5	-

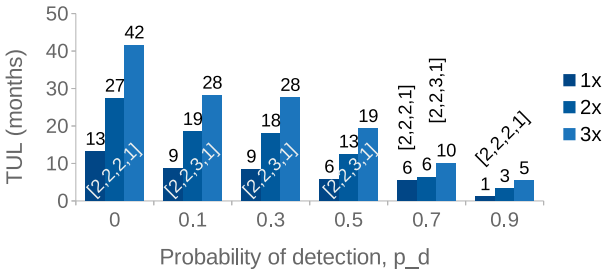
Notes: c_{as}/c_s=1.05, $P(A|R) = 0.5$, β = 0.001, t_r = 3

The main observation relating to the results in Table 10 is that **the optimal allocation is rather stable in face of changes to the scenario weights.** In 14/16 of the cases tested for the table, the optimal allocation was found to be $[2, 2, 3, 1]$ (using attack cost minimization). In the remaining two cases, where a high weight was given to Scenario 2 ($p_{S2} = 1$, and $[p_{S2} = 0.75, p_{S3} = 0.25]$), the optimum was the non-diversified allocation $[2, 0, 3, 1]$.



Case 1, β = 0.001; ref. alloc. [2,0,2,1]; penalty multip. 1x-3x

(a) Minimum-cost reference allocation



Case 1, β = 0.001; ref. alloc. [3,0,2,1]; penalty multip. 1x-3x

(b) Original case-study reference allocation

Fig. 5. Months until cumulative maintenance cost exceeds attack-time benefit

13 Impact on TUL from changing reference allocation

Fig. 5 shows the impact a different reference allocation has on the TUL. The difference between the two subplots of Fig. 5 is the reference allocation used for the TUL calculation.

The reference allocation has a sizable impact on the TUL, with values in Fig. 5b generally higher than those in Fig. 5a by several months, and more than double at the extreme ends of p_d values (0 and 0.9). The reference allocation influences both the numerator and the denominator of the TUL metric, and the large effect observed arises as the additional server in the reference allocation of Fig. 5b causes a large reduction in the excess maintenance cost (the denominator of TUL).

Because of the large impact the reference allocation has on the estimated excess maintenance cost, it is important that the reference point chosen is reasonable for the organization. For example, we can easily find the cheapest allocation providing the minimum required service level during normal times, but this may not reflect the need for redundancy to deal with random faults. In our case study, the minimum-cost allocation that yields the required performance level during normal times is $[2, 0, 2, 1]$. However, the baseline used by [1] for the case study was $[3, 0, 2, 1]$, perhaps reflecting fault tolerance or spare capacity requirements. Using the latter as the reference leads to significantly larger TUL estimates than the former, as shown in Fig. 5.

14 TUL optimization result tables

Tables 11 and 12 show results for TUL optimization. The former table shows results with attack penalty multiplier 1x

($50 per customer), while the latter uses penalty multiplier 3x ($150 per customer).

Note that, due to the condition in the TUL maximization problem (equation (6) in the main document) to consider only allocations with more servers than in the reference allocation, in cases where adding redundancy is not beneficial the optimization may fail to find a solution that is reasonable, i.e. the chosen allocation has a higher cost than the reference. In such cases the reference allocation should be chosen instead. This situation occurs in Tables 11 and 12 for the non-diversified cases with $p_d = 0$ and $p_d = 0.9$, and for $p_d = 0.8$ in Table 11.

TABLE 11
Sensitivity to detection probability p_d, TUL optimization, Case 1 defense

Detection prob. (p_d)	Alloc. type	Optimum allocation	Exp. cost	Cost difference		% diff.
				penalty	alloc.	
β = 0.001, t_r = 3						
0	Div.	[2, 2, 2, 1]	11425	-2889	389	-18.0
	N.D.	[2, 0, 2, 1]	13925	-	-	-
	Ref	[2, 0, 2, 1]	13925	0	0	0
0.1	Div.	[2, 1, 2, 1]	9721	-1142	9	-10.4
	N.D.	[2, 0, 3, 1]	10854	-	-	-
	Ref	[2, 0, 2, 1]	11143	474	-185	+2.7
0.3	Div.	[2, 1, 2, 1]	5630	-826	9	-12.7
	N.D.	[2, 0, 3, 1]	6447	-	-	-
	Ref	[2, 0, 2, 1]	6931	669	-185	+7.5
0.5	Div.	[2, 1, 2, 1]	3263	-700	9	-17.5
	N.D.	[2, 0, 3, 1]	3954	-	-	-
	Ref	[2, 0, 2, 1]	4175	406	-185	+5.6
0.7	Div.	[2, 1, 2, 1]	1995	-658	194	-18.9
	N.D.	[2, 0, 2, 1]	2459	-	-	-
	Ref	[2, 0, 2, 1]	2459	0	0	0
0.9	Div.	[2, 2, 2, 1]	1322	-434	389	-3.3
	N.D.	[2, 0, 2, 1]	1367	-	-	-
	Ref	[2, 0, 2, 1]	1367	0	0	0

Notes: c_{as}/c_s=1.05, $P(A|R) = 0.5$

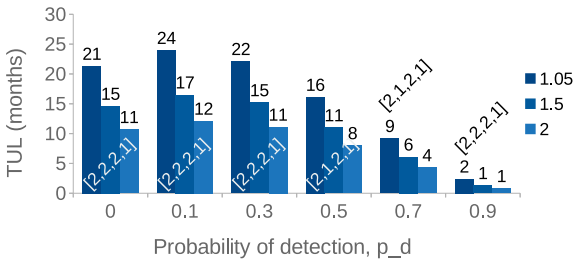
TABLE 12
Sensitivity to detection probability p_d, TUL optimization, Case 1 defense

Detection prob. (p_d)	Alloc. type	Optimum allocation	Exp. cost	Cost difference		% diff.
				penalty	alloc.	
β = 0.001, t_r = 3						
0	Div.	[2, 2, 2, 1]	31647	-8667	389	-20.7
	N.D.	[2, 0, 2, 1]	39925	-	-	-
	Ref	[2, 0, 2, 1]	39925	0	0	0
0.1	Div.	[2, 2, 2, 1]	22270	-8276	204	-26.6
	N.D.	[2, 0, 3, 1]	30342	-	-	-
	Ref	[2, 0, 2, 1]	31579	1422	-185	+4.1
0.3	Div.	[2, 2, 2, 1]	10362	-6963	204	-39.5
	N.D.	[2, 0, 3, 1]	17121	-	-	-
	Ref	[2, 0, 2, 1]	18943	2007	-185	+10.6
0.5	Div.	[2, 1, 2, 1]	7552	-2098	9	-21.7
	N.D.	[2, 0, 3, 1]	9641	-	-	-
	Ref	[2, 0, 2, 1]	10675	1219	-185	+10.7
0.7	Div.	[2, 1, 2, 1]	3746	-1607	9	-29.9
	N.D.	[2, 0, 3, 1]	5344	-	-	-
	Ref	[2, 0, 2, 1]	5527	368	-185	+3.4
0.9	Div.	[2, 2, 2, 1]	1337	-1286	204	-44.7
	N.D.	[2, 0, 2, 1]	1082	-	-	-
	Ref	[2, 0, 2, 1]	1082	0	0	0

Notes: c_{as}/c_s=1.05, $P(A|R) = 0.5$, ccharge=$150

15 TUL SENSITIVITY TO DIVERSIFICATION COSTS c_{as}/c_s

Fig. 6 shows the sensitivity of TUL to changes in server cost ratio c_{as}/c_s, for the TUL maximization problem. The results



Case 1, β = 0.001; ref. alloc. [2,0,2,1]; penalty multip. 3x

Fig. 6. TUL sensitivity to server price ratio c_{as}/c_s

show that the relative cost of diversified servers can have a large impact on TUL, which is due to the maintenance costs during normal times. Taking this together with the findings for other parameters, some of the parameters and assumptions have opposing impacts on the results, so the final result is not easily predictable with rules of thumb. That is, whether diversity is beneficial or not depends on the balance of all costs, and is impacted by multiple parameters, so modeling is recommended to support decision making.

16 ATTACKER CAPABILITIES

When adding diversity to the system, additional vulnerabilities may be added, thus increasing the threat surface. In our analysis we assume that both server types involved, the regular and the alternative one, have vulnerabilities, and the attacker has capabilities to exploit these vulnerabilities with certain probabilities. We denote by $P(R)$ the probability that the attacker has the capability to exploit a vulnerability in the regular server type, and by $P(A)$ the probability that the attacker is capable of exploiting a vulnerability in the alternative type that is used for adding diversity. This means that, when considering diversity, we are interested in four joint probabilities, $P(R \cap A)$, $P(R \cap A')$, $P(R' \cap A)$, and $P(R' \cap A')$, where event R' denotes the complement of event R, i.e. when the attacker does not have the exploit for the vulnerability in the regular server type. These four cases determine the probabilities of scenario variants which represent the full attack to both server types, partial attacks to type R only and to type A only, and no attack to either server type. For example, for attack Scenario 1 $[A \rightarrow P1, P1 \rightarrow P2]$, the diversified case leads to the scenario variants shown in Table 13. Although shown in the table for completeness, when setting the relative weights of the variants in our impact evaluation we do not consider the case $(R' \cap A')$ where the attacker lacks the capability to compromise either server type. Consequently, the weights relating to the three other cases are rescaled to sum to 1, as shown in the "Weight" column of the table.

To make comparisons to the case with no diversification, where only event R is relevant, we are most interested in the probability $P(R \cap A)$ relative to the probability $P(R)$. Here we test the impact of assuming that the events R and A are not independent, which could arise if attackers are more likely to be able to exploit a vulnerability if they already know how to exploit a similar vulnerability. We are thus interested in how the conditional probability $P(A|R)$ affects

TABLE 13
Scenario variants for attack Scenario 1

#	Variant path	Probability	Weight
Non-diversified case			
1	$[A, P_1, P_2]$	$P(R)$	1
2	$[A]$	$P(R')$	0
Diversified case			
1a	$[A, P_1, P_2, P_{1A}, P_{2A}]$	$0.5 \cdot P(R \cap A)$	$0.5 \cdot P(R \cap A)/\zeta$
1b	$[A, P_{1A}, P_{2A}, P_1, P_2]$	$0.5 \cdot P(R \cap A)$	$0.5 \cdot P(R \cap A)/\zeta$
2	$[A, P_1, P_2]$	$P(R \cap A')$	$P(R \cap A')/\zeta$
3	$[A, P_{1A}, P_{2A}]$	$P(R' \cap A)$	$P(R' \cap A)/\zeta$
4	$[A]$	$P(R' \cap A')$	0

Note: The scaling parameter $\zeta = P(R \cap A) + P(R \cap A') + P(R' \cap A)$

the benefits from diversification. Tables 14 and 15 show how the results from diversification are affected by different assumptions with regard to the conditional probability $P(A|R)$, in the case where detection applies to all attack steps, including to replica privileges. The tables assume $P(R) = 0.5$. We also change the conditional probability $P(A|R')$ so that we maintain $P(R \cap A') = P(R' \cap A)$, i.e. both the attacks with partial capabilities are equally likely. This last assumption is made to simplify the amount of parameters to change, while the focus is on $P(A|R)$ and its impact on $P(R \cap A)$.

From Tables 14 and 15 we can see that for the case study system, while the cost impact found is lower if the events R and A are dependent than in the independent case, the benefit from applying diversity remains sizable even with high levels of dependence.

In the analysis included in the paper, we assume that the probability that an attacker has a capability to exploit a given vulnerability is independent from the probability of them having the capability to exploit a different one. This is because vulnerabilities can be very specific to a particular component, and often require specific conditions to be exploitable. Another reason for this assumption is that we are not aware of a database that would provide information on the relations in exploitability between vulnerabilities,[3] which would be required to test to what extent the exploitability of different vulnerabilities is independent or not.

17 Other costs from attacks

If the overall costs from an attack are underestimated, so will the TUL estimate be. This could lead to under-investment in redundancy. The modeling showed above has assumed attack costs only arise when system performance is below contractual levels for a specific time. However, costs related to business loss due to reputation impacts can behave very differently. First, reputational costs may occur in response to an attack event even if an SLA condition is not breached. This can increase the benefit from limiting disruptions from attacks even if the length of disruption would not breach the SLA. Second, reputation costs can keep increasing with the length of disruption even when the SLA penalty is capped at some limit. Third, even in the case that reputational costs would only occur if an SLA was breached, they would add to the cost of the attack impact. Thus, excluding them from the analysis means the TUL estimate is biased down.

We use loss of business as an approximation for reputational costs. The loss of revenue due to a cyber attack was estimated by [7] to represent 20% of all costs of a cyber attack among large international companies, and around 24% of the full cost for DoS attacks. However, as this

3. MITRE ATT&CK® [6] provides the closest approximation to such information which we know, as the joint occurrence of specific attack techniques in known attacker groups' arsenal could be analyzed. However, this is based on known high-profile attacker groups, and on their capabilities that they have been known to exhibit in previous attacks, which mean the data would be unlikely to represent attacker groups more generally, or even the latest level of capabilities of the known attacker groups.

TABLE 14
Sensitivity to conditional probability of exploits $P(A|R)$, Case 1 defense

β = 0.001, t_r = 3

$P(A\|R)$	Optimum allocation	Exp. cost	Cost difference penalty	Cost difference alloc.	% diff.	$P(R \cap A)$
$p_d = 0.3$						
1	[2, 2, 3, 1]	4910	-1926	389	-23.8	0.5
0.9	[2, 2, 3, 1]	4606	-2230	389	-28.6	0.45
0.8	[2, 2, 3, 1]	4352	-2484	389	-32.5	0.4
0.7	[2, 2, 3, 1]	4138	-2698	389	-35.8	0.35
0.6	[2, 2, 3, 1]	3954	-2882	389	-38.7	0.3
0.5	[2, 2, 3, 1]	3795	-3041	389	-41.1	0.25
Comparison point – Non-diversified optimum						
N/A	[2, 0, 3, 1]	6447	-	-	-	N/A
Comparison point – reference allocation						
N/A	[2, 0, 2, 1]	6931	669	-185	+7.5	N/A
$p_d = 0.5$						
1	[2, 2, 3, 1]	2514	-1829	389	-36.4	0.5
0.9	[2, 2, 3, 1]	2406	-1937	389	-39.2	0.45
0.8	[2, 2, 3, 1]	2317	-2026	389	-41.4	0.4
0.7	[2, 2, 3, 1]	2241	-2102	389	-43.3	0.35
0.6	[2, 2, 3, 1]	2176	-2167	389	-45.0	0.3
0.5	[2, 2, 3, 1]	2119	-2224	389	-46.4	0.25
Comparison point – Non-diversified optimum						
N/A	[2, 0, 3, 1]	3954	-	-	-	N/A
Comparison point – reference allocation						
N/A	[2, 0, 2, 1]	4175	406	-185	+5.6	N/A

Notes: c_{as}/c_s=1.05, $P(R)$=0.5, $P(R \cap A') = P(R' \cap A)$

TABLE 15
Sensitivity to conditional probability of exploits $P(A|R)$, Case 2 defense

β = 0.001, t_r = 3

$P(A\|R)$	Optimum allocation	Exp. cost	Cost difference penalty	Cost difference alloc.	% diff.	$P(R \cap A)$
$p_d = 0.3$						
1	[5, 1, 3, 4]	3167	0	9	+0.3	0.5
0.9	[4, 2, 3, 4]	3140	-37	19	-0.6	0.45
0.8	[3, 2, 3, 4]	3078	87	-167	-2.5	0.4
0.7	[3, 2, 3, 4]	3023	32	-167	-4.3	0.35
0.6	[3, 2, 3, 4]	2977	-14	-167	-5.7	0.3
0.5	[2, 2, 3, 4]	2911	105	-352	-7.8	0.25
Comparison point – Non-diversified optimum						
N/A	[6, 0, 3, 4]	3158	-	-	-	N/A
Comparison point – reference allocation						
N/A	[2, 0, 2, 1]	6931	5253	-1480	+119.5	N/A
$p_d = 0.5$						
1	[3, 1, 3, 2]	2030	0	9	+0.4	0.5
0.9	[2, 2, 3, 2]	2012	-28	19	-0.4	0.45
0.8	[2, 2, 3, 2]	1989	-51	19	-1.6	0.4
0.7	[2, 2, 3, 2]	1970	-70	19	-2.5	0.35
0.6	[2, 2, 3, 2]	1953	-87	19	-3.4	0.3
0.5	[2, 2, 3, 2]	1939	-101	19	-4.1	0.25
Comparison point – Non-diversified optimum						
N/A	[4, 0, 3, 2]	2021	-	-	-	N/A
Comparison point – reference allocation						
N/A	[2, 0, 2, 1]	4175	2894	-740	+106.6	N/A

Notes: c_{as}/c_s=1.05, $P(R)$=0.5, $P(R \cap A') = P(R' \cap A)$

estimate is relative to the total cost of an attack, which is not clear in advance and therefore hard to use as a model input, we approximate the impact based on loss of customers. Based on a study of the cost of data breaches of large international companies, [8] find the average abnormal churn in customers after an attack to be 3.4% of customers. Some industries faced significantly higher rates of churn, including 6.7% for health, 6.1% for financial, and 4.6% for tech companies. Industrial companies were below average at 3.1% abnormal churn. As these are estimates of loss of customers for data breaches, which cause larger than average loss of revenue than general attacks, we scale them down using loss of revenue estimates. As these estimates apply to data breaches, which cause larger than average loss of revenue than general attacks, we scale them down using loss of revenue estimates. The share of total attack cost arising from loss of revenue was 40% across all firms in [8] (data breaches only), but 20% for across cyber attack types in [7]. Thus, we obtain an estimate for loss of customers from attacks that do not lead to a data breach by halving the [8] churn rates. That is, to 1.7% on average across industries.

We test two approaches, one where the loss of customers is binary, occurring at the 1.7% average rate for any successful attack, and a second where the customer loss varies based on the extent of disruption following a logistic function.

$$f(x) = \frac{0.034}{1 + e^{-500(x-0.01)}} \quad (1)$$

where $f(x)$ is the rate of loss of customers as a function of the size of the disruption x, measured by the share of time when system performance is below the required level ("disruption-time share"). This provides a smooth sigmoid shape with values between 0 and 0.034, where the midpoint of 0.017 is reached at a disruption-time share of 0.01.

A key parameter here is the length of time for which the number of customers will be dampened. We assume that the impact of the attack on customer numbers will be temporary, so eventually numbers will return to the level that would have been expected in the absence of the attack. We consider durations of 18 and 36 months, and estimate the loss as the duration (in months) times the monthly loss based on the revenue that would have been made from the customers. We discount the losses during future years (from month 13 onward) at a rate of 3% per year.

Tables 16 and 17 show results for when we use an added cost in terms of loss of customers, using fixed loss share in the former and variable loss share in the latter table. The tables show the diversified allocation results for both the attack cost minimization (shown as "Div. (a)") and TUL maximization ("Div. (t)"). The "N.D" allocations were the same with both optimization approaches, so shown in the tables only once for each case.

As we can see from the top part of table 16 with $\beta = 0.01$, with a fixed cost from attack incidents redundancy could reduce the costs from attacks even if they do not lead to breach in the SLA conditions. However, with the parameter values used in Table 16, the TUL is low, so the longer-term cost of the redundancy makes investing at these parameter values unwise unless attacks are expected very frequently. By comparison, with the variable incident costs shown in Table 17, when $\beta = 0.01$, the added cost from the loss of customers is so small that diversification is loss-making, as was the case when we only considered SLA penalty costs.

TABLE 16
Sensitivity to loss of customers, fixed loss share

Loss months	Alloc. type	Optimum allocation	Exp. cost	Cost difference incid.	Cost difference alloc.	% diff.	TUL
β = 0.01, t_r = 3							
18	Div. (a)	[2, 2, 3, 1]	2196	-924	389	-19.6	1.0
	Div. (t)	[2, 1, 2, 1]	2490	-250	9	-8.8	1.3
	N.D.	[2, 0, 3, 1]	2731	-	-	-	-
	Ref.	[2, 0, 2, 1]	2749	203	-185	+0.7	-
36	Div. (a)	[2, 2, 3, 1]	2863	-1808	389	-33.1	2.8
	Div. (t)	[2, 2, 2, 1]	3106	-1380	204	-27.5	3.6
	N.D.	[2, 0, 3, 1]	4282	-	-	-	-
	Ref.	[2, 0, 2, 1]	4495	398	-185	+5.2	-
β = 0.001, t_r = 3							
18	Div. (a)	[2, 2, 3, 1]	4493	-3964	389	-44.3	7.4
	Div. (t)	[2, 1, 2, 1]	7001	-1076	9	-13.2	9.0
	N.D.	[2, 0, 3, 1]	8068	-	-	-	-
	Ref.	[2, 0, 2, 1]	8755	872	-185	+8.5	-
36	Div. (a)	[2, 2, 3, 1]	5160	-4848	389	-46.4	9.3
	Div. (t)	[2, 1, 2, 1]	8312	-1316	9	-13.6	11.3
	N.D.	[2, 0, 3, 1]	9619	-	-	-	-
	Ref.	[2, 0, 2, 1]	10501	1067	-185	+9.2	-

Notes: c_{as}/c_s=1.05, $P(A|R) = 0.5$, Case 1 defense

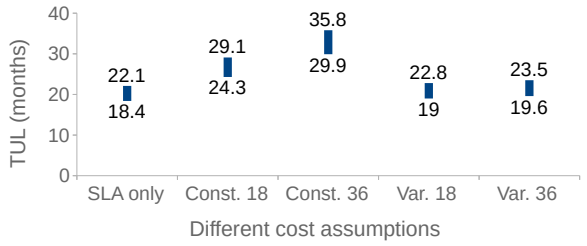
Apart from the slight changes in the expected costs and TUL values, the results are qualitatively similar to those from previous sections. The key impact of ignoring these added costs would be an underestimation of the benefits of redundancy, as the cost differences and TUL would be underestimated. Accordingly, we shall now focus on how the TUL estimates are affected by the two types of additional cost.

TABLE 17
Sensitivity to loss of customers, variable loss share

Loss months	Alloc. type	Optimum allocation	Exp. cost	Cost difference incid.	Cost difference alloc.	% diff.	TUL
β = 0.01, t_r = 3							
18	Div. (a)	[1, 1, 2, 1]	1122	1	9	+0.9	-1.1
	Div. (t)	[2, 0, 2, 1]	1112	-	-	-	-
	N.D.	[2, 0, 2, 1]	1112	-	-	-	-
	Ref.	[2, 0, 2, 1]	1112	-	-	-	-
36	Div. (a)	[1, 1, 2, 1]	1301	0	9	+0.7	-1
	Div. (t)	[2, 0, 2, 1]	1292	-	-	-	-
	N.D.	[2, 0, 2, 1]	1292	-	-	-	-
	Ref.	[2, 0, 2, 1]	1292	-	-	-	-
β = 0.001, t_r = 3							
18	Div. (a)	[2, 2, 3, 1]	3867	-3136	389	-41.5	5.7
	Div. (t)	[2, 1, 2, 1]	5771	-852	9	-12.7	6.9
	N.D.	[2, 0, 3, 1]	6614	-	-	-	-
	Ref.	[2, 0, 2, 1]	7118	689	-185	+7.6	-
36	Div. (a)	[2, 2, 3, 1]	3935	-3227	389	-41.9	5.9
	Div. (t)	[2, 1, 2, 1]	5906	-876	9	-12.8	7.2
	N.D.	[2, 0, 3, 1]	6614	-	-	-	-
	Ref.	[2, 0, 2, 1]	7298	710	-185	+7.8	-

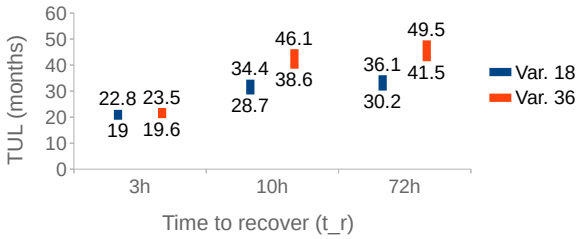
Notes: c_{as}/c_s=1.05, $P(A|R) = 0.5$, Case 1 defense

Fig. 7 shows the ranges in TUL using the two optimization approaches when the cost assumptions are changed. The lower ends of the columns show the TUL when attack impact cost is minimized without considering TUL (Eq. 4), and the top ends are the values with TUL maximization (Eq. 6). The five categories are "SLA-only" with attack costs only arising from SLA penalties, "Const. 18" and "Const. 36" show cases with a constant loss of customers in addition to SLA penalty, and "Var. 18" and "Var. 36" are with the customer loss varying with the disruption. The 18 and 36 in the labels refer to the duration of the customer loss in months. The situation pictured is for $3x$ penalty multiplier,



Case 1; penalty multiplier 3x; p_d = 0.3; ref. alloc. [2,0,2,1].
Top of line: TUL maximization; lower end: A.C. minimization.

Fig. 7. TUL ranges, varying assumptions on customer loss; β = 0.001



Case 1; penalty multiplier 3x; p_d = 0.3; ref. alloc. [2,0,2,1].
Top of line: TUL maximization; lower end: A.C. minimization.

Fig. 8. TUL ranges varying t_r; variable customer loss; β = 0.001

$p_d = 0.3$, $t_r = 3$ and $\beta = 0.001$.

It is clear from Fig. 7 that the TUL estimates are affected by the inclusion or exclusion of added cost estimates. The impact is especially large with the constant loss assumption, as this causes a large cost impact from loss of customers if any level of disruption is experienced due to a cyber attack. This added cost allows large savings on attack incident costs relative to the reference allocation, enabling large TUL values (up to 29.1 and 35.8 months for 18-month and 36-month customer losses, respectively). The impact from the variable customer loss ("Var. 18" and "Var. 36") is not as large, as the extent of disruption is not large due to the 3h recovery time, at the low part of our logistic function (Eq. 1), which keeps the loss of customers small.

Section 6.2.1 of the main document showed that for the case where attack costs are only from SLA penalties, varying t_r only impacted the results if it causes a move between the classes of SLA tolerance and penalty levels. The same is true if other costs are added which are invariant to disruption level, as with the constant loss of customers assumptions, but not when the additional costs vary with the extent of disruption. Figure 8 shows how the results with our variable customer loss assumption are impacted by t_r. The graph has two categories of columns, one for 18-month and another for 36-month loss of customers, with the three categories of time to recover t_r on the x-axis. We observe that when the costs from disruption vary according to our logistic function, the room to benefit from redundancy can increase dramatically if the speed of recovery is slow.

18 Discussion of Potential Limitations

Potential limitations of the modeling shown here:

- The SLA used here only uses one penalty level, not multiple levels – this should still capture the key effects on the diversification choice, but some nuance may be lost. This could be true especially with Case 2 defense, which has more scope for redundancy to mitigate attacks.
- The defensive actions after a detection are not modeled. This is to simplify an already complex model.
- The performance metric considered, average throughput X, may not be of direct interest to an individual customer who cares about their individual experience. It is true that throughput would mainly be of interest to the company providing the services, while a metric such as response rate would be more relevant for a customer. However, estimation of the response rate by simulation models is less accurate, so the study would have required an assessment of the accuracy on this part, which is left for future work. Also, as the results from attacks cause very drastic impacts with complete unavailability of resources, using throughput as the metric should not bias our results much. Additionally, the consideration of the SLA conditions makes the distinction between the metrics less relevant, as a generic, non-customized, SLA could specify a metric which is more in line with the general performance than the case of a specific customer.
- The case study is based on a dated system. We required a case study with an existing published queueing network, to provide a neutral reference point for the modeling. The case study still adequately demonstrates the application and usefulness of our model. Figures may vary significantly from sector to sector or organization to another. We have chosen some figures giving a rationale for them, but we invite organizations to apply this methodology with the parameters they deem suitable.
- The recovery time t_r (per server) fully determines the duration of the disruption, as there is no competition between the defense and attackers on recovery vs further compromises. Such a result is reasonable to the extent that it follows as a consequence of what we believe are reasonable modeling assumptions, although the result is somewhat more extreme in the extent of the impact of t_r than one might expect in practice.
- The expected cost related to attacks is calculated against an amalgam of attack scenarios, with weights assigned to represent relative probabilities of the various alternative outcomes. There are several weaknesses in the assignment of the weights to the different attack scenarios. First, the assignment of the weights can be complicated, as attacker priorities with regard to the system in question can be hard to estimate. Second, the priorities of the attackers could change based on the defense posture. These are problems that warrant extensive study themselves. However, we believe that the use of such a probability framework is beneficial to estimating the benefits from redundancy as part of defense, as it allows comparisons of expected costs which can help guide actions, so the estimation of the weights is useful even if they are not perfect. Further, once estimates of the relative probabilities are assigned, the impact of errors in these estimates can be assessed via sensitivity analysis. Therefore, a model for updating

the probabilities based on defense would also need to reflect the imperfect information of the attackers, or cost estimates might end up being lead by changes in the estimates of attackers' priorities that may have no relation to reality. Given this, we feel it is appropriate to leave the relative scenario weights as exogenous inputs to the model, and to trust the defenders to update the estimates outside of the model if they feel a change in expected attacker behavior has taken place.

- The attack model does not consider attacks where the impact is a slowing down of the servers, which could be due to increased traffic being directed to the servers (DDoS) or malware impacting processing speeds.

REFERENCES

[1] S. Kounev and A. P. Buchmann, "Performance modeling and evaluation of large-scale J2EE applications," in *Int. CMG Conf.*, vol. 11, 2003.

[2] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., 1984.

[3] Ponemon Institute, "The Cost of Denial-of-Service Attacks," Ponemon Institute, Tech. Rep., 2015.

[4] K. Finnerty, S. Fullick, H. Motha, J. Navin Shah, M. Button, and V. Wang, *Cyber Security Breaches Survey 2019*. Department for Digital, Culture, Media & Sport, Apr. 2019.

[5] J. G. Koomey, C. Belady, M. Patterson, A. Santos, and K.-D. Lange, "Assessing trends over time in performance, costs, and energy use for servers," Analytics Press, Oakland, CA, Tech. Rep., 2009.

[6] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "Mitre ATT&CK: Design and philosophy," MITRE Corporation, Tech. Rep., 2018.

[7] K. Bissell, R. LaSalle, and P. Dal Cin, "The Cost of Cybercrime—Ninth Annual Cost of Cybercrime Study," Accenture, Tech. Rep., 2019.

[8] Ponemon Institute, "2018 Cost of a Data Breach Survey: Global Overview," IBM, Tech. Rep., 2018.