Redundancy Planning for Cost Efficient Resilience to Cyber Attacks

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

Abstract—We investigate the extent to which redundancy (including with diversity) can help mitigate the impact of cyber attacks that aim to reduce system performance. Using analytical techniques, we estimate impacts, in terms of monetary costs, of penalties from breaching Service Level Agreements (SLAs), and find optimal resource allocations to minimize the overall costs arising from attacks. Our approach combines attack impact analysis, based on performance modeling using queueing networks, with an attack model based on attack graphs. We evaluate our approach using a case study of a website, and show how resource redundancy and diversity can improve the resilience of a system by reducing the likelihood of a fully disruptive attack. We find that the cost-effectiveness of redundancy depends on the SLA terms, the probability of attack detection, the time to recover, and the cost of maintenance. In our case study, redundancy with diversity achieved a saving of up to around 50 percent in expected attack costs relative to no redundancy. The overall benefit over time depends on how the saving during attacks compares to the added maintenance costs due to redundancy.

Index Terms—cyber security, redundancy, diversity, performance, cyber resilience

1 Introduction

Cyber attacks cost businesses millions of dollars every year, with [1] reporting an average overall annual cost from cybercrime of US$13 million for large international businesses. The same report names the cost of business disruption arising from downtime or unplanned outages as the second largest consequence of cybercrime, costing on average $4 million per annum, or 31% of the total cost related to cybercrime. The economic cost of an attack extends beyond the damage caused directly by the attacker and includes its consequences on the availability of the system taking into account system inter-dependencies and cascading effects. Ensuring resilience to cyber attacks must thus consider investments in redundancy to minimize attack impacts. This requires methodologies that allow analyzing together the attack progression and its impact, to enable decision makers to balance investment to minimize business cost. However, by and large, there is a lack of such methodologies.

We propose a new methodology combining attack modeling with performance modeling to optimize server capacity and minimize the cost of service provision given that attacks occur. We investigate the extent to which server redundancy, with and without diversity, can help improve system resilience during cyber attacks, and thus reduce the business costs from downtime.

We introduce a model to evaluate the expected cost impacts of a set of attack scenarios under different server allocations. Then, we propose two optimization problems for finding the most appropriate allocation for the scenarios considered: one that minimizes the expected cost during the attack, and another that ensures redundancy is affordable during times without attacks. The approach includes an attack model representing the progress of an attacker, with simple modeling of attack detection. Modeling of recovery is used to approximate the duration of the disruption due to the attacks. The attack impacts are estimated using a Queueing Network (QN) model which estimates the performance impact of downtime taking into account interdependencies and cascading effects. Finally, a cost model is used to assign a monetary value to the disruption.

We illustrate our model in an e-commerce case study. Our focus is on selecting server allocations such that the system performance, measured in terms of throughput of jobs, is maintained within the requirements of a Service Level Agreement (SLA). The model is used to find the allocations that balance the cost of redundancy with the costs from attacks, including penalties for breaching the SLA. Beyond the case study considered, the analysis we propose is applicable to broader classes of systems where performance and availability are important, and/or where there are SLAs. This is the case, among others, in transactional systems, supply chains, and manufacturing systems.

The main contributions of the paper are as follows:

1) We investigate the effectiveness of redundancy to mitigate attack impacts, taking the service capacity analysis angle to redundancy and diversity. This differs from previous works on the use of diversity for security such as [2], [3], [4], which focus on identifying which parts of a system to diversify, but without considering overall performance. On the capacity analysis side, where performance modeling is common, our work differs in our focus on malicious attacks. Attacks change the problem considerably, as malicious compromises are not random but follow a correlated approach to maximize damage, so simple redundancy is less effective than against random failures.
2) Attack modeling combined with detailed performance modeling is itself novel. Only a few works such as [5],

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.

[6], [7] and [8] consider both attacks and availability. Further, investigating the impact of multi-step attacks with detailed performance modeling using QNs has, to our knowledge, not been presented in the literature.

3) We introduce a novel metric for the cost-effectiveness of redundant allocations, Time Until Loss (TUL), reflecting the attack frequency required to make a redundant allocation financially viable compared to a reference.

Our findings, summarized below, have wider implications for the resilience of systems, their design, operation and regulation. Whilst some may seem intuitive, the methodology we introduce provides the means to quantify them and to enable organizations to provision their investment in resilience to cyber attacks. The key findings are:

Adding redundancy with diversity can improve system performance during attacks and help organizations avoid SLA penalties. By comparison, redundancy without diversity provides limited benefits.

A key parameter for determining the benefits from a redundancy strategy is the probability of detection of individual attack steps, as this significantly impacts the overall attack success likelihood.

Benefits from redundancy critically depend on whether attacks can trigger SLA compensation and thus losses.

The frequency of attacks and the expected loss determine the long-term financial viability of a redundancy strategy. The benefit from the strategy must exceed the excess maintenance costs due to it.

The SLA structure has a large impact on the appetite to invest in security, other things being equal: it can cap the compensation payable to customers, and thus the financial impact on the company.

We find that redundancy with diversity can reduce the costs arising from cyber attacks if diversification reduces the likelihood of an attack affecting services enough to trigger SLA penalties. However, the benefits from diversification during attacks must exceed the consequent excess maintenance costs incurred over time. Therefore, the final choice of whether to apply redundancy with diversity relies on a balance of various parameters, and the estimated frequency of attacks. Our methodology can aid such decisions.

The rest of the paper is structured as follows: Section 2 discusses related literature; Section 3 introduces our general methodology, after which a running case is introduced in Section 4; A specific instantiation of our methodology onto the running case is shown in Section 5; The evaluation of our method is discussed in Section 6, and Section 7 concludes.

2 Related work

Resource planning and redundancy: We investigate resource allocation planning for networked systems under attack scenarios. The topic has similarities to works using queueing networks for reliability and system performance [9], resource provisioning [10], capacity planning [11], and resource management [12], [13]. The key difference is that we focus on impacts from attacks (as opposed to failures or environmental effects), and how these could be mitigated with capacity planning using redundancy with diversity. Cyber attacks cause a different pattern of component unavailability that requires different modeling. Attackers do not act randomly but are goal-oriented, and likely to target whole services rather than individual components.

Existing works where redundancy is used for security often employ approaches similar to N-version programming [14] for fault tolerance, e.g. in [15], [16] and [17]. Such works use redundancy solely to assess or improve the integrity of an output. By contrast, we consider the use of redundant capacity to maintain system availability during attacks.

Ge *et al.* [7] use redundancy for availability considering the security impact of patching cycles and investigate the effect of different redundancy strategies. Although their work is closely related to ours, key differences exist in the specific problem addressed and the approach. For example, they model attack impacts relying on CVSS [18] impact metrics, which are static, whereas we model impacts on system performance over time. Thus we can evaluate changes to impacts even when vulnerabilities do not change.

Attack graphs: We use Attack Graphs (AG) to model the propagation of attacks. Generally, an AG is a representation of the paths an attacker can take through a system. They have been widely used in research and are used in several products although there are variations in how they are defined and applied. Recent surveys on attack graphs include [19] and [20]. In contrast to some of these, we explicitly consider in our AG model the time it takes an attacker to exploit vulnerabilities, rather than basing the analysis on externally set exploit probabilities. Although this approach has been taken by others who analyze mean-time-to-compromise in attack models, such as [21], [22], [23], [24], these studies focus on evaluating the time required to compromise a system, and do not model impacts and cascading effects in interdependent systems as we do.

Combining attack modeling and performance: Attack modeling has rarely been combined with performance evaluation. For example, while AGs have been used for attack impact evaluation by [25], [26], [27], their impact evaluations have not used detailed performance models, but simpler models of component dependencies. Similarly, while the Möbius tool [28] supports both performance evaluation with Stochastic Activity Networks (SANs) and attack modeling with ADVISE [23], no published works that use Möbius combine these capabilities of the tool to investigate attack impacts on system performance.

Chen *et al.* [5] combine workflow models with security information in an *Argument Graph*, to generate quantitative metrics of how the workflow performs with respect to a security goal. Their model is at a higher abstraction level than our QN-based approach in terms of performance evaluation, and less suited to model bottlenecks arising from downtime.

Some studies have examined the impact of DDoS attacks using QN models. For example, Shen *et al.* [8] investigate the impact of DDoS attacks on web applications using a QN model, and [29], [30] examined how cloud platforms can maintain availability in the face of DDoS attacks. However, DDoS attacks represent a simple special case of an attack as they present themselves as increases in system load, without requiring attacker actions within the system. By comparison, we consider attacks where multiple stages take place inside the system, exploiting its vulnerabilities, and hence require a model of how the attack can propagate within the system.

Diversity: Existing studies on the use of diversity to

increase robustness to attacks (e.g. [2], [3], [4]) focus on optimizing network service diversity to resist attack propagation. By comparison, we determine the optimal number of (diverse) servers to be used for the services provided by a system in order to limit the impacts from attacks, with diversity arising in the different server types used for a given service. Therefore, our work is complementary to the literature on network diversity, not competing with it. Furthermore, network diversity literature optimizes metrics based on the properties of the network structure, various diversity metrics [3] and *'h safety metric'* in [2]. In contrast, we model the performance of a system while under attack, and estimate attack costs under different server allocations, including redundancy with diversity, and optimize costs.

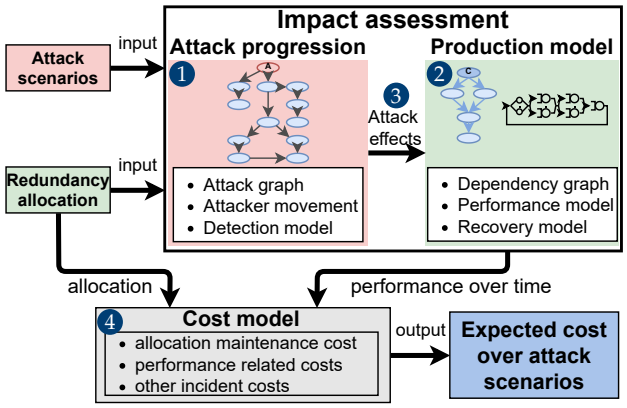
3 Methodology

We propose a methodology to estimate the optimal level of redundancy for production components in a system whose output performance can be affected by cyber attacks. Our approach aims to quantify the cyber resilience [31] of a system, and uses economic costs as the basis for choices over actions. At the core of this method is the ability to quantify the impact of attacks, which we approach from the perspective of the financial costs incurred before, during and after the attack. Both direct and indirect costs are considered, including the costs of preparation (investment into redundant capacity) and the losses due to disruption (loss of system availability or performance).

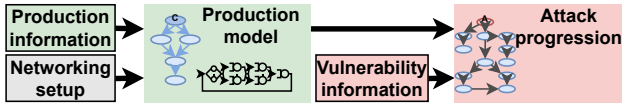
Our approach to impact assessment, summarized in Fig. 1a, integrates a model of attack progression ((1) in Fig. 1a) with a production model of the system ((2) in Fig. 1a), linking the privileges obtained during the attack with the damage they can be used to inflict upon the system ((3) in Fig. 1a). Finally, we evaluate the cost of the attack impacts and redundancy provision ((4) in Fig. 1a).

We model *attack progression* ((1) in Fig. 1a) with an AG, showing the paths an attack can take, i.e., how vulnerabilities can be exploited in successive attack steps to acquire more privileges. To analyze the expected cost of attacks to a system, an estimate of the relative likelihood of different attack outcomes is required. We thus model the detection of individual attack steps, splitting an attack into different outcomes according to the number of steps that succeed without being detected. We assume that once detected, the attack can be contained, but can still disrupt the production system using the privileges obtained by that point. This models the damage that can be achieved even if the attacker does not reach the maximum level of privileges aimed for.

The *production* process ((2) in Fig. 1a) can be modeled using a Dependency Graph (DG) of the different components required to produce the final output, or another production model appropriate for the system of interest. Our running example is a multi-tier application where the output is a web-service, and the production model components are the servers processing requests to the application. We use a DG to visualize the dependencies between components, and a QN model to evaluate *system performance*. The QN modeling enables us to estimate the detailed performance impacts from changes in the system component allocation and from damage due to attacks, which is required for redundancy



(a) Attack cost evaluation and key modeling components



(b) Deriving the models from system information

Fig. 1. Summary of our attack impact analysis approach

analysis. The benefits of using a QN model in our case are described in detail in Section 2 of the supplement document.

To calculate the cumulative loss of production, a major part of the costs arising from cyber attacks, we must establish the *duration* of a disruption and thus need to model how the system can be recovered after an attack. *Recovery* in our model involves both recovering the services affected by an attack, and removing the privileges held by the attacker. Removing the privileges is important, as failing to do so would allow the attacker to disrupt the system again.

The *links between the attack and production models* ((3) in Fig. 1a) occur when an obtainable privilege can be used to disrupt a production component. In our running example below, these are shown as links between the AG and the DG.

Finally, we need to quantify the losses due to the disruption as well as the direct costs (acquisition and maintenance) of the servers and redundancy, i.e. define a *cost model* ((4) in Fig. 1a). In the case study used here, the cost of disruption is based on penalties for failing to meet an SLA, but our method applies with other ways of evaluating production loss due to the disruption, e.g. the cumulative value of transactions lost.

This impact analysis approach is mainly intended for systems where attacks can disrupt the production process of the system. This is because we aim to expose the trade-offs between cyber resilience and costs, and use them to find cost-efficient choices for resilience improvement. Hence, it is best suited for analyses where the availability and integrity of the system are disrupted.

Using this approach to impact estimation, we propose a method to evaluate the optimal level of redundancy for production components in a system. That is, a level of redundancy investment that provides sufficient robustness to mitigate the losses due to attacks while being cost effective.

We focus here on the system design and resource allocation before attacks occur, using redundant capacity to increase the robustness of the system to cyber attacks. As

we consider redundancy planning, the defensive actions of interest are *additions* of redundant servers, with and without *diversity*. Additions take place before the attacks, and redundant capacity is instantly functional when needed. We model diversity by adding "alternative" servers that do not have the same vulnerabilities as the "regular" servers (a regular server refers to the server type that would be the first choice if only one type was used).[1] Thus the redundancy state is described by a "component allocation", which specifies the number of components for each function and whether they are homogeneous or diverse.

Our analysis is conducted over an extensive set of the possible outcomes of a set of attack scenarios, based on which an expected cost of attack impacts is calculated. We use an optimization algorithm to find a cost-efficient component allocation, which uses a genetic algorithm to find the optimal solution. This approach was chosen because we have an integer programming problem with a non-linear objective function.

Fig. 1 summarizes our process for evaluating attack costs, based on which the optimization is done. As shown in Fig. 1a, attack scenarios and a component allocation are input to an impact assessment that determines system performance over time. The cost model is then applied to evaluate the expected attack costs. The optimal component allocation given the attack scenarios is found by optimizing based on these costs. Fig. 1b shows how the production and attack progression models are derived from system information: the production model builds upon knowledge of the components required for production and their connectivity; attack progression is built to describe how the production can be attacked, using the production model and vulnerability information as inputs.

In the rest of the paper we instantiate this methodology in a detailed case study and demonstrate its application.

3.1 Threat model

In this paper we focus on attacks that aim to *reduce system performance* via *targeted multi-step attacks* that compromise system components and disrupt their *availability*, thus impacting the ability of the system to produce its output.

An attack requires a sequence of steps within the system, exploiting vulnerabilities and gaining privileges. Some of the privileges enable the attacker to disrupt system components, and impact production in the system. Disruption to the availability of a component can be achieved in different ways: by damaging the integrity of the software/hardware, by deleting/encrypting data necessary for the component to operate, etc. We do not make assumptions about how this is achieved. Several past security incidents have impaired system production in this way. For example, a 2014 cyber attack on a German steel mill caused breakdowns of control components which resulted in an uncontrolled shutdown of a furnace and damage to the system [33]. Even ransomware attacks' main impact is often on system availability.

We do not consider attacks that only aim to steal data, as they do not directly affect system performance. While their impact on a business can be considerable, their costs can be modeled without a production model for the system, e.g. based on the number of records lost [34]. If data breach losses need to be considered in addition to damages to production, our method can be adapted to add these costs, as explained in Section 6.3.2. We also do not consider attacks where the system is not breached, such as DDoS attacks on the external web-interface of a system. The impact of such attacks can be characterized solely in terms of the availability of the end-point, and the attack steps prior to the impact stage, e.g. recruitment of a botnet, do not take place within the system attacked. Such external steps cannot be detected or acted upon from within the system, and thus do not fit our attack progression and detection modeling.

We assume that the attacker progresses through the system by exploiting vulnerabilities of the system components and that each exploitation of a vulnerability has a time required for exploitation. Intrinsically, vulnerabilities do not need to be software vulnerabilities but can be of any type, i.e. cyber, physical or human, as long as they have a probability of success, probability of being detected, and lead to the attacker acquiring more privileges.

We distinguish between attack steps performed to obtain privileges, and using these privileges to disrupt system production. We assume that an attacker will seek to maximize the privileges obtained before causing disruption, but if detected, will cause what disruption it can with the privileges obtained before detection. We assume that detection leads to containment of the attack, i.e. being stopped from obtaining further privileges. The assumptions we make about attack scenarios are discussed in Section 4.1.

4 Running case

Our case study is based on an e-commerce setting using J2EE services. We chose it as its architecture is representative of widespread service-based systems and its performance model is well understood. The system and workload characteristics we use are based on the queueing model in [35] on a multi-tier application benchmark by SPEC [36].

The basic structure of the QN, its workloads and parameter values are as in [35]. Fig. 2 shows the QN used. Customers (represented by node C) send jobs to the application servers in S_A, which process them and pass them on to the database. The jobs then go through the database processors in S_{DB} before the queries reach the database DB itself, from which a response is returned to the customer. The figure shows that the application and DB processor stations have server multiplicities m_{SA} and m_{SD}, respectively. In our model we identify the appropriate number of each server type for the most cost-efficient performance during expected attacks. This also includes the number of databases in DB.

Details of the QN parameters and the workload are included in the supplementary material, Section 1, including the five different job classes processed in the system, the service demands for the jobs at the three processing components (application server, database server CPU, and database I/O). Our analysis uses 260 concurrent clients, corresponding to the moderate workload case in [35].

While the performance modeling details of the system largely remain as in [35], our case study adds assumptions

1. To check whether servers share known vulnerabilities, databases such as NVD [32] can be used. Extending the model to servers that share some vulnerabilities but not others could be done by e.g. using the similarity between vulnerabilities as in [4].

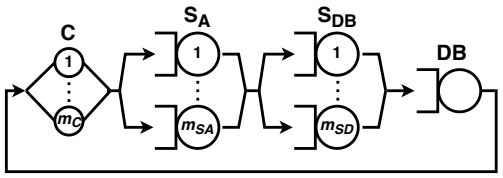


Fig. 2. Queueing network for the J2EE case study

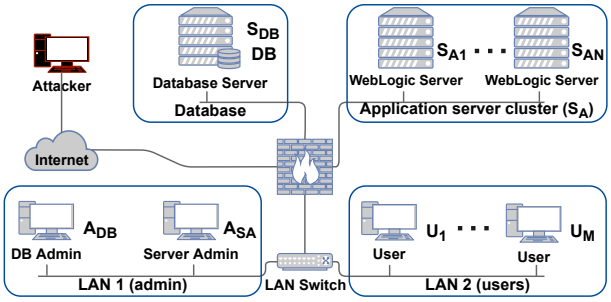


Fig. 3. Network topology for the J2EE case study

about the specific network topology and vulnerabilities present in order to model attacks. The application servers provide clients access to the services over the internet, and the database resides in a separate subnetwork, as shown in Fig. 3. The rest of the network is split into two subnetworks containing administrators and users, respectively.

4.1 Summary of key assumptions

A set of assumptions are necessary to model the behavior of the attacks, their detection, the recovery of the system and the costs. This enables us to reason about the resilience of the system and analyze investments in capacity to withstand attacks. However, no set of assumptions can fully represent all possibilities. When applied in the context of a specific system, an organization and an industry sector, prior experience offers a useful guide on the type of threats, damages and recovery to consider, and the model can customized accordingly. In our case, we have chosen these assumptions with a view of ensuring the consistency and validity of the analysis in the case study. We have also explored the sensitivity to various parameters to ensure that our conclusions hold even if some assumptions are relaxed.

We consider that the attackers have well defined goals and sufficient knowledge of the system to progress towards those goals. This is appropriate, as the technology and typical deployment of the system it seeks to disrupt will be known to the attacker. We also assume that attacks seek to maximize the privileges acquired before launching the disruptive step, unless they are detected (or cannot progress), in which case they immediately cause the disruption they can. Thus, to evaluate the impact of the attacks, we consider a set of attack scenarios that are reasonably expected to cause significant damage, and evaluate the resulting system performance under the different outcomes of those scenarios. We believe this is more realistic than assuming the attack always chooses the shortest path (attackers have different skills, tooling, and preferences) or assuming the attack chooses its next steps randomly (which is both not realistic and computationally expensive). We have chosen a number of scenarios for our case study, shown in Fig. 4 but do not place any limitation on the scenarios considered. Whist this approach may omit some variations of the scenarios, this affects the calculation of probabilities for the different outcomes, but does not change the set of outcomes considered for impact. Therefore, the assumptions could be relaxed by providing a different model for the attack behavior, without affecting the rest of our approach. Finally, we assume that the disruptive actions are taken simultaneously on all servers to which the attacker has privileges. This is reasonable, as causing disruption in steps would likely be detected.

If an attack step was successfully performed without being detected, we assume that it will not be detected later in useful time, i.e., before the attack reaches the privileges it seeks to obtain. We believe this holds in most cases where attacks aim to cause disruption rather than simply to remain embedded in the system. If an attack only seeks to establish permanence and capability, it could be detected at a much later stage, but no disruption would have been caused. As a result, attack detection in our model is based on the latest attacker activity, i.e. the latest attack step attempted. In the sample analysis, the detection probability is assumed equal at each attack step, regardless of where in the system it occurs. This is a simplifying assumption. Although, the methodology could be extended to consider a different likelihood of detection for each vulnerability, determining these values is difficult as they are context specific. Choosing the likelihood of detection randomly is also not appropriate. Once an attack is detected, we consider that the attack is *contained* i.e., that further attack steps in the attack graph are not possible. This may seem a strong assumption but it is reasonable as, typically, compromised systems are either disconnected or quarantined. If an attack is detected we assume the attack executes all the possible disruption steps with the privileges it has acquired so far. This is consistent with the assumption that the attack seeks to cause disruption to the production system.

We assume that the organization can recover compromised servers and purge the attack from the system, e.g. via re-imaging and patching, and do not consider the recovery process to be over until the attack has been entirely removed. This is a practical assumption in most cases, as recovering a system without removing the attack would allow the attack to simply repeat its previous actions. We have considered that recovery occurs simultaneously at all disrupted servers. This disregards that recovery resources may be limited, but ensures our analysis is conservative as it minimizes recovery time. A longer recovery would entail even larger losses and further favor the use of redundancy. Often, in major security incidents additional help will be contracted from external specialists to assist with forensics, attack removal, and recovery. It is thus not possible to model the resources available accurately.

In our cost model, the costs considered in the optimization include server costs (per month, including purchase/rental and maintenance) and losses due to attacks, in the form of penalties for SLA breaches. Other losses can also be considered, as explained in Section 6.3.2, but they do not affect the optimization.

In addition, further assumptions were made on the parameter values used in the case study, but these are example values and do not limit the modeling itself. These are explained under the relevant subsections of Section 5, and their values listed in Table 2 of Section 6. In general such parameter values should be set based on the system and context in which the methodology is applied.

TABLE 1
Privileges in the case study

P_1: U, WebLogic 7	P_5: U, SuSE 8 (AS)	P_9: A, SuSE 8 (DS)	P_{1A}:U, alt. server
P_2: A, WebLogic 7	P_6: A, SuSE 8 (AS)	P_{10}: U, Oracle 9i DB	P_{2A}:A, alt. server
P_3: U, LAN 2	P_7: A, DB Admin	P_{11}: A, Oracle 9i DB	P_{5A}:U, alt. OS (AS)
P_4: A, Server Admin	P_8: U, SuSE 8 (DS)		P_{6A}:A, alt. OS (AS)

U: user; A: admin; AS: application server; DS: DB Server; DB: database; alt.: alternative

5 MODEL

5.1 Background definitions

Attack Graphs: We use Attack Graphs (AGs) [37], [20] as the basis for our attack modeling. Our AG definition is a simple directed graph with system privileges as nodes, and edges as vulnerabilities whose exploits enable an attacker holding one privilege to obtain others. While a more detailed logical AG could be used, it is not required for our analysis.

Definition 5.1. (Attack Graph) *Given a set of system privileges P and a set of vulnerabilities V, an attack graph G is the directed graph $G = (P, V)$, where P is the set of nodes and $V \subseteq P \times P$ is the set of edges.*

Dependency Graphs: A Dependency Graph (DG) describes the dependencies between the components of the production model.

Definition 5.2. (Dependency Graph) *Given a set of system components N, and a relation $E \subseteq N \times N$ with $(s, t) \in E$ meaning that component s depends on component t, a dependency graph D is the directed graph $D = (N, E)$, where N is the set of nodes and E is the set of edges.*

5.2 Attack progression

Fig. 4a shows the AG and the DG for the case study, visualized with connections between the graphs following the approach proposed in [25], [26]. The DG provides a view of the service dependencies for the customer-facing services, with the cluster of application servers treated as one node, as is the database server cluster. The edges from AG nodes to DG identify the system components whose operation can be disrupted if the attacker holds the given privileges. For example, an attacker obtaining the privilege P_2 can disable a server in the application server cluster S_A.

The privileges represented by the AG nodes are listed in Table 1. The vulnerabilities that can be exploited to obtain these privileges (AG edges) are shown as labels on the edges of Fig. 4a, e.g. V_1. For our model, the key is how the vulnerabilities relate to privileges, i.e. what privilege they require and what privilege can they be used to obtain, while the specific vulnerability identifier is not very important. The identifiers of the specific vulnerabilities assumed in the case study are provided in Section 3 of the supplement. The AG is a directed acyclic graph, with three leaf nodes representing the attack targets of interest: P_2, P_6 and P_{11}. Often AGs used for risk assessment have a single goal node. However, in our analysis the AG is used to describe the paths an attacker can take to disrupt services, so a graph with multiple leaf (goal) nodes allows a simple summary of the attack paths through the system. A similar analysis could equally be conducted by using three separate AGs with one goal each.

The network configuration in this case allows two distinct attack methods. First, direct attacks over the Internet to the WebLogic servers in the application servers subnet are possible. These could be traffic to the appropriate application port, exploiting vulnerabilities in the application (V_1 and V_2 in the AG). Second, if an attacker obtains access to the network itself, other attacks become possible. For example, a phishing attack to a machine on LAN 2 can allow an attacker to obtain privileges in the admin machines (DB Admin or Server Admin) in LAN 1, enabling the exploitation of vulnerabilities in the Application server and Database subnets. The network is configured so that administrator machines can be remotely accessed via SSH (e.g. for remote access with a VPN), and thus have port 22 open to traffic from within the network. The computers have a vulnerability enabling an attacker to obtain privileges on the admin machines, which then allows compromising the application servers or the database.

The diversified case: To investigate the effect of implementing *redundancy with diversity*, we evaluate the impact of adding *alternative* application servers with a sufficiently different configuration that they do not share vulnerabilities with the *regular* application servers otherwise used. While the term *diversity* can be used in regard to different system aspects, in our case it simply means that the components are sufficiently diverse not to share vulnerabilities. That is, the vulnerabilities in the alternative servers are not the same as those in the regular servers.

This is shown in Fig. 4b, where a darker color is used in nodes where servers provide redundancy with diversity to existing application servers. The DG has an additional node S_{AA}, representing these added servers which provide the same service as those in S_A but are of the alternative type. Similarly, the AG has additional privilege nodes relating to servers in S_{AA}. These privileges are obtainable via vulnerabilities that differ from those in servers in S_A, but form similar patterns of attack.

With diversity, the attacker requires two different exploits to acquire privileges in all the servers providing a service, rather than a single one. To model the added attack difficulty due to requiring a further exploit, we set a probability that the attacker has the capability to exploit one of the vulnerabilities, the other, or both. An attack scenario to a diversified service is thus split into three cases, as the case of not having capabilities to exploit either is disregarded. We assume that the probability that an attacker can exploit a given vulnerability is independent from the probability of them being able to exploit a different one. In our case study we also assume, for simplicity, that these three cases of capabilities are equally likely to occur in attacks on a system that is diversified. However, we have also examined the impact of relaxing these assumptions; see Section 16 of the supplement for the results.

(a) AG and DG for the J2EE case study (b) Diversi ed AG and DG in the case study (c) Attack scenarios in the case study

Fig. 4. AG and DG in the J2EE case study, with attack scenarios

Attack steps: An attack consists of "move steps" and "disruptive steps". In a move step the attacker moves in the AG by exploiting vulnerabilities to obtain privileges. After obtaining a privilege, the attacker has the ability to take a disruptive step to make a server (or servers) unavailable. This is represented with the dashed lines from AG nodes to DG.

In our case study the privileges relate to speci c server instances, so a copy of a privilege applies to one server copy, not several. For example if there are two servers in S_A, the attacker requires two copies of the privilege P_2 (or of P_6) to disrupt both servers in S_A.

Time-to-exploit: The time taken by the move steps of an attacker is determined based on estimates of the time it takes to exploit the vulnerabilities required. In general, these would be based on the capabilities, skills and system knowledge of the attacker. As we are concerned with attacks on system performance rather than objectives that require attackers to persist in the system, we assume that attackers take steps as fast as their abilities allow, i.e. steps take t_r time, and there is no delay between steps.

Based on [21], [22], [24], the time to exploit a vulnerability is in the order of hours or days. In the running case, we use the following values for time to exploit: 24h for the initial exploit of each vulnerability; 6h for using the same exploit elsewhere in the network. Once the attacker has obtained a privilege on a server, the time to replicate the exploit on other servers of the same type is 4 hours.

In contrast to move steps, disruptive steps are assumed instant, so take place immediately after the attacker has obtained the privileges required for its goal. Disruptive steps are launched simultaneously from all privileges obtained.

5.3 Attack detection

We make three assumptions in our detection modeling. First, intrusion detection acts on the latest activity, so detection can only occur in an AG node where the attacker is currently acting; thus not taking into account delayed detection of earlier attack steps. Second, detection happens with equal probability at every node. Third, if an attack is detected during a move step (before disruption), the defense can and will stop it from moving further along the AG, but not from disrupting the system using the privileges already acquired. Whilst the system could be modeled in more detail, these assumptions simplify the model and allow us to focus the explanations and illustrations on the resilience aspects rather than the modeling.

The detection affects the probability of an attack reaching its intended goal, and leads to partial attacks. In these cases, we assume that the attacker launches disruptive steps from the privileges it holds, simultaneously from all the nodes.

This detection model leads to a simple solution for the relative probabilities of the different outcomes of an attack. For an attack with a full length of n steps, the probability of complete success is $(1 - p_d)^{n-1}$. Any sub-path with length $m > 0$ steps occurs at the probability $(1 - p_d)^{m-1} p_d$. In the outcomes yielding partial paths, the attack is contained at the point where it was detected.

Defense after detection: We consider two cases for analysis. In Case 1, attacks will impact all servers of the same type in a cluster if they compromise one of the servers, i.e. the defense cannot stop the attacker between repeating the exploits in identical servers. This re ects situations where stopping the attack requires countermeasures which themselves cause unavailability in other servers of the same type, e.g. blocking connectivity to a part of the network. In Case 2, it is possible to stop the attack between exploits of the same vulnerability on different server instances. Thus, when an attack step is detected, the attacker becomes unable to obtain further privileges.

Attack scenarios in the case study: We consider three possible attack scenarios, and their respective partial success outcomes. The scenarios considered, shown in Fig. 4c, are:

> Scenario 1: attack to application servers over the internet (target: P_2); [$A \to P_1; P_1 \to P_2$]
> Scenario 2: attack to DB, as direct as possible (target: P_{11}); [$A \to P_3; P_3 \to P_7; P_7 \to P_8; P_8 \to P_9; P_9 \to P_{11}$]
> Scenario 3: attack to both the application servers (using privilege P_6) and DB, reusing vulnerabilities V_4 - V_6 that occur along both paths (targets: P_6 and P_{11}); [$A \to P_3; P_3 \to P_4; P_4 \to P_5; P_5 \to P_6; P_3 \to P_7; P_7 \to P_8; P_8 \to P_9; P_9 \to P_{11}$]

The relative weights given to these three scenarios should, in practice, be based on their likelihood of occurring. Such estimates could made based on past experience of the company or the industry sector. To simplify, we have chosen to assign equal relative probabilities to the three scenarios.[2] The weights of all the sub-paths (attack outcomes) within a given scenario must sum to the weight for that scenario,

2. We investigated the sensitivity of our results to this equal-weight assumption in the supplementary material, Section 12. Although the expected cost values change, the results in terms of the optimal choice of server allocation remain stable, changing to a different choice only in cases with extremely uneven weighting where no weight was given to one of the scenarios. Thus the equal-probability assumption is not affecting the overall characteristics of our results.

1=3 in our case, so that the total weight across all attack outcomes is $W = \sum_{M \in A} w_M = 1$.

5.4 Recovery modeling

When an attack has been detected, the defense will recover the servers affected, and purge the attacker from the system in t_r time. This recovery time consists of the time to reboot servers, and the time to purge the attacker from the system. Failing to purge the attacker would likely lead to a re-occurrence of the impact and a longer overall downtime. While server reboot times are fast, in the order of seconds or minutes, purging the attacker requires detailed analysis and could take considerably longer, in the order of several hours. Thus our recovery times here are based on estimates on recovery from attacks, instead of server reboot times.

We combined findings from cyber security reports [38], [39] to arrive at two recovery time assumptions to use in our model. Based on a report by Ponemon Institute on DoS attacks [38], we find an estimate of 3h for the average downtime from a DoS attack. However, the attacks in [38] likely consist of cases where the attacker is flooding the servers with requests, while we are mainly interested in the service availability impact of attacks penetrating the network of a company. Thus, we use a second value to represent the recovery time from more general attacks, 72h, based on a finding by [39] on recovery from attacks that had an economic impact. Specifics of how we derived these two figures are provided in Section 7.1 of the supplement. These values are approximations, used to illustrate the approach. The recovery times could vary across different systems and attacks, based on factors such as vulnerability types, patch availability, and manpower capacity. Consequently, the values applied should reflect the application context. In practice, values could be set based on previous experience, or blue-team exercises.

5.5 Performance modeling

We assess the performance of the production model using QN models, solved using the LINE tool [40], [41]. For this purpose, the impact of an attack is expressed in terms of the availability of servers in the nodes of the production model. Impacts are evaluated for the different outcomes of the attack, including partial successes caused by detection.

The availability impact of a specific attack outcome is represented by a matrix M_{attack}, specifying the numbers of active (available) servers in different production nodes across the different stages of impact from that outcome. That is, M_{attack} is a $I \times J$ matrix with elements $m_{i;j}$ specifying the number of active (available) servers at stage i of the impacts of attack outcome, in node j of the performance model. The total number of rows I is the number of distinct impact stages that occur in response to a specific attack outcome, and the number of columns J is the number of production nodes in the system for which redundancy is considered. The impact stages (rows) start from the normal system state, followed by the states reached after an attacker has taken disruptive steps, and those after defensive actions were taken. The last row is the state that prevails when the time horizon t_h is reached. For example, in our diversified system with four server types, with the server allocation $[m_{SA}; m_{SAA}; m_{SD}; m_{DB}]$ the attack outcome where two servers in S_{AA} are disrupted is expressed as:

$$M_{attack} = \begin{pmatrix} m_{SA} & m_{SAA} & m_{SD} & m_{DB} \\ m_{SA} & m_{SAA} - 2 & m_{SD} & m_{DB} \\ m_{SA} & m_{SAA} & m_{SD} & m_{DB} \end{pmatrix}$$

where: row 1 is the initial state; row 2 shows the available servers after two servers in S_{AA} have been disrupted; and row 3 represents the situation after recovery.

The performance evaluation for an attack outcome uses the rows of M_{attack} as server multiplicity inputs to a sequence of QN models used to estimate performance across the impact stages. The duration of the impact stages (rows of M_{attack}) is represented by the column vector $_{attack}$, whose values are computed from the time-to-exploit and time-to-recover estimates. For the example above, the related duration vector is $^{0}_{attack} = [56; 3; 661]$. The effect from the attack materializes after 56 hours as it requires an exploit to obtain a stepping-stone privilege (24h), another to obtain the privilege enabling disruption of the first server (24h), and further two exploits to obtain these privileges on another server copy. These repeat exploits are assumed to take only 4h each as they are re-exploits of the same vulnerabilities as for the first server. After this the disruptive action is launched. The second state is active for $t_r = 3$ hours before recovery of the services takes place, and the final state stays in effect for the time remaining until the time horizon, which is 720h (one month) here, so $t_h - 59 = 661$ hours.

The number of stages in M_{attack} depends on the attack and recovery models. In our case study, all attacks yield one impact stage as all disruptive steps are taken in one go, and recovery occurs simultaneously in all servers requiring it. Thus we require a total of three stages including the starting and final situations. If the attacker caused disruption in several steps instead of simultaneously, the matrix would contain one stage for each change in the number of active servers. However, this is not effective attacker behavior in general, as it increases the likelihood of detection before the objective is reached. Similarly, recovery done in multiple steps would add stages. However, it would not lead to a marked difference in results as the key performance impact from recovery is the time to return to the required performance. We capture this time with the parameter t_r.

5.6 Costs

The costs relating to the provision of a service comprise the direct costs of server capacity (monthly per-unit costs), and penalty costs from failing to meet SLAs. The capacity cost is:

$$_{i} = c_i \; x_i + c_{a;i} \; x_{a;i} \qquad (1)$$

where x_i is the number of regular servers used for a given service (or cluster) i and c_i is the monthly per-unit cost for a regular server (equivalent monthly cost, including server purchase/rental and maintenance costs). The alternative servers used for diversification are assumed to have a higher cost $c_{a;i}$ per server, and $x_{a;i}$ is the number of such servers. Fixed costs of capacity and diversification do not feature in (1), as they do not impact the optimization. They can be added afterward if an overall cost-benefit analysis is desired.

Our SLA has only one credit level, for 100% service credit. The SLA breach condition is: 100% service credit if $X(t;x;M) < 0.9\ X_{ref}$ for a disruption time greater than in a month. Here, x is a server allocation vector, M is an attack outcome matrix, and $X(t;x;M)$ is the system throughput at time t observed when server allocation is x and the attack outcome M. X_{ref} is the reference throughput that has been marketed to the client (the published performance), and $X_{req} = 0.9X_{ref}$ the required throughput. The 90% limit is as in the performance SLAs in [42]. is the limit value (as a share of time in a month) that performance can be below X_{req} before the SLA is breached. The SLA penalty costs:

$$SLAp(x;M) = 1_{DT(x;M)>} \quad N \quad cc \tag{2}$$

where $DT(x;M)$ is a disruption-time share (defined below), is as described above, N is the number of clients, and cc is the client charge for the overall service. The equation states that a penalty of N cc is to be paid if $DT(x;M) >$, otherwise the penalty is 0. $DT(x;M)$ is defined as:

$$DT(x;M) = \frac{\sum_{s=1}^{t_h S} 1_{X(s;x;M)<X_{req}}}{t_h \quad S} \tag{3}$$

where $s \in S$ is a sub-unit of time used in the performance modeling, with each time unit t split into S fractions. $DT(x;M)$ is the share of time during which throughput is less than the required throughput, $X_{req} = 0.9X_{ref}$, out of all time periods up to the time horizon t_h. The time horizon is set to the SLA service commitment period, one month.

Costs in the case study: We assume a server operating cost of \$185 per month for regular servers. Detail on how this was derived is given in Section 7.2 of the supplement. 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. With 260 concurrent client requests this penalty equates to \$13 000.

5.7 Cost minimization over attack scenarios

The optimization problem for minimizing costs over multiple attack scenarios is complex due to the non-linearities arising from the SLA penalties, as these are estimated using performance modeling. Our optimization problem is:

$$\begin{aligned} &\min_x (c^T x + \sum_{M \in A} w_M \quad SLAp(x;M)) \\ &\text{s.t.} \\ &\quad \text{variables in } x \text{ are non-negative integers} \end{aligned} \tag{4}$$

where c is a vector of cost coefficients applying to the variables in the server allocation vector x; M is an attack outcome matrix in the set A of attack outcome matrices for all attack scenarios considered; $SLAp(x;M)$ is the SLA penalty in attack outcome M for allocation x; and w_M is the weight assigned to attack outcome M, with $\sum_{M \in A} w_M = 1$.

As apparent in (4), the SLA penalty cost is dependent on the average throughput level from the attacks, which itself depends on the variables to be optimized. This cost term must be estimated during the optimization. Thus, we cannot use solvers for standard MILP problems, and rely instead on a genetic algorithm solver (here, MATLAB's ga solver).

5.7.1 Accounting for excess maintenance costs

The cost optimization problem (4) does not consider potential added maintenance costs during times without attacks. We therefore account for the excess maintenance costs with a novel metric that measures the time until maintenance costs exceed the benefits from the redundant allocation (relative to a reference allocation). We call this new metric Time Until Loss (TUL) and measure it in months for our case study as this coincides with the SLA commitment period. TUL is the longest time between attacks for which the benefits of the redundant allocation exceed the additional costs involved. If attacks are less frequent than the TUL estimate for an allocation, that allocation is not worth investing in (the reference is more cost effective). For a given server allocation vector x, TUL is calculated as:

$$TUL(x) = \frac{(x_{ref}) \quad (x)}{c^T x \quad c^T x_{ref}} \tag{5}$$

where $(x) = c^T x + \sum_{M \in A} w_M \quad SLAp(x;M)$, and x_{ref} is the server capacity vector for the reference allocation. The numerator includes the overall cost during a month with an attack, while the denominator only concerns the maintenance costs for the allocation x.

Whilst we can estimate TUL for the optima to the problem in (4), that optimization may not yield results that ensure the most favorable values for TUL as it focuses on the costs during attacks, ignoring the longer term. Thus, we form an alternative optimization problem, max TUL:

$$\begin{aligned} &\max_x \quad \frac{(x_{ref}) \quad (x)}{c^T x \quad c^T x_{ref}} \\ &\text{s.t.} \\ &\quad \|x\|_1 > \|x_{ref}\|_1 \\ &\quad \text{variables in } x \text{ are non-negative integers} \end{aligned} \tag{6}$$

The condition $\|x\|_1 > \|x_{ref}\|_1$, where $\|x\|_1$ is the l_1-norm, states that the optimization considers only allocations with more servers than in x_{ref}, i.e. with added redundancy.

5.7.2 Speeding up the optimization

As the evaluation of the queueing model is done as part of the objective function, we speed up the optimization by multi-stage optimization and by using 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. This evaluation is fast, and is used to run the optimization across a large problem space to identify a region to focus on with the more expensive evaluation in the second stage. In the second stage we estimate the throughput of the model in the reduced search space using the fluid solver provided by LINE [40]. In this stage we also employ memoization to reduce the number of times the queueing model is solved, as some configurations of the QN reoccur often during the optimization. More detail is included in Section 8 of the supplementary material.

6 Evaluation

Our evaluation focuses on the optimal choices for redundancy in our running case. Sec. 6.1 discusses the results using our baseline values for model parameters; Sec. 6.2

TABLE 2
Parameter values and sensitivity ranges

Parameter name	Parameter description	baseline value	sensitivity range
t_r	recovery time	3 (hours)	{3;5;10;72}
p_d	detection probability	0.3	[0;1]
c_s	regular server cost	$185	$185
c_{as}	alt. server cost	$1.05 \cdot c_s$	$[c_s; 10 \cdot c_s]$
X_{req}	required throughput	$X_{req} = 0.9 \cdot X_{ref}$	-
	SLA disruption-time limit	0.001	{0.001, 0.01}
customers*		260	-
penalty cost		$50 per customer	{50;100;150}

*Each customer has continuous demand of the system, i.e. there are 260 jobs circulating in the QN.

examines how the different parameters impact the results; Sec. 6.3 investigates long-term maintenance costs using our TUL metric, and costs from reputation loss. Table 2 lists the parameter values and their ranges.

6.1 Baseline results

Fig. 5 shows the optimal server allocations in the "standard" non-diverse case (Fig. 5a), and with diversification (Fig. 5b). The optimal server allocation is given as a tuple, e.g. [2,0,3,1], where the first element denotes the number of application servers (in DG node S_A) of the regular type, and the second element with alternative servers (for diversity). The third and fourth elements are the numbers of database servers (in S_{DB}) and databases (in DB), respectively.

The figures depict the breach metric value for a given attack outcome with blue bars (left y-axis) and the costs during the outcome using orange bars (right y-axis). The number of outcomes depends on the number of exploits that can be detected, which increases with the number of servers in the allocation, and with diversification. In Fig. 5a there are 14 different outcomes to the three attack scenarios, while the diversified case in Fig. 5b leads to 59 outcomes.

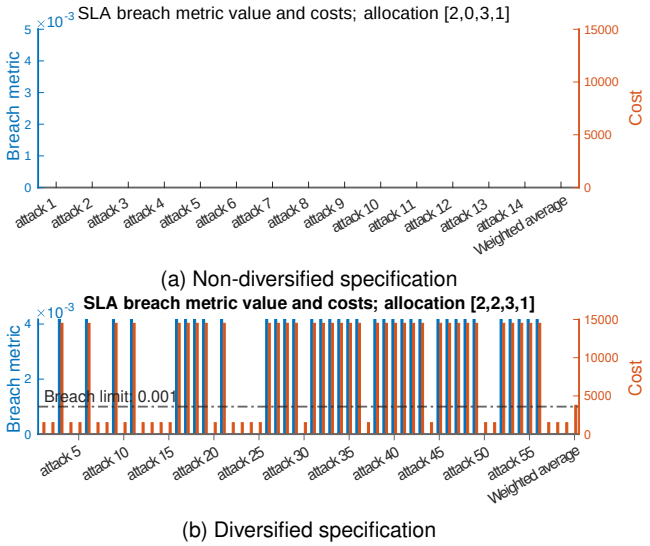
Diversification helps bring down the expected cost of the attack scenario by around 40%. While under the non-diversified configuration only three of the attack outcomes (21%) avoid an SLA breach, diversification leads to 24 outcomes (41%) without a breach. Although the outcome likelihoods are not equal but vary as explained in Sec. 5.3, this pattern is still indicative of how diversification mitigates costs.

6.2 Sensitivity to parameter changes

6.2.1 Time to recover

The relationship between the time to recover servers (t_r) and the level of disruption tolerance in the SLA ( ) is a key determinant of the effectiveness of redundancy strategies. As attackers cause a disruption only when they have acquired all the privileges they require, the speed of recovery determines the duration of disruption. Whether this duration is long enough to breach the SLA depends on t_r in relation to .

Fig. 6 shows the effect of different recovery times for a given server allocation, other parameters being kept constant at their baseline levels. The figure shows how the impact of one specific attack outcome changes when t_r is varied among $\{3,5,10,72\}$. The figure format is similar to Fig. 5, but focusing on one attack outcome (full success of



(a) Non-diversified specification

(b) Diversified specification

Fig. 5. Optimal allocations in the case study, baseline, = 0:001

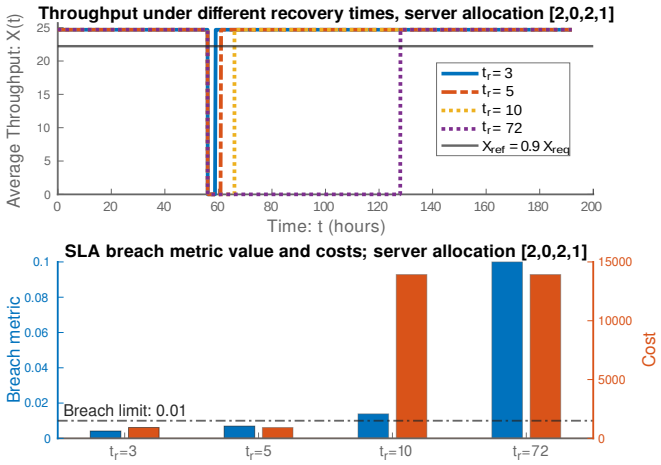


Fig. 6. Impact of varying recovery time t_r in the case study

Scenario 1) and varying t_r. We see that the higher recovery times of 10 and 72h cause breaches of the SLA. Note that, as the SLA penalty does not vary with the extent of the disruption, once the SLA is breached the duration of the recovery no longer affects the cost.

Table 3 shows results when t_r varies between 3 and 72 hours. The table considers two SLA disruption limits: up to 1% of the 1-month SLA window (7h 12min) in columns three to five (= 0.01), and up to 0.1% of the time window (43min) in the last three columns (= 0.001). It shows the optimal allocations for each recovery time in both the diversified (Div.) and non-diversified (N.D.) configurations. For each allocation, the expected overall cost is shown along with the cost savings obtained from diversification relative to the optimum in the non-diversified case. Due to the higher cost of the alternative servers, the allocation cost in the diversified case is often higher than in the non-diversified one. However, the non-diversified configuration has a lower overall expected cost than the diversified one only when = 0.01 and t_r is 3 or 5. In these cases, the disruption from the attacks is not large enough to breach the SLA. In contrast,

TABLE 3
Sensitivity to recovery time t_r

		= 0.01			= 0.001		
t_r	Alloc. type	Optimum allocation	Exp. cost	% diff.	Optimum allocation	Exp. cost	% diff.
3;5	Div.	[1; 1; 2; 1]	934	+1.0	[2; 2; 3; 1]	3795	-41.1
	N.D.	[2; 0; 2; 1]	925	-	[2; 0; 3; 1]	6447	-
	Ref.	[2; 0; 2; 1]	925	-	[2; 0; 2; 1]	6931	+7.5
10;72	Div.	[2; 2; 3; 1]	3795	-41.1	[2; 2; 3; 1]	3795	-41.1
	N.D.	[2; 0; 3; 1]	6447	-	[2; 0; 3; 1]	6447	-
	Ref.	[2; 0; 2; 1]	6931	+7.5	[2; 0; 2; 1]	6931	+7.5

Notes: p_d = 0.3, c_{as}=c_s=1.05

when the recovery time is longer or the disruption limit is lower, the attacks cause an SLA breach. In these cases, *diversification provides savings of 41.1% in the cost of the attack relative to the non-diversified optimum*, and 45.2% relative to the reference.

Note that the results obtained are the same in all cases where the SLA is breached. This is due to: a) the compensation being based on credit levels instead of varying in proportion to the extent of performance reduction, and b) the SLA having a single credit level. Thus t_r and only impact whether the contract is breached or not, not the extent of the penalty. Often, in real cases, SLAs can have multiple credit levels, so the penalty increases in steps until the maximum credit level is reached. After that, higher disruption duration no longer increases the penalty, as in our results.

The key finding from Table 3 is that the benefit from redundancy depends on whether the speed of recovery is faster than the SLA disruption tolerance . When t_r is short enough to be within the disruption tolerance, diversity is not beneficial as no penalty is incurred. When t_r is longer than the disruption tolerance, a penalty is possible under some attack outcomes, and redundancy can yield cost savings. Moreover, diversification (Div.) provides benefits beyond those from redundancy without diversification (N.D.). Diversity gives greater flexibility to reduce the expected penalty costs by avoiding the penalty in some cases, and by reducing the probability of successful attacks. Thus, when planning for resilience, attack impacts can be reduced by improving robustness to the attack by adding redundancy with diversity, or by improving recovery speed. Alternatively, if the company can insist on an SLA with a loose disruption tolerance , there is no incentive to invest in mitigating attack impacts, as attacks do not breach the SLA.

6.2.2 Detection probability

The probability of detection of attack steps p_d affects the relative likelihoods of the attack outcomes. As such, it only impacts the optimal allocation if some attack outcome breaches the SLA. If an optimal allocation can avoid penalties altogether, varying p_d has no impact on the optimum. We investigated the impact of p_d in both Case 1 and Case 2, as described in Section 5.2. Here we focus the discussion on Case 1, whilst Case 2 is described in Section 10 of the supplement. As previously discussed, the values of and t_r determine if a penalty is possible. Thus, we discuss only cases when an SLA breach can occur; the no-penalty cases always favor keeping the reference allocation.

TABLE 4
Sensitivity to detection probability p_d, Case 1 defense.

Detection prob. (p_d)	Alloc. type	Optimum allocation	Exp. cost	Cost difference		% diff.
				penalty	alloc.	
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, $P(AjR)$ = 0:5, = 0:001, t_r = 3, Case 1

The results for the Case 1 defense model are shown in Table 4, which for each p_d (1st column) gives the results for three types of allocation: diversified optimum (Div.), non-diversified optimum (N.D.), and a reference allocation (Ref.) that has not been optimized. Column 2 gives the respective allocation type, whilst column 3 shows the optimal server allocation tuple. The expected costs are shown in the 4th column, whilst the 5th and 6th columns show cost differences relative to N.D. arising from penalty and allocation costs, respectively. Finally, we show the percentage difference of the expected cost relative to the optimum without diversification (N.D.) in column 7.

Table 4 shows that there are no benefits to adding identical redundant servers to a cluster which can be a final target of an attack, which is the case for S_A and DB. However, there is benefit to redundancy for intermediate servers, which are only used as a stepping stone in the attack. For example, for server S_{DB} one redundant server ensures two functional copies remain during an attack.[3] Consequently, when the attack scenarios can lead to SLA penalties, having three DB servers is optimal when p_d 2 $f0{:}1; 0{:}3; 0{:}5g$. Additionally, when diversification is beneficial, the optimal allocation is one where there are as many alternative servers as regular ones, thus ensuring that the alternative servers cluster can meet the performance requirements on its own.

To summarize, *redundancy with diversity is beneficial whenever SLA penalties cannot be fully avoided.* Additionally, the probability of detection affects the benefits obtained from redundancy. *A higher detection probability reduces the expected costs from attacks, and thus the benefit obtained from redundancy.* This suggests that companies could benefit from increasing their detection capabilities when cost effective, although this is difficult to achieve in practice due to the cost of dealing with high levels of false positive alerts.

6.2.3 Cost of diversification

We analyzed the sensitivity of our results to different levels of diversification costs, varying the c_{as}=c_s ratio, i.e. the per-

3. This is because the attacker needs only one server as a stepping stone, and will not risk detection to obtain additional ones.

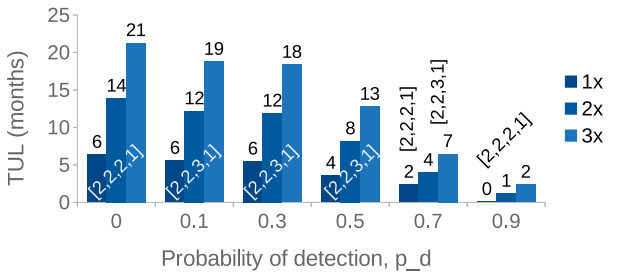


Fig. 7. Months until cumulative maintenance cost exceeds attack-time benefit (TUL). Optimal diversified allocation shown as label. The bar colors correspond to different multipliers on the baseline SLA penalty.

unit costs of the alternative servers relative to the regular ones. The detailed results are shown in Section 11 of the supplement. In summary, we find that the level of diversification cost does not impact which allocation is found optimal unless the cost difference between the server types is large, which in our analysis occurred once the cost ratio c_{as}=c_s reached 9, i.e. when the alternative server is 9 times as expensive as the regular type. However, the benefit from diversification is affected, as the excess maintenance costs of the alternative servers need to be balanced against the benefits observed during attacks. The impact of costs over time is explored further below.

6.3 Long-term maintenance costs

The analysis thus far has considered the expected cost during an attack, but not the costs during normal operation when no attacks occur. However, choosing a redundant allocation over one that is cheaper depends on the expected attack likelihood/frequency, and the excess maintenance cost. Thus, the choice depends on the expected cost of the allocations over a longer period mostly spent without attacks. The key parameter in this evaluation is the frequency of attacks in the chosen time frame.

We estimate the longer term benefit of the redundancy approaches by estimating the longest time between attacks before the cumulative excess maintenance costs exceed the benefit during attack periods. For this we use the TUL metric we introduced in (5). Its advantage is that it does not require an estimate of the frequency of attacks as model input before analysis.

The TUL value of an allocation is sensitive to the various model parameters, the key ones being the penalty for breaching the SLA and the reference allocation. Fig. 7 shows how TUL varies with p_d and the SLA penalty. We can see that higher p_d correlates with lower TUL because the benefit from diversity gets smaller with p_d, as shown in Sec. 6.2.2, reducing the numerator in (5). Similarly, higher levels of penalty correspond to higher TUL estimates. This is because the increased penalty makes the expected loss from an attack higher, which provides greater scope for cost savings from diversity, and increases the numerator of the TUL.

The reference allocation has a sizable impact on TUL, because of its effect on the excess maintenance cost in the denominator of (5). We tested this by varying our reference allocation. In our case study, the minimum-cost allocation yielding the required performance during normal times is $[2; 0; 2; 1]$. However, the baseline used by [35] was $[3; 0; 2; 1]$. Using the latter as the reference leads to significantly larger TUL estimates. Because of this, *organizations must choose carefully their reference point*: the cheapest allocation may not reflect the need for redundancy and may bias the results.

To put TUL values into perspective, [43] found that there were on average 1.37 successful attacks per year with significant ("very high profile") impact and 5.94 with moderate impact, for large international companies who were not among the top security performers. This amounts to *one significant breach every nine months, and a moderate breach every two months*. For top security performers, moderate breaches occurred once every 13 months, while significant ones were rare (0.05 per year). This shows that attack frequencies vary across companies and having a reasonable estimate is important to determine if server diversification is worthwhile.

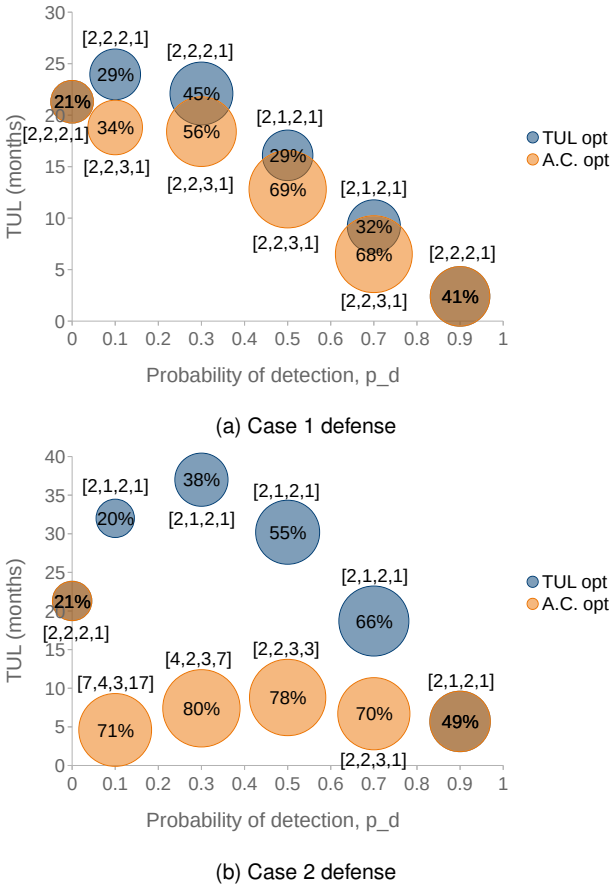
6.3.1 TUL optimization

The results in the previous sections aim to minimize cost during attack periods i.e., during the month of the attack. However, this ignores the excess maintenance paid during months when attacks do not occur. This is why Sec. 5.7.1 introduced a second optimization problem: maximizing the TUL metric. This section focuses on how the results from maximizing TUL differ from minimizing attack incident costs. Apart from the differences pointed out here, the results for the two optimization problems are qualitatively similar and thus shown in the supplement, in Section 14.

The comparison of the two optimization approaches is shown in Fig. 8. The key observation is that TUL maximization typically leads to the use of fewer servers than attack cost minimization, as this saves on maintenance costs and allows for a higher TUL. For Case 1 defense, in Fig. 8a, the difference in TUL is 5.2 months for $p_d = 0.1$, and decreases to 2.7 months for $p_d = 0.7$. This increased TUL comes with a higher during-attack cost, with the saving relative to the reference allocation reduced by 5-40%-points compared to attack cost optimization. In general the two optimizations show the extremes between which a suitable approach can be chosen based on a desired TUL level or preference for savings during attacks.

The results for Case 2 defense, in Fig. 8b tell a similar tale, but with more extreme differences between the two approaches. This is due to the defense being able to stop attacks between exploits within clusters of servers of the same type. In this context, attack cost minimization leads to very high redundancy when the detection probability is small but non-zero. This results in large attack cost savings but limited TUL. The TUL maximization approach finds optimal allocations with an even lower number of servers than with Case 1 defense, with a higher TUL but often lower attack cost savings than in Fig. 8a.

Taking the two subfigures together, we see that the extent to which benefits from diversity can be realized depends on how frequent attacks are expected to be. If the detection capability is very high, $p_d = 0.9$, diversity is only beneficial if attacks are very frequent (on average every 2.4 months for Case 1, and 5.7 months for Case 2). Such frequencies may appear excessive, but do occur in some cases [43]. With



(a) Case 1 defense

(b) Case 2 defense

Fig. 8. TUL and cost savings when minimizing attack costs (A.C. opt, orange bubbles) or maximizing TUL (blue). Bubble sizes represent attack cost saving relative to the reference allocation [2,0,2,1]. Labels show the server allocation. The results show are for SLA credit level of $150 per customer (3x multiplier), with = 0:001 and t_r = 3.

lower levels of detection, a high TUL is possible, so diversity is viable for a wide range of attack frequencies. Moreover, when the expected attack frequency is between the TUL values provided by the two optimization approaches, there is scope to choose an allocation between the two optima, trading-off some TUL for added cost savings during attack.

6.3.2 *Other costs from attacks*

Underestimating the costs from an attack will lead to underestimating the TUL and thus under-investment in redundancy. We have considered that attack incident costs arise only from SLA breaches. However, costs related to regulatory penalties or business lost due to impacts on reputation can behave very differently. First, such added costs can occur after an attack even if the SLA is not breached. Second, these costs can increase with the length of disruption even if the SLA penalty is capped. Third, even if these costs occur only when an SLA is breached, they would add to the cost of the incident, so excluding them would bias TUL down.

We investigated the impact of including the cost of lost business in our model; the analysis is in the supplement, in Section 17. The key results are: a) *Including loss of business costs can increase TUL estimates significantly*: in our tests, increases were up to 63% when a constant loss of customers (1.7% of customers for three years) was assumed, and up to 124% when loss of customers varied with disruption length; b) *If business loss varies with the length of disruption, speed of recovery has a large impact*: the TUL increase ranged from 3% with t_r = 3 up to 124% with t_r = 72.

6.4 Scalability

Earlier on, we have illustrated our method and evaluated its output on a small scale case study. We built our case-study on this particular system as it is well known [36], and its performance characteristics and model are understood and validated [35]. It is difficult to implement much larger case studies because the attack and system information required to build attack graphs are usually confidential and not publicly available. However, our method can scale in practice to much larger systems for reasons pertaining to the granularity of the model, the theoretical complexity of the impact evaluation algorithm, and our observed computational time in the analysis reported in this paper.

Model granularity: Each node in our graph represents either a set of privileges acquired by the attacker (in the attack progression model) or a component with an arrival queue in the production model. Thus, each node can represent components at various levels of aggregation, e.g. an individual process on a computer, a cluster of servers, or even an entire sub-network. The same model (with different parameters) can therefore be used to represent a system where a server is an entire cluster of servers and a database is a distributed database. This can be taken advantage of where the attack progression allows, e.g. if the attack spreads and impacts servers in groups of 10, then modeling at the level of individual servers is not needed.

Theoretical complexity: The most computationally expensive part of our impact estimation is the performance evaluation. While the fluid approximation used in the QN evaluations itself scales well, it needs to be done numerous times as a performance estimate is required for each different attack outcome under each candidate allocation considered during the optimization. Consequently, the complexity of our optimization depends mainly on two factors: 1. The solution space determined by the upper and lower bounds given to the optimization; 2. The number of attack outcomes to evaluate for each candidate allocation.

The number of the attack outcomes to investigate depends substantially on whether attacks can be detected (and prevented) when moving laterally from one server to another. Servers that do not share vulnerabilities require the attacker to use new attack steps, and thus the attack can be potentially detected and prevented. For servers that share the same vulnerabilities, and in particular identical servers, we distinguish two cases: **Case 1** where the attack cannot be detected or prevented when moving from one server to the other – this results in all identical servers being treated as a cluster, they are either all compromised or not; and **Case 2** where we assume that it is possible to detect and prevent the attack when moving from one server copy to another. Case 2 requires accounting for attack steps being detected or not detected for every server instance. The complexity therefore increases combinatorially when this assumption is considered for server allocations at all layers (application, processing, databases). Complexity wise, this is a worst-case

assumption that rarely occurs in practice. If an attack cannot be detected when it compromises an instance of a server, it will typically not be possible to detect it on identical copies of that server as the attack steps can be replicated identically. The only exception are alerts that are triggered when a certain volume of events has been reached, for example, connections to unopened ports or failed authentication attempts. Repeating these actions too often may result in an alert, whilst a small number of such events may be considered benign. However, such circumstances are rare, and even then each individual server instance would not make a sufficient difference. A more realistic view would be to group servers according to the alert thresholds. This would be, in essence, a case somewhere between Case 1 and Case 2 depending on the size of the groups. Our complexity analysis shows that Case 1 scales well to larger systems, Case 2 becomes difficult to scale, and solutions in-between depend on the size of the server groups considered. More detail is provided in Section 6 of the supplement.

Observed run time: The time required to compute the system performance under a given "effective server allocation" (servers available after a stage of disruption) in our case study was on average 3.08s with the QN model fluid approximation, and 0.0056s with the bounds analysis used in the first stage of the optimization. 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. Note that our analysis is run "off-line", i.e. it does not need to be run in real time during the attack.

QN model accuracy: Large-scale QN models pose challenges in achieving high accuracy in performance prediction. While simulations may be carried out for increased precision, they are often too slow for a single scenario evaluation. Our approach based on fluid solvers offers a solution to this issue by the mean-field approximation theory from [40], which guarantees that as the system scale grows, intended as the total number of jobs and servers (i.e., processing units) in the model, then the model precision also grows. Namely, asymptotically the sample paths of the exact model converge to the fluid approximation [40, Thm. 1].

Since for increasing number of jobs and servers the number of ordinary differential equations does not change, our approach can scale efficiently. However, if more stations are added to the QN, the number of equations will grow linearly. In models with many equations, lumping approaches may be applied to control the cost of the model solution [44].

7 Conclusion

Amongst our findings, two stand out as the most important for using redundancy (and diversity) for attack mitigation. First, when the losses are based on SLA penalties, there is a cut-off point for when redundancy can be beneficial and when it cannot, depending on the speed of recovery relative to the disruption tolerance of the SLA. A sufficiently fast recovery can avoid SLA penalties. However, when penalties can occur in response to attacks, redundancy with diversity can mitigate losses.

Second, choosing a server allocation (with and without diversity) depends on both the cost during an attack, and the added maintenance costs over time. We have introduced the Time Until Loss (TUL) metric to track the balance of these two types of costs. It essentially measures the frequency of attacks required for a given server allocation to be financially viable in comparison to a reference allocation. The size of the potential loss from an attack has a large impact on the TUL values observed. Large losses make diversification viable even at lower attack frequencies.

The detection probability has a significant impact. Increases in p_d effectively tighten the limits of affordability for diversity, reducing expected costs under all allocations and thus limiting the scope for diversification to yield sufficient attack cost savings to justify the added maintenance costs.

Finally, the various parameters can impact the results in different directions. There are no clear rules-of-thumb and there is a paucity of tools and models to analyze the economic impact from cyber attacks, leading to ad-hoc security investment decisions and poor provisioning of systems to ensure resilience under attack. Our work provides a first such methodology and analysis leading the way for other tools and models to be developed. While the parameter values used in this paper are drawn from the available literature, they may differ across industry sectors and organizations. Conducting the analysis in specific contexts would be relatively straightforward by using context specific parameters, and the methodology can be adapted to accommodate variations in the cost or attack models.

References

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

[2] D. Borbor, L. Wang, S. Jajodia, and A. Singhal, "Surviving unpatchable vulnerabilities through heterogeneous network hardening options," *J. of Computer Security*, vol. 26, no. 6, pp. 761–789, 2018.

[3] ——, "Optimizing the network diversity to improve the resilience of networks against unknown attacks," *Computer Communications*, vol. 145, pp. 96–112, 2019.

[4] T. Li, C. Feng, and C. Hankin, "Scalable approach to enhancing ics resilience by network diversity," in *IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN)*. IEEE, 2020, pp. 398–410.

[5] B. Chen, Z. Kalbarczyk, D. M. Nicol, W. H. Sanders, R. Tan, W. G. Temple, N. O. Tippenhauer, A. H. Vu, and D. K. Yau, "Go with the flow: Toward workflow-oriented security assessment," in *New Security Paradigms Workshop*, 2013, pp. 65–76.

[6] A. H. Vu, N. O. Tippenhauer, B. Chen, D. M. Nicol, and Z. Kalbarczyk, "Cybersage: A tool for automatic security assessment of cyber-physical systems," in *Int. Conf. on Quantitative Evaluation of Systems*. Springer, 2014, pp. 384–387.

[7] M. Ge, H. K. Kim, and D. S. Kim, "Evaluating security and availability of multiple redundancy designs when applying security patches," in *47th IEEE/IFIP Int. Conf. on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2017, pp. 53–60.

[8] H. Shan, Q. Wang, and C. Pu, "Tail Attacks on Web Applications," in *Proc. 2017 ACM SIGSAC Conf. Computer and Communications Security*, 2017, pp. 1725–1739.

[9] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[10] Y. Hu, J. Wong, G. Iszlai, and M. Litoiu, "Resource provisioning for cloud computing," in *Conf. of the Center for Advanced Studies on Collaborative Research*. IBM Corp., 2009, pp. 101–111.

[11] Y. Kouki and T. Ledoux, "SLA-driven capacity planning for cloud applications," in *4th IEEE Int. Conf. on Cloud Computing Technology and Science*. IEEE, 2012, pp. 135–140.

[12] Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan, and L. Liu, "Service level agreement based energy-efficient resource management in cloud data centers," *Computers & Electrical Engineering*, vol. 40, no. 5, pp. 1621–1633, 2014.