RESEARCH MONOGRAPH

An Author's Framework for Applying Machine Learning in Multi-Layer Enterprise Cybersecurity Systems

Pre-emptive Architecture, Combined Labeling, and Ensemble Integration

Rustam Kolchin
Head of Cybersecurity Software Product Development
SoftLine PJSC, Almaty, Kazakhstan
ORCID: 0009-0002-8054-4552

The American Journal of Engineering and Technology
ISSN: 2689-0984
PUBLISHED 2026

An Author's Framework for Applying Machine Learning in Multi-Layer Enterprise Cybersecurity Systems

Pre-emptive Architecture, Combined Labeling, and Ensemble Integration

Rustam Kolchin

Head of Cybersecurity Software Product Development

SoftLine PJSC, Almaty, Kazakhstan

ORCID: 0009-0002-8054-4552

PUBLICATION INFO: The American Journal of Engineering and Technology (ISSN: 2689-0984)

ISBN: - 978-1-957653-72-3

CROSSREF DOI: - https://doi.org/10.37547/tajet/book-26-04

PUBLISHED DATE: - 05 August 2026

Preface

This book sets out a framework for applying machine learning in corporate cybersecurity that the author developed across fifteen years of building and operating defensive systems. The framework did not begin as a theory. It began as a set of recurring engineering decisions, made and remade across several security products, that turned out to matter more than the choice of any individual model: when a model is allowed to act, what evidence it is taught from, and where it sits among the other controls in a defense. Naming those decisions as principles, and showing that their combination can be verified, is the purpose of the book.

The intended reader is a practitioner. The book is addressed to the engineers, architects, and security leaders who design machine-learning systems for enterprise defense and who must make these decisions whether or not they are made deliberately. For that reader the book aims to be useful rather than exhaustive: it states each principle, gives the architectural requirements it imposes, and grounds it in a system that was built and measured. The quantitative core of the book is an email security platform whose results were established in a peer-reviewed study, and the book is careful throughout to separate what has been measured on that system from what is projected for the adjacent domains where the same principles apply.

The book treats the published literature as a working tool rather than as a survey to be completed. Sources are cited where they establish a mechanism, corroborate a claim, or mark a limit, and the reader who wishes to go deeper will find the trail. The aim is a book that reads as the distilled judgment of practice, supported by the literature, rather than as a compilation of it.

List of Abbreviations

Abbreviation	Expansion
API	Application Programming Interface
AUC	Area Under the (ROC) Curve
BEC	Business Email Compromise
CTI	Cyber Threat Intelligence
DKIM	DomainKeys Identified Mail
DMARC	Domain-based Message Authentication, Reporting and Conformance
DNS	Domain Name System
DNSBL	DNS-based Blackhole List
DRPS	Digital Risk Protection System
EASM	External Attack Surface Management
EPSS	Exploit Prediction Scoring System
F1	F1-score (harmonic mean of precision and recall)
IDS	Intrusion Detection System
ML	Machine Learning
MX	Mail Exchange (DNS record)
NLP	Natural Language Processing
PAM	Privileged Access Management
RBL	Real-time Blackhole List
ROC	Receiver Operating Characteristic
SEG	Secure Email Gateway
SIEM	Security Information and Event Management
SMOTE	Synthetic Minority Over-sampling Technique
SMTP	Simple Mail Transfer Protocol
SOAR	Security Orchestration, Automation, and Response
SOC	Security Operations Center
SPF	Sender Policy Framework
TF-IDF	Term Frequency-Inverse Document Frequency
UEBA	User and Entity Behavior Analytics
ZTNA	Zero Trust Network Access

Content

Introduction 6
Chapter 1. Theoretical Foundations of Machine Learning in Enterprise Cybersecurity 10
1.1 The evolution of machine learning in security tasks 10
1.2 Categories of security tasks addressed by machine learning 12
1.3 Architectural patterns for integrating machine learning into defenses 13
1.4 Limitations of current approaches 15
1.5 Problem statement 16
Chapter 2. Pre-emptive Architecture as the First Principle 18
2.1 The concept of pre-emptive defense 18
2.2 Architectural requirements of pre-emptive systems 20
2.3 Applying the principle to email protection 21
2.4 Applying the principle to external digital-risk protection 23
2.5 Applying the principle to vulnerability management 24
Chapter 3. Combined Data Labeling as the Second Principle 26
3.1 The problem of systematic bias in homogeneous labeling 26
3.2 The methodology of combined labeling 28
3.3 Implementing combined labeling in machine-learning pipelines 29
3.4 Comparative analysis of labeling strategies 31
Chapter 4. Ensemble Integration in Multi-Layer Defense 33
4.1 The architectural model of multi-layer defense 33
4.2 Weighted aggregation in the decision engine 34
4.3 Integrating machine-learning and non-machine-learning controls 36
4.4 Robustness of the ensemble architecture to attacks 37
4.5 Integration with monitoring and response 38
Chapter 5. End-to-End Verification of the Framework on a Practical Case 40
5.1 The object of verification 40
5.2 Architectural realization of the three principles 41
5.3 Experimental methodology 42
5.4 Results 43
5.5 Transferability of the framework to adjacent security tasks 47
Conclusion 49
Glossary 51
Bibliography 52

Introduction

Machine learning has become a standard component of corporate cybersecurity, yet most of the published work and most production deployments treat it as a problem of model selection: which classifier, which feature set, which accuracy figure. This monograph takes a different position. The effectiveness of machine learning in enterprise defense is governed less by the choice of algorithm than by three architectural decisions that surround it, namely when the model is allowed to act relative to the threat, how its training data are labeled, and where the model sits within the larger system of controls. These decisions are made before any classifier is trained, and they bound the performance that any classifier can reach. The central argument advanced here is that machine learning in corporate cybersecurity should be designed as an architectural discipline rather than as a modeling exercise.

The argument is grounded in fifteen years of applied practice in enterprise information security and in the development of operational security products. The author leads cybersecurity software product development at SoftLine PJSC in Almaty, where a portfolio of defensive systems was designed around a common set of architectural principles. One of those systems, an email security platform discussed in detail in the final chapter, has been documented in a peer-reviewed study and supplies the verified evidence on which the framework rests (Kolchin, 2025). The remaining systems in the portfolio illustrate the same principles across adjacent security domains and are described qualitatively, as design examples rather than as measured results.

The motivation for an architectural reframing is concrete. Electronic mail remains the dominant delivery channel for spam, phishing, business email compromise, and malware in enterprise environments, and the volume of phishing activity recorded by industry monitoring bodies has continued to rise across recent reporting periods (Anti-Phishing Working Group, 2023). The three attack categories share a single traffic stream but demand different detection logics, which makes single-modality controls structurally inadequate (Ahmed et al., 2022). Business email compromise, in particular, exploits trusted communication relationships and resists conventional content filtering, so that detection depends on combining heterogeneous signals rather than on scanning message text alone (Atlam & Oluwatimilehin, 2023). The threat landscape is also changing in kind. Large language models have lowered the cost of producing fluent, individually tailored phishing at scale, and measurement studies report that machine-generated messages already account for a substantial share of observed malicious traffic (Hao et al., 2025). A defense whose machine learning component is trained on yesterday's lexical patterns degrades precisely when the adversary's content quality rises.

Against this background, three limitations recur across the literature and across industrial practice. The first is a reactive posture. Machine learning is typically applied after a threat has been delivered or activated: mail is analyzed after it reaches the server, phishing domains are flagged after the first complaints, vulnerabilities are scored after disclosure. Each case opens a window of compromise in which the attack can succeed before detection occurs. Research on spammer strategy documents systematic exploitation of exactly these detection latencies, together with the dataset-shift problem that erodes model accuracy as threats evolve (Jáñez-Martino et al., 2022). The second limitation is homogeneous labeling. Most production models are trained on a single source of ground truth, whether user feedback, sandbox verdicts, or threat intelligence, and each source carries a characteristic bias: user feedback under-represents the phishing that succeeded, sandbox labels apply only to attachment-bearing messages, and threat intelligence lags newly emerging campaigns. A model inherits the blind spots of whichever source labeled its data. The third limitation is isolated deployment. Machine learning is treated as a self-contained product, a standalone anti-spam engine or a separate behavioral analytics module, which ignores the layered nature of enterprise defense. Modern security architecture follows the zero

trust principle of continuous per-request verification across independent controls (Syed et al., 2022; Kang et al., 2023), and a model that operates in isolation cannot benefit from, or contribute to, that layered evidence.

The framework proposed in this monograph addresses these limitations through three connected principles. The first is pre-emptive architecture: the point at which machine learning acts is moved to the stage before delivery or activation, so that messages are filtered before the SMTP handshake completes, phishing domains are identified before they are weaponized in campaigns, and vulnerabilities are prioritized before they are exploited. This compresses the window of compromise to values that a reactive design cannot reach. The second principle is combined labeling: training data are labeled through several independent evidence streams at once, so that each source compensates for the blind spots of the others and the systematic bias of single-source ground truth is removed. The third principle is ensemble integration: machine learning is embedded as one weighted layer among several, and the decision to deliver, quarantine, or block is taken by aggregating signals from reputation filtering, protocol authentication, content analysis, dynamic sandboxing, and the model itself. This removes the brittleness of a design that an attacker can defeat by evading a single control. Figure 0.1 shows how the three principles compose into a single framework and how that framework is realized as a multi-layer, pre-emptive pipeline terminating in a weighted decision engine.

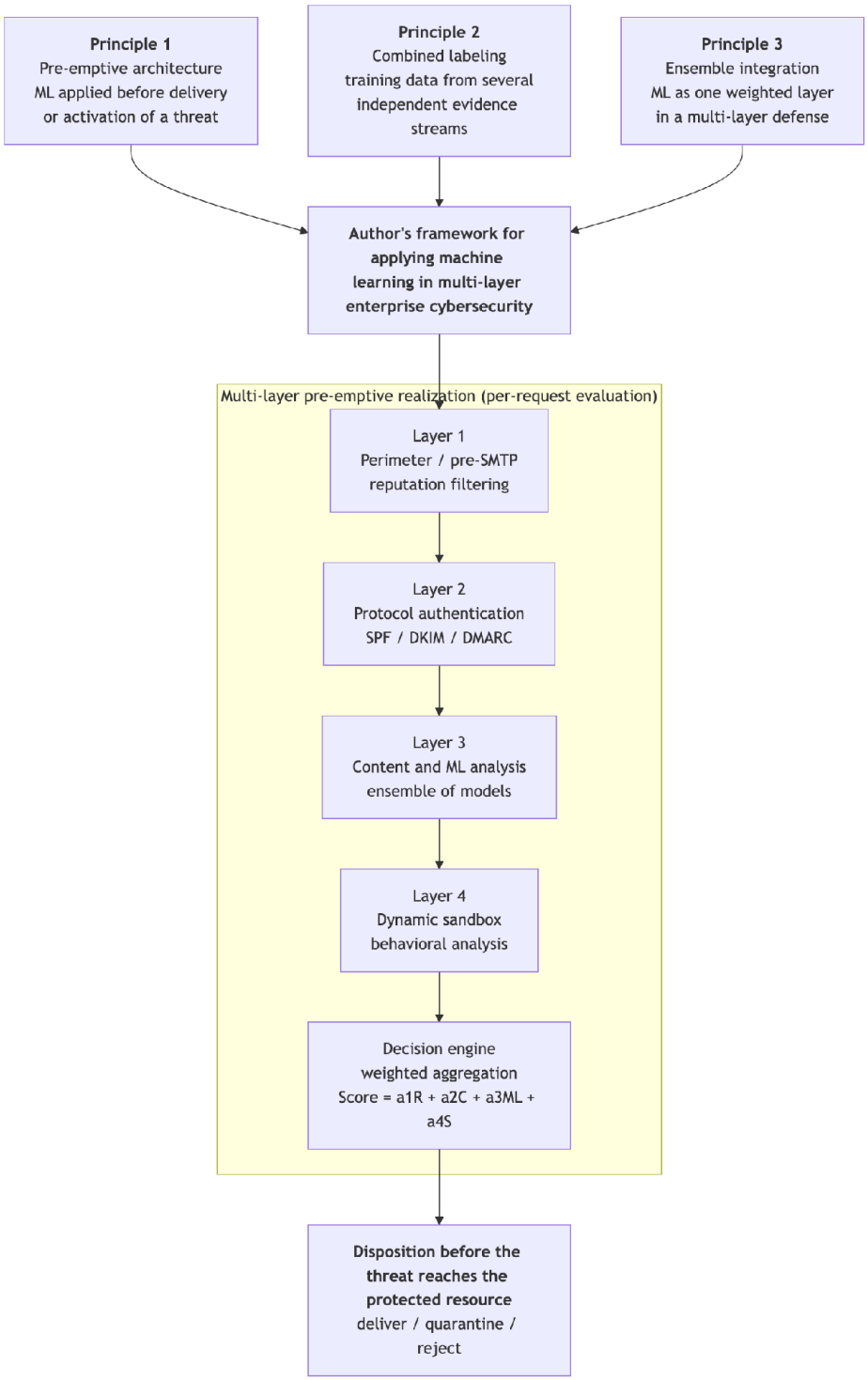

Figure 0.1. *The author's framework: three methodological principles and their realization as a multi-layer, pre-emptive architecture with a weighted decision engine. Source: author's construction.*

The three principles are individually present, in scattered form, in prior work; their integration into a single methodology with demonstrated verification is the contribution claimed here. That verification is not hypothetical. The email security platform built on these principles was evaluated on 3.6 million messages collected over six months from a corporate infrastructure of approximately 2,500 users. Its machine learning pipeline reached an F1-score of 0.985 and an area under the ROC curve of 0.996, an improvement of 7.5 percentage points over the conventional secure email gateway baseline of 0.91 measured on the same data, and its production deployment reduced spam, phishing, and malware

prevalence by 89.4, 92.9, and 97.6 percent respectively (Kolchin, 2025). These figures, examined in full in the final chapter, are the empirical anchor of the framework rather than a marketing claim about a product.

The monograph pursues six objectives. It analyzes the limitations of current machine-learning-in-security practice; it formulates the three principles of the framework; it develops each principle with its architectural requirements and design consequences; it demonstrates an end-to-end verification of the framework on a documented case; it maps the transferability of the principles to adjacent security tasks; and it identifies directions for further development. The scientific novelty lies in the integration. No published source presents pre-emptive application, combined labeling, and ensemble integration as a single coherent methodology supported by verified results across a complete system. Individual principles appear in isolation across the literature, but their combination into an author's framework, with the verification that practice makes possible, constitutes an independent contribution to industrial security engineering.

The work is organized into five chapters. Chapter 1 establishes the theoretical foundations, tracing the evolution of machine learning in security tasks, the categories of problem it addresses, the architectural patterns through which it is integrated, and the limitations that motivate the framework. Chapters 2, 3, and 4 develop the three principles in turn: pre-emptive architecture, combined labeling, and ensemble integration, each with its requirements, design decisions, and applications across email defense, external digital-risk protection, and vulnerability management. Chapter 5 presents the end-to-end verification of the framework on the documented email security case, including the experimental methodology, the confirmed performance, and an analysis of how the principles transfer to neighboring tasks. The conclusion consolidates the contribution and outlines where the framework can be extended, including federated learning, defense against adversarial machine learning, the integration of large-language-model components, and post-quantum considerations in labeling and trust.

Chapter 1. Theoretical Foundations of Machine Learning in Enterprise Cybersecurity

The framework proposed in this monograph rests on a reading of how machine learning entered enterprise defense, what classes of problem it now solves, and how it is wired into operational systems. This chapter sets that foundation. It traces the progression from signature matching to statistical and learned models, organizes the security tasks that machine learning addresses, examines the architectural patterns through which models are integrated into defensive systems, and identifies the structural limitations that the three principles of the framework are designed to remove. The chapter closes by stating the problem the rest of the book answers.

1.1 The evolution of machine learning in security tasks

Enterprise threat detection began as an exercise in pattern matching. Early antivirus engines, intrusion detection systems, and spam filters compared incoming artifacts against a database of known-bad signatures, and a defense was only as current as its last signature update. This design is captured in the standard division of intrusion detection into signature-based detection, which recognizes previously catalogued attacks, and anomaly-based detection, which models normal behavior and flags departures from it (Khraisat et al., 2019). Signature methods are precise against known threats and nearly blind to novel ones; anomaly methods generalize to the unseen at the cost of higher false-positive rates. The history of machine learning in security is, in large part, the history of shifting weight from the first approach to the second without surrendering the precision that operations require.

The first move beyond static signatures was statistical. Bayesian spam filtering, reputation scoring, and frequency-based heuristics introduced probability into the delivery decision and allowed a system to act on likelihood rather than on exact match. The second move was the adoption of classical supervised learning, in which engineered features drawn from headers, payloads, and network flows are fed to decision trees, support vector machines, and tree ensembles. Surveys of the decade that followed document the displacement of hand-written rules by trained classifiers across intrusion detection, malware analysis, and email security, together with a recurring finding that conventional, signature-bound systems fail against polymorphic and previously unseen attacks (Shaukat et al., 2020). Support vector machines and kernel methods were prominent in this phase, and comprehensive surveys document their wide application before deep learning displaced them as the default for high-dimensional inputs (Cervantes et al., 2020). The third move was representational. Deep learning removed part of the manual feature-engineering burden by learning representations directly from raw or lightly processed inputs, and reviews of its application to cybersecurity catalogue its spread across network intrusion detection, malware classification, and traffic analysis (Mahdavifar & Ghorbani, 2019; Liu & Lang, 2019; Vinayakumar et al., 2019; Ahmad et al., 2020). Anomaly-based intrusion detection in particular has been surveyed extensively as deep architectures matured (Aldweesh et al., 2019; Gamage & Samarabandu, 2020), and the general progress of deep-learning concepts and architectures that these security applications drew on is itself the subject of broad review (Alzubaidi et al., 2021). Around the same time, the field began to describe itself in data-science terms, framing detection as the extraction of incident patterns from security data and the construction of data-driven models on top of them (Sarker et al., 2020).

A fourth transition is now underway, and it is the one this monograph is concerned with. The earlier shifts changed the model; the current shift changes the model's place in the system. Detection is moving from isolated classifiers, each guarding a single surface, toward integrated architectures in which a model

is one evidence source among several and the decision is taken by combining sources. The email security literature illustrates the turn: bulk spam, credential phishing, and malware delivery travel through one channel yet demand different detection logics, which makes any single-modality control structurally incomplete (Ahmed et al., 2022). Table 1.1 summarizes the four generations and the limitation that motivated each successor.

Generation	Detection basis	Representative tasks	Defining limitation
1. Signature and rule	Exact match against known-bad catalogue	Antivirus, early IDS, blocklist spam filters	Blind to novel and zero-day threats; update latency
2. Statistical and classical ML	Probability from engineered features	Bayesian spam scoring, reputation, tree/SVM classifiers	Feature engineering effort; limited semantic reach
3. Deep learning and ensembles	Learned representations; combined learners	Network IDS, malware classification, phishing NLP	Data hunger; opacity; adversarial fragility
4. Integrated architectures	Model as one weighted layer among controls	Multi-layer email defense, zero-trust evidence aggregation	Requires architectural design, not model choice alone

Table 1.1. Four generations of machine learning in enterprise security and the limitation that motivated each transition. Source: author's synthesis based on Khraisat et al. (2019), Shaukat et al. (2020), Mahdavifar and Ghorbani (2019), and Sarker et al. (2020).

The generational arc is visible within a single domain. Email defense began with signature and blocklist filtering, added Bayesian and reputation scoring, then incorporated supervised classifiers over header, content, and url features, and now combines those classifiers with protocol authentication and dynamic analysis inside layered gateways. Comparative evaluations of machine-learning algorithms for phishing detection report that gradient-boosted trees and related ensembles match or exceed more expensive deep models on structured url and domain features (Almujahid et al., 2024; Kumar et al., 2024), while deep and transformer architectures add discriminative power on message text that lexical features alone do not capture (Altwaijry et al., 2024; Otieno et al., 2023; Devlin et al., 2019). The same trajectory is visible in spam detection and in malware classification, where comprehensive surveys trace the move from hand-crafted features to learned representations (Karim et al., 2019; Gopinath & Sethuraman, 2022; Bensaoud et al., 2024), and where deep architectures such as densely connected networks have been applied to malware classification with strong reported results (Hemalatha et al., 2021). The conclusion the present framework draws from this body of work is not that one model family wins, but that the families are complementary and belong inside the same system, each contributing the signal it captures best.

The progression is cumulative rather than substitutive. Production systems still run signature and reputation checks, still compute statistical scores, and now add learned classifiers, because each generation answers a question the others answer poorly. What distinguishes the fourth transition from the three before it is the unit of design. The first three improved the model in place: a better filter, a better feature set, a better network. The fourth changes what is being designed, from a model to a system in which the model is a component with a defined role, a defined input, and a defined weight in a combined

decision. This is the difference between asking which classifier to deploy and asking how detection evidence should be produced, combined, and acted upon. The framework developed here treats the coexistence of generations as the normal case and asks how a learned model should be positioned among the older controls rather than whether it should replace them. That positioning question is the subject of the three principles, and it is why the book is organized around architecture rather than around algorithms.

1.2 Categories of security tasks addressed by machine learning

Machine learning in enterprise defense is not one task but a family of them, and the architectural decisions that matter differ by family. Four categories cover most production use.

The first is threat classification, the assignment of an artifact to a class such as legitimate, spam, phishing, or malware. The formulation may be binary, multiclass, or hierarchical, and the input may be message text, a uniform resource locator, an attachment, or a combination. Natural-language and lexical features carry much of the signal: a systematic review of phishing email detection through natural-language processing identifies content representation and url-derived features as the dominant predictors across the literature (Salloum et al., 2022). Business email compromise sits at the difficult end of this category, because it imitates legitimate organizational correspondence and offers little malicious content to classify, which is why systematic reviews of the problem stress the combination of behavioral and relational signals over text alone (Atlam & Oluwatimilehin, 2023). Ensemble classifiers that combine heterogeneous base learners are now the standard answer for phishing classification, since no single learner captures the full feature space (Innab et al., 2024).

The second category is anomaly and behavior detection, which models the normal activity of a user, host, or account and flags deviations. User and entity behavior analytics applies this idea to insider risk and account compromise, profiling baseline behavior and scoring departures from it (Khan et al., 2022). The category is operationally distinct from classification because it has no fixed catalogue of bad classes; it learns what is normal and treats the rest as suspect. Its central difficulty is the cost of false positives, since legitimate behavior is diverse and a naive anomaly model floods analysts with benign deviations. The organizational stakes are high: studies of insider threat document its disproportionate impact on critical business functions relative to its frequency (Saxena et al., 2020).

The third category is risk prioritization, the ranking of items by expected harm so that finite remediation effort is spent where it matters. Vulnerability management is the canonical case. Severity scores alone are weak predictors of which vulnerabilities will actually be exploited, and data-driven exploit prediction improves on them by incorporating signals that correlate with real-world attacker behavior (Jacobs et al., 2023). Ensemble models trained on vulnerability and threat-intelligence features can estimate the probability of exploitation before it occurs, converting a backlog of undifferentiated findings into a ranked queue (Fang et al., 2020). This category supplies one of the clearest illustrations of the pre-emptive principle developed in Chapter 2, because its entire value lies in acting before exploitation.

Each of the first three families rewards a different architectural emphasis, and the comparative literature makes the differences concrete. In threat classification, performance gains come less from a single superior model than from widening the feature space: studies that compare algorithms on phishing detection find that the ranking of models is unstable across datasets, while the inclusion of url, header, and reputation features alongside content is a consistent source of improvement (Almujahid et al., 2024; Alnemari & Alshammari, 2023). This is an argument for combining modalities, not for chasing a single best classifier. In anomaly detection, the architectural pressure runs toward controlling the false-positive rate, because the absence of a fixed bad-class catalogue means the model must distinguish rare-but-benign from rare-and-malicious; behavior analytics systems manage this by scoring deviations

continuously and escalating only aggregated risk rather than individual events (Khan et al., 2022). In prioritization, the pressure runs toward calibration and timing, because a ranked queue is only useful if its probabilities are trustworthy and available before the attacker acts; exploit-prediction models are evaluated precisely on whether their estimates anticipate real exploitation rather than restate static severity (Jacobs et al., 2023; Fang et al., 2020).

The fourth category is natural-language analysis of unstructured threat data, which applies text models to sources outside the protected perimeter such as forums, marketplaces, and leak sites. The same content-analysis techniques that classify phishing also classify and rank threat chatter, and they have acquired new urgency as adversaries adopt large language models to generate fluent malicious text at scale (Hazell, 2023). Controlled studies of commercial language models confirm that such systems can be induced to produce convincing phishing content with little manual effort, which raises both the volume and the average quality of malicious text that a defense must process (Roy et al., 2024). The defensive response is itself a classification task, now applied to adversary-generated language, and it inherits all of the data-bias and adversarial-fragility problems discussed in Section 1.4. Table 1.2 organizes the four families with their formulations, applications, and representative methods.

Task family	**Formulation**	**Example security application**	**Representative methods**
Threat classification	Binary / multiclass / hierarchical labeling	Spam, phishing, malware, BEC triage	TF-IDF + tree ensembles; transformer text models
Anomaly and behavior detection	Deviation from learned baseline	UEBA, insider risk, account compromise	Autoencoders, sequence models, clustering
Risk prioritization	Ranking by expected harm	Vulnerability and threat prioritization	Gradient-boosted exploit prediction
NLP of unstructured threat data	Classification / ranking of free text	Leak monitoring, threat-chatter triage	Language models, lexical and entropy features

Table 1.2. A taxonomy of enterprise security tasks addressed by machine learning. Source: author's synthesis based on Salloum et al. (2022), Khan et al. (2022), Jacobs et al. (2023), and Fang et al. (2020).

The families share statistical machinery but differ in the architectural question each raises. Classification asks which features and which combination of learners. Anomaly detection asks how to bound false positives. Prioritization asks how to act before the event. Unstructured analysis asks how to keep pace with adversary-generated content. A framework that treats machine learning as architecture, rather than as model selection, must answer all four, and it does so through the same three principles applied with task-specific emphasis.

1.3 Architectural patterns for integrating machine learning into defenses

Once a model exists, a system designer faces a separate decision: where the model sits and how its output is combined with everything else the defense knows. Three integration patterns recur, and they differ in robustness, latency, and failure behavior.

The standalone pattern deploys a model as a self-contained control, such as an independent anti-phishing service or a separate malware classifier. It is simple to build and to reason about, and surveys of machine-learning malware detection treat the standalone classifier as the default unit of analysis (Singh & Singh, 2020). Its weakness is structural: a standalone model exposes a single decision surface, so an adversary who defeats that one model defeats the control. The chained pattern arranges several controls in sequence, each able to reject an artifact, which improves coverage but couples the stages, since a miss early in the chain removes an artifact from later scrutiny and the chain inherits the latency of its slowest stage.

The ensemble pattern combines the outputs of several models, or of models and non-model controls, into a single decision. Ensemble learning is mature as a modeling technique. Bagging, boosting, and stacking each combine base learners to reduce variance or bias, and the family includes random forests (Breiman, 2001), gradient boosting, and the regularized gradient-boosted tree implementations that dominate tabular security features (Chen & Guestrin, 2016; Mienye & Sun, 2022). The architectural insight of this monograph is to apply the ensemble idea one level higher, not only across models of the same kind but across independent modalities: reputation, authentication, content, and dynamic analysis. Dual-layer designs that combine content classification with url and metadata analysis already report lower false-negative rates than content-only classification, which demonstrates the value of crossing modalities rather than stacking homogeneous learners (Doshi et al., 2023). Stacking heterogeneous base learners likewise improves accuracy over any single learner in spam classification (Adnan et al., 2024). Table 1.3 compares the three patterns.

Integration pattern	**Decision locus**	**Robustness to single-point evasion**	**Latency behavior**	**Representative use**
Standalone	One model	Low: one surface defeats the control	Single-model cost	Independent anti-phishing or AV classifier
Chained	Sequential gates	Moderate: early miss removes later scrutiny	Sum of stage latencies	Gateway stages applied in order
Ensemble (cross-modality)	Aggregated evidence	High: bypassing one layer leaves others	Parallelizable across layers	Multi-layer evidence aggregation

Table 1.3. Architectural patterns for integrating machine learning into enterprise defenses. Source: author's synthesis based on Mienye and Sun (2022), Doshi et al. (2023), and Adnan et al. (2024).

The choice among patterns is not only about accuracy; it is about failure behavior. A standalone control fails closed or open as a unit: when it is wrong, nothing else compensates. A chained control fails forward: an artifact dismissed by an early stage never reaches a later one that might have caught it, so the chain is only as alert as its first inattentive gate. A cross-modality ensemble fails gracefully: when one layer is blinded, whether by evasion, outage, or an unfamiliar attack, the remaining layers still contribute evidence and the aggregate decision degrades rather than collapses. This property, examined in detail in Chapter 4, is the architectural reason the framework places the model inside an ensemble of independent modalities rather than in a single decision path. It also changes how the model must be built, because a layer that contributes to a weighted decision is tuned for calibrated, comparable output rather than for a standalone verdict, and its threshold is set jointly with the other layers rather than in isolation.

The gap between research and practice is itself an architectural problem. A holistic account of the role of machine learning in cybersecurity attributes the slow operational uptake of otherwise strong models to a mismatch between how models are evaluated in isolation and how they must behave inside a layered system under adversarial pressure (Apruzzese et al., 2022). A classifier reporting high accuracy on a balanced benchmark may be unusable in production, where classes are imbalanced, the threat distribution shifts week to week, and a single percentage point of false positives translates into thousands of misrouted legitimate messages per day. The ensemble-integration principle of Chapter 4 addresses that mismatch directly by making the model a calibrated contributor to a combined decision rather than an autonomous judge, and by evaluating it under the temporal and class-imbalance conditions that production imposes rather than under benchmark conditions that flatter it.

1.4 Limitations of current approaches

Three limitations recur across the categories and patterns above, and together they bound the performance that model choice alone can deliver.

The first is systematic bias in training data. A supervised model inherits the blind spots of whatever process labeled its examples, and in production security the labeling process is rarely neutral. The dataset-shift problem, in which the distribution of threats drifts away from the training distribution, is identified as a primary obstacle in spam detection, alongside the spammer strategies that deliberately accelerate that drift (Jáñez-Martino et al., 2022). When the labels themselves come from a single source, the bias compounds, a point that motivates the combined-labeling principle of Chapter 3.

The second limitation is adversarial fragility. The phenomenon was established for deep classifiers in general, where small, deliberately crafted input perturbations flip confident predictions (Goodfellow et al., 2014), and it has since been catalogued for the security setting in dedicated surveys of attacks and defenses (Qiu et al., 2019; Chakraborty et al., 2021). Machine-learning classifiers degrade under inputs crafted to deceive them, and the degradation can be severe: text classifiers that perform well on clean data lose substantial accuracy under adversarial perturbation, and surveys of text adversarial attacks catalogue both the attack families and the defenses, identifying adversarial training as the most durable countermeasure at a modest cost in clean accuracy (Han et al., 2022). Phishing detection models specifically have been shown to be vulnerable to adversarial manipulation of message content (Gholampour & Verma, 2023). A standalone model is most exposed to this failure mode, since the adversary need only defeat one decision surface; an ensemble that aggregates independent modalities raises the cost of evasion because content perturbation does not move reputation or authentication evidence.

The third limitation is dependence on the timeliness of external knowledge. Reputation feeds, blocklists, and threat intelligence age, and detection that leans on them inherits their latency. Evasion research shows that adversaries exploit exactly these update gaps, for example by timing campaigns to the window before a signature or reputation entry exists, and malware uses sandbox-evasion techniques to avoid generating the behavioral evidence on which detection depends (Afianian et al., 2019). Evasive samples delay or suppress their malicious behavior until they detect a real execution environment, so that a time-limited sandbox observes nothing; countermeasures that extend or instrument dynamic analysis recover some of this lost evidence but at additional cost (Aboaoja et al., 2023; Liu et al., 2022). The pre-emptive principle of Chapter 2 attacks this limitation by moving the decision point earlier, so that the defense acts on structural and behavioral signals that do not require a prior catalogue entry, and by treating dynamic analysis as one bounded layer among several rather than as the sole arbiter of an attachment.

A fourth limitation cuts across the other three: the opacity of learned models and its effect on operational trust. A security analyst must be able to justify why a message was blocked or an account suspended, and a model that returns a probability without an account of which evidence produced it is hard to defend in an incident review and hard to tune when it errs. The research-practice gap noted above is partly a trust gap, since operators are reluctant to cede an irreversible action to a control they cannot interrogate (Apruzzese et al., 2022). The architectural answer is again positional rather than algorithmic. When the model is one weighted layer within an aggregated decision, its contribution is visible alongside reputation, authentication, and sandbox evidence, the decision can be traced to the layers that drove it, and the model is relieved of the burden of being the sole and unexplained judge.

These limitations are not defects of particular models; they are properties of how models are positioned. Each is addressed not by a better classifier but by an architectural choice about when the model acts, how its data are labeled, and where it sits among other controls. That correspondence, one limitation answered by one principle, is the organizing logic of the three chapters that follow.

1.5 Problem statement

The preceding sections support a single claim. The performance of machine learning in enterprise cybersecurity is bounded less by the choice of algorithm than by three architectural decisions that surround it: the point in the threat lifecycle at which the model is allowed to act, the diversity of the evidence used to label its training data, and the model's position within a system of independent controls. Conventional practice answers these three questions in the costliest way, by acting after delivery, labeling from a single source, and deploying models in isolation. The result is the recurring pattern documented above, namely strong benchmark accuracy that does not survive contact with novel, adversarial, and evolving threats.

The framework advanced in this monograph answers the three questions differently. It applies machine learning before delivery or activation rather than after, a pre-emptive architecture that compresses the window of compromise. It labels training data through several independent evidence streams rather than one, a combined-labeling methodology that removes single-source bias. And it embeds the model as one weighted layer within a multi-layer defense rather than as a standalone judge, an ensemble integration that survives the evasion of any single control. These three principles align with the zero-trust direction of enterprise security, which replaces implicit trust with continuous per-request verification across independent checks (Syed et al., 2022; Kang et al., 2023), and which systematic review associates with reduced breach dwell time and contained lateral movement (Gambo & Almulhem, 2025). The principles are developed individually in Chapters 2 through 4 and verified together, on a documented system, in Chapter 5, where the design choices argued here are shown to produce a measurable improvement over the conventional baseline (Kolchin, 2025).

A word on the standard of evidence used in this book is in order before the principles are developed. The argument is addressed to practitioners who build and operate machine-learning security systems, and it is held to a practitioner's standard: a principle is credible to the degree that it can be realized in a deployable architecture and shown to change a measured outcome under production conditions. For that reason the three principles are not presented as abstractions and then illustrated with hypothetical examples. Each is stated, given its architectural requirements, and tied to a concrete realization, and all three are then verified jointly on a single system evaluated on millions of real messages with a temporal split that mirrors operational deployment. Where the book discusses systems for which production metrics are not available, it is explicit that the claim is one of design intent rather than measured result, and it confines quantitative verification to the case where the data exist. This discipline, separating what

is engineered from what is demonstrated, is itself part of the methodology, because a framework that cannot distinguish a design goal from a verified outcome cannot be trusted to guide either research or practice.

Chapter 2. Pre-emptive Architecture as the First Principle

The first principle of the framework concerns timing. It holds that the value of a machine-learning control depends, before anything else, on where in the lifecycle of a threat the control is permitted to act. Conventional practice places the model late, after a message has been delivered, a domain has been weaponized, or a vulnerability has been disclosed and exploited. The pre-emptive principle moves the model forward, to the stage before delivery or activation, and in doing so it changes the quantity that ultimately governs damage: the length of time during which an attack is live but unaddressed. This chapter defines the principle, derives the architectural requirements it imposes, and shows how it applies across three security domains, using the author's products as concrete design illustrations.

2.1 The concept of pre-emptive defense

A defensive posture can be characterized by the moment at which it commits to a decision relative to the threat lifecycle. That lifecycle has four stages relevant to enterprise defense: a threat is created, it reaches the protected perimeter, it is delivered or activated within the environment, and it produces impact. Three postures map onto this lifecycle. A reactive posture decides after delivery or activation, when the artifact is already inside the environment; it detects compromise rather than preventing it. A proactive posture anticipates threats through intelligence and hunting, narrowing the gap by acting on knowledge of adversary behavior, yet it frequently still operates on artifacts that have already arrived. A pre-emptive posture commits its decision before delivery or activation, at or before the perimeter, so that a malicious artifact is refused entry rather than detected after entry.

The distinction is not rhetorical. Reactive detection inherits a structural latency, because it can only recognize what has already been catalogued, and adversaries exploit the interval before a signature or reputation entry exists. The review of spammer strategy and the dataset-shift problem documents the deliberate exploitation of these update gaps as a core tactic rather than an incidental weakness (Jáñez-Martino et al., 2022). Malware compounds the problem by withholding its behavior until it confirms a real execution environment, so that the very evidence reactive controls depend on is suppressed during analysis (Aboaoja et al., 2023; Liu et al., 2022). The catalogue of dynamic-analysis evasion techniques shows how systematically this is done (Afianian et al., 2019). Each of these tactics targets the reactive assumption that a threat can be recognized once it has acted. The pre-emptive posture refuses that assumption by acting first.

The governing metric of the pre-emptive principle is the window of compromise, defined here as the interval between the moment a threat becomes capable of causing harm within the environment and the moment the defense neutralizes it. Under a reactive posture this window is open by construction: the threat is delivered, it acts, and only then is it detected, so the window spans the entire delivery-to-detection interval. Under a pre-emptive posture the window is closed before it opens, because the decision is taken at the perimeter and a refused artifact never enters the environment to begin its harmful action. Figure 2.2 illustrates the two decision points against the threat lifecycle and the window each leaves open.

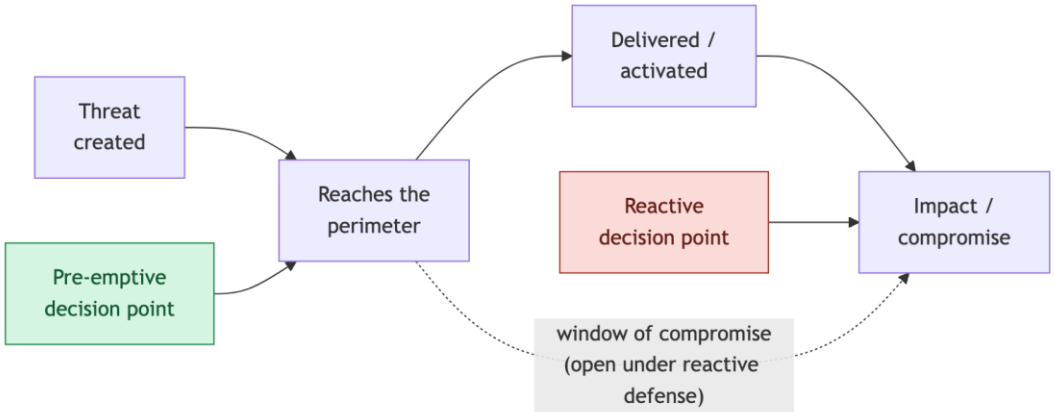

Figure 2.2. *The window of compromise under reactive and pre-emptive postures. A reactive control decides after the threat has been delivered and activated, leaving the delivery-to-impact interval open; a pre-emptive control decides at the perimeter, before delivery, and closes the window before it opens. Source: author's construction.*

Table 2.1 sets the three postures side by side. The progression from reactive to proactive to pre-emptive is a progression in how early the defense commits and in what evidence it is willing to act on. A reactive control waits for certainty and pays for it with an open window; a pre-emptive control accepts uncertainty in exchange for closing the window, and the rest of this chapter is concerned with how to make that trade safely.

Posture	**Decision point in lifecycle**	**Evidence relied on**	**Window of compromise**	**Characteristic difficulty**
Reactive	After delivery or activation	Signatures, reputation entries, observed behavior	Full delivery-to-detection interval	Blind to novel threats until catalogued
Proactive	Ahead of attack, by anticipation	Threat intelligence, hunting hypotheses	Reduced, but action is often still post-delivery	Depends on intelligence coverage and freshness
Pre-emptive	At or before the perimeter, before delivery	Structure, provenance, reputation, early behavior	Closed before it opens	Must decide on incomplete information

Table 2.1. Reactive, proactive, and pre-emptive postures compared by decision point, evidence, and the window of compromise each leaves open. Source: author's construction, drawing on Jáñez-Martino et al. (2022), Sun et al. (2023), and Gambo and Almulhem (2025).

This reframing aligns the principle with the broader direction of enterprise security. Zero-trust architecture replaces implicit trust with continuous per-request verification, and its central claim, codified in the reference architecture published by the national standards body, is that authorization should be re-established for every request rather than inherited from prior context (Rose et al., 2020; Syed et al., 2022; Kang et al., 2023). A pre-emptive control is the temporal expression of the same idea: it treats every

inbound artifact as untrusted until it has passed evaluation, and it performs that evaluation before granting the artifact any foothold. Systematic review associates zero-trust adoption with reduced breach dwell time and contained lateral movement (Gambo & Almulhem, 2025), which are precisely the quantities a compressed window of compromise improves. Proactive cyber-threat-intelligence practice supplies the knowledge that makes early decisions possible, mining indicators and adversary patterns ahead of attacks, and surveys of the field frame this intelligence as the input that turns anticipation into action (Sun et al., 2023). The pre-emptive principle operationalizes that intelligence at the moment of decision rather than leaving it as situational awareness.

2.2 Architectural requirements of pre-emptive systems

Moving the decision point forward is not free. A system that must decide before delivery faces three requirements that a reactive system can avoid, and the architecture of any pre-emptive control is shaped by how it meets them.

The first requirement is interception before the point of delivery or activation. The defense must sit in the path of the artifact and hold authority to refuse it, rather than observing a copy after the fact. For email this means the gateway must receive inbound connections on behalf of the protected domain and decide whether to accept each message before passing it onward, which in turn requires that mail routing direct traffic through the gateway rather than to the destination server. For external digital risk it means monitoring registration and infrastructure signals as they appear, before a malicious domain is used in a campaign. For vulnerability management it means assessing exposure continuously, before a weakness is exploited. In every case interception is an architectural precondition: a control that cannot stand in the path cannot be pre-emptive, regardless of how good its model is.

The second requirement is low decision latency. Because the artifact is held while the decision is made, the evaluation must complete within a budget that does not disrupt legitimate flow. A reactive analyzer can take minutes to study a delivered sample; a pre-emptive gateway deciding whether to accept a connection has milliseconds before the delay becomes visible to senders and recipients. This constraint forces a layered evaluation in which cheap, high-recall filters run first and expensive analysis runs only on the residue that earlier layers could not clear. The comparative literature supports this division of labor, since gradient-boosted models on structured features achieve strong accuracy at low inference cost and are well suited to early, high-throughput stages (Almujahid et al., 2024; Kumar et al., 2024), while heavier content models are reserved for the cases that need them.

The third requirement is the capacity to decide on incomplete information. A reactive system can wait until a threat has fully manifested before judging it; a pre-emptive system must commit before the threat acts, which means it must extract enough signal from structure, provenance, and reputation to decide without observing the harmful behavior itself. This is why pre-emptive designs lean on signals that are available early and do not depend on a prior catalogue entry: the lexical structure of a url, the registration age of a domain, the authentication status of a sender, the reputation of the originating infrastructure. Phishing-domain research shows that such structural features generalize to newly registered domains absent from any blocklist, which is exactly the property a pre-emptive control needs (Alnemari & Alshammari, 2023; Salloum et al., 2022). The requirement to decide early is therefore also a requirement to choose features that carry signal before the threat has done anything observable.

A fourth consideration is the cost of being early, which the architecture must budget for explicitly. Acting before a threat manifests means acting on suspicion, and suspicion produces two errors that a pre-emptive control must keep within operational bounds. The first is the false positive at the gate: a legitimate message refused, a benign domain flagged, a harmless vulnerability elevated. Because a pre-emptive

control refuses rather than merely alerts, its false positives carry a higher operational cost than a reactive alert that an analyst can dismiss, so the gate cannot be tuned for recall alone. The second error is the availability risk: a control that stands in the path of all traffic becomes a single point through which everything must pass, so it must be engineered for throughput and for graceful behavior when overloaded. These costs explain why a responsible pre-emptive design does not block on a single aggressive model. It distributes the decision across layers, reserves outright refusal for the cases where independent signals agree, and routes ambiguous cases to a holding state rather than to deletion. The window of compromise is closed not by a single confident classifier at the gate but by a layered evaluation that is collectively confident, which is the bridge from this principle to the ensemble integration of Chapter 4.

These requirements interact. Interception creates the latency budget; the latency budget forces layering; layering with incomplete information forces a reliance on early structural signals and on the combination of several weak-but-fast indicators rather than one slow-but-certain verdict; and the cost of early errors forces that combination to be calibrated rather than aggressive. The pre-emptive principle thus already points toward the ensemble integration developed in Chapter 4, because deciding early and quickly on partial evidence, without paying an unacceptable false-positive price, is best done by aggregating independent fast signals rather than by waiting for, or gambling on, a single definitive one.

2.3 Applying the principle to email protection

Email is the domain in which the pre-emptive principle is most fully realized in the author's practice, and it is the domain verified in Chapter 5. The architectural move is to perform all analysis before the message reaches the corporate mail server, a pre-delivery model in which the gateway accepts inbound connections on behalf of the protected domain, evaluates each message through sequential layers, and forwards only those that pass (Kolchin, 2025). The routing precondition is met by directing the domain's mail-exchange records to the gateway rather than to the internal server, so that every inbound message is subject to evaluation before the server ever sees it.

The contrast with a conventional secure email gateway is instructive. A traditional gateway often applies signature matching and reputation lookups at or after delivery, which embeds the reactive limitation directly into the architecture: a threat is only stopped if a corresponding signature or reputation entry already exists, and zero-day malware, newly registered phishing domains, and socially engineered business email compromise pass unimpeded. Post-delivery remediation, the retrieval of a message from a recipient's mailbox after the fact, is operationally awkward and frequently impossible once the message has been read. The pre-emptive design removes the remediation problem by never delivering the malicious message in the first place.

The pre-SMTP stage is the architectural heart of the principle in this domain. Before message content is even transmitted, the gateway evaluates the connection itself: reputation lookups against multiple feeds produce an aggregate sender score, greylisting rejects first-contact connections that do not retry in the standards-compliant way that legitimate mail transfer agents do, and rate limiting suppresses burst delivery from compromised hosts. A connection that fails this stage is refused at the transport level, before any content crosses the wire. This is the purest form of pre-emption, because the decision precedes not only delivery but transmission. Subsequent layers add protocol authentication and content analysis. Authentication verification of the sender-policy, signature, and domain-alignment standards establishes a provenance chain and detects domain spoofing, and the evidence it produces is used both to enforce policy and as input features for later classification regardless of the policy level the sending domain declares (Ragheb et al., 2023). The limits of authentication are themselves an argument for layering:

rejection policies eliminate direct spoofing but leave lookalike-domain attacks untouched, which is why content and url analysis must follow (Hureau et al., 2024). Figure 2.3 shows the full pre-delivery pipeline.

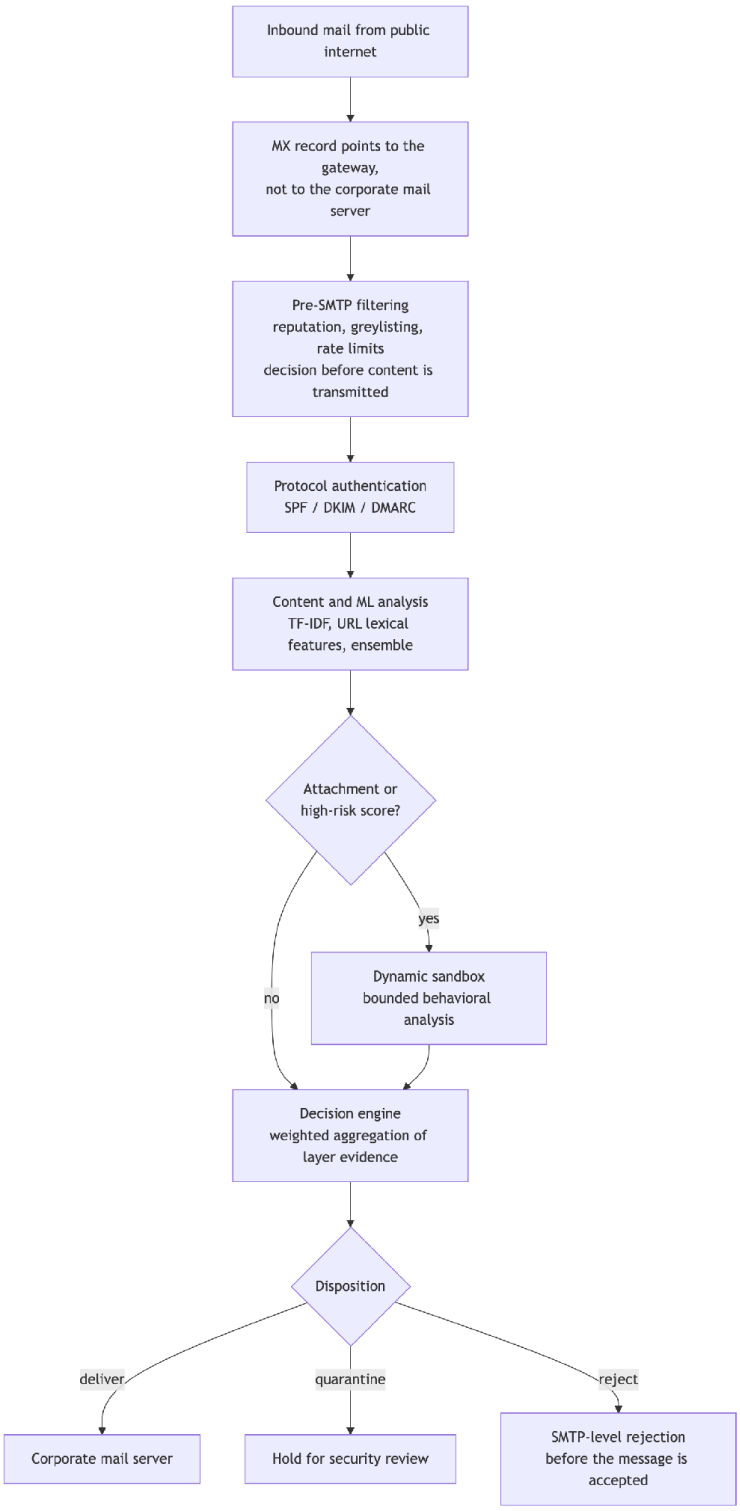

Figure 2.3. *Pre-delivery email processing under the pre-emptive principle. Mail routing directs inbound traffic to the gateway rather than to the corporate server; sequential layers evaluate the connection and message, and only passing messages are forwarded. Source: author's construction.*

The ordering of layers expresses the latency requirement of Section 2.2. The cheapest decisions, made on connection metadata, run first and dispose of the bulk of obvious abuse before any expensive analysis

begins. Authentication runs next, at low cost. Content and machine-learning analysis run on the survivors, and dynamic sandbox analysis, the most expensive layer, runs only on attachment-bearing or high-risk messages. This is the same complementarity that multi-modal email research reports, in which combining metadata with content outperforms either alone (Doshi et al., 2023; Ahmed et al., 2022). The pre-emptive realization adds the constraint that all of this must complete before delivery, which is what distinguishes the architecture from a post-delivery analyzer that happens to use the same models.

Two cases test the principle in this domain and show why layering is essential to it. The first is business email compromise, the targeted-deception attack that imitates legitimate internal correspondence and carries little overtly malicious content. A content classifier alone has almost nothing to grip, which is why systematic review of the problem stresses behavioral and relational signals over message text (Atlam & Oluwatimilehin, 2023). The pre-emptive architecture handles this not by a stronger text model but by the layers that precede content analysis: authentication exposes a spoofed or misaligned sender, reputation exposes anomalous originating infrastructure, and the aggregate of these early signals carries the decision that content alone cannot. Business email compromise is thus a case where pre-emption succeeds precisely because the decision is taken from provenance evidence available before the message body is even considered. The second case is the malicious attachment, where the relevant evidence is behavioral and would ordinarily require execution. Here the pre-emptive design routes attachment-bearing or high-risk messages to a bounded dynamic-analysis layer that executes the sample in an instrumented environment before delivery, so that behavioral evidence is obtained while the message is still held rather than after it has reached a mailbox. The evasive-malware problem of Section 1.4 bounds what this layer can achieve, since samples that withhold behavior may pass, which is why dynamic analysis is one weighted layer rather than the sole arbiter (Aboaoja et al., 2023). In both cases the architecture closes the window by combining what each layer can establish early, rather than by demanding that any single layer be certain.

2.4 Applying the principle to external digital-risk protection

The second domain is the space outside the corporate perimeter, where threats are assembled before they are aimed at the organization. The pre-emptive principle here means detecting and neutralizing a threat during its preparation rather than during its execution. The author’s digital-risk protection system is designed to operate in this space, and it illustrates the principle without supplying production metrics, which remain outside the scope of this monograph.

The clearest application is the detection of phishing and fraudulent domains before they are used in campaigns. Attackers register lookalike domains that differ from a legitimate brand by one or two character operations, exploiting visual and typographic confusion. These domains can be identified from their structure alone, before any phishing message is sent, using lexical signals such as character entropy, subdomain depth, the presence of address literals, and edit distance to a watchlist of protected brand domains (Alnemari & Alshammari, 2023; Salloum et al., 2022). A large body of work confirms that machine-learning models built on url and domain features detect phishing infrastructure effectively, spanning multidimensional deep-learning detectors (Yang et al., 2019), character-level convolutional models (Aljofey et al., 2020), transformer-based url classifiers (Elsadig et al., 2022), and hybrid url systems (Karim et al., 2023), and recent surveys map the breadth of these approaches and their open challenges (Basit et al., 2020; Alabdan, 2020; Zieni et al., 2023). Because these features do not depend on the domain having yet been used, a system that monitors registrations can flag a malicious domain during its dormant preparation window, which is the pre-emptive ideal: the threat is neutralized before it becomes operational. The same lexical machinery that classifies phishing urls inside an email also

classifies candidate domains in the wild, which is why the digital-risk system is designed to apply url and domain analysis to external monitoring feeds rather than only to inbound mail.

The mechanism deserves a closer look because it shows what makes a signal pre-emptive. A homoglyph attack substitutes visually similar characters, and a typosquatting attack substitutes adjacent or omitted ones, so that a registered domain reads as a trusted brand to a hurried human eye. Both manipulations are detectable from the string itself: edit distance to a brand watchlist captures the one- or two-character substitution, character entropy captures the algorithmically generated label, and the presence of an address literal or an unusual subdomain depth captures structural anomaly (Alnemari & Alshammari, 2023; Salloum et al., 2022). None of these features requires the domain to have sent a single message, which is what makes the detection pre-emptive rather than reactive: the decision is available during the dormant interval between registration and weaponization. A monitoring system that scores newly registered domains against the protected brand watchlist can therefore raise an alert, and where policy permits initiate a takedown or pre-emptive block, before the first phishing email is sent. This is the external mirror of the pre-SMTP decision: in both cases the defense acts on structure during a preparation window rather than on behavior during an attack.

A second application is the monitoring of shadow sources, the forums, marketplaces, and leak sites where compromised data and attack plans circulate. Early detection of a leak or of preparatory chatter lets an organization respond before the exposed material is weaponized. This is a natural-language problem, and it inherits both the techniques and the new pressures of adversary-generated text. Large language models now produce fluent malicious content at scale, raising the volume and quality of material that monitoring must triage (Hazell, 2023; Roy et al., 2024). Proactive threat-intelligence mining is the discipline that turns this monitoring into pre-emptive action, extracting indicators and adversary patterns from open and closed sources ahead of attacks so that defensive controls can be updated before the corresponding campaign launches (Sun et al., 2023). The digital-risk system is designed to feed such indicators back into the perimeter controls, closing a loop in which external intelligence shortens the window at the gateway.

The transferability is the point. The pre-emptive principle is not specific to email; it is a stance toward timing that applies wherever a threat has a preparation phase that can be observed. Email offers a perimeter at which to intercept; external digital risk offers a preparation phase at which to detect. In both cases the architecture is organized around acting before the threat is operational, and in both cases the early-available structural signal, rather than the post-hoc behavioral signal, carries the decision.

2.5 Applying the principle to vulnerability management

The third domain is the management of the organization’s own exposures, where the pre-emptive principle means acting on a vulnerability before it is exploited rather than after. Vulnerability management is overwhelmed by volume: the number of disclosed vulnerabilities rises each year, and organizations cannot patch everything at once, so they must prioritize. The reactive failure mode is to prioritize by static severity and to react to exploitation after it occurs. The pre-emptive alternative is to predict which vulnerabilities will be exploited and to remediate those first, before the exploitation happens.

Static severity scores are weak predictors of actual exploitation because they do not incorporate the evolving signals that drive attacker behavior. Data-driven exploit prediction improves on them by learning from features that correlate with real-world exploitation, producing a probability of exploitation that can rank a remediation queue ahead of any attack (Jacobs et al., 2023). Ensemble models trained on vulnerability and threat-intelligence features estimate exploitation likelihood and convert an

undifferentiated backlog into an ordered queue (Fang et al., 2020). This is the pre-emptive principle in its prioritization form: the model acts on the probability of a future event and directs finite effort to close the exposures most likely to be used, before they are used.

The author's continuous penetration-testing and asset-monitoring system is designed around this stance for the external perimeter. It is engineered to maintain a continuous inventory of an organization's externally visible and hidden assets, including the shadow infrastructure that escapes manual tracking, and to assess that surface continuously rather than at periodic intervals. The architectural logic mirrors the email case: continuous interception of the changing attack surface replaces periodic post-hoc audit, and automated prioritization directs attention before exploitation rather than after a breach.

Asset discovery is the precondition that makes the rest pre-emptive, and it is the part most often neglected. An organization cannot prioritize an exposure it does not know it has, and external attack surfaces grow through forgotten subdomains, abandoned services, and infrastructure stood up outside the change-management process. A periodic audit captures the surface as it was on the audit date and is stale by the next deployment; a continuous inventory captures the surface as it changes, which is the only basis on which exploitation can be anticipated rather than discovered after the fact. The system is therefore engineered to enumerate visible and hidden assets continuously and to feed that inventory into the exploit-prioritization model, so that a newly exposed and likely-to-be-exploited asset is surfaced while it is still merely exposed. This is the vulnerability-management expression of the same loop seen in the email and digital-risk domains: continuous observation of a changing surface, early structural and predictive signals, and action taken during the preparation window. The system is designed to combine automated assessment with expert verification, which reflects a practical limit of the principle discussed next.

The limit is that pre-emptive prioritization is probabilistic, and acting on probability means accepting both false alarms and missed predictions. A model that ranks exploitation likelihood will sometimes elevate a vulnerability that is never exploited and sometimes under-rank one that is. The architectural response is the same combination of automated speed and human judgment that the framework applies elsewhere: automated prioritization handles scale and provides the ranked queue, while expert verification adjudicates the high-stakes and ambiguous cases. This pairing reappears in Chapter 4 as the general pattern of combining machine speed with non-machine judgment, and it is the reason the pre-emptive principle does not stand alone but composes with the ensemble-integration principle. Acting early and on incomplete information is only safe when the early decision is one input to a layered process rather than an irreversible verdict, which is the architecture the next two chapters build.

The principle developed in this chapter has a precondition that the chapter has so far left implicit, and naming it sets up the chapter that follows. A pre-emptive control decides on early signals: the structure of a url, the provenance of a sender, the reputation of an originating host, the predicted likelihood that a vulnerability will be exploited. Every one of these decisions is produced by a model that learned, from labeled examples, what an early signal of compromise looks like. The quality of a pre-emptive decision is therefore bounded by the quality of the labels the model was trained on, and a model trained on biased or single-source labels will be confidently wrong at exactly the early, low-information moment when the pre-emptive architecture asks it to act. The window of compromise can be closed in principle by deciding early, but it is only closed in practice if the early decision is correct, and correctness at the perimeter depends on how the training data were labeled. That dependency is the subject of the second principle. Where this chapter asked when the model should act, the next asks how the data that teach the model should be assembled, so that the early decisions on which pre-emption rests are trustworthy rather than merely fast.

Chapter 3. Combined Data Labeling as the Second Principle

The second principle concerns the data on which a model learns. It holds that the ground truth used to train a security model should be drawn from several independent sources at once rather than from any single one, because each source of labels carries a characteristic and systematic bias, and only the combination of independent sources cancels those biases. Where the first principle governs when a model acts, the second governs what the model knows, and the two are linked: a pre-emptive control that must decide early on incomplete information is only as trustworthy as the labels that taught it what early evidence means. This chapter establishes the bias problem, sets out the methodology of combined labeling, describes its implementation in a machine-learning pipeline, and compares the behavior of homogeneously and heterogeneously labeled models.

3.1 The problem of systematic bias in homogeneous labeling

A supervised model learns the function it is shown. If the examples it is shown are labeled by a single process, the model inherits not only the knowledge of that process but also its blind spots, and in production security each labeling process has blind spots that are systematic rather than random. Random labeling error degrades a model gently and symmetrically; systematic labeling bias degrades it in a specific, exploitable direction, teaching the model to be confidently wrong about exactly the cases the labeling process cannot see. Three labeling sources dominate operational email and threat detection, and each fails in a different direction.

The first source is user feedback, the verdicts that recipients supply when they report a message as malicious or mark it as safe. User feedback is valuable because it reflects real human judgment on real messages, but it is biased by what users notice. A successful phishing message, by definition, is one the recipient did not recognize as phishing, so it is under-reported precisely when it matters most. User feedback therefore over-represents clumsy, obvious attacks and under-represents the sophisticated deception that evades human attention, and a model trained on it learns to catch what people already catch while remaining blind to what fools them. Business email compromise, which imitates legitimate correspondence and offers no obvious tell, is the clearest victim of this bias (Atlam & Oluwatimilehin, 2023).

The second source is sandbox analysis, the verdicts produced by executing a suspicious artifact in an instrumented environment and observing its behavior. Sandbox labels are precise for what they cover, but their coverage is narrow in two ways. They apply chiefly to attachment-bearing or executable content, leaving text-only phishing and credential-harvesting links largely outside their reach. And they are defeated by evasion: malware that withholds its behavior until it detects a real execution environment generates no malicious evidence in the sandbox and is labeled benign (Aboaoja et al., 2023; Liu et al., 2022). The catalogue of evasion techniques shows how routinely this is done (Afianian et al., 2019). A model trained on sandbox labels alone therefore learns a confident but partial notion of malice, accurate on the attachments it has seen detonate and blind to both non-executable attacks and evasive samples.

The third source is threat intelligence, the externally curated indicators of compromise that mark known-malicious domains, addresses, and hashes. Threat intelligence is authoritative once it exists, but it exists only after a threat has been observed and catalogued somewhere, so it lags new campaigns. The review of spammer strategy and the dataset-shift problem documents this lag, and the deliberate exploitation of it, as central to how spam and phishing evade detection (Jáñez-Martino et al., 2022). Proactive threat-intelligence research treats the same lag as the problem to be solved, mining indicators ahead of attacks

precisely because catalogued intelligence arrives too late to label the campaign that produced it (Sun et al., 2023). A model trained on threat-intelligence labels alone learns yesterday's threats well and today's not at all, which is the same reactive limitation that motivated the first principle, now reappearing in the training data rather than in the decision point. This is the concept-drift problem viewed from the labeling side: as the threat distribution shifts, labels anchored to a single lagging source drift out of date, and the literature on learning under concept drift treats exactly this nonstationarity as a first-class obstacle for deployed classifiers (Lu et al., 2018). Table 3.1 sets out the three sources, what each labels well, and the systematic blind spot each carries.

Labeling source	**Labels well**	**Systematic blind spot**	**Temporal character**
User feedback	Obvious, recognizable attacks	Successful deception that users miss (e.g., BEC)	Immediate but selective
Sandbox analysis	Attachment and executable behavior	Non-executable attacks; evasive, environment-aware malware	Per-sample, bounded by analysis time
Threat intelligence	Known, catalogued indicators	New campaigns before they are catalogued	Lagging
Combined (this principle)	Union of the above coverage	Reduced: each source covers another's gap	Mixed; refined retrospectively

Table 3.1. Systematic bias of single-source labeling and the coverage gain from combination. Source: author's synthesis based on Atlam and Oluwatimilehin (2023), Aboaoja et al. (2023), Afianian et al. (2019), and Jáñez-Martino et al. (2022).

The structure of the three biases is the argument for the principle. The blind spots do not overlap. User feedback misses what sandbox analysis and threat intelligence can catch; sandbox analysis misses what user feedback and intelligence can catch; intelligence misses what the other two catch in real time. A label set drawn from all three covers, in union, what no single source covers alone, and the residual blind spot of the combination is far smaller than that of any member.

A concrete sequence shows how the bias becomes self-reinforcing when it is left uncorrected. Consider a credential-harvesting campaign that uses a newly registered lookalike domain and a message body crafted to read as a routine internal notice. At the moment of arrival the domain is absent from threat intelligence, the message carries no attachment for the sandbox to detonate, and the recipients who act on it do not report it because it did not look wrong. All three single sources independently label the campaign benign, and a model trained on any one of them is reinforced in its error: the next variant of the same campaign is scored as safe with even higher confidence, because the model has now seen confirming benign labels. This is the practical danger of systematic bias. It does not merely leave a gap; it actively trains the model to defend the gap, because the biased labels look like clean ground truth. Only a source that sees the campaign from a different angle, the structural anomaly of the domain or a later intelligence indicator, breaks the cycle, and combined labeling is the mechanism that guarantees such a source is present.

The distinction between systematic and random error deserves emphasis because it determines what can fix the problem. Random labeling noise is reduced by more data and by averaging, since errors in different directions cancel. Systematic bias is not reduced by more data, because every additional example is mislabeled in the same direction, so a larger single-source dataset produces a more confident biased model rather than a more accurate one. The only remedy for systematic bias is a source of labels whose bias points elsewhere, which is exactly what combination provides and what scale alone cannot.

3.2 The methodology of combined labeling

Combining label sources is not the same as concatenating them. Independent sources disagree, they carry different reliabilities, and they arrive at different times, so a methodology for combined labeling must specify how sources are weighted, how conflicts are resolved, and how labels are revised as new information arrives. The author's approach, realized in the email security platform verified in Chapter 5, rests on three commitments.

The first commitment is independence. The sources combined must fail in genuinely different ways, because combining correlated sources adds volume without adding coverage. User feedback, sandbox execution, and threat intelligence qualify because their failure modes are unrelated: a message that fools a user need not evade a sandbox, and a sample that evades a sandbox need not be absent from intelligence. This is the labeling analogue of the ensemble principle of Chapter 4, where independent decision layers are combined for the same reason, and the conceptual link is exact: diversity of error is what makes combination worthwhile, whether the things combined are labels or decisions. The ensemble-learning literature makes the same point about models, holding that combining base learners helps to the degree that their errors are uncorrelated (Mienye & Sun, 2022); the present principle applies that result one level earlier, to the sources that produce the labels rather than to the models that consume them.

The second commitment is a trust hierarchy with explicit conflict resolution. When two sources assign different labels to the same artifact, the disagreement must be resolved by a rule rather than by accident of pipeline ordering. The rule follows from the reliability structure of the sources for the case at hand. A confirmed sandbox detonation or a verified intelligence indicator is strong positive evidence and outweighs an absence of user reports, because the absence of a report is weak evidence of safety. A user report of a missed phishing message, once verified by security operations, outweighs a benign sandbox result, because the sandbox may simply not have covered the relevant content. The semi-supervised-learning literature provides the general frame for reasoning about labels of differing reliability and for propagating confident labels to related unlabeled instances (van Engelen & Hoos, 2019), and the trust hierarchy is the security-specific instance of that reasoning. Table 3.2 states the resolution rules used.

Conflict scenario	**Resolution**	**Rationale**
Sandbox malicious vs no user report	Label malicious	Absence of a report is weak evidence of safety
Verified user report vs sandbox benign	Label malicious	Sandbox may not cover the relevant content
Threat-intel indicator vs benign content score	Label malicious	Catalogued indicator is high-confidence positive
All sources benign	Label legitimate, retain for retrospective review	Combined absence of evidence is stronger than any single source

Conflict scenario	Resolution	Rationale
Sources benign now, intel arrives later	Relabel retrospectively	New information supersedes the earlier provisional label

Table 3.2. Trust hierarchy and conflict-resolution rules for combined labeling. Source: author's methodology, with the reliability-weighting frame from van Engelen and Hoos (2019).

The third commitment is retrospective relabeling. A label assigned at the moment of delivery is provisional, because the evidence that would settle an ambiguous case may not arrive until later. When a threat-intelligence indicator is published days after a message was delivered, the messages that carried that indicator can be relabeled, converting a set of formerly ambiguous examples into confident training data. This closes part of the threat-intelligence lag inside the training set rather than at the decision point: the model cannot benefit from intelligence that did not exist when a message arrived, but it can learn from that intelligence at the next retraining, so that the lag shortens with each cycle. Retrospective relabeling is what makes the combined label set improve over time rather than merely aggregate at a single instant.

These three commitments place combined labeling in the broader family of methods that build training data from imperfect, multiple signals rather than from a single curated source. Semi-supervised and weakly supervised learning address the same underlying scarcity, that confidently labeled examples are expensive while weak or partial signals are abundant, and they share the same core technique of propagating high-confidence labels to related instances and reconciling signals of differing reliability (van Engelen & Hoos, 2019). Combined labeling is the security-specific member of this family, with the important difference that its weak signals are not noisy versions of one ground truth but genuinely independent observers of different facets of the same artifact. That independence is what allows the trust hierarchy to be more than a voting scheme. A simple majority vote across sources would let two correlated sources outvote one independent source that happens to be the only one seeing the threat, which is the failure the bias analysis warns against. The hierarchy instead weights a source by its reliability for the specific kind of evidence at issue, so that a single authoritative positive, a confirmed detonation or a verified indicator, can override a chorus of weak negatives. Designing that weighting is the central craft of the principle, and getting it wrong, by over-trusting the most convenient source, reintroduces precisely the single-source bias the principle exists to remove.

3.3 Implementing combined labeling in machine-learning pipelines

A methodology becomes a principle only when it is realized in a pipeline that other practitioners can build. Three implementation problems must be solved: collecting labels from distributed sources, handling the class imbalance that production traffic imposes, and splitting data for evaluation without leaking information across the split.

Collection is an architectural problem because the three sources live in different systems and arrive on different schedules. User reports come from mail clients and ticketing systems, sandbox verdicts from the analysis cluster, and intelligence indicators from external feeds, and the pipeline must reconcile them onto a common identifier for each message so that a message's eventual label reflects all three. The pipeline must also retain provisional labels and the evidence behind them, because retrospective relabeling requires the ability to revisit an earlier decision when new evidence arrives. This is a data-engineering commitment as much as a modeling one, and it is the part of the principle most often skipped in deployments that default to whichever single source is easiest to collect. The reconciliation requires a

stable identifier for each artifact and a store that holds both the provisional label and the evidence that produced it, so that when a late-arriving indicator changes a verdict the affected examples can be found and relabeled. A pipeline that discards the evidence after assigning a label cannot relabel retrospectively, and a pipeline that cannot relabel cannot shorten the intelligence lag in its training data. The unglamorous decision to retain evidence and identifiers is therefore what makes the methodology work, and its absence is why many deployments collapse to a single source despite knowing better.

Class imbalance is the second problem. In production email traffic, legitimate messages outnumber malicious ones by a wide margin, and a classifier trained naively on that distribution learns to predict the majority class and miss the minority that matters. The standard remedies adjust the training distribution: synthetic minority oversampling, the technique that generates plausible interpolated minority examples rather than duplicating them (Chawla et al., 2002), undersampling that reduces the majority, and cost-sensitive weighting that penalizes minority errors more heavily during optimization. The same logic of synthesizing additional minority examples underlies data-augmentation practice in deep learning more broadly (Shorten & Khoshgoftaar, 2019), and the security-specific resampling literature confirms its value for imbalanced detection data. Research on resampling for intrusion-detection datasets confirms that these techniques materially improve a classifier's ability to recognize minority attack classes that it would otherwise ignore (Bagui & Li, 2021), and studies on imbalanced and current intrusion datasets reach the same conclusion for machine-learning detectors generally (Karataş et al., 2020). Combined labeling interacts well with imbalance handling, because a richer label set supplies more genuine minority examples before any synthetic augmentation, reducing the burden placed on oversampling to manufacture them.

The third problem is the split between training and evaluation data, and it is where combined labeling meets the first principle most directly. The intuitive approach, a random split of all labeled messages into training and test sets, introduces a subtle and serious error in security evaluation: threat indicators that appear in both partitions let the classifier recognize a test message because it memorized a near-duplicate in training, inflating measured performance through memorization rather than generalization. This is a form of data leakage, and the reproducibility literature identifies leakage as a pervasive cause of overoptimistic machine-learning results across many fields of science (Kapoor & Narayanan, 2023). The remedy in security is a temporal split: the evaluation set consists exclusively of messages received after every training message, so the classifier is tested on threats it could not have seen during training, which is the operational condition it will actually face. Table 3.3 contrasts the two approaches.

Dimension	**Random split**	**Time-based (temporal) split**
Partition rule	Random assignment of all examples	Test set strictly after all training examples
Leakage risk	High: near-duplicate threats span both sets	Low: no future information in training
What it measures	Memorization plus generalization	Generalization to unseen future threats
Operational realism	Weak: does not mirror deployment	Strong: mirrors detecting novel variants
Effect on reported metrics	Inflated	Conservative and trustworthy

Table 3.3. Random versus time-based data splitting for security model evaluation. Source: author's synthesis based on Kapoor and Narayanan (2023) and the temporal-holdout protocol of Kolchin (2025).

The temporal split is not a refinement of measurement; it is a precondition for trusting any number a security model reports. A model whose accuracy was measured under a random split has been graded on a test it partly saw in advance, and its reported performance does not predict its behavior on tomorrow's threats. The combined-labeling methodology and the temporal split together produce a training and evaluation regime in which the labels are drawn from independent sources, refined retrospectively, balanced against the production class distribution, and evaluated on a strictly future test set. That regime is what the verification in Chapter 5 implements, and it is why the numbers reported there can be read as estimates of operational rather than benchmark performance (Kolchin, 2025).

Temporal leakage is the most consequential leak in security evaluation, but it is not the only one, and a disciplined pipeline guards against the others as well. Feature leakage occurs when a feature available at training time would not be available, or would carry different information, at decision time; a reputation score computed after a campaign has been catalogued, for instance, encodes future knowledge that the model will not have when it must decide on the next campaign. The reproducibility literature catalogues several such leakage forms and ties them directly to overoptimistic reported results (Kapoor & Narayanan, 2023). Combined labeling interacts with this risk in a specific way: because retrospective relabeling deliberately uses future information to correct labels, the pipeline must keep that future information out of the features. The label may be improved with hindsight, but the feature vector must reflect only what was knowable at decision time, or the evaluation will measure the model's access to the future rather than its ability to predict it. Separating retrospective label correction from feature construction is therefore a precise discipline that the combined-labeling methodology imposes, and conflating the two is a subtle way to manufacture impressive and meaningless numbers.

3.4 Comparative analysis of labeling strategies

The value of combined labeling is clearest when the behavior of a homogeneously labeled model is set against that of a heterogeneously labeled one on the cases where their training differs. The comparison is qualitative here, because the production figures belong to Chapter 5, but its structure can be stated precisely.

A model trained on a single source performs well on the region its labels cover and degrades sharply outside it. A user-feedback model catches obvious attacks and misses sophisticated phishing; a sandbox model catches detonating attachments and misses text-only deception and evasive samples; an intelligence model catches catalogued threats and misses novel ones. Each is locally strong and globally fragile, and its fragility is invisible under a test set drawn from the same biased source, because the test inherits the training's blind spot and never probes it. This is the mechanism by which single-source models report high benchmark accuracy and then fail in production: the benchmark shares the blind spot.

A model trained on combined labels trades a small amount of peak accuracy on any single region for a large gain in coverage across regions. Its advantage is largest exactly where single-source models are weakest, namely on the threats that one source misses but another catches: the evasive attachment that the sandbox missed but intelligence later flagged, the sophisticated phishing that users missed but content analysis caught, the novel campaign that intelligence lacked but anomalous structure revealed. The gain is smallest, and combination adds least, when threats are homogeneous and well covered by a single source, which is the benign case that production rarely presents. The semi-supervised perspective explains why the combination can exceed the sum of its parts: confident labels from one source propagate

to related unlabeled instances, extending coverage beyond the directly labeled examples (van Engelen & Hoos, 2019).

This comparison has a measurement consequence that is easy to misread. A combined-labeled model may report a lower headline accuracy on a conventional benchmark than a single-source model reports on its own benchmark, and an unwary evaluator might conclude that the single-source model is better. The opposite is true. The single-source benchmark shares the training source's blind spot and therefore never tests the cases the model fails, so its high number is an artifact of a biased test rather than evidence of a strong model. The combined-labeled model is tested on a broader, harder distribution that includes the cases single sources miss, so its lower headline number is earned against a fairer test and predicts production behavior more honestly. The right comparison is not benchmark-to-benchmark but both models against a single combined, temporally split test set, on which the combined-labeled model's coverage advantage becomes visible as the single-source model's hidden failures are finally exercised. This is the labeling counterpart of the evaluation-realism argument of Section 1.3: a number is only as trustworthy as the distribution it was measured on.

Combined labeling also contributes to adversarial robustness, which links the second principle to the fourth. An adversary who studies a defender's single label source can craft attacks that fall into that source's blind spot, confident that the resulting examples will be mislabeled benign and will train the model to accept the attack. Against a defender who labels from three independent sources, the adversary must evade all three simultaneously, because an attack caught by any one source enters the training set with a correct malicious label and teaches the model to recognize the technique. This matters because content-level adversarial manipulation, the perturbation of message text to flip a classifier's verdict, is both effective and well documented (Han et al., 2022; Gholampour & Verma, 2023); a defender whose labels come only from content-derived signals is most exposed to it, while a defender whose labels also reflect provenance and behavior retains a correct signal even when the text is adversarially tuned. Diversity of labeling thus raises the cost of the data-poisoning-by-omission that single-source labeling invites, in the same way that diversity of decision layers, examined in Chapter 4, raises the cost of evasion at decision time. The two principles defend against the same adversarial strategy at two different stages, labeling and decision, which is why the framework treats them as complementary rather than independent.

The principle has limits, and honesty about them is part of the methodology. Combined labeling raises engineering cost, because three collection paths must be built and reconciled rather than one. It introduces conflict-resolution decisions that, if specified poorly, can import a new bias rather than cancel an old one, since a trust hierarchy that systematically over-trusts one source recreates that source's blind spot. And it depends on the genuine independence of its sources, so combining three views that share a common upstream dependency yields less than their number suggests. These limits do not undermine the principle; they specify the conditions under which it pays, which are precisely the conditions of production enterprise traffic: diverse threats, imbalanced classes, and adversaries who study and exploit whichever single source a defender relies on. Under those conditions the combined-labeling principle removes a systematic bias that no amount of model tuning can fix, and it prepares the trustworthy training data that the ensemble architecture of the next chapter turns into trustworthy decisions.

Chapter 4. Ensemble Integration in Multi-Layer Defense

The third principle concerns position. It holds that a machine-learning model should operate as one weighted layer within a multi-layer defense rather than as a standalone control, and that the decision to allow, hold, or block an artifact should be taken by aggregating the evidence of several independent layers rather than by trusting any one of them. Where the first principle governs when the model acts and the second governs what it learns from, the third governs where its output goes and how it is combined with everything else the defense knows. This chapter develops the architectural model of layered defense, formalizes the weighted aggregation that combines the layers, examines how machine-learning and non-machine-learning controls complement one another, analyzes the robustness that aggregation buys, and connects the decision layer to the monitoring and response systems that close the loop.

4.1 The architectural model of multi-layer defense

A layered defense is organized as a sequence of independent controls, each examining an artifact from a different vantage and each producing evidence rather than a final verdict. The vantages run from the outer perimeter inward: reputation and infrastructure signals at the connection level, identity and authentication signals at the protocol level, content and machine-learning signals at the message level, and behavioral signals from dynamic analysis at the execution level. The defining commitment of the model is that no single layer decides alone. Each contributes a calibrated signal, and a separate decision stage combines those signals into the disposition. Figure 4.1 shows the structure, including the monitoring and feedback paths developed in Section 4.5.

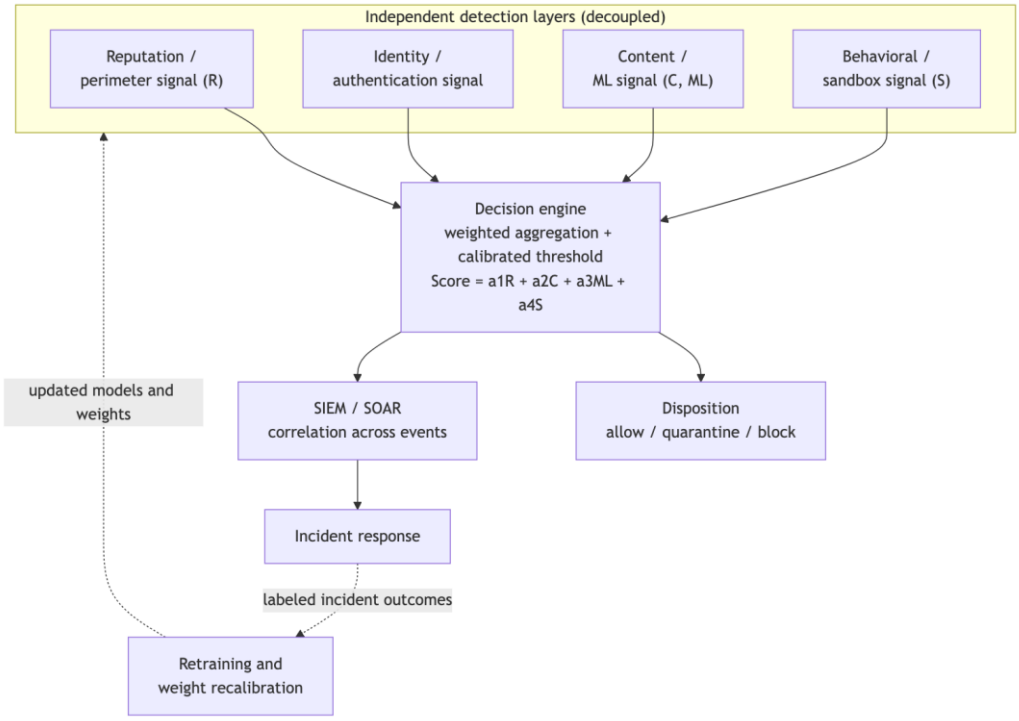

Figure 4.1. *The multi-layer defense model. Independent, decoupled layers each contribute a calibrated signal to a decision engine that aggregates them under a calibrated threshold; dispositions flow to monitoring and response, and labeled incident outcomes feed retraining and recalibration. Source: author's construction.*

The architectural property that makes this work is decoupling. The layers must be independent in implementation and in failure, so that the conditions that blind one do not blind the others. A reputation layer fails when an attacker uses fresh infrastructure; an authentication layer fails against a correctly authenticated but malicious sender; a content model fails under adversarial perturbation; a sandbox fails against environment-aware malware. Because these failure conditions are unrelated, an artifact that defeats one layer is unlikely to defeat all of them, and the aggregate retains signal even when a member is blind. This is the decision-time counterpart of the labeling-time argument of Chapter 3: diversity of error is what makes combination worthwhile, and the same independence that made combined labeling cancel bias makes combined decisions resist evasion. The zero-trust literature describes the same posture as continuous verification across independent checks, in which trust is never inherited and every request is re-evaluated against multiple controls (Syed et al., 2022; Kang et al., 2023), and systematic review ties this layered, non-perimeter stance to reduced breach impact (Gambo & Almulhem, 2025).

Decoupling also has an operational benefit beyond robustness: it lets each layer scale and evolve independently. A computationally heavy sandbox can be scaled separately from a lightweight reputation lookup, a content model can be retrained without touching the authentication layer, and a new layer can be added without redesigning the others, provided it produces a signal in the form the decision engine expects. The model is therefore not only more robust than a monolithic classifier but more maintainable, because the boundaries between layers are also the boundaries along which the system can be changed.

The abstract layers correspond to concrete controls, and the author's product portfolio illustrates the mapping without supplying production metrics for the controls outside the email case. At the outer perimeter, continuous attack-surface monitoring and digital-risk protection supply reputation and exposure signals about infrastructure and external threats before any specific request arrives. At the identity layer, privileged-access management and zero-trust network access supply authentication and authorization signals, enforcing per-request verification of who is asking and from what device posture. At the message and content layer, the email security platform supplies the content and machine-learning signals examined in detail in Chapter 5. At the behavioral layer, dynamic analysis supplies execution evidence. These are not a single product but an ecosystem of decoupled controls designed around a common architectural stance, and their combined deployment is what produces defense in depth: a request must pass perimeter, identity, content, and behavioral scrutiny rather than a single gate. The architectural point is that defense in depth is not achieved by owning many products but by combining their signals in a decision stage; a portfolio of standalone tools that never aggregate their evidence is a collection of single points of failure, whereas the same tools feeding one decision engine form a genuine multi-layer defense. The framework is the discipline that turns the former into the latter.

4.2 Weighted aggregation in the decision engine

The decision engine is the component that turns several layer signals into one disposition, and its design is where the ensemble principle becomes concrete. The author's realization aggregates layer evidence through a weighted linear scoring function of the form $\text{Score} = \alpha_1 R + \alpha_2 C + \alpha_3 ML + \alpha_4 S$, where R, C, ML, and S denote the reputation, content, machine-learning, and sandbox signals and the coefficients α weight each layer's contribution (Kolchin, 2025). The score is compared against thresholds that define disposition zones, so that a sufficiently low score is delivered, a sufficiently high score is rejected, and the ambiguous middle is held for review. The form is deliberately simple, because the engine's job is not to be a sophisticated model in its own right but to combine the sophisticated signals beneath it in a way that is calibrated, interpretable, and tunable.

Three design decisions govern the engine. The first is the choice of weights, which encode how much each layer's evidence counts. Weights are calibrated on a validation set rather than assigned by intuition, and they are updated through periodic retraining as the relative reliability of layers drifts. A layer that becomes noisier, perhaps because adversaries have learned to manipulate its signal, can be down-weighted without being removed, which lets the system degrade a compromised layer's influence gracefully rather than trusting or discarding it wholesale. The second decision is the placement of thresholds, which trade detection against disruption. A lower rejection threshold catches more threats at the cost of more false positives, and because a pre-emptive control refuses rather than alerts, the threshold must respect the operational cost of a false rejection. The third decision is the treatment of the ambiguous zone, which exists precisely so that the engine is not forced into a binary verdict on insufficient evidence; routing uncertain cases to a quarantine for human review is what lets the engine be aggressive about clear threats without being reckless about ambiguous ones. Table 4.2 sets out the components of the aggregation and how each is calibrated.

Layer signal	**Symbol**	**Contributes**	**Calibration basis**
Reputation / perimeter	R	Infrastructure and sender trust before content	Validation set; down-weighted as feeds age
Content analysis	C	Lexical, URL, and structural message features	Validation set; retrained on combined labels
Machine-learning classifier	ML	Ensemble model probability over full feature set	Validation set; recalibrated each retraining cycle
Sandbox / behavioral	S	Dynamic execution evidence for attachments	Validation set; weighted by coverage of the case
Decision threshold	θ	Boundaries of allow / quarantine / reject zones	Tuned to operational false-positive tolerance

Table 4.2. Components of the weighted decision engine and their calibration. Source: author's synthesis based on the decision-engine formulation of Kolchin (2025).

The calibration discipline is what distinguishes a principled ensemble from an ad hoc one. The research-practice gap discussed in Chapter 1 is in large part a calibration gap: models evaluated in isolation report probabilities that are not comparable across layers, so naively summing them produces a meaningless score (Apruzzese et al., 2022). The engine therefore requires each layer to emit a calibrated signal on a common scale, and it sets weights and thresholds jointly on held-out data under the temporal-split discipline of Chapter 3, so that the combined decision is evaluated under the conditions it will face in production rather than under benchmark conditions that flatter each layer in isolation.

The choice of a linear combiner over a more elaborate meta-model is deliberate and worth defending, because the obvious alternative is to train a second-stage model that learns the best non-linear combination of layer outputs. A learned combiner can fit the validation data more tightly, but it sacrifices three properties the security setting values. It sacrifices interpretability, because a linear score lets an analyst read off how much each layer contributed to a disposition, while a non-linear combiner obscures

that attribution. It sacrifices robustness, because a complex combiner adds its own attack surface and its own opportunity to overfit the validation distribution, reintroducing at the aggregation level the fragility the architecture removed at the layer level. And it sacrifices operational tunability, because a linear weight can be adjusted by an operator who understands its meaning, whereas a learned combiner must be retrained to change its behavior. The linear form keeps the engine simple enough to reason about, audit, and tune, and it places the system's sophistication where it belongs, in the independent layers, rather than in a combiner that would become a new monolith. This is a case where the architecturally correct choice is the less powerful model, because the property being optimized is the trustworthiness of the combined decision rather than its fit to a validation set.

4.3 Integrating machine-learning and non-machine-learning controls

A central claim of the ensemble principle is that the layers combined need not all be machine-learning models. The strongest defenses combine learned and unlearned controls, because the two kinds fail in different ways and therefore reinforce one another. Three pairings illustrate the complementarity.

The first is machine learning with reputation filtering. Reputation is a non-learned signal, a lookup against feeds of known-bad infrastructure, and it is fast, cheap, and precise on catalogued threats while blind to novel ones. A machine-learning content model is the reverse, capable of generalizing to unseen threats but slower and less certain. Combining them lets reputation dispose of obvious abuse cheaply at the perimeter while the model handles the residue, which is the division of labor that the latency budget of Chapter 2 requires and that multi-modal email research shows outperforms either control alone (Doshi et al., 2023; Ahmed et al., 2022).

The second pairing is machine learning with protocol authentication. The sender-policy, signature, and domain-alignment standards are deterministic cryptographic and policy checks, not learned models, and they establish provenance with certainty that no classifier can match: a failed signature is a fact, not a probability. Their limit is scope, because authentication detects spoofing but not a correctly authenticated malicious sender, and rejection policies stop direct domain abuse but leave lookalike-domain attacks untouched (Ragheb et al., 2023; Hureau et al., 2024). The ensemble uses authentication outcomes both as hard policy and as input features to the classifier, so that the deterministic certainty of the protocol check and the probabilistic generalization of the model are combined rather than chosen between.

The third pairing is machine learning with dynamic sandbox analysis. The sandbox is a behavioral oracle that observes what an artifact does rather than what it looks like, and it catches malware that static and content analysis miss. Its limits are cost and evasion, since detonation is expensive and environment-aware malware withholds its behavior (Aboaoja et al., 2023; Liu et al., 2022; Afianian et al., 2019). The ensemble routes only high-risk or attachment-bearing artifacts to the sandbox and treats its verdict as one weighted signal, so that an evasive sample that produces no sandbox evidence is not thereby cleared, because reputation, content, and authentication evidence still contribute to its score. A fourth pairing extends the principle from message defense to access control, and it is where the identity-layer products of the portfolio fit. Privileged-access management and zero-trust network access are largely deterministic policy controls: they verify identity, evaluate device posture, and grant access strictly within the scope of a task, with every request authenticated and logged (Syed et al., 2022; Kang et al., 2023). Machine learning complements these deterministic controls by scoring the behavior that policy alone cannot judge, flagging an authenticated session whose activity departs from the user's established baseline, which is the behavior-analytics task of Chapter 1. The deterministic control answers whether a request is permitted; the learned control answers whether a permitted request is behaving normally. Combining them gives an access decision that is both policy-compliant and behaviorally monitored, which neither

control achieves alone, and it extends the ensemble principle from the question of what enters the organization to the question of what authenticated insiders and credentials do once inside.

The general lesson of all four pairings is that a deterministic control and a learned control, combined, cover each other's characteristic failure, and that the decision engine is the place where deterministic facts and learned probabilities are reconciled into a single disposition.

4.4 Robustness of the ensemble architecture to attacks

The robustness argument for ensemble integration is its most important property, and it follows directly from decoupling. An attacker facing a standalone control needs to defeat one decision surface; an attacker facing an ensemble needs to defeat several independent surfaces at once, which is harder by construction. Three attack scenarios make the difference concrete, and Table 4.3 contrasts the standalone and ensemble outcomes for each.

The first scenario is adversarial manipulation of the machine-learning layer. Content classifiers degrade under inputs crafted to deceive them, and phishing detectors specifically have been shown to lose accuracy under adversarial perturbation of message text (Gholampour & Verma, 2023; Han et al., 2022). The vulnerability is general to learned classifiers, established for deep models by the demonstration that small crafted perturbations flip confident predictions (Goodfellow et al., 2014), and surveys of adversarial examples in machine-learning-enabled cybersecurity catalogue the breadth of these attacks and the difficulty of fully defending against them within a single model (Macas et al., 2023; Qiu et al., 2019; Chakraborty et al., 2021). In a standalone deployment, defeating the content model defeats the control. In the ensemble, an adversarially perturbed message that fools the content model still carries unaltered reputation, authentication, and behavioral evidence, so the perturbation moves only one term of the aggregate score and the disposition can still be correct. The second scenario is evasion of the sandbox by environment-aware malware, which produces no malicious behavioral evidence; here the reputation, content, and authentication layers remain informative, so the missing sandbox signal degrades the score rather than nullifying the decision. The third scenario is a stale or poisoned reputation feed, against which the content model and sandbox still operate, so the ensemble tolerates the degraded layer.

Failure scenario	**Standalone outcome**	**Ensemble outcome**
ML content layer defeated by adversarial perturbation	Control bypassed; threat delivered	Other layers retain signal; score degrades, decision often holds
Sandbox evaded by environment-aware malware	Attachment cleared as benign	Reputation, content, authentication still contribute
Reputation feed stale or poisoned	Known-bad infrastructure admitted	Content and behavioral layers still operate
One layer offline or overloaded	Coverage gap until restored	Remaining layers sustain a degraded but live decision

Table 4.3. Graceful degradation: standalone versus ensemble outcomes under single-layer failure. Source: author's synthesis based on Gholampour and Verma (2023), Han et al. (2022), Macas et al. (2023), and Aboaoja et al. (2023).

The property that unifies the scenarios is graceful degradation. A standalone control fails as a unit, so its failure is total; an ensemble fails one layer at a time, so its failure is partial and the aggregate decision

degrades smoothly rather than collapsing. This is the architectural reason the framework refuses to let any model be the sole judge, and it is also why adversarial robustness is treated here as a property of system position rather than of model hardening alone. Hardening a single model against adversarial inputs is valuable and is pursued through adversarial training and statistical detection of generated text (Han et al., 2022), but it remains a defense of one surface. Ensemble integration adds a structural defense that does not depend on anticipating the attacker's specific perturbation, because it forces the attacker to defeat several uncorrelated surfaces rather than to find one model's weakness.

The robustness of the ensemble is real but not unconditional, and the principle is weakened by anything that correlates the layers' failures. If two layers share an upstream dependency, a single compromise of that dependency blinds both at once, and the apparent diversity is illusory; a reputation feed and a content blocklist drawn from the same intelligence provider, for instance, fail together when that provider is wrong. Common-mode failure of this kind is the chief threat to a layered defense, and guarding against it is a design obligation: the layers must be audited for shared dependencies, and genuine independence must be engineered rather than assumed. A second limit is that the decision engine itself becomes a target. An adversary who cannot defeat the individual layers may attack the aggregator, probing the thresholds to find the score region in which a malicious artifact is delivered, which is why the engine's weights and thresholds are themselves sensitive configuration and why the ambiguous-zone quarantine matters as a safety margin. A third limit is cost: running several layers per artifact is more expensive than running one, and the latency budget of Chapter 2 constrains how many layers can be applied before delivery. These limits do not refute the principle; they specify the engineering it demands, namely audited independence, a protected aggregator, and a layer budget matched to the latency the deployment can afford.

4.5 Integration with monitoring and response

The decision engine does not end the lifecycle; its outputs feed the systems that correlate events, coordinate response, and produce the labeled outcomes that train the next model. Two integrations close the loop.

The first is the flow of decisions into security monitoring. Dispositions and the evidence behind them are forwarded to security information and event management systems, where they are correlated with events from other controls to reconstruct multi-stage attacks that no single decision could reveal. The operational difficulty here is alert fatigue: contemporary monitoring generates a volume of alerts that overwhelms analysts, and a large fraction are false, which hinders effective response (Ban & Takahashi, 2023). The ensemble helps with this problem rather than adding to it, because a calibrated aggregate decision carries more information than a raw model alert, and because its quarantine zone already separates the confident cases from the ambiguous ones that warrant human attention. Security operations centers, whose importance has grown as centralized security operations have been adopted, are the organizational locus where these correlated decisions become coordinated action (Vielberth et al., 2020). Orchestration and automated response platforms extend this from correlation to action, automating containment steps that would otherwise wait on manual intervention (Ismail & Kurnia, 2025), and the automation of layered, zero-trust controls is itself an active area of architecture research (Cao & Pokhrel, 2024).

The second integration is the feedback loop from incidents back to the models. When an incident is investigated and resolved, its outcome is a high-quality label, confirming or correcting the disposition the engine produced. Feeding these labeled outcomes back into the combined label set of Chapter 3 closes the loop: the system that decided is also the system that learns from how its decisions turned out, and its weights and models are recalibrated on outcomes drawn from its own operation. This is where the three

principles become one system. Pre-emptive decisions are taken early on combined-labeled evidence, aggregated across independent layers, and the results of those decisions become new combined labels that refine the next cycle. The architecture is not three separate techniques but a single loop in which timing, labeling, and aggregation reinforce one another, and the explainability that the aggregated decision provides, with each layer's contribution visible alongside the others, is what makes the loop auditable and the system trustworthy to the analysts who operate it (Zhang et al., 2022). The linear aggregation is interpretable by construction, and where per-feature attribution within a layer is needed, model-agnostic explanation methods such as additive feature attribution supply it (Lundberg & Lee, 2017), so that an analyst can trace a disposition not only to the layer that drove it but, within that layer, to the features that mattered.

The loop has a cadence, and setting it is an operational decision with its own trade-offs. Retraining and recalibration too rarely lets the model drift away from the current threat distribution, the dataset-shift problem of Chapter 1 reasserting itself as the world moves on from the training data; retraining too often is expensive and risks chasing noise, recalibrating on short-term fluctuations that do not represent a real change in the threat. A disciplined deployment therefore monitors for drift rather than retraining on a fixed clock alone, tracking the divergence between recent traffic and the training distribution and the stability of each layer's calibration, and triggering recalibration when the divergence exceeds a threshold. The combined-labeling pipeline of Chapter 3 supplies the labeled incident outcomes that make each retraining cycle meaningful, and the weighted engine of this chapter is what lets a single drifting layer be down-weighted between full retrains, so that the system can respond to a degrading layer without a complete rebuild. Drift monitoring thus sits at the junction of all three principles: it watches the timeliness that the pre-emptive principle depends on, it consumes the labels the second principle produces, and it adjusts the weights the third principle aggregates. Chapter 5 verifies that this loop, realized in a single deployed system, produces a measurable improvement over the conventional baseline.

Chapter 5. End-to-End Verification of the Framework on a Practical Case

The three principles developed in the preceding chapters are claims about how machine learning should be applied in enterprise defense. A claim of this kind is credible only if a system built on all three, together, produces a measured improvement under production conditions. This chapter supplies that verification. It describes the object on which the framework was apprised, shows how each of the three principles is realized in the system's architecture, sets out the experimental methodology, reports the confirmed results, and analyzes how far the verified principles transfer to adjacent security tasks. The system is the Mail Security Guardian email security platform, and the figures reported here are drawn from the author's peer-reviewed study of it (Kolchin, 2025); they are presented as the empirical anchor of the framework, not as claims about a commercial product.

5.1 The object of verification

The verification was conducted on the inbound electronic mail of a corporate infrastructure of approximately 2,500 users, processing on the order of 120,000 messages per day. The choice of email as the verification domain is deliberate, because email concentrates the conditions the framework is designed for: it is the dominant enterprise attack surface, it carries spam, phishing, and malware through a single stream that demands different detection logics, and it offers a perimeter at which a pre-emptive decision can be taken. The infrastructure is described in anonymized terms throughout, consistent with the confidentiality under which the deployment was conducted, and no client identity, sector detail, or deployment geography is required for the verification, which concerns the architecture and its measured performance rather than the identity of the organization protected.

The threat landscape on this infrastructure is the ordinary one of a large enterprise mailbox estate. Bulk spam imposes the highest volume, targeted phishing and business email compromise the highest per-incident risk, and malware-bearing attachments the most severe potential impact. The class distribution is heavily imbalanced toward legitimate mail, which is the production condition that defeats naively trained classifiers and that the methodology of Chapter 3 is built to handle. The evaluation conditions are stated plainly so that the results can be read for what they are: a six-month observation of a single production deployment, with a three-month pre-deployment reference period and a three-month post-deployment measurement period, rather than a controlled multi-site trial. This is the scope within which the framework is verified, and the transferability analysis of Section 5.5 is careful not to claim beyond it.

The control that the framework replaced is the natural baseline for the verification. Before deployment the infrastructure was protected by a conventional secure email gateway of the kind described in Chapter 2, applying signature matching and reputation lookups at or after delivery. This baseline is not a strawman; it is the standard production control for enterprise mail, and the fact that the comparison is against a real deployed gateway rather than against a textbook classifier is what makes the improvement meaningful. The pre-deployment period measured the threat background that this conventional control allowed through, and the post-deployment period measured the background under the framework, so the comparison is between two production configurations of the same infrastructure rather than between a laboratory model and an abstraction. Holding the infrastructure, the user population, and the traffic constant across the two periods is what isolates the architectural change as the cause of the measured difference.

5.2 Architectural realization of the three principles

The system realizes all three principles in a single distributed architecture, which is what makes it a verification of the framework rather than of any one principle. The implementation is a microservice architecture in which each analytical layer is an independently deployable service, the services communicate through a message queue, and containerization with orchestration manages their lifecycle and scaling (Kolchin, 2025). This implementation substrate is what allows the layers to be decoupled in the sense Chapter 4 requires, scaled independently, and combined at a final decision stage.

The first principle, pre-emptive architecture, is realized as a pre-delivery model. The gateway receives inbound connections on behalf of the protected domain, evaluates each message through sequential layers, and forwards only passing messages to the internal mail server, so that all analysis completes before the message reaches its destination. The layers are those introduced in Chapter 2: pre-SMTP reputation filtering at the connection level, protocol authentication, content and machine-learning analysis, dynamic sandbox analysis for attachment-bearing or high-risk messages, and a final decision engine. The pre-SMTP stage decides on connection metadata before content is transmitted, which is the earliest possible decision point and the purest expression of the pre-emptive principle.

The second principle, combined labeling, is realized in the construction of the training data. The evaluation corpus of 3.6 million messages collected over six months was labeled through three independent evidence streams: user-reported verdicts verified by security operations personnel, sandbox analysis outcomes cross-referenced against multi-engine threat intelligence, and retrospective threat-intelligence enrichment that reclassified initially ambiguous messages once corresponding indicators were published (Kolchin, 2025). This is precisely the combined-labeling methodology of Chapter 3, including the retrospective relabeling that shortens the intelligence lag inside the training set, and the deployment confirms that the methodology is implementable at the scale of millions of messages rather than only in principle.

The third principle, ensemble integration, is realized in both the modeling and the decision stages. The core classifier is an ensemble of four architectures spanning linear, tree-based, and neural families: logistic regression as a linear baseline, random forest as a bootstrapped tree ensemble (Breiman, 2001), regularized gradient-boosted trees (Chen & Guestrin, 2016), and a multilayer perceptron for non-linear interactions. Above the classifier, the decision engine aggregates the evidence of all layers through the weighted scoring function of Chapter 4, combining reputation, content, machine-learning, and sandbox signals into a single disposition. The system thus integrates ensembles at two levels, within the content classifier and across the independent layers, which is the architecture the third principle prescribes. The implementation uses established components for each role, with the high-throughput proxy, the message queue, the model-training libraries, the audit store, and the search index each chosen for its layer's requirements, so that the architecture is reproducible from standard building blocks rather than dependent on bespoke infrastructure.

The feature representation on which the classifier operates is itself an expression of the ensemble principle, because it concatenates signals from every layer rather than from message content alone. The feature vector spans five categories (Kolchin, 2025). Reputation and network features encode the sender infrastructure, including aggregate blocklist reputation, geographic source classification, and connection behavior. Authentication features encode the outcomes of the sender-policy, signature, and domain-alignment checks, including signature-domain alignment. Text features represent the message body through term-frequency vectorization of normalized content. Attachment features capture file type, extension, archive nesting depth, and payload size. Url and behavioral features encode the count and

structure of links, character-entropy statistics, the presence of address literals, and the minimum edit distance from each link to a watchlist of protected brand domains. The classifier therefore sees provenance, authentication, content, attachment, and link evidence in a single vector, so that the model itself already fuses modalities before its output is combined with the other layers at the decision engine. This is why the framework integrates ensembles at two levels: the classifier fuses feature modalities, and the decision engine fuses layer verdicts, and the two levels of fusion are what give the system its robustness to any single evaded signal.

The decoupled microservice substrate is what makes this multi-level fusion operationally viable. Because each layer is an independent service communicating through a message queue, the compute-intensive layers, model inference and sandbox detonation, scale separately from the lightweight reputation and authentication services, and a surge in attachment-bearing mail can be absorbed by scaling the sandbox cluster alone. The same decoupling allows the content model to be retrained and redeployed without disturbing the authentication or reputation layers, which is the maintainability property of Chapter 4 realized in the deployment. The architecture thus reproduces, in running code, the decoupling that the third principle requires in theory.

5.3 Experimental methodology

The methodology follows the evaluation discipline argued for in Chapters 1 and 3, and its central choice is the temporal split. The evaluation set consists exclusively of messages received after every training message, so the classifier is tested on threats that postdate its training data (Kolchin, 2025). As Chapter 3 argued, this is not a stylistic preference but a precondition for trusting the result: a random split would let near-duplicate threats appear in both partitions and inflate measured performance through memorization, whereas the temporal split confronts the classifier with the novel variants it will actually face in deployment. The reported numbers are therefore estimates of operational generalization rather than of benchmark fit.

The decision to evaluate this way carries a cost that is worth naming, because it explains why the reported figures should be read as conservative. A temporal split is harder to score well on than a random split, since the classifier is denied any glimpse of the test-period threats during training and must generalize to genuinely novel variants. A system that reports an F1-score of 0.985 under a temporal split is therefore reporting a stronger result than the same number would represent under a random split, where part of the score would be memorization. Many published benchmarks use random splits and report figures that do not survive the move to temporal evaluation, which is one reason laboratory accuracy so often fails to reproduce in production. By measuring under the harder protocol, the verification trades a higher headline number for a trustworthy one, and the comparison against the baseline is fair because both systems were evaluated under the identical temporal split on the identical data.

Class imbalance was handled as Chapter 3 prescribes, through synthetic minority oversampling (Chawla et al., 2002) applied to the training partition combined with cost-sensitive weighting during optimization, so that the heavy majority of legitimate mail did not train the classifier to ignore the malicious minority. Performance was quantified with the standard classification metrics: precision, the fraction of messages flagged malicious that are in fact malicious; recall, the fraction of malicious messages that are correctly flagged; the F1-score, their harmonic mean; and the area under the receiver-operating-characteristic curve, which summarizes class separation across all decision thresholds. Operational performance was measured separately through end-to-end latency and per-node throughput, because a pre-emptive control that holds messages while it decides must meet a latency budget as well as an accuracy target. Table 5.1 summarizes the dataset and evaluation conditions.

Property	Value
Evaluation corpus	3.6 million inbound messages
Collection period	6 months
Protected infrastructure	approximately 2,500 users
Daily traffic volume	approximately 120,000 messages per day
Labeling	Combined: user feedback, sandbox, threat intelligence (with retrospective relabeling)
Class balancing	Synthetic minority oversampling plus cost-sensitive weighting
Train/test split	Temporal holdout (test strictly after training)
Deployment measurement	3-month pre-deployment reference, 3-month post-deployment

Table 5.1. Dataset and evaluation conditions for the framework verification. Source: Kolchin (2025).

5.4 Results

On the temporal holdout set, the machine-learning pipeline achieved a precision of 0.992, a recall of 0.978, an F1-score of 0.985, and an area under the ROC curve of 0.996 (Kolchin, 2025). The precision of 0.992 corresponds to a false-positive rate of 0.8 percent, a level at which flagged messages can be reviewed through quarantine without an unmanageable burden on security staff, and the recall of 0.978 means that 97.8 percent of malicious messages were intercepted. The area under the ROC curve of 0.996 indicates near-complete class separation across the threshold range, which is what allows operators to move the decision boundary to match their risk tolerance without collapsing performance. Table 5.2 lists the classifier metrics.

Metric	Value
Precision	0.992
Recall	0.978
F1-score	0.985
AUC-ROC	0.996
False-positive rate (derived)	0.8%

Table 5.2. Machine-learning pipeline performance on the temporal holdout set. Source: Kolchin (2025).

The binary confusion matrix at the production threshold confirms the balance between the two error types. Of legitimate messages, 98.6 percent were correctly delivered and 1.4 percent were flagged as malicious; of malicious messages, 99.3 percent were correctly intercepted and 0.7 percent were missed (Kolchin, 2025). Figure 5.3 shows the matrix. The asymmetry is operationally appropriate: the system is tuned so that its residual error falls slightly more on the side of over-caution, where a quarantined legitimate message can be released on review, than on the side of a missed threat, which cannot be recalled once delivered.

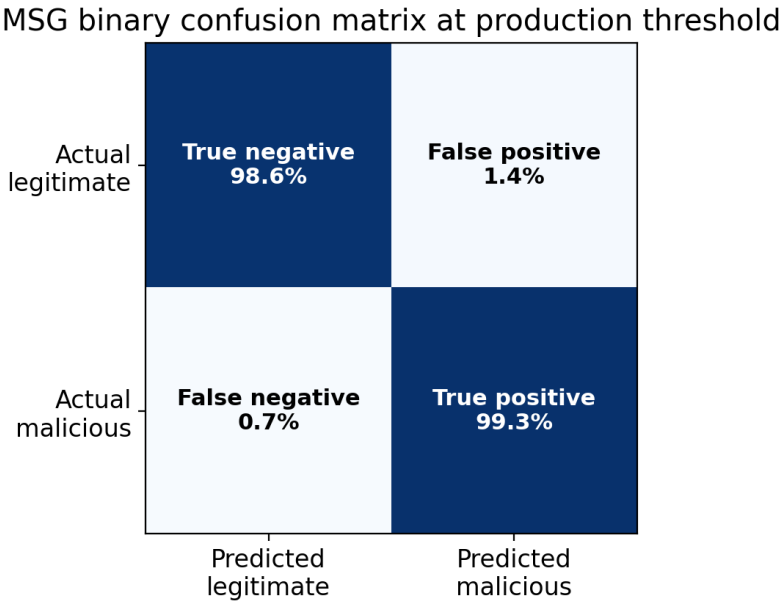

Figure 5.3. *Binary confusion matrix at the production threshold. Source: author's figure from Kolchin (2025).*

The value of the framework is clearest in comparison with the conventional control it replaced. Evaluated under identical conditions on the same temporal holdout set, a conventional secure email gateway achieved an F1-score of 0.910, against the framework's 0.985, an improvement of 7.5 percentage points (Kolchin, 2025). Figure 5.5 shows the comparison. On the production traffic volume, this difference corresponds to roughly 9,000 additional messages classified correctly each day, which converts an abstract gain in F1-score into a concrete daily reduction in threats delivered and legitimate messages misrouted. The improvement is attributable to the architecture rather than to any single model: the gateway baseline applies signature and reputation logic reactively, while the framework supplies its classifier with pre-processed reputation and authentication features and aggregates the classifier with independent layers, which is the ensemble-integration advantage of Chapter 4 made measurable.

Placed against the wider literature, the result is strong but not anomalous, which is the appropriate outcome for an architectural rather than a modeling contribution. Recent deep-learning studies of phishing email detection report F1-scores in the upper-0.9 range, and transformer-based classifiers, building on the pretrained language-model architectures that reshaped text classification (Devlin et al., 2019), reach comparable figures on balanced datasets (Altwaijry et al., 2024; Jamal et al., 2024). The framework's F1-score sits at the upper end of that range, but the relevant point is not that it edges out individual models on their own benchmarks; it is that the framework reaches this level under a temporal split on production traffic and as the byproduct of an architecture whose advantages are robustness and graceful degradation rather than raw accuracy. Multi-modal integration, the combination of metadata with content, consistently outperforms single-modality classification in the literature (Doshi et al., 2023; Innab et al., 2024), and the framework is the systematic realization of that finding: its accuracy is what multi-modal, multi-layer integration produces when it is engineered end to end rather than assembled from a single content model. The contribution is the architecture that makes the result reproducible and robust, not a novel classifier that happens to score highly.

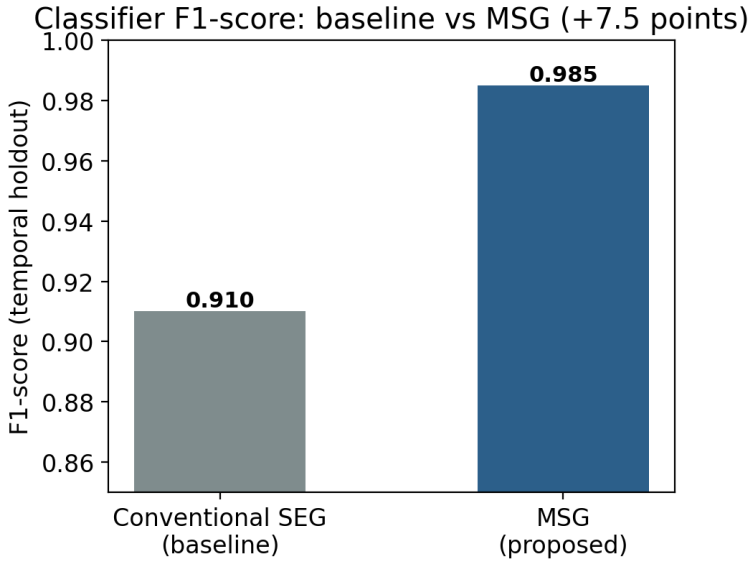

Figure 5.5. *Classifier F1-score, conventional secure email gateway baseline versus the framework, on the same temporal holdout set. Source: author's figure from Kolchin (2025).*

The production deployment translated the classifier's accuracy into a measured reduction in the threat background reaching users. Across the pre- and post-deployment periods, spam prevalence in inbound mail fell from 18.0 to 1.9 percent, phishing from 4.2 to 0.3 percent, and malware from 2.1 to 0.05 percent, reductions of 89.4, 92.9, and 97.6 percent respectively (Kolchin, 2025). Figure 5.4 shows the before-and-after prevalence. The pattern across categories reflects the multi-layer architecture directly. Spam reduction combines pre-SMTP reputation filtering, which blocks bulk delivery at the connection level, with content classification that extends coverage to campaigns absent from reputation feeds. Phishing reduction demonstrates multi-layer complementarity at its clearest, since pre-SMTP filtering, authentication verification, url analysis, and content classification each address a different facet of the attack. Malware reduction reflects the sandbox layer catching both known samples, through static matching, and novel variants, through behavioral analysis.

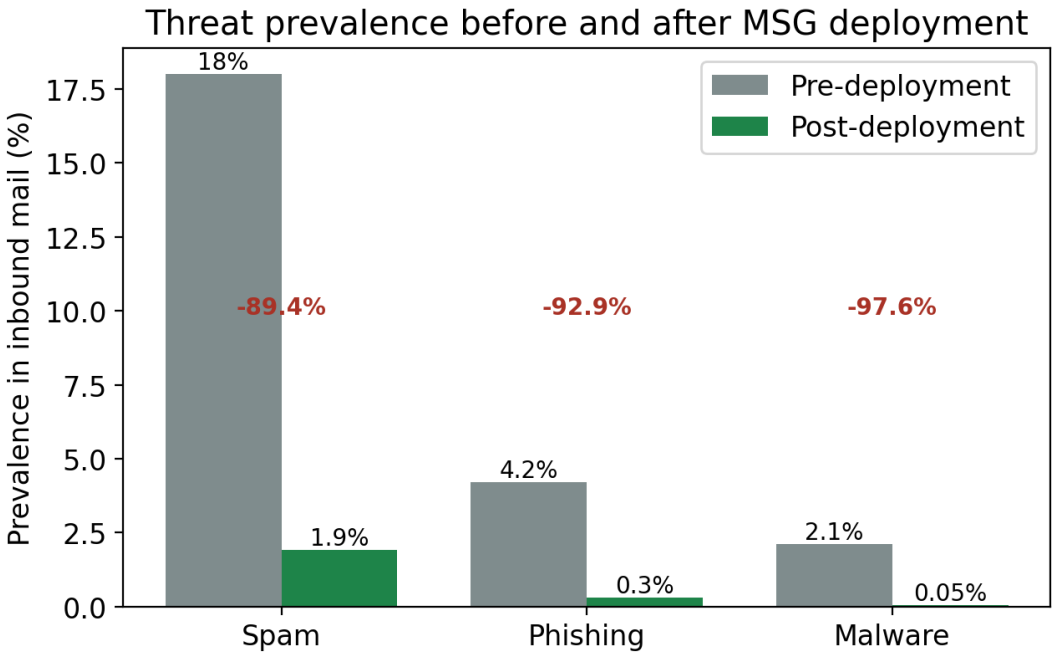

Figure 5.4. *Threat prevalence in inbound mail before and after deployment, with relative reductions. Source: author's figure from Kolchin (2025).*

The framework also met its operational requirements, which matters because a pre-emptive control that is accurate but slow is unusable. End-to-end analysis latency for messages processed without sandbox engagement ranged from 120 to 350 milliseconds, sandbox-routed messages incurred bounded additional analysis time of up to 5 seconds per sample, and per-node throughput reached 50,000 messages per hour, scaling linearly with node count because the microservice architecture decouples the compute-intensive layers from the lightweight ones (Kolchin, 2025). Table 5.6 lists the operational figures. Together with the accuracy results, these confirm that the framework satisfies the latency budget the pre-emptive principle imposes while delivering the accuracy the ensemble principle promises.

Operational metric	**Value**
End-to-end latency (no sandbox)	120–350 milliseconds
Sandbox analysis (per sample)	bounded at 5 seconds
Per-node throughput	50,000 messages per hour
Scaling	linear with node count

Table 5.6. Operational performance of the framework. Source: Kolchin (2025).

The operational meaning of these numbers is worth making explicit, because a security result is only as good as the workload it imposes on the people who run it. A false-positive rate of 0.8 percent on a six-figure daily volume is not negligible in absolute terms, but the architecture handles it without overwhelming analysts because the decision engine routes uncertain cases to quarantine rather than deleting them, so a flagged legitimate message is held and can be released on review rather than lost. The recall of 0.978 means a small fraction of malicious messages still reach users, which is why the framework is a layer in a defense rather than a guarantee, and why the feedback loop of Chapter 4 matters: each missed message that is later reported becomes a labeled example that improves the next retraining cycle. The deployment results should therefore be read not as a claim of perfection but as a measured, sustained reduction in the threat background, achieved at a false-positive cost that a quarantine workflow

can absorb. That combination, a large reduction in delivered threats at a manageable review burden, is the practical outcome the framework is designed to produce.

The results meet the standard of evidence set out in Chapter 1. The improvement is measured under a temporal split that mirrors deployment, against a baseline evaluated on identical data, on a corpus of millions of real messages, and it is confirmed both in classifier metrics and in the production threat background. The numbers are the author's own, drawn from a peer-reviewed study of the system, and they verify the framework as a whole rather than any single principle in isolation.

5.5 Transferability of the framework to adjacent security tasks

The verification was conducted on email, and intellectual honesty requires distinguishing what has been demonstrated from what is projected. What has been demonstrated is that the three principles, realized together in a single email security system, produce a measured improvement over a conventional baseline under production conditions. What is projected is that the same three principles transfer to adjacent security tasks, and this section sets out the basis for that projection and its limits, without claiming for the other domains the verification that only the email case carries.

The strongest transfer is to external digital-risk protection, because it shares email's structure most closely. The pre-emptive principle applies directly: phishing and fraudulent domains can be detected from their structure during the dormant interval before they are used, exactly as malicious messages are filtered before delivery. The combined-labeling principle applies, because external threat detection benefits from the same fusion of user reports, automated analysis, and threat intelligence. The ensemble principle applies, because a digital-risk decision is best taken by aggregating structural, reputational, and content signals rather than by any one. The transfer is therefore well-founded in architecture, and the author's digital-risk system is engineered around these principles, though production metrics for it are outside the scope of this monograph and the transfer is presented as design rather than as a measured result.

A second transfer is to continuous vulnerability and attack-surface management. Here the pre-emptive principle takes its prioritization form, acting on the predicted likelihood of exploitation before exploitation occurs, and the ensemble principle takes the form of combining automated prioritization with expert verification. The combined-labeling principle transfers less directly, because vulnerability data are labeled differently from message traffic, but the underlying idea, that ground truth should be drawn from several independent signals rather than from static severity alone, holds. The author's continuous penetration-testing and asset-monitoring system is engineered around continuous interception and predictive prioritization, again as a design realization rather than a metrically verified one.

The limits of transfer must be stated as plainly as its basis. The verified result is specific to email, to one infrastructure, and to one six-month period, and the transfer to other domains rests on architectural analogy rather than on independent measurement. Domains that lack a clean perimeter at which to intercept, or that cannot assemble independent label sources, or whose latency budgets cannot accommodate several layers, will realize the principles less fully, and the framework does not claim a uniform gain across all security tasks. What the framework does claim, and what the email verification supports, is that where a task has a preparation phase that can be observed before impact, where ground truth can be drawn from independent sources, and where a decision can aggregate independent layers, the three principles applied together will outperform the reactive, single-source, standalone alternative. The email case shows that claim realized and measured; the adjacent domains show it as a design direction whose verification is the natural subject of future work.

What such verification would require is worth specifying, because it defines the research program the framework implies. A verification of the digital-risk transfer would measure the lead time between a malicious domain's registration and its structural detection, and compare the campaign success rate against a reactive baseline that acts only after the first phishing report, holding the monitored brand set and observation window constant in the way the email comparison held the infrastructure constant. A verification of the vulnerability-management transfer would measure whether predictive prioritization, acting before exploitation, reduces the number of exploited exposures relative to severity-based prioritization over a matched period, again with a clean before-and-after or controlled comparison. In each case the methodology would mirror the email verification: a real deployment, a genuine baseline, a temporally honest evaluation, and metrics that capture the quantity the principle is meant to improve. The email result is one instance of that methodology carried to completion, and it establishes both that the framework can be verified and how the verification should be done. Extending it to the adjacent domains is the path from a verified single-domain framework to a verified general one, and it is the direction the concluding chapter identifies for the framework's continued development.

Conclusion

This monograph has argued that the effectiveness of machine learning in enterprise cybersecurity is governed by architecture rather than by the choice of model, and it has developed and verified a framework built on that premise. The argument began from a diagnosis: across the literature and across industrial practice, machine-learning controls fail in three recurring ways that no improvement in the model itself can repair. They act reactively, after a threat has been delivered or activated. They learn from homogeneously labeled data that carries the systematic bias of a single source. And they are deployed in isolation, as standalone judges exposed at a single decision surface. Each failure is a property of how the model is positioned in time, in its data, and in the system, and each is therefore answerable only by an architectural choice.

The framework answers the three failures with three connected principles. Pre-emptive architecture moves the point of decision to the stage before delivery or activation, closing the window of compromise that reactive designs leave open. Combined labeling draws training ground truth from several independent evidence streams at once, cancelling the systematic bias that any single source imposes. Ensemble integration embeds the model as one weighted layer among independent controls, so that the decision survives the evasion of any single layer and degrades gracefully rather than collapsing. The principles are not three separate techniques but three facets of one stance, and they reinforce one another in a single loop: pre-emptive decisions are taken early on combined-labeled evidence, aggregated across independent layers, and the outcomes of those decisions become new combined labels that refine the next cycle. The unity of the three is the contribution. Each principle has antecedents in the literature, but their integration into one coherent, verified methodology is what the published record did not contain before this work.

The contribution is verified rather than asserted. The framework was realized in a complete email security system and evaluated on 3.6 million messages over six months from a corporate infrastructure of approximately 2,500 users, under a temporal split that mirrors operational deployment. Its machine-learning pipeline reached an F1-score of 0.985 and an area under the ROC curve of 0.996, an improvement of 7.5 percentage points over the conventional secure email gateway it replaced, and its production deployment reduced spam, phishing, and malware prevalence by 89.4, 92.9, and 97.6 percent respectively (Kolchin, 2025). These figures are the empirical anchor of the framework. They show that the three principles, applied together and engineered end to end, produce a measured and substantial improvement over the reactive, single-source, standalone alternative, under the production conditions that defeat laboratory accuracy.

For the practitioner who designs machine-learning security systems, the framework offers a discipline rather than a product. It directs the designer to ask, before selecting any model, three questions that determine the ceiling the model can reach: at what point in the threat lifecycle the system will be permitted to act, from how many independent sources its training data will be labeled, and how its output will be combined with the other controls in the defense. A system that answers these questions well will outperform one that answers them poorly regardless of the sophistication of its classifier, because the questions set the ceiling and the classifier only approaches it. The framework is also architectural in a second, practical sense: because it positions the model as one layer among independent controls, it can be integrated into an existing security estate without replacing it, adding a calibrated machine-learning layer to reputation, authentication, and behavioral controls already in place rather than demanding their removal. This is what makes the framework adoptable in the messy, layered environments that real enterprises run, rather than only in a system built from scratch.

The framework also marks the boundaries of what it claims. Its quantitative verification is specific to email, to one infrastructure, and to one six-month period. The transfer of the principles to external digital-risk protection and to continuous vulnerability management rests on architectural analogy and on systems engineered around the principles, not on independent measurement, and the monograph has been explicit throughout in separating what is demonstrated from what is designed. Verifying the transfers, by the same disciplined methodology that the email case followed, is the most immediate direction for further work, and the form such verification should take has been specified in Chapter 5.

Beyond verifying the transfers, four directions extend the framework. The first is federated learning, which would let independent deployments improve a shared model without centralizing raw message data, expanding the diversity of training data that the combined-labeling principle depends on while preserving the confidentiality that enterprise mail requires; federated averaging established that models can be trained across decentralized data without centralizing it (McMahan et al., 2016), and the open problems of federated learning at scale remain an active research area (Kairouz et al., 2021; Wen et al., 2022). The second is defense against adversarial machine learning, which grows more pressing as attackers craft inputs specifically to defeat learned layers; the framework's ensemble integration already raises the cost of such attacks structurally, but combining it with model-level adversarial hardening is a natural extension (Macas et al., 2023). The third is the integration of large-language-model components, on both sides of the contest: adversaries already use language models to generate fluent malicious content at scale (Hao et al., 2025; Roy et al., 2024), and the defense will need language-model-aware detection layers and, increasingly, language-model assistance in triage and response. The fourth is the application of post-quantum considerations to the trust and authentication signals on which the framework's early decisions rest, so that the provenance evidence that anchors a pre-emptive decision remains sound as cryptographic assumptions evolve.

The thesis of this monograph is finally simple. Machine learning earns its place in enterprise defense not by being the most accurate classifier on a benchmark but by being correctly positioned in time, correctly taught from diverse evidence, and correctly combined with the controls around it. The three principles make that positioning explicit, and the verified result shows what their combination achieves. The work of the field ahead is to extend the same architectural discipline to the security tasks that email here exemplifies, so that the gap between what machine learning can do in a paper and what it does in a production defense continues to close.

Glossary

Combined labeling. The second principle of the framework: the practice of labeling training data through several independent evidence streams at once (for example user feedback, sandbox analysis, and threat intelligence), so that the systematic bias of any single source is cancelled by the others.

Concept drift (dataset shift). The change over time in the statistical distribution of the data a model sees, so that a model trained on past data degrades as the threat distribution moves away from its training distribution.

Decision engine. The component that aggregates the calibrated signals of several independent defense layers into a single disposition, typically through a weighted scoring function compared against thresholds.

Decoupling. The architectural property by which defense layers are independent in implementation and in failure, so that the conditions that blind one layer do not blind the others.

Defense in depth. A security posture in which an artifact must pass several independent controls rather than a single gate, so that the failure of one control does not breach the defense.

Ensemble integration. The third principle of the framework: embedding a machine-learning model as one weighted layer within a multi-layer defense, with the final decision taken by aggregating independent layers rather than by trusting any one.

External attack surface management (EASM). The continuous discovery, inventory, and assessment of an organization's externally visible and hidden assets, used to anticipate exposure before exploitation.

Graceful degradation. The behavior of a layered system whose performance declines smoothly when one layer is blinded or offline, rather than collapsing as a standalone control does when defeated.

Pre-emptive architecture. The first principle of the framework: applying machine learning before a threat is delivered or activated rather than after, so as to close the window of compromise.

Pre-SMTP filtering. Evaluation of an inbound mail connection at the transport and envelope level, before message content is transmitted, allowing a connection to be refused before any content crosses the wire.

Temporal (time-based) split. An evaluation protocol in which the test set consists exclusively of data postdating all training data, so that the model is assessed on genuinely novel threats and data leakage is avoided.

Window of compromise. The interval between the moment a threat becomes capable of causing harm within an environment and the moment the defense neutralizes it; the metric the pre-emptive principle is designed to minimize.

Zero trust. A security model that replaces implicit, perimeter-based trust with continuous per-request verification of identity and posture across independent controls.

Bibliography

1. Aboaoja, F. A., Zainal, A. B., Ali, A. M., Ghaleb, F. A., Alsolami, F. J., & Rassam, M. A. (2023). Dynamic extraction of initial behavior for evasive malware detection. *Mathematics, 11*(2), 416. https://doi.org/10.3390/math11020416

2. Adnan, M., Imam, M. O., Javed, M. F., & Murtza, I. (2024). Improving spam email classification accuracy using ensemble techniques: A stacking approach. *International Journal of Information Security*, *23*, 505–517. https://doi.org/10.1007/s10207-023-00756-1

3. Afianian, A., Niksefat, S., Sadeghiyan, B., & Baptiste, D. (2019). Malware dynamic analysis evasion techniques: A survey. *ACM Computing Surveys, 52*(6), 126. https://doi.org/10.1145/3365001

4. Ahmad, Z., Khan, A. S., Shiang, C. W., Abdullah, J., & Ahmad, F. (2020). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1). https://doi.org/10.1002/ett.4150

5. Ahmed, N., Amin, R., Aldabbas, H., Koundal, D., Alouffi, B., & Shah, T. (2022). Machine learning techniques for spam detection in email and IoT platforms: Analysis and research challenges. *Security and Communication Networks, 2022*, 1862888. https://doi.org/10.1155/2022/1862888

6. Alabdan, R. (2020). Phishing Attacks Survey: Types, Vectors, and Technical Approaches. *Future Internet*, 12(10), 168. https://doi.org/10.3390/fi12100168

7. Aldweesh, A., Derhab, A., & Emam, A. (2019). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, 189, 105124. https://doi.org/10.1016/j.knosys.2019.105124

8. Aljofey, A., Jiang, Q., Qu, Q., Huang, M., & Niyigena, J. (2020). An Effective Phishing Detection Model Based on Character Level Convolutional Neural Network from URL. *Electronics*, 9(9), 1514. https://doi.org/10.3390/electronics9091514

9. Almujahid, N. F., Haq, M. A., & Alshehri, M. (2024). Comparative evaluation of machine learning algorithms for phishing site detection. *PeerJ Computer Science, 10*, e2131. https://doi.org/10.7717/peerj-cs.2131

10. Alnemari, S., & Alshammari, M. (2023). Detecting phishing domains using machine learning. *Applied Sciences, 13*(8), 4649. https://doi.org/10.3390/app13084649

11. Altwaijry, N., Al-Turaiki, I., Alotaibi, R., & Alakeel, F. (2024). Advancing phishing email detection: A comparative study of deep learning models. *Sensors, 24*(7), 2077. https://doi.org/10.3390/s24072077

12. Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A. Q., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal Of Big Data*, 8(1), 53. https://doi.org/10.1186/s40537-021-00444-8

13. Anti-Phishing Working Group. (2023). *Phishing activity trends report: 4th quarter 2023*. APWG. https://docs.apwg.org/reports/apwg_trends_report_q4_2023.pdf

14. Apruzzese, G., Laskov, P., de, E. M., Mallouli, W., Rapa, L. B., Grammatopoulos, A. V., & Franco, F. D. (2022). The Role of Machine Learning in Cybersecurity. *Digital Threats Research and Practice*, 4(1), 1–38. https://doi.org/10.1145/3545574

15. Atlam, H. F., & Oluwatimilehin, O. (2023). Business email compromise phishing detection based on machine learning: A systematic literature review. *Electronics*, *12*(1), 42. https://doi.org/10.3390/electronics12010042

16. Bagui, S., & Li, K. (2021). Resampling imbalanced data for network intrusion detection datasets. *Journal Of Big Data*, 8(1). https://doi.org/10.1186/s40537-020-00390-x

17. Ban, T., Takahashi, T., Ndichu, S., & Inoue, D. (2023). Breaking alert fatigue: AI-assisted SIEM framework for effective incident response. *Applied Sciences*, *13*(11), 6610. https://doi.org/10.3390/app13116610

18. Basit, A., Zafar, M., Liu, X., Javed, A. R., Jalil, Z., & Kifayat, K. (2020). A comprehensive survey of AI-enabled phishing attacks detection techniques. *Telecommunication Systems*, 76(1), 139–154. https://doi.org/10.1007/s11235-020-00733-2

19. Bensaoud, A., Kalita, J., & Bensaoud, M. (2024). A survey of malware detection using deep learning. *Machine Learning with Applications*, 16, 100546. https://doi.org/10.1016/j.mlwa.2024.100546

20. Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32. https://doi.org/10.1023/a:1010933404324

21. Cao, Y., & Pokhrel, S. R. (2024). Automation and orchestration of zero trust architecture: Potential solutions and challenges. *Machine Intelligence Research*, *21*, 294–317. https://doi.org/10.1007/s11633-023-1456-2

22. Cervantes, J., García-Lamont, F., Rodríguez-Mazahua, L., & López-Chau, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408, 189–215. https://doi.org/10.1016/j.neucom.2019.10.118

23. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2021). A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1), 25–45. https://doi.org/10.1049/cit2.12028

24. Chawla, N. V., Bowyer, K. W., Hall, L., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321–357. https://doi.org/10.1613/jair.953

25. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Pages 785 - 794. https://doi.org/10.1145/2939672.2939785

26. Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 4171–4186. https://doi.org/10.18653/v1/n19-1423

27. Doshi, J., Parmar, K., Sanghavi, R., & Shekokar, N. M. (2023). A comprehensive dual-layer architecture for phishing and spam email detection. *Computers & Security*, *133*, 103378. https://doi.org/10.1016/j.cose.2023.103378

28. Elsadig, M., Ibrahim, A. O., Basheer, S., Alohali, M. A., Alshunaifi, S., Alqahtani, H. M., Alharbi, N., & Nagmeldin, W. (2022). Intelligent Deep Machine Learning Cyber Phishing URL Detection Based on BERT Features Extraction. *Electronics*, 11(22), 3647. https://doi.org/10.3390/electronics11223647

29. Engelen, J. E. v., & Hoos, H. H. (2019). A survey on semi-supervised learning. *Machine Learning*, 109(2), 373–440. https://doi.org/10.1007/s10994-019-05855-6

30. Fang, Y., Liu, Y., Huang, C., & Liu, L. (2020). FastEmbed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm. *PLoS ONE*, 15(2), e0228439. https://doi.org/10.1371/journal.pone.0228439

31. Gamage, S., & Samarabandu, J. (2020). Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 169, 102767. https://doi.org/10.1016/j.jnca.2020.102767

32. Gambo, L., & Almulhem, A. (2025). Zero trust architecture: A systematic literature review. *Journal of Network and Systems Management*. https://doi.org/10.1007/s10922-025-09998-x

33. Gholampour, P. M., & Verma, R. M. (2023). Adversarial robustness of phishing email detection models. In *Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics (IWSPA '23)* (pp. 67–76). ACM. https://doi.org/10.1145/3579987.3586567

34. Goodfellow, I., Shlens, J., & Szegedy, C. (2014). Explaining and Harnessing Adversarial Examples. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1412.6572

35. Gopinath, M. A., & Sethuraman, S. C. (2022). A comprehensive survey on deep learning based malware detection techniques. *Computer Science Review*, 47, 100529. https://doi.org/10.1016/j.cosrev.2022.100529

36. Han, X., & Zhang, T. (2022). Text adversarial attacks and defenses: Issues, taxonomy, and perspectives. *Security and Communication Networks, 2022*, 6458488. https://doi.org/10.1155/2022/6458488

37. Hao, W., Tran, V., Rideout, V., Wang, Z., Dasbach-Prisk, A., Afifi, M. H., Yang, J., Katz-Bassett, E., Ho, G., & Cidon, A. (2025). Do spammers dream of electric sheep? Characterizing the prevalence of LLM-generated malicious emails. In *Proceedings of the 2025 ACM Internet Measurement Conference (IMC '25)* (pp. 192–203). ACM. https://doi.org/10.1145/3730567.3732922

38. Hazell, J. (2023). Spear phishing with large language models. *arXiv preprint* arXiv:2305.06972. https://doi.org/10.48550/arXiv.2305.06972

39. Hemalatha, J., Roseline, S. A., Geetha, S., Kadry, S., & Damaševičius, R. (2021). An Efficient DenseNet-Based Deep Learning Model for Malware Detection. *Entropy*, 23(3), 344. https://doi.org/10.3390/e23030344

40. Hureau, O., Bayer, J., Duda, A., & Korczyński, M. (2024). Spoofed emails: An analysis of the issues hindering a larger deployment of DMARC. In *Passive and Active Measurement (PAM 2024)*, Lecture Notes in Computer Science (Vol. 14537, pp. 232–261). Springer. https://doi.org/10.1007/978-3-031-56249-5_10

41. Innab, N., Osman, A. A. F., Ataelfadiel, M. A. M., Abu-Zanona, M., Elzaghmouri, B. M., Zawaideh, F. H., & Alawneh, M. F. (2024). Phishing attacks detection using ensemble machine learning algorithms. *Computers, Materials & Continua, 80*(1), 1325–1345. https://doi.org/10.32604/cmc.2024.051778

42. Ismail, I., Kurnia, R., Brata, Z. A., Nelistiani, G. A., & Kim, H. (2025). Toward robust security orchestration and automated response in security operations centers. *Information, 16*(5), 365. https://doi.org/10.3390/info16050365

43. Jacobs, J., Romanosky, S., Suciu, O., Edwards, B., & Sarkar, A. (2023). Enhancing vulnerability prioritization: Data-driven exploit predictions with community-driven insights. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (pp. 194–206). IEEE. https://doi.org/10.1109/eurospw59978.2023.00027

44. Jamal, S., Wimmer, H., & Sarker, I. H. (2024). An improved transformer-based model for detecting phishing, spam, and ham emails: A large language model approach. *Security and Privacy, 7*(3), e402. https://doi.org/10.1002/spy2.402

45. Jáñez-Martino, F., Alaiz-Rodríguez, R., González-Castro, V., Fidalgo, E., & Alegre, E. (2022). A review of spam email detection: Analysis of spammer strategies and the dataset shift

problem. *Artificial Intelligence Review, 56*, 1145–1173. https://doi.org/10.1007/s10462-022-10195-4

46. Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., … Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and Trends in Machine Learning, 14*(1–2), 1–210. https://doi.org/10.1561/2200000083

47. Kang, H., Liu, G., Wang, Q., Meng, L., & Liu, J. (2023). Theory and application of zero trust security: A brief survey. *Entropy*, *25*(12), 1595. https://doi.org/10.3390/e25121595

48. Kapoor, S., & Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, 4(9), 100804. https://doi.org/10.1016/j.patter.2023.100804

49. Karataş, G., Demir, Ö., & Şahingöz, Ö. K. (2020). Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset. *IEEE Access*, 8, 32150–32162. https://doi.org/10.1109/access.2020.2973219

50. Karim, A., Azam, S., Shanmugam, B., Kannoorpatti, K., & Alazab, M. (2019). A Comprehensive Survey for Intelligent Spam Email Detection. *IEEE Access*, 7, 168261–168295. https://doi.org/10.1109/access.2019.2954791

51. Karim, A., Shahroz, M., Mustofa, K., Belhaouari, S. B., & Joga, S. R. K. (2023). Phishing Detection System Through Hybrid Machine Learning Based on URL. *IEEE Access*, 11, 36805–36822. https://doi.org/10.1109/access.2023.3252366

52. Khan, N., Lee, K., & Houng, A. Y. (2022). Anomaly detection and enterprise security using user and entity behavior analytics (UEBA). In *2022 3rd International Conference on Next Generation Computing Applications (NextComp)*. IEEE. https://doi.org/10.1109/iconics56716.2022.10100596

53. Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1). https://doi.org/10.1186/s42400-019-0038-7

54. Kolchin, R. (2025). Development and implementation of the Mail Security Guardian (MSG) system for multi-layer proactive email protection against spam, phishing and malware. *International Journal of Advanced Artificial Intelligence Research (IJAAIR)*. [Author's published study; full volume/issue/pages/DOI to be confirmed at submission.]

55. Kumar, P., Antony, K., Banga, D., & Sohal, A. S. (2024). PhishNet: A phishing website detection tool using XGBoost. *arXiv preprint* arXiv:2407.04732. https://doi.org/10.48550/arXiv.2407.04732

56. Liu, H., & Lang, B. (2019). Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Applied Sciences*, 9(20), 4396. https://doi.org/10.3390/app9204396

57. Liu, S., Feng, P., Wang, S., Sun, K., & Cao, J. (2022). Enhancing malware analysis sandboxes with emulated user behavior. *Computers & Security, 115*, 102613. https://doi.org/10.1016/j.cose.2022.102613

58. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, 1. https://doi.org/10.1109/tkde.2018.2876857

59. Lundberg, S., & Lee, S. (2017). A Unified Approach to Interpreting Model Predictions. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1705.07874

60. Macas, M., Wu, C., & Fuertes, W. (2023). Adversarial examples: A survey of attacks and defenses in deep learning-enabled cybersecurity. *Expert Systems with Applications, 238*, 122223. https://doi.org/10.1016/j.eswa.2023.122223

61. Mahdavifar, S., & Ghorbani, A. A. (2019). Application of deep learning to cybersecurity: A survey. *Neurocomputing*, 347, 149–176. https://doi.org/10.1016/j.neucom.2019.02.056

62. McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2016). Communication-Efficient Learning of Deep Networks from DecentralizedData. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1602.05629

63. Mienye, I. D., & Sun, Y. (2022). A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects. *IEEE Access*, 10, 99129–99149. https://doi.org/10.1109/access.2022.3207287

64. Otieno, D. O., Namin, A. S., & Jones, K. S. (2023). The application of the BERT transformer model for phishing email classification. In *Proceedings of the 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 1303–1308). IEEE. https://doi.org/10.1109/COMPSAC57700.2023.00198

65. Qiu, S., Liu, Q., Zhou, S., & Wu, C. (2019). Review of Artificial Intelligence Adversarial Attack and Defense Technologies. *Applied Sciences*, 9(5), 909. https://doi.org/10.3390/app9050909

66. Ragheb, M., Elmedany, W., & Sharif, M. (2023). The effectiveness of DKIM and SPF in strengthening email security. In *Proceedings of the 10th International Conference on Future Internet of Things and Cloud (FiCloud 2023)* (pp. 422–426). IEEE. https://doi.org/10.1109/FiCloud58648.2023.00068

67. Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero Trust Architecture. National Institute of Standards and Technology Special Publication 800-207. https://doi.org/10.6028/nist.sp.800-207

68. Roy, S. S., Thota, P., Naragam, K. V., & Nilizadeh, S. (2024). From chatbots to phishbots? Phishing scam generation in commercial large language models. In *Proceedings of the 2024 IEEE*

Symposium on Security and Privacy (SP) (pp. 36–54). IEEE. https://doi.org/10.1109/SP54263.2024.00182

69. Salloum, S. A., Gaber, T., Vadera, S., & Shaalan, K. (2022). A systematic literature review on phishing email detection using natural language processing techniques. *IEEE Access, 10*, 65703–65727. https://doi.org/10.1109/ACCESS.2022.3183083

70. Sarker, I. H., Kayes, A. S. M., Badsha, S., Alqahtani, H., Watters, P., & Ng, A. (2020). Cybersecurity data science: an overview from machine learning perspective. *Journal Of Big Data*, 7(1). https://doi.org/10.1186/s40537-020-00318-5

71. Saxena, N., Hayes, E., Bertino, E., Ojo, P., Choo, K. R., & Burnap, P. (2020). Impact and Key Challenges of Insider Threats on Organizations and Critical Businesses. *Electronics*, 9(9), 1460. https://doi.org/10.3390/electronics9091460

72. Shaukat, K., Luo, S., Varadharajan, V., Hameed, I. A., & Xu, M. (2020). A Survey on Machine Learning Techniques for Cyber Security in the Last Decade. *IEEE Access*, 8, 222310–222354. https://doi.org/10.1109/access.2020.3041951

73. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal Of Big Data*, 6(1). https://doi.org/10.1186/s40537-019-0197-0

74. Singh, J., & Singh, J. (2020). A survey on machine learning-based malware detection in executable files. *Journal of Systems Architecture*, 112, 101861. https://doi.org/10.1016/j.sysarc.2020.101861

75. Sun, N., Ding, M., Jiang, J., Xu, W., Mo, X., Tai, Y., & Zhang, J. (2023). Cyber Threat Intelligence Mining for Proactive Cybersecurity Defense: A Survey and New Perspectives. *IEEE Communications Surveys & Tutorials*, 25(3), 1748–1774. https://doi.org/10.1109/comst.2023.3273282

76. Syed, N. F., Shah, S. W., Shaghaghi, A., Anwar, A., Baig, Z. A., & Doss, R. R. M. (2022). Zero trust architecture (ZTA): A comprehensive survey. *IEEE Access, 10*, 57143–57179. https://doi.org/10.1109/ACCESS.2022.3174679

77. Vielberth, M., Böhm, F., Fichtinger, I., & Pernul, G. (2020). Security operations center: A systematic study and open challenges. *IEEE Access, 8*, 227756–227779. https://doi.org/10.1109/ACCESS.2020.3045514

78. Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, 7, 41525–41550. https://doi.org/10.1109/access.2019.2895334

79. Wen, J., Zhang, Z., Lan, Y., Cui, Z., Cai, J., & Zhang, W. (2022). A survey on federated learning: Challenges and applications. *International Journal of Machine Learning and Cybernetics, 14*, 513–535. https://doi.org/10.1007/s13042-022-01647-y

80. Yang, P., Zhao, G., & Zeng, P. (2019). Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning. *IEEE Access*, 7, 15196–15209. https://doi.org/10.1109/access.2019.2892066

81. Zhang, Z., Hamadi, H. A., Damiani, E., Yeun, C. Y., & Taher, F. (2022). Explainable Artificial Intelligence Applications in Cyber Security: State-of-the-Art in Research. *IEEE Access*, 10, 93104–93139. https://doi.org/10.1109/access.2022.3204051

82. Zieni, R., Massari, L., & Calzarossa, M. C. (2023). Phishing or Not Phishing? A Survey on the Detection of Phishing Websites. *IEEE Access*, 11, 18499–18519. https://doi.org/10.1109/access.2023.3247135