Contemporary Mathematics
http://ojs.wiserpub.com/index.php/CM/

Research Article

Optimization of Diversity Redundancy Systems Based on Modified Shannon Index

George Popov, Antoaneta Popova, Nikolay Hinov*

Faculty of Computer Science and Technology, Technical University of Sofia, Sofia, Bulgaria
E-mail: hinov@tu-sofia.bg

Received: 24 December 2025; **Revised:** 27 March 2026; **Accepted:** 27 March 2026

Abstract: Diversity redundancy in information systems aims to reduce systemic risk by distributing critical resources (e.g., servers, communication channels, and services) across heterogeneous component types so that a single vulnerability is less likely to compromise many assets simultaneously. This paper proposes a vulnerability-aware diversification framework based on an entropy-inspired indicator H_s that combines (i) deployment shares of component types and (ii) time-accumulated resilience derived from compromise probabilities over a planning horizon. The approach includes a practical allocation principle that prioritizes more resilient technologies while allowing minimum-diversity constraints when required, as well as a deterministic integer allocation procedure (floor-and-carry) suitable for layered architectures under a fixed instance budget. Using a stationary compromise model for comparative planning, we illustrate how resilience-prioritized allocations can improve H_s relative to uniform allocation. To avoid substituting the optimization target, we also report the corresponding system-level survival probability under the same assumptions, showing that improvements in H_s are accompanied by improved system survival in the examples. The proposed methodology is intended as a decision-support tool for architecture planning and what-if analysis; practical use requires parameter estimation from operational data and assessment of modeling assumptions, especially when strong dependencies or correlated compromises are present.

***Keywords*:** cyber security, diversity, modeling, reliability, Shannon

MSC: 94A17, 68M25, 94A60

1. Introduction

The increasing complexity of digital infrastructure has led to a paradigm shift in how cybersecurity is conceptualized and implemented. In today's interconnected environments, where distributed architectures, cloud computing, and software-defined networks dominate, traditional perimeter-based defense mechanisms are no longer sufficient [1]. Security must now be embedded into the design of system architecture, embracing proactive strategies that anticipate, absorb, and adapt to evolving threats. One such emerging strategy is architectural diversity, which aims to reduce systemic vulnerabilities through the deliberate use of heterogeneous technological components [2].

At its core, architectural diversity draws inspiration from ecological systems, where biodiversity has long been correlated with ecosystem resilience [3]. In a natural ecosystem, a wider range of species offers protection against disease outbreaks, environmental changes, and predator pressures. The same logic applies to digital ecosystems: systems

Copyright ©2026 Nikolay Hinov, et al.
DOI: https://doi.org/10.37256/cm.7420269474
This is an open-access article distributed under a CC BY license
(Creative Commons Attribution 4.0 International License)
https://creativecommons.org/licenses/by/4.0/

composed of varied components (e.g., operating systems, software frameworks, hardware configurations) are less likely to be uniformly affected by a single exploit. This heterogeneity disrupts the propagation of threats and reduces the probability of simultaneous compromise, thereby increasing the overall system robustness [4].

Several empirical studies support this rationale. For instance, injecting diversity into homogeneous systems has demonstrated reduced attack success rates and greater containment efficiency [5]. In studies on networked systems, the introduction of platform heterogeneity has been shown to significantly impede malware propagation [6]. Similarly, architectural designs that incorporate multiple, non-overlapping layers of defense—each with unique failure modes—exhibit greater resistance to targeted attacks [7]. Moreover, frameworks such as moving target defense (MTD) explicitly rely on introducing unpredictability and variability to enhance defense mechanisms [8].

Recent developments in machine learning-driven threat detection also underscore the importance of diversity. Systems trained on varied datasets and models from heterogeneous sources have shown superior performance in identifying novel attack patterns [9]. Furthermore, research in adversarial machine learning highlights that ensemble models composed of diverse learners are more robust against adversarial evasion [10]. These findings illustrate that diversity not only enhances baseline robustness but also strengthens advanced detection capabilities.

Despite these theoretical and empirical validations, practical implementation remains complex. System designers must manage trade-offs between resilience and manageability. Heterogeneous systems, while more resilient, introduce operational burdens, including increased complexity, compatibility challenges, and resource demands [11]. This tension is particularly pronounced in large-scale critical infrastructures where legacy constraints, certification standards, and interoperability requirements may hinder the adoption of diverse architectures [12].

To address these limitations, recent approaches advocate for semi-automated diversification, guided by intelligent orchestration platforms and cyber-risk analytics [13]. These platforms leverage threat intelligence feeds, vulnerability databases, and behavioral telemetry to inform diversification decisions dynamically. Such approaches move toward adaptive and context-aware resilience, aligning security policies with real-time risk assessments.

This paper contributes to this evolving domain by proposing a quantitative framework based on the Shannon entropy index. Originally developed to measure uncertainty in communication systems [14], the entropy index is reformulated here to capture the degree of technological diversity within IT infrastructures. Importantly, we integrate real-world threat data, component-specific compromise probabilities, and temporal risk accumulation. This transforms entropy from an abstract metric into a concrete, decision-support instrument.

Additionally, we discuss how the entropy model can be operationalized using optimization algorithms that target multi-layered architectures. These architectures typically span operating systems, virtualization layers, application platforms, and databases. By structuring diversity allocation across layers, and weighting component selection based on vulnerability history and exposure profiles, system architects can derive maximally resilient configurations that are also cost-effective and manageable [15].

2. Related work and literature review

Diversity has been studied as a cross-cutting mechanism for increasing system dependability and cybersecurity resilience. Relevant literature spans (i) design diversity and redundant implementations in dependable computing, (ii) quantitative diversity metrics (often borrowed from ecology and information theory), and (iii) practical cyber-defense paradigms such as moving-target defense and cloud-native heterogeneous deployments. This section synthesizes these strands and clarifies the methodological gap addressed by the proposed modified Shannon-based optimization framework.

2.1 *Diversity and redundancy in dependable and secure systems*

The rationale for using diversity originates from dependable computing, where correlated or common-mode failures limit the effectiveness of pure redundancy. The N-version programming paradigm formalized the use of multiple independently developed implementations to reduce the probability of simultaneous failure [16]. Empirical studies, however, show that independence assumptions can be violated, motivating explicit modeling of failure correlation and

systematic design diversification [17]. A broad overview of reliability and redundancy allocation methods can be found in standard references such as the Handbook of Software Reliability Engineering [18]. In the security domain, redundancy and diversity have been framed as a means to mitigate shared vulnerabilities and to increase the attacker's workload, particularly against monoculture effects [4, 5]. Network-level configuration diversity and redundancy have also been investigated as a defense strategy to reduce large-scale compromise probability and to constrain malware propagation [6].

From a systems engineering perspective, applying diversity introduces practical trade-offs: heterogeneous components increase operational complexity, integration effort, and testing/verification workload. Modern security engineering guidance therefore emphasizes systematic architecture design and assurance arguments that make assumptions explicit and traceable [11]. These trade-offs are particularly pronounced in regulated or legacy-heavy environments, including industrial control systems (ICS) [12].

2.2 *Diversity metrics: from ecology and information theory to cybersecurity*

Quantifying diversity requires metrics that capture both richness (how many component types exist) and evenness (how uniformly components are distributed). The Shannon entropy measure [14] and classical ecological indices (e.g., Simpson and related Hill numbers) have been widely used to quantify biodiversity and resilience [19–23]. In cybersecurity, these indices are attractive because they provide a compact scalar measure of heterogeneity that can be optimized under resource constraints [2]. Entropy-like measures have also been used operationally in security analytics, for example to characterize traffic distributions for anomaly detection [24].

A key limitation of using classical entropy directly is that it treats all component types as equally risky, which is rarely true in practice. Cyber components differ in exposure, patch cadence, vulnerability history, and attacker interest. Recent surveys therefore highlight the need for security-aware diversity metrics that incorporate vulnerability and dependency structure rather than measuring heterogeneity alone [2]. The modified index developed in this paper follows this direction by embedding time-accumulated compromise probabilities into the diversity weights.

2.3 *2.3 Quantitative security metrics, threat modeling, and vulnerability data*

Quantitative cybersecurity assessment frequently relies on structural threat modeling and vulnerability scoring. Attack-graph approaches provide a systematic representation of multi-step intrusions and enable computation of security metrics under assumptions about exploitability and reachability [25, 26]. In operational practice, vulnerability scoring frameworks such as the Common Vulnerability Scoring System (CVSS) provide standardized severity measures that are incorporated into security programs and vulnerability management pipelines [27, 28]. Public repositories such as the National Vulnerability Database (NVD) provide widely used data feeds, although data quality issues and bias can affect quantitative conclusions, motivating data-cleaning and calibration efforts [29, 30].

Machine learning has also been applied to intrusion detection and threat analytics, but its performance and robustness depend strongly on data quality, feature engineering, and deployment assumptions [9, 31]. Moreover, adversarial machine learning shows that models can be manipulated at inference time; ensembles of diverse learners and defense strategies are frequently proposed to improve robustness under adaptive adversaries [10]. These observations reinforce the value of architectural diversity as a system-level mitigation complementing detection and response.

2.4 *Optimization of redundancy and diversity allocation*

Deciding how many instances of each component type to deploy is a resource-allocation problem that resembles classical reliability redundancy allocation under cost and performance constraints. Reliability engineering literature provides optimization formulations and heuristics (including integer programming and evolutionary methods) for allocating redundant components to maximize system reliability [32, 33]. From an optimization standpoint, convex and constrained optimization principles provide a general toolbox for deriving tractable surrogate objectives and stable allocation rules [15].

In cybersecurity settings, allocation must additionally account for time-dependent compromise processes and the fact that vulnerabilities are not uniform across technologies. The proposed model bridges this gap by (i) expressing diversity

through a modified Shannon index that is weighted by resilience, and (ii) providing a deterministic layered allocation algorithm that transforms fractional weights into integer deployments while preserving total instance constraints.

2.5 *Dynamic and cloud-native diversification*

Cloud-native architectures (virtualization, containers, and microservices) make diversity and redundancy more practical by lowering provisioning overhead and enabling rapid reconfiguration. At the same time, they create new attack surfaces and dependency chains that require systematic security engineering and platform-specific hardening [11, 34, 35]. Moving-target defense (MTD) explicitly introduces runtime variability to increase attacker uncertainty, complementing static architectural diversity through dynamic reconfiguration and address/route/platform changes [8, 13]. Finally, modern security paradigms such as Zero Trust emphasize continuous verification and assume network compromise, further motivating compartmentalization and heterogeneity at multiple layers [1]. Complementary application-focused studies also explore diversity for distributed systems and sector-specific contexts [36–38].

In summary, prior work establishes diversity as a powerful mechanism, but there remains a practical need for metrics and allocation procedures that are simultaneously (a) quantitatively grounded, (b) vulnerability-aware, and (c) usable for multi-layer system design under fixed resource budgets. The model and algorithm proposed in the following sections aim to provide this missing link, by connecting entropy-based diversity measurement with time-accumulated compromise probability and a concrete deployment optimization rule.

3. Multi-layered diversity as a cybersecurity strategy

In the context of layered system architectures, cybersecurity must account for the interdependencies between different functional tiers such as hardware, operating systems, network services, middleware, applications, and data management. A breach at any layer can compromise the integrity of the entire system, particularly if homogeneous components are replicated across tiers. The strategic deployment of multi-layered diversity represents a refined defense-in-depth approach, reducing systemic risk by ensuring heterogeneity within and across each layer.

Figure 1 illustrates this principle conceptually, depicting an IT infrastructure where diverse technologies, vendors, and configurations coexist.

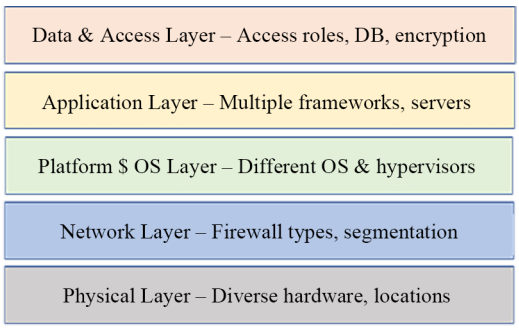


Figure 1. Multi-layered diversity of IT infrastructure

Each architectural layer employs components selected based on differing vulnerability profiles and attack surfaces. For instance, using different operating systems (e.g., Linux, Windows, FreeBSD) at the OS layer hinders the reuse of a single exploit across all servers. Similarly, alternating between web frameworks (e.g., Apache, NGINX, IIS) or database systems (e.g., PostgreSQL, MySQL, Oracle) at the application layer further compartmentalizes risk exposure [2].

The main idea is that diversity at each architectural layer enhances overall resilience to attacks, as it reduces the likelihood of simultaneous compromise through shared vulnerabilities:

1. Physical Layer—Hardware and location diversity

Employing different hardware vendors and architectures, along with geographic distribution, mitigates risks from single points of physical failure, including natural disasters, physical sabotage, and supply chain attacks [39].

2. Network Layer—Segmentation and firewall variety

Utilizing multiple types of firewalls (e.g., NGFW, proxy-based, DPI) and segmentation strategies (e.g., VLANs, microsegmentation) hinders lateral movement by attackers and limits the blast radius of successful intrusions [40].

3. Platform & OS Layer—Operating system and hypervisor diversity

Deploying different OS platforms (e.g., Linux, Windows, FreeBSD) and hypervisors (e.g., VMware, KVM, Hyper-V) minimizes the risk of widespread exploitation via a single vulnerability. Systems remain isolated in their exposure surfaces [41].

4. Application Layer—Framework and web server heterogeneity

Using multiple web frameworks (e.g., Django, Laravel, Express.js) and web servers (e.g., Apache, NGINX, IIS) decreases the effectiveness of large-scale, automated attacks targeting known platform-specific vulnerabilities [42].

5. Data & Access Layer—Roles, databases, and encryption diversity

Diverse databases (e.g., PostgreSQL, MySQL, Oracle), access control models, and encryption schemes (e.g., AES, RSA, ECC) increase resilience against data breaches or manipulation. Role-Based Access Controls (RBAC) also reduce the impact of compromised accounts [43].

Diversity within multi-layered systems is not just a qualitative attribute but can be measured quantitatively. Our extended Shannon-based entropy model accounts for layered diversity by weighting the contribution of each component by its vulnerability and its role in the dependency tree.

Empirical results suggest that optimal diversity is not achieved by maximizing entropy uniformly across all layers but by targeting the most critical and vulnerable tiers. For example, studies show that exploiting diversity at the OS and application layers yields greater marginal benefits in security metrics than diversity at the hardware layer alone [2].

This implies that entropy optimization in multi-layer systems should be guided by a layered prioritization strategy, informed by historical breach data, dependency graphs, and exposure likelihoods. The result is a flexible yet robust architecture that actively deters common-mode failures and zero-day escalations.

4. Methodology for assessment of security in diversity systems

Assessing security in diversified IT architectures requires a quantitative metric that captures the degree of heterogeneity among system components. In this work, we start from the Shannon diversity index, widely used in ecology to measure biodiversity [19], and adapt it for cybersecurity settings by incorporating time-accumulated compromise probabilities and component resilience.

$$H = -\sum_{i=1}^{n} p_i \log(p_i), \tag{1}$$

where $p_i = \frac{n_i}{N}$ is the proportion of component type i ($i = 1, \ldots, k$) among the total N deployed instances, and $\sum_{i=1}^{k} p_i = 1$. Unless stated otherwise, $\log(\bullet)$ denotes the natural logarithm.

Notation summary: p_i is the deployment share (a proportion), q_i is the per-attempt success probability of an attack, $P_i(t)$ is the cumulative compromise probability over horizon t, and $R_i(t) = 1 - P_i(t)$ is the corresponding resilience.

4.1 *Adaptation of the Shannon index*

To address this, we define a modified version of entropy H_s which incorporates security weightings:

$$H_s = -\sum_{i=1}^{n} w_i \log(p_i), \tag{2}$$

with

$$w_i = p_i \cdot R_i(t). \tag{3}$$

Interpretation of H_s. In the classical Shannon index, the weights coincide with a probability distribution (the shares p_i). In (2)–(3), the weights are modified by resilience, $w_i = p_i \cdot R_i(t)$, and therefore $\{w_i\}$ does not generally form a probability distribution (in particular, $w_i = \sum p_i R_i(t) \leq 1$). Consequently, H_s should not be interpreted as information-theoretic entropy in the strict sense, but as an entropy-inspired surrogate indicator that combines (i) architectural diversity via p_i and (ii) achieved resilience via $R_i(t)$. For this reason, in Section 6 we also report the system-level survival probability $P_{sys}(t)$ defined in (6), so that improvements in H_s can be directly related to improvements in system security.

4.2 *Modeling vulnerability over time*

In the following, t denotes the planning horizon. Unless stated otherwise, the examples use a fixed horizon $T = 30$ days (720 h), which is representative for short-term security planning and vulnerability assessment cycles. We assume that attacks arrive according to a stationary process with mean attempt intensity λ_i for component type i, and that each attempt has success probability q_i. The vulnerability over a time horizon is modeled by:

$$P_i(t) = 1 - (1 - \lambda_i q_i)^t. \tag{4}$$

Scope of applicability. Equation (4) assumes independent and stationary attack attempts over the planning horizon t, which provides a tractable first-order approximation for comparative allocation studies. In dynamic environments (e.g., evolving attacker behavior, cloud workloads, or zero-day waves), the parameters λ_i and q_i can change over time. The framework can be applied in a rolling-horizon manner by periodically re-estimating λ_i and q_i from telemetry and threat intelligence, or extended to non-stationary processes (e.g., time-varying $\lambda_i(t)$, $q_i(t)$) as future work.

This formulation captures both frequency and impact of threats, enabling time-sensitive risk analysis for long-term system planning.

4.3 *Diversity allocation principle*

For optimal security, the share of p_j each component type is allocated proportionally to its resilience:

$$p_j = \frac{1 - P_j}{\sum_{i=1}^{k} (1 - P_i)} = \frac{R_j}{\sum_{i=1}^{k} R_i}. \tag{5}$$

Diversity-resilience trade-off. The proportionality rule in (5) prioritizes more resilient technologies, which can reduce evenness if resilience differs strongly across types. This reflects an explicit trade-off: architectural diversity values heterogeneity, while resilience optimization rewards stronger components. In practice, the allocation is therefore interpreted as resilience-prioritized diversification under a minimum-diversity constraint (e.g., enforcing at least one instance per type when $N \geq k$, and/or imposing an upper bound on any single share). These constraints prevent degeneration into a quasi-monoculture while preserving the main optimization benefit.

Under the above minimum-diversity constraint, this ensures representation of each type while prioritizing more secure components.

4.4 *Composite security of the system*

Assuming conditional independence between compromises of different component instances/types, the probability that the system remains uncompromised over time is:

$$P_{sys}(t) = \prod_{i=1}^{k} (1 - P_i(t))^{n_i}, \tag{6}$$

where n_i is the number of instances of component type. This allows the evaluation of security outcomes under both uniform and optimized allocations, enabling comparative resilience analysis.

This independence assumption is a simplifying approximation used here for transparent comparison between uniform and optimized allocations. In real deployments, correlations (e.g., shared credentials, common management planes, supply-chain dependencies, or cross-layer privilege escalation) can reduce absolute security levels. Section 7 discusses how dependency-aware models (e.g., attack graphs or common-cause factors) can be incorporated as an extension.

To provide a simple quantitative illustration of this effect, we introduce a common-cause dependence parameter $\delta \in [0,1]$, epresenting the strength of common-cause exposure in a first-order stress approximation. A dependence-adjusted approximation of the system survival probability can then be written as:

$$P_{sys}^{dep}(t) = (1 - \delta) \prod_{i=1}^{k} (1 - P_i(t))^{n_i}. \tag{7}$$

Here, $\delta = 0$ corresponds to the baseline independent case, while larger values of δ represent stronger common-cause exposure. This simplified extension is not intended as a universal dependence model, but as a transparent robustness check allowing assessment of whether the comparative advantage of the optimized allocation persists when absolute system survival is degraded by shared-risk effects. Illustrative scenarios based on this approximation are discussed in Section 6 and revisited in Section 7.

5. Resilience-oriented layered allocation algorithm

To implement the methodology efficiently in multi-layered systems, we propose the resilience-oriented layered allocation algorithm. It is designed to ensure optimal distribution of component instances across all system layers while respecting component security characteristics and maintaining structural diversity.

The algorithm consists of the following steps:

Step A1: Initialization.

For each system layer L_i, identify all available component types $j = 1,2,...k_i$ and input their vulnerability profiles: λ_{ij} and q_{ij}.

Step A2: Calculate Component Security.

For a given operational time window t_i , compute compromise probability and resilience:

$$P_{ij}(t) = 1 - (1 - \lambda_{ij} \cdot q_{ij})^t, \tag{8}$$

$$R_{ij}(t) = 1 - P_{ij}(t). \tag{9}$$

Step A3: Derive Allocation Weights.

Calculate ideal fractional share of each component by (5).

Step A4: Integer Allocation with Carry Propagation.

Transform fractional shares into integers for a fixed total N_i instances per layer:

• For each j, assign:

$$n_{ij} = Floor(N_i \cdot p_{ij} + carry_j). \tag{10}$$

• Accumulate the rounding remainder as:

$$carry_{j+1} = carry_j + (N_i \cdot p_{ij} - n_{ij}). \tag{11}$$

• At final iteration, adjust last n_{ij} to ensure:

$$\sum_j n_{ij} = N_i. \tag{12}$$

Rationale and basic properties. We use a floor-and-carry scheme (rather than standard independent rounding) to preserve the total instance count exactly ($\sum n_i = N$) in a deterministic way and to avoid rounding drift across layers. The carry term accumulates fractional remainders, ensuring that the final integer allocation closely tracks the fractional targets while satisfying the hard budget constraint N. If a minimum diversity level is required (e.g., $n_i \geq 1$ for all i when $N \geq k$), this can be enforced by seeding one instance per type and distributing the remaining $N - k$ instances using the same procedure.

Step A5: Recursion Across Layers.

For each assigned component j at layer L_i, recursively build valid combinations with components from layer L_{i+1} using the same strategy.

Step A6: Output Final Configuration Tree.

Generate the final layered architecture as a hierarchy (tree) with weighted component instances per layer that reflect resilience priorities and total diversity constraints.

This algorithm ensures reproducible, numerically stable distribution that adheres to both diversity requirements and security optimization goals. It is particularly suited for automated security provisioning in cloud-native, containerized, or virtualized infrastructures.

Accuracy of integer approximation.

Let the fractional target allocation be $x_i = Nw_i$, where $\sum_{i=1}^{k} x_i = N$, and let n_i be the integer allocation produced by the floor-and-carry procedure in (9)–(11). By construction, the method preserves the total number of allocated components exactly, i.e.

$$\sum_{i=1}^{k} n_i = N. \tag{13}$$

In addition, the procedure keeps the integer allocation close to the fractional target, because at each step it assigns the integer part of the current carried value and forwards the remaining fractional part to the next component type. This avoids the budget inconsistency that may arise from independent rounding of the values x_i.

To quantify the discretization effect, we consider the deviation measures

$$E_1 = \sum_{i=1}^{k} |n_i - x_i|, \quad E_{\max} = \max_i |n_i - x_i| \tag{14}$$

In the numerical examples in Section 6, these deviations remain small and do not affect the qualitative comparison between uniform and optimized allocations. Thus, the floor-and-carry procedure provides a practical integer approximation that preserves the total budget while remaining close to the intended fractional optimum.

6. Examples and validation

Example 1: One layer, 9 devices of 3 types

Let's look at a practical scenario with 9 devices of 3 types, which have cumulative compromise probabilities $P_A(t) = 0.8$, $P_B(t) = 0.5$ and $P_C(t) = 0.2$ over a fixed planning horizon $T = 30$ days (720 h).

$$p_A = p_B = p_C = \frac{3}{9} = 0.333. \tag{15}$$

Calculated by (2)–(3), the achieved security through diversity is: $H_s(t) = 0.7674$

Case 2—optimal distribution. The fractional shares are obtained from (5), and the integer allocation is obtained using (10)–(12):

$$p_A = 0.1515, \quad p_B = 0.2424, \quad p_C = 0.6061. \tag{16}$$

The number of devices of each type is determined:

$$x_i = \text{round}\,(N \cdot p_i)\,. \tag{17}$$

Therefore, in the redundant diversity system there will be a corresponding number of devices of each type:

$$x_1 = \text{round}(1.36) = 1, \quad x_2 = \text{round}(2.18) = 2, \quad x_3 = \text{round}(5.45) = 6. \tag{18}$$

For the distribution according to (2)–(3) , the security achieved by optimal redundancy is $H_s(T) = 0.8316$. It can be seen that the optimal distribution increases the entropy-inspired indicator H_s by over 8% compared to the uniform one. To directly relate the entropy-based indicator to system security, we also compute $P_{sys}(T)$ from (6). For the uniform allocation $(n_A, n_B, n_C) = (3, 3, 3)$, $P_{sys}(T) = (1 - 0.8)^3\ (1 - 0.5)^3\ (1 - 0.2)^3 = 5.12 \times 10^{-4}$. For the optimized integer allocation $(n_A, n_B, n_C) = (1, 2, 6)$, $P_{sys}(T) = (1 - 0.8)^1\ (1 - 0.5)^2\ (1 - 0.2)^6 \approx 1.31 \times 10^{-2}$. Thus, the observed increase in H_s is accompanied by a substantial improvement in system-level survival probability.

Numerical check of rounding accuracy.

For the obtained fractional targets $x_i = Nw_i$, the final integer allocation n_i generated by the floor-and-carry procedure remains close to the continuous optimum. The corresponding deviations $|n_i - x_i|$ are small for all component types, and the total allocation constraint is satisfied exactly. This confirms that the proposed rounding procedure introduces only limited discretization error while preserving the practical interpretability of the solution.

Table 1 quantifies the deviation between the fractional allocation targets and the final integer allocation obtained by the floor-and-carry procedure in Example 1.

Table 1. Numerical accuracy of the floor-and-carry integer approximation in Example 1

Component type	Fractional share	Fractional target	Integer allocation	Deviation
A	0.1515	1.36	1	0.36
B	0.2424	2.18	2	0.18
C	0.6061	5.45	6	0.55
Total	1.0000	9.00	9	
E_1				1.09
E_{max}				0.55

As seen from Table 1, the floor-and-carry procedure preserves the total allocation exactly and introduces only limited deviations from the fractional targets. This supports its use as a practical deterministic discretization method under a strict integer budget.

Figure 2 compares the modified entropy index for uniform and optimized allocations in Example 1.

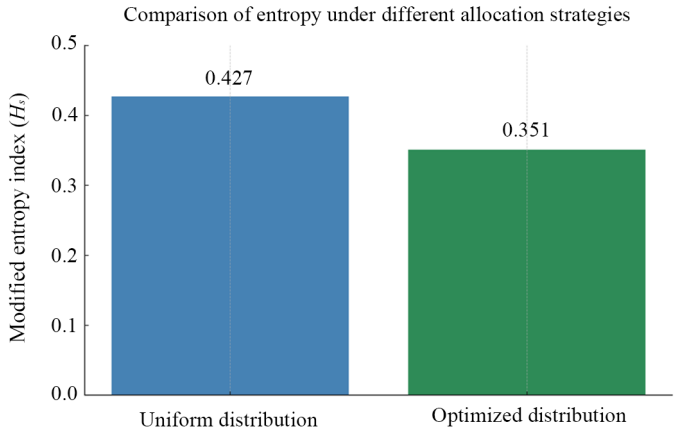


Figure 2. Comparison of entropy index H_s for uniform vs optimized distribution

Example 2: Global diversification of web services

Instead of averaging probabilities, each configuration is evaluated according to formulas (4) and (5), based on attack intensity μ_i and attack success probability (vulnerability) $s_i (q_i = \mu_i s_i)$.

Table 2 shows the vulnerabilities of various types of software and the probability of a breach occurring in 30 days. Each of the systems A, B and C consists of an operating system, a web server and a database.

Table 2. Values for vulnerability of individual systems

System	Compo-nent	μ_i [1/h]	s_i	q_i [1/h]	P_i (720 h)	P_{sys} (720 h)
A	Linux	0.0002	0.3	0.00006	0.042281	0.1723
	Apache	0.0003	0.4	0.00012	0.082777	
	MySQL	0.0003	0.3	0.00009	0.062748	
B	Windows	0.0005	0.5	0.00025	0.164749	0.3658
	IIS	0.0004	0.5	0.00020	0.134125	
	MSSQL	0.0004	0.5	0.00020	0.134125	
C	FreeBSD	0.0001	0.2	0.00002	0.014297	0.0551
	Nginx	0.0001	0.2	0.00002	0.014297	
	PostgreSQL	0.0002	0.2	0.00004	0.028390	

P_i (720 h) is computed by Eq. (4) for $t = 720$; values are rounded to 6 decimals.
Graphical representation of above calculation is shown at Figure 3.

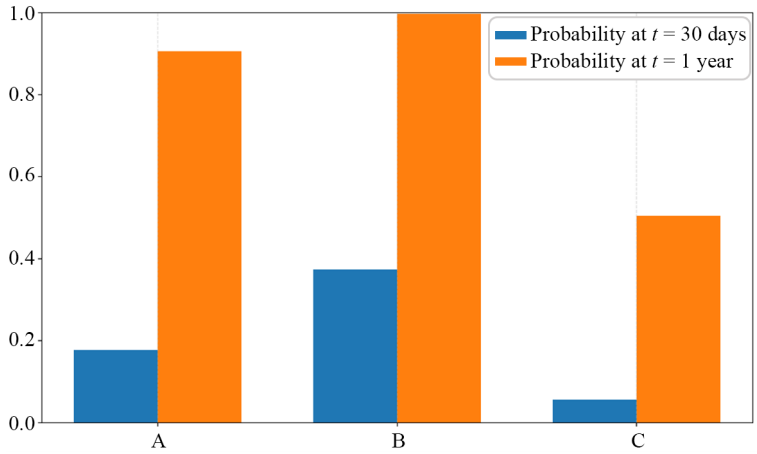


Figure 3. Comparison of the vulnerability of the three system configurations for $T = 30$ days (720 h) and $T = 1$ year

Sensitivity analysis of illustrative parameters.

Since the parameter values used in the examples are illustrative and partly based on expert judgment, an additional sensitivity analysis was performed to examine the robustness of the comparative results. Specifically, the products $\lambda_i q_i$, which determine the compromise probabilities $P_i(t)$, were varied within a range of ±20% around their baseline values while keeping the same planning horizon.

For each scenario, the values of $P_i(t)$, the corresponding resilience indicators, and the resulting allocation outcomes were recomputed. The analysis shows that, although the absolute numerical values change as expected, the qualitative comparison between uniform and optimized allocation remains stable over the examined range. In particular, the optimized allocation continues to provide better system-level survival than the uniform allocation, even though the magnitude of improvement varies across scenarios.

These results support the intended role of the examples in this study: the numerical values are illustrative rather than predictive, but the comparative conclusions of the proposed allocation approach are robust under moderate perturbations of the input parameters.

A more rigorous empirical calibration of λ_i and q_i based on operational telemetry or incident statistics remains an important direction for future work.

Table 3 summarizes four illustrative parameter scenarios used to test the robustness of the comparative result between uniform and resilience-prioritized allocation.

Table 3. Sensitivity of comparative results to variation of illustrative vulnerability parameters

Scenario	Variation applied to $\lambda_i q_i$	Effect on absolute $P_i(T)$ values	Preferred allocation	Comparative conclusion
Baseline	None	Reference case	Optimized	Optimized allocation improves $P_{sys}(T)$ relative to uniform allocation
Optimistic	All $\lambda_i q_i$ decreased by 20%	All compromise probabilities decrease	Optimized	Ranking preserved
Pessimistic	All $\lambda_i q_i$ increased by 20%	All compromise probabilities increase	Optimized	Ranking preserved
Asymmetric stress	+20% increase applied to the weaker configurations only	Relative risk gap increases	Optimized	Ranking preserved

The scenarios in Table 3 indicate that the absolute probability values vary, as expected, but the qualitative comparison between uniform and optimized allocation remains unchanged across the examined parameter range. Thus, the proposed allocation rule appears robust as a comparative design tool under moderate perturbations of the illustrative inputs.

The values shown in the table are hypothetical and based on expert assessment with reference to the NIST CVE database (NVD) and available empirical studies on the vulnerability of various components, e.g. NVD Metrics [29], reports by Rapid7, Qualys, etc. They are not provided in direct form, but are modeled in the spirit of good practices for risk assessment in the absence of complete empirical data.

If the vulnerabilities of the components (e.g. OS, web server, database) are independent, they are consistent in terms of security, since a breach anywhere makes the system vulnerable.

Figure 4 shows the optimized allocation weights among configurations A, B, and C in Example 2.

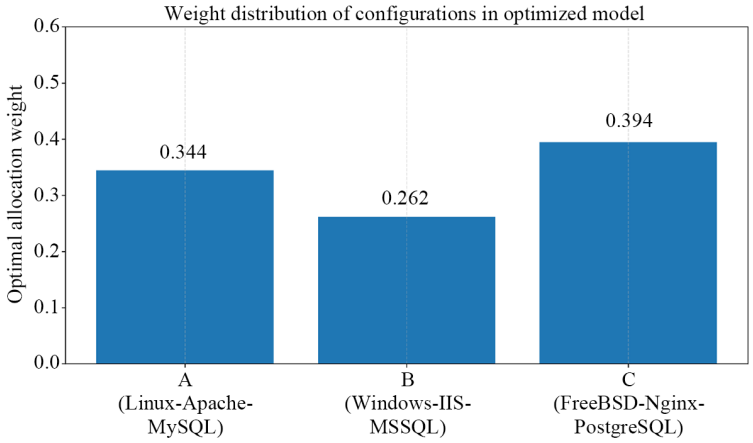


Figure 4. Optimal allocation weights among three configurations

Example 3: Multilevel (partial) diversification of web services

To ensure a resilient and diverse system architecture, the following selection rules were applied when allocating $N = 9$ servers to configuration groups composed of OS, Web Server, and Database components:

1. Each major component category (e.g., OS, Web Server, DB) must appear at least once in the final configuration pool. This ensures that the Shannon diversity entropy remains strictly positive for each layer.

2. The number of instances per configuration is proportional to its intrusion resilience, defined as $1 - P_j$. This guarantees that more secure configurations are prioritized in the allocation process.

3. The total number of allocated servers is fixed (in this example $N = 9$). Rounded integer allocations are computed based on relative weights. If the sum differs from 9, the remaining server is assigned to the most resilient configuration.

Vulnerabilities were calculated for 27 configurations built from combinations of 3 operating systems $A_i \in \{A1, A2, A3\}$, 3 web servers $B_i \in \{B1, B2, B3\}$ and 3 databases $C_i \in \{C1, C2, C3\}$ The combined vulnerability was calculated using the system vulnerability formula and optimal allocation weights were derived for a total of 9 servers.

The example is hypothetical because the compatibilities of individual OS, Servers and DB are not taken into account.

The vulnerabilities and intrusion resistances of the individual modules are given in Table 4.

Table 4. Examples values for vulnerability of individual systems

Module	P_j	R_j
A1	0.015	0.985
A2	0.033	0.967
A3	0.072	0.928
B1	0.101	0.899
B2	0.009	0.991
B3	0.075	0.925
C1	0.055	0.945
C2	0.034	0.966
C3	0.075	0.925

Table 5 shows the distribution of the different types of OS, according to their security, as agreed upon (17).

Table 5. Determining the number of different types of OS

OS Type	P_j	w	% from 9	Number
A1	0.015	0.601367	5.412301	5
A2	0.033	0.273349	2.460137	3
A3	0.072	0.125285	1.127563	1

Similarly, Table 6 shows the calculated distribution of the different types of servers.

Table 6. Global Servers distribution

Server	P_j	W	% from 9	Number
B1	0.101	0.073698	0.663282	1
B2	0.009	0.827055	7.443498	7
B3	0.075	0.099247	0.89322	1

Table 7 shows the partitioning of the different servers for the OS A1 instances. Last column shows Carry for the next distribution.

Table 7. Servers distribution for separate OS A1

A1:B[i]	P_j	w	from 9	N	Carry
A1: B1	0.101	0.073698	0.36849	0	0.36849
A1: B2	0.009	0.827055	4.135277	4	0.135277
A1: B3	0.075	0.099247	0.496233	1	−0.50377

This is followed by the completion of Layer 2 (servers), which determines the number of servers for each OS (See Table 8)

Table 8. Servers distribution for OS A1 &A2

A2:B[i]	N	Carry	A3:B[i]	N
A2:B1	0	0.36849	A3: B1	0
A2:B2	4	0.135277	A3: B2	1
A2:B3	1	−0.50377	A3:B3	0

Table 9 shows the partition definition and the exact number of databases.

Table 9. Global databases distribution

Server	P_j	W	% from 9	Number
C1	0.055	0.298420129	2.685781159	3
C2	0.034	0.482738444	4.344645992	4
C3	0.075	0.218841428	1.96957285	2

Table 10 is calculated similarly to Table 8. It gives the distribution of partitions and the specific number of databases, relative to the instances of the operating systems and servers A[i]:B[i].

Table 10. Final distribution by number of operating systems (OS), servers (SRV) and databases (DB)

OS:SRV:DB	Number
A1:B2:C1	1
A1:B2:C2	2
A1:B2:C3	1
A1:B3:C1	1
A2:B1:C2	1
A2:B2:C1	1
A2:B2:C2	1
A2:B2:C3	1
A3:B2:C2	1

Figure 5 illustrates the reduction in aggregated system risk achieved by optimized diversity allocation in Example 3.

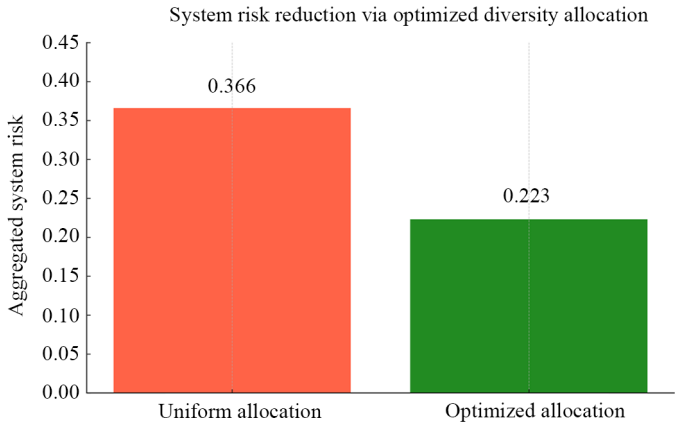


Figure 5. Reduction in systemic risk via optimal distribution

7. Discussion and limitations

The proposed modified Shannon index and allocation algorithm are intended as a decision-support method for architectural diversification, rather than a replacement for full threat modeling, penetration testing, and security operations. This section highlights key modeling assumptions, practical considerations for deployment, and limitations that inform future work.

7.1 *Modeling assumptions and parameter estimation*

The compromise probability model assumes that attack attempts against component type i arrive according to a stochastic process with mean intensity λ_i, and that each attempt succeeds with probability q_i. Over a planning horizon T, these parameters determine the cumulative compromise probability $P_i(T)$ and the corresponding resilience $R_i(T) = 1 - P_i(T)$. In practice, λ_i and q_i can be estimated from incident telemetry (e.g., IDS/IPS and SIEM data), vulnerability management records (e.g., patch lead times), and external threat intelligence feeds. When empirical data are sparse, expert elicitation may be used, in which case uncertainty should be reflected through sensitivity analysis across plausible ranges of λ_i and q_i.

Another simplifying assumption is the conditional independence of compromise events across components of different types. Real-world attacks can violate this assumption (e.g., credential reuse, shared management planes, supply chain dependencies, or cross-layer privilege escalation). When such dependencies are significant, the resilience weights should be adjusted using dependency-aware risk models (e.g., attack graphs or common-cause factors) before applying the allocation rule [44].

To improve transparency, the revised manuscript supplements the baseline examples with three simple robustness checks: a dependence-stress scenario for common-cause exposure, a sensitivity analysis of the illustrative vulnerability inputs, and a numerical verification of the deviation introduced by integer rounding. These additions do not eliminate the need for deployment-specific calibration, but they show that the main comparative conclusions are not artifacts of a single nominal parameter setting.

As a direction for future work, correlated compromises can be modeled via copula-based dependence structures or explicitly through attack graphs that capture shared prerequisites and cross-component propagation paths. Such dependency models can be used to adjust the effective resilience weights before applying the allocation rule.

7.2 *Compatibility, operational constraints, and costs*

The multi-layer examples in this paper intentionally abstract away compatibility constraints among operating systems, web servers, and databases. In production systems, compatibility, certification, and operational maturity impose hard

constraints on feasible combinations. The allocation algorithm can accommodate such constraints by restricting the allowed component set per layer and by excluding invalid cross-layer combinations during configuration tree generation.

Diversity also has a cost dimension: heterogeneous stacks require broader expertise, more complex CI/CD and monitoring pipelines, and potentially higher licensing or support costs. Future work will extend the optimization objective to include explicit cost, availability, and performance constraints, enabling multi-objective trade-off analysis between resilience and operational efficiency.

7.3 *Practical implications for cloud-native systems*

In cloud-native deployments, diversity can be implemented at multiple levels: OS images, container base images, language runtimes, service meshes, and managed database engines. Infrastructure-as-code and automated orchestration make it feasible to maintain heterogeneous fleets, while also enabling rapid rebalancing after new vulnerability disclosures. The proposed framework can be used periodically (or event-driven) to recompute optimal allocations as λj and $p j$ change over time.

Finally, architectural diversity complements Zero Trust and Moving Target Defense (MTD): Zero Trust reduces reliance on network perimeter assumptions, while MTD introduces runtime variability. Combining static diversity allocation (this work) with runtime adaptation is a promising direction for further improving system-level resilience.

8. Conclusions

This paper presented a vulnerability-aware diversification framework for diversity redundancy in information systems. The main contribution is an entropy-inspired indicator H_s that combines technological heterogeneity (deployment shares) with time-accumulated resilience derived from compromise probabilities over a planning horizon. On this basis, we proposed a resilience-prioritized allocation rule and a deterministic integer allocation procedure (floor-and-carry) suitable for layered architectures under a strict instance budget.

The illustrative examples show that resilience-prioritized diversification can improve H_s compared to uniform allocation. To avoid substituting the target metric, we also report the system-level survival probability P_{sys} under the same modeling assumptions; in the provided examples, improvements in H_s are accompanied by improvements in P_{sys}. This supports the use of H_s as a compact, engineering-oriented indicator for comparative design and what-if analysis, while keeping P_{sys} as the primary security quantity when the underlying assumptions hold.

From a practical perspective, the approach is intended as a decision-support tool for architecture planning in environments where deploying heterogeneous stacks is feasible (e.g., virtualized and cloud-native infrastructures, hosting environments, and modular service deployments). In such settings, the model can be used to periodically recompute allocations as estimated parameters change (e.g., via rolling-horizon updates from vulnerability management and operational telemetry), and it can be combined with feasibility constraints (compatibility, performance, or policy constraints) to restrict the configuration space.

The numerical results rely on simplifying assumptions, most notably stationary attack parameters over the horizon and conditional independence of compromise events. In real deployments, correlated compromises (shared credentials, common management planes, supply-chain effects, or cross-layer propagation) can reduce absolute security levels; therefore, parameter estimation and sensitivity analysis are essential, and dependency-aware modeling should be considered before applying the allocation rule.

Future work will focus on (i) incorporating explicit feasibility and operational constraints (compatibility, performance, and cost), (ii) extending the vulnerability model to time-varying parameters and event-driven re-estimation, (iii) integrating dependency and correlation modeling to relax independence assumptions, and (iv) empirical validation on real infrastructures and datasets.

Acknowledgment

This work was supported by the Operational Programme "Research, Innovation and Digitalization for Smart Transformation 2021–2027" under Project No. BG16RFPR002-1.014-0006, "National Centre of Excellence Mechatronics and Clean Technologies", and the APC was funded by Project No. BG16RFPR002-1.014-0006.

Conflict of interest

The authors declare no competing financial interest.

References

[1] Rose S, Borchert O, Mitchell S, Connelly S. *Zero Trust Architecture*. NIST Special Publication 800-207. National Institute of Standards and Technology; 2020. Available from: https://doi.org/10.6028/NIST.SP.800-207.

[2] Zhang M, Wang L, Jajodia S, Singhal A, Albanese M. Network diversity: a security metric for evaluating the resilience of networks against zero-day attacks. *IEEE Transactions on Information Forensics and Security*. 2016; 11(5): 1071–1086. Available from: https://doi.org/10.1109/TIFS.2016.2516916.

[3] Tilman D. Biodiversity: population versus ecosystem stability. *Ecology*. 1996; 77(2): 350–363. Available from: https://doi.org/10.2307/2265614.

[4] Littlewood B, Strigini L. Redundancy and diversity in security. In Proceeding of the 9th European Symposium on Research Computer Security; 2004 Sep. 13–15; Sophia Antipolis, Francep. p. 423–438. Available from: https://doi.org/10.1007/978-3-540-30108-0_26.

[5] Forrest S, Somayaji A, Ackley DH. Building diverse computer systems. In Proceedings of the 6th Workshop on Hot Topics in Operating Systems (HotOS VI); 1997 May 5–6; Cape Cod, MA, USA. p. 67–72. Available from: https://doi.org/10.1109/HOTOS.1997.595185.

[6] Laszka A, Abbas W, Vorobeychik Y, Koutsoukos X. Integrating redundancy, diversity, and hardening to improve security of industrial internet of things. *Cyber-Physical Systems*. 2020; 6(1): 1–32. Available from: https://doi.org/10.1080/23335777.2019.1624620.

[7] Cleghorn L. Network defense methodology: a comparison of defense in depth and defense in breadth. *Journal of Information Security*. 2013; 4(3): 144–149. Available from: https://doi.org/10.4236/jis.2013.43017.

[8] Jajodia S, Ghosh AK, Swarup V, Wang C, Wang XS. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. Springer New York, NY; 2011. Available from: https://doi.org/10.1007/978-1-4614-0977-9.

[9] Buczak AL, Guven E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys and Tutorials*. 2016; 18(2): 1153–1176. Available from: https://doi.org/10.1109/COMST.2015.2494502.

[10] Biggio B, Roli F. Wild patterns: ten years after the rise of adversarial machine learning. *Pattern Recognition*. 2018; 84: 317–331. Available from: https://doi.org/10.1016/j.patcog.2018.07.023.

[11] Sgandurra D, Lupu E. Evolution of attacks, threat models, and solutions for virtualized systems. *ACM Computing Surveys*. 2016;48(3): 1–38. Available from: https://doi.org/10.1145/2856126.

[12] Stouffer K, Pillitteri V, Lightman S, Abrams M, Hahn A. *Guide to Industrial Control Systems (ICS) Security*. NIST Special Publication 800-82 Revision 2. National Institute of Standards and Technology; 2015. Available from: https://doi.org/10.6028/NIST.SP.800-82r2.

[13] Okhravi H, Hobson T, Bigelow D, Streilein W. Finding focus in the blur of moving-target techniques. *IEEE Security and Privacy*. 2014; 12(2): 16–26. Available from: https://doi.org/10.1109/MSP.2013.137.

[14] Shannon CE. A mathematical theory of communication. *Bell System Technical Journal*. 1948; 27(3): 379–423. Available from: https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

[15] Boyd S, Vandenberghe L. *Convex Optimization*. Cambridge University Press; 2004.

[16] Avizienis A. The N-version approach to fault-tolerant software. *IEEE Transactions on Software Engineering*. 1985; SE-11(12): 1491–1501. Available from: https://doi.org/10.1109/TSE.1985.231893.

[17] Knight JC, Leveson NG. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*. 1986; SE-12(1): 96–109. Available from: 10.1109/TSE.1986.6312924.

[18] Lyu MR, editor. *Handbook of Software Reliability Engineering*. IEEE Computer Society Press and McGraw-Hill; 1996.

[19] Magurran AE. *Measuring Biological Diversity*. Blackwell Publishing; 2004.

[20] Jost L. Entropy and diversity. *Oikos*. 2006; 113(2): 363–375. Available from: https://doi.org/10.1111/j.2006.0030-1299.14714.x.

[21] Hill MO. Diversity and evenness: a unifying notation and its consequences. *Ecology*. 1973; 54(2): 427–432. Available from: https://doi.org/10.2307/1934352.

[22] Simpson EH. Measurement of diversity. *Nature*. 1949; 163: 688. Available from: https://doi.org/10.1038/163688a0.

[23] Pielou EC. The measurement of diversity in different types of biological collections. *Journal of Theoretical Biology*. 1966; 13: 131–144. Available from: https://doi.org/10.1016/0022-5193(66)90013-0.

[24] Lakhina A, Crovella M, Diot C. Diagnosing network-wide traffic anomalies. *ACM SIGCOMM Computer Communication Review*. 2004; 34(4): 219–230. Available from: https://doi.org/10.1145/1030194.1015492.

[25] Wang L, Singhal A, Jajodia S. Toward measuring network security using attack graphs. In 14th ACM Conference on Computer and Communications Security; 2007 Oct. 29; Alexandria, VA, USA. p. 49–54. Available from: https://doi.org/10.1145/1314257.1314273.

[26] Sheyner O, Haines J, Jha S, Lippmann R, Wing JM. Automated generation and analysis of attack graphs. In Proceedings of the 2002 IEEE Symposium on Security and Privacy; 2002 May 12–15; Berkeley, CA, USA. p. 273–284. Available from: https://doi.org/10.1109/SECPRI.2002.1004377.

[27] Mell P, Scarfone K, Romanosky S. A complete guide to the Common Vulnerability Scoring System (CVSS). 2006. Available from: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51198 [Accessed 20 March 2026].

[28] Forum of Incident Response and Security Teams. *Common Vulnerability Scoring System v3.1: Specification Document*. FIRST; 2019. Available from: https://www.first.org/cvss/v3.1/specification-document [Accessed 20 March 2026].

[29] National Institute of Standards and Technology. National Vulnerability Database (NVD). Available from: https://nvd.nist.gov [Accessed 20 March 2026].

[30] Anwar A, Abusnaina A, Chen S, Li F, Mohaisen D. Cleaning the NVD: comprehensive quality assessment, improvements, and analyses. *IEEE Transactions on Dependable and Secure Computing*. 2021; 19(6): 4255-4269. Available from: https://doi.org/10.1109/TDSC.2021.3125270.

[31] Sommer R, Paxson V. Outside the closed world: on using machine learning for network intrusion detection. In 2010 IEEE Symposium on Security and Privacy; 2010 May 16–19; Berkeley, CA, USA. p. 305–316. Available from: https://doi.org/10.1109/SP.2010.25.

[32] Kuo W, Prasad VR. An annotated overview of system-reliability optimization. *IEEE Transactions on Reliability*. 2000; 49(2): 176–187. Available from: https://doi.org/10.1109/24.877336.

[33] Coit DW, Smith AE. Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Transactions on Reliability*. 1996; 45(2): 254–260. Available from: https://doi.org/10.1109/24.510811.

[34] Souppaya M, Morello J, Scarfone K. *Application Container Security Guide*. NIST Special Publication 800-190. National Institute of Standards and Technology; 2017. Available from: https://doi.org/10.6028/NIST.SP.800-190.

[35] Dragoni N, Giallorenzo S, Lluch-Lafuente A, Mazzara M, Montesi F, Mustafin R, et al. Microservices: yesterday, today, and tomorrow. In: Mazzara M, Meyer B. (eds.). *Present and Ulterior Software Engineering*. Springer; 2017. p. 195–216. Available from: https://doi.org/10.1007/978-3-319-67425-4_12.

[36] Popov G, Popova A. Application of system diversity for increasing security and reliability of distributed systems. In Proceedings of the XXXI International Scientific Conference Electronics (ET); 2022 September 13–15; Sozopol, Bulgaria. Available from: https://doi.org/10.1109/ET55967.2022.9920304.

[37] Popov G, Popova A. Structural model of an innovative information system for prevention and detection of financial customs violations. In 2020 III International Conference High Technology for Sustainable Development (HiTech); 2020 October 8–9; Sofia, Bulgaria. Available from: https://doi.org/10.1109/HiTech51434.2020.9364001.

[38] Andonova AS, Atanasova NG. Accelerated reliability growth of electronic devices. In 27th International Spring Seminar on Electronics Technology: Meeting the Challenges of Electronics Technology Progress; 2004 May 13–16; Bankya, Bulgaria. p. 242–246. Available from: https://doi.org/10.1109/ISSE.2004.1490427.

[39] Swanson M, Bowen P, Phillips AW, Gallup D, Lynes D. *Contingency Planning Guide for Federal Information Systems*. NIST Special Publication 800-34 Revision 1. National Institute of Standards and Technology; 2010. Available from: https://doi.org/10.6028/NIST.SP.800-34r1.

[40] Scarfone K, Hoffman P. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publication 800-41 Revision 1. National Institute of Standards and Technology; 2009. Available from: https://doi.org/10.6028/NIST.SP.800-41r1.

[41] Scarfone K, Souppaya M, Hoffman P. *Guide to Security for Full Virtualization Technologies*. NIST Special Publication 800-125. National Institute of Standards and Technology; 2011. Available from: https://doi.org/10.6028/NIST.SP.800-125.

[42] Weinberger J, Saxena P, Akhawe D, Finifter M, Shin R, Song D. A systematic analysis of XSS sanitization in web application frameworks. In 16th European Symposium on Research in Computer Security; 2011 September 12–14; Leuven, Belgium. p. 150–171. Available from: https://doi.org/10.1007/978-3-642-23822-2_9.

[43] Ferraiolo DF, Kuhn DR, Chandramouli R. *Role-based access control*. Boston: Artech House; 2003.

[44] Boyens J, Paulsen C, Moorthy R, Bartol N. *Supply Chain Risk Management Practices for Federal Information Systems and Organizations*. NIST Special Publication 800-161. National Institute of Standards and Technology; 2015. Available from: https://doi.org/10.6028/NIST.SP.800-161.