JDEBM
Journal of Digital Engineering and Business Managemet

Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

HYBRID AI ARCHITECTURE FOR PREDICTIVE CYBER RISK INTELLIGENCE

[#1]**KOLAGANI SOWJANYA**, *Assistant professor, Department of CSE(AI&ML)*
[#2]**K.SHILPIKA**, *Assistant Professor, Dept of CSE,*
[#3]**Dr.K SRIDHAR,** Associate professor, Department of CSE
[#1,2]Vaageswari College of Engineering, Karimnagar, TG.
[#3]Malla Reddy(MR) Deemed to be University, Medchal,Telangana
[#1]Kamunisowjanyasanthosh@gmail.com
[#2]shilpika.kanaparthi@gmail.com
[#3]sridhark529reddy@gmail.com

Abstract: Organizations must decide which of thousands of known vulnerabilities and alerts to act on first, yet common practice ranks them by static severity scores that do not estimate how likely an attack is or what it would cost. Data-driven methods have improved individual tasks: machine learning predicts which vulnerabilities will be exploited, deep anomaly detectors flag suspicious network and host behavior, and attack graphs model how an adversary can move between hosts. These capabilities are rarely combined, so their outputs remain separate scores that do not add up to a forecast of business risk. This paper proposes HyCRI, a hybrid AI architecture for predictive cyber risk intelligence that couples symbolic and neural components. A security knowledge graph and rule-based attack-graph generation supply structure; a calibrated exploit-likelihood model, an autoencoder-ensemble anomaly detector, and a relational graph convolutional network supply learned probabilities; and a Bayesian attack graph fuses both into the probability that each critical asset is compromised within a planning horizon, expressed as expected loss. A decision layer ranks mitigations by risk reduction per unit cost and explains each recommendation through attack paths and feature attributions. We formalize each layer, give the risk-update algorithm and a reference implementation stack, and define a leakage-aware evaluation protocol. Empirical results are not yet reported; the protocol and results template specify how the architecture should be validated.

Keywords**:** *cyber risk; threat intelligence; exploit prediction; Bayesian attack graph; anomaly detection; neuro-symbolic AI*

1. INTRODUCTION

Security teams face more findings than they can address. Vulnerability scanners report many open issues per host, intrusion detection systems raise alerts continuously, and threat-intelligence feeds publish new indicators daily. The practical question is not whether a weakness exists but which weaknesses, if left unaddressed, are most likely to lead to a costly incident in the near future. Husák et al. [1] describe this shift from detecting attacks to projecting, predicting, and forecasting them as a central direction for cyber defense.

Current practice answers the question poorly. Vulnerabilities are often prioritized by the Common Vulnerability Scoring System (CVSS) base score [3], which measures technical

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

severity but not the likelihood of exploitation. Risk-quantification frameworks such as FAIR [2] express risk as the product of loss event frequency and loss magnitude, but the frequency estimates are usually elicited from experts rather than learned from data. Four gaps separate today's tools from predictive risk intelligence.

1. **Severity is not likelihood.** Only a small share of published vulnerabilities is ever exploited in the wild [4], [5], so severity-based ranking spends effort on many issues that attackers ignore.
2. **Signals are isolated.** Exploit predictions, anomaly alerts, and threat reports describe the same adversary but are scored separately, with no common probability scale.
3. **Context is missing.** Whether a vulnerability matters depends on network reachability and on what it allows an attacker to reach next. Attack graphs capture this structure [12], [13], but their probabilities are usually fixed by hand.
4. **Outputs are not decisions.** Analysts need to know which action reduces the most risk for the effort, and why.

Purely data-driven models handle uncertain, high-volume signals well but ignore known structure, such as network topology and the preconditions of an exploit. Purely symbolic models encode that structure but cannot learn from data. A hybrid design can use each where it is strong.

This paper proposes **HyCRI** (Hybrid Cyber Risk Intelligence), a layered architecture in which symbolic reasoning provides the structure of possible attacks and neural models provide the probabilities that populate it. The research objectives are to (i) define interfaces through which learned probabilities enter a probabilistic attack model; (ii) express the result as forecast business risk over a horizon; and (iii) specify an evaluation protocol that avoids the temporal leakage and unrealistic data that undermine much security machine learning [8], [16].

The main contributions are:

- A five-layer hybrid architecture combining a security knowledge graph, rule-based attack-graph generation, a calibrated exploit-likelihood model, an anomaly detector, a relational graph network, and Bayesian risk fusion.
- A formal specification of the probability calibration, graph propagation, evidence update, risk, and mitigation-selection equations, with an online risk-update algorithm.
- A reference implementation stack built from established open-source tools.
- A reproducible, leakage-aware evaluation protocol with baselines, ablations, metrics, and a results template.

Section 2 reviews related work. Section 3 presents the architecture, and Section 4 the implementation and workflow. Section 5 defines the evaluation protocol and discusses limitations. Section 6 concludes.

2. LITERATURE REVIEW

Relevant prior work falls into four strands: exploit and incident prediction, anomaly-based detection, attack-graph risk models, and graph learning. Table 1 compares representative approaches.

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

2.1 Exploit and Incident Prediction

Sabottke et al. (2015) [5] showed that features drawn from Twitter discussion of vulnerabilities, combined with vulnerability metadata, can predict which vulnerabilities will be exploited in the real world. They also showed that such social-media features can be manipulated by an adversary, a weakness for any system relying on open signals. Jacobs et al. (2021) [4] introduced the Exploit Prediction Scoring System (EPSS), a data-driven model that estimates the probability that a vulnerability will be exploited within the following 30 days, using vulnerability attributes and observed exploitation activity. EPSS gives a probability rather than a severity, but it is defined per vulnerability and does not consider the network in which the vulnerability sits. Liu et al. (2015) [6] forecast security incidents at the organization level from externally observable indicators, such as network misconfiguration and malicious activity originating from the organization's address space. Their approach predicts whether an organization will suffer a breach, but not which assets or paths are responsible.

2.2 Anomaly-Based Detection

Buczak and Guven (2016) [7] surveyed machine-learning and data-mining methods for intrusion detection and noted the difficulty of obtaining representative labeled data. Sommer and Paxson (2010) [8] argued that anomaly detection is harder in security than in other domains because of high error costs, diverse benign traffic, and the semantic gap between an anomaly and an attack. Mirsky et al. (2018) [9] proposed Kitsune, an online ensemble of small autoencoders that flags network traffic with high reconstruction error and runs without labels. Du et al. (2017) [10] proposed DeepLog, which models system logs as sequences with an LSTM and flags deviations from learned execution patterns. Liu et al. (2008) [11] introduced Isolation Forest, a tree-based anomaly detector that is a common unsupervised baseline. These detectors produce scores but not calibrated probabilities of compromise, and they do not reason about which assets an anomaly puts at risk.

2.3 Attack Graphs and Risk Models

Ou et al. (2005) [12] proposed MulVAL, which encodes vulnerability preconditions, network configuration, and attacker capabilities as Datalog rules and derives logical attack graphs automatically. Poolsappasit et al. (2012) [13] modeled attack graphs as Bayesian networks, computed the probability that each security condition is reached, and updated those probabilities when attack evidence was observed. Their exploit probabilities, however, were derived from CVSS metrics [3] rather than learned. MITRE ATT&CK [14] organizes adversary behavior into tactics and techniques and is widely used to label both threat reports and detections. Freund and Jones (2014) [2] formalized risk in the FAIR model as loss event frequency multiplied by loss magnitude, providing a monetary scale on which security and business decisions can be compared.

2.4 Graph Learning

Schlichtkrull et al. (2018) [15] proposed relational graph convolutional networks (R-GCN), which learn node representations in multi-relational graphs and score missing links. Security knowledge graphs are multi-relational, with relations such as *runs*, *has-vulnerability*, and *can-reach*, so R-GCN can estimate edges that scanners and configuration data miss.

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

Table 1. Capabilities of representative approaches (✓ provided; Partial; — not addressed)

Approach	Learned likelihood	Network context	Runtime evidence	Business risk	Explanation
CVSS [3]	—	—	—	—	Partial
EPSS [4]; Sabottke et al. [5]	✓	—	—	—	Partial
Liu et al. [6]	✓	—	Partial	Partial	—
Kitsune [9]; DeepLog [10]	✓	—	✓	—	Partial
MulVAL [12]	—	✓	—	—	✓
Bayesian attack graph [13]	—	✓	✓	Partial	✓
FAIR [2]	—	—	—	✓	✓
HyCRI (proposed)	✓	✓	✓	✓	✓

2.5 Research Gap

Learned predictors provide likelihoods without network context; attack-graph models provide context with hand-set likelihoods; risk frameworks provide a business scale without data-driven frequencies. No approach in Table 1 connects all three. HyCRI differs by feeding calibrated, learned probabilities into a Bayesian attack graph derived from symbolic rules, updating it with runtime evidence, and expressing the result as expected loss over a stated horizon.

3. ARCHITECTURE / SYSTEM DESIGN

HyCRI has five layers, shown in Fig. 1. Layers L2 (symbolic) and L3 (neural) run in parallel and exchange information; L4 fuses their outputs into risk; L5 turns risk into ranked actions.

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Management
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

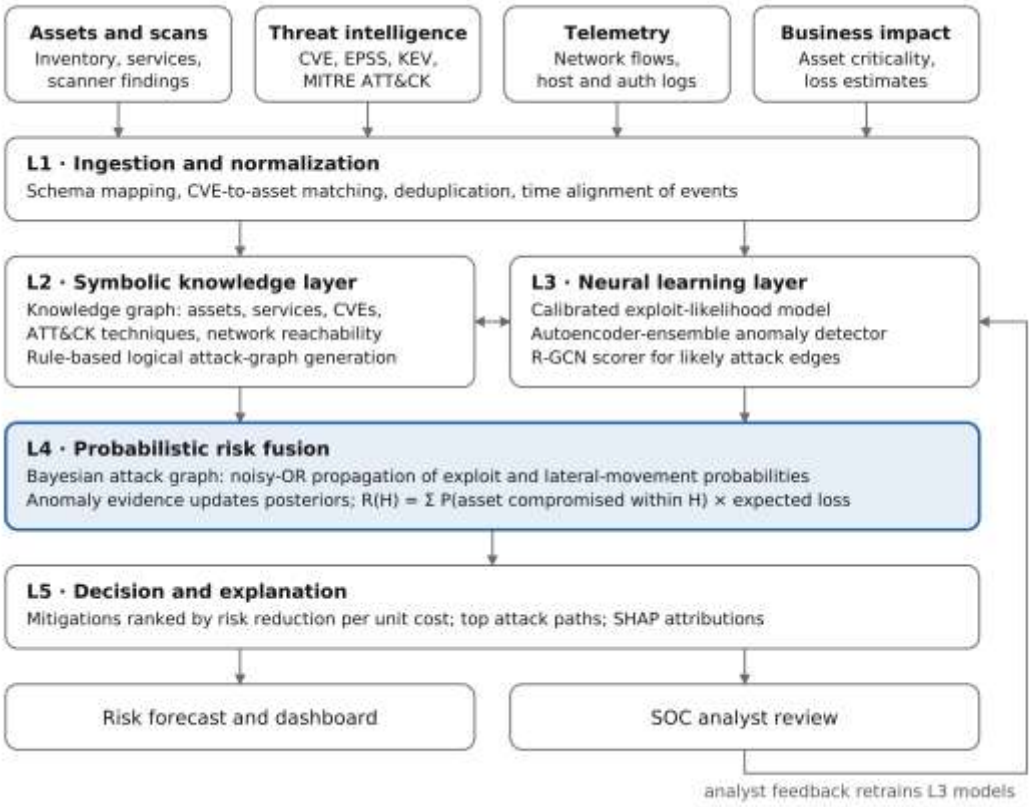


Fig. 1. HyCRI architecture. The symbolic layer supplies the structure of possible attacks and the neural layer supplies probabilities; both feed the Bayesian risk-fusion layer. Analyst decisions are fed back as labels for the neural models.

3.1 Problem Formulation

The environment is described by a set of assets $\mathcal{V}_A$ (hosts, services, accounts), a set of known vulnerabilities $\mathcal{V}_V$ present on those assets, and a subset of critical assets $\mathcal{C} \subseteq \mathcal{V}_A$ whose compromise causes a business loss. Each critical asset c has an expected loss magnitude L_c, estimated as in FAIR [2]. Given the data available at time t, the system must estimate the probability $P_c(H)$ that c is compromised within a horizon of H days, the total expected loss $R(H)$, and a set of mitigations that most reduces $R(H)$ within a budget B.

3.2 Layer 1: Ingestion and Normalization

L1 collects four kinds of input: asset inventory and scanner findings; threat intelligence, including CVE records, EPSS scores [4], the CISA Known Exploited Vulnerabilities (KEV) catalog [20], and MITRE ATT&CK mappings [14]; network flows and host and authentication logs; and business-impact data. It maps each record to a common schema, matches CVEs to assets by product and version, removes duplicates, and timestamps every fact so that later layers can reconstruct exactly what was known at any past time.

3.3 Layer 2: Symbolic Knowledge Layer

L2 stores the normalized facts in a security knowledge graph $\mathcal{G} = (\mathcal{N}, \mathcal{R}, \mathcal{E})$ with typed nodes $\mathcal{N}$, relation types $\mathcal{R}$, and edges $\mathcal{E}$. Node types include asset, service, vulnerability, account, and ATT&CK technique; relations include *runs*, *has-vulnerability*, *can-reach*, and *enables-technique*. Rule-based reasoning in the style of MulVAL [12] then derives a logical attack graph: a directed acyclic graph of privilege and execution conditions, connected by exploit and lateral-movement steps whose preconditions are satisfied by the facts in $\mathcal{G}$. L2 contributes

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

structure and constraints only: it states which attack steps are possible, not how likely they are.

3.4 Layer 3: Neural Learning Layer

L3 supplies probabilities for the steps produced by L2.

Exploit likelihood. For each vulnerability v, a gradient-boosted classifier receives features $\phi(v)$: CVSS metrics, the CVE description embedded with a pretrained text encoder, exploit-code availability, vendor and product, and the current EPSS score [4]. The output logit $f_\theta(\phi(v))$ is calibrated with temperature scaling [19], giving the 30-day exploitation probability in (1). Under a constant-hazard assumption, it is converted to the planning horizon H as in (2).

$$p_v = \sigma\left(\frac{f_\theta(\phi(v))}{T}\right)$$

$$p_v(H) = 1 - (1 - p_v)^{H/30}$$

Runtime anomaly evidence. For each asset a, an ensemble of E autoencoders, in the manner of Kitsune [9], reconstructs the feature vector $\mathbf{x}_a(t)$ built from that asset's flows and log events in a time window. The anomaly score is the mean reconstruction error, standardized against the asset's own history to give $z_a(t)$. A logistic calibration fitted on labeled incidents converts it to the probability $q_a(t)$ that the anomaly reflects attack activity:

$$s_a(t) = \frac{1}{E}\sum_{e=1}^{E} \| \mathbf{x}_a(t) - \mathrm{dec}_e\left(\mathrm{enc}_e(\mathbf{x}_a(t))\right) \|_2^2, \qquad q_a(t) = \sigma(\beta_0 + \beta_1 z_a(t))$$

Here enc_e and dec_e are the encoder and decoder of ensemble member e, and β_0, β_1 are calibration parameters.

Missing attack edges. Scanners and configuration data are incomplete. An R-GCN [15] learns node representations on $\mathcal{G}$ by the propagation rule in (4), where $\mathcal{N}_i^r$ is the set of neighbors of node i under relation r and $c_{i,r}$ is a normalization constant. A bilinear decoder scores candidate *can-reach* and *has-vulnerability* edges; edges above a threshold are added to the attack graph with their scores as prior probabilities and are marked as inferred.

$$\mathbf{h}_i^{(l+1)} = \mathrm{ReLU}\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}} W_r^{(l)}\mathbf{h}_j^{(l)} + W_0^{(l)}\mathbf{h}_i^{(l)}\right)$$

3.5 Layer 4: Probabilistic Risk Fusion

L4 converts the attack graph into a Bayesian network, following Poolsappasit et al. [13], but uses the learned probabilities from L3 in place of fixed CVSS-derived values. Each attack step $j \to n$ has a success probability π_{jn}: $p_v(H)$ for an exploit step, the R-GCN score for an inferred edge, and an expert prior for steps such as credential reuse. A condition node n with parents $\mathrm{pa}(n)$ is reached under disjunctive (OR) or conjunctive (AND) semantics:

$$P_{\mathrm{OR}}(n) = 1 - \prod_{j\in\mathrm{pa}(n)}\left(1 - \pi_{jn}\,P(j)\right), \qquad P_{\mathrm{AND}}(n) = \prod_{j\in\mathrm{pa}(n)} \pi_{jn}\,P(j)$$

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

Entry nodes, such as Internet-facing services, receive a prior attempt probability. Runtime evidence enters as soft evidence: the anomaly probability $q_a(t)$ defines a likelihood ratio $\Lambda_a = q_a(t)/(1-q_a(t))$ relative to the prior odds, and the posterior odds of compromise of the condition node for asset a are updated as in (6). Posteriors then propagate to downstream nsodes by belief propagation, so an anomaly on a jump host raises the risk of the database it can reach.

$$\frac{P(n_a \mid e)}{1-P(n_a \mid e)} = \Lambda_a \cdot \frac{P(n_a)}{1-P(n_a)}$$

Total forecast risk is the expected loss over critical assets, as in (7). Because L_c and the probabilities are uncertain, L4 also draws Monte Carlo samples of both to report a loss distribution and its 95th percentile alongside the mean.

$$R(H) = \sum_{c \in C} P_c\,(H \mid e)\,L_c$$

3.6 Layer 5: Decision and Explanation

A mitigation m, such as applying a patch, closing a firewall rule, or adding detection for an ATT&CK technique, removes or weakens edges in the attack graph and has an estimated cost $\kappa(m)$ in hours or money. Its benefit is the reduction in risk it produces, re-evaluated in the graph, and mitigations are selected greedily by benefit per unit cost until the budget B is exhausted:

$$\Delta R(m) = R(H) - R(H \mid m), \qquad m^* = \operatorname*{argmax}_{m \in \mathcal{M}} \frac{\Delta R(m)}{\kappa(m)}$$

Each recommendation is explained in two ways: by the most probable attack paths it breaks, read directly from the Bayesian network, and by SHAP attributions [18] for the exploit-likelihood predictions on those paths. The analyst's decision to accept, reject, or reclassify an alert or recommendation is logged and becomes a training label for L3.

3.7 Reliability, Scalability, and Security

Attack-graph size can grow quickly with the number of hosts. HyCRI therefore groups hosts with identical configuration and reachability into equivalence classes, prunes paths below a probability threshold, and recomputes only the subgraph affected by an update. The system is itself a high-value target: its stores are access-controlled and encrypted, and its outputs reveal attack paths, so they are restricted to authorized staff. Open signals can be manipulated [5]; features from untrusted sources are therefore capped in influence and monitored for sudden shifts.

4. IMPLEMENTATION TECHNIQUES AND WORKFLOW

This section specifies a reference implementation from established open-source tools. HyCRI is a proposed architecture: no deployment or measured performance is claimed.

4.1 Software Stack

Layer	Function	Library or tool
L1 Ingestion	Collection, schema mapping, CVE-to-asset matching	Python 3.10+, pandas, Apache Kafka

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Management
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

L2 Symbolic	Knowledge graph storage; rule-based attack-graph generation	Neo4j; MulVAL-style Datalog rules
L3 Neural	Exploit model; autoencoder ensemble; R-GCN	XGBoost, PyTorch, PyTorch Geometric
L4 Fusion	Bayesian network inference; Monte Carlo loss sampling	pgmpy, NumPy
L5 Decision	Mitigation ranking; SHAP explanations; dashboard	SHAP, FastAPI, a web dashboard

Table 2. Reference implementation stack

Hardware requirements depend on network size and telemetry volume and will be reported with the experiments. Autoencoder and R-GCN training benefit from a GPU; Bayesian inference and ranking run on CPU.

4.2 Initial Configuration

Default settings, to be tuned on validation data, are: planning horizon $H = 30$ days, matching the EPSS prediction window [4]; an ensemble of $E = 5$ autoencoders per asset group; two R-GCN layers with 64-dimensional representations; a path-pruning threshold of 10^{-4}; and 10,000 Monte Carlo samples for the loss distribution. These values are design defaults, not results.

4.3 Offline Workflow

1. **Build the graph.** Load inventory, scanner, configuration, and threat-intelligence data into the knowledge graph, keeping a timestamp on every fact.
2. **Generate the attack graph.** Apply the derivation rules to produce the logical attack graph for the current network.
3. **Train the exploit model.** Label each vulnerability by whether exploitation was observed within 30 days of a reference date, for example by inclusion in KEV [20]. Train on earlier periods only and fit the temperature T on a later validation period.
4. **Train the anomaly detectors.** Fit the autoencoders on telemetry from periods without known incidents; fit β_0, β_1 on labeled incidents and analyst-confirmed alerts.
5. **Train the R-GCN.** Hide a random share of known edges, train the model to recover them, and select the edge threshold on validation data.
6. **Calibrate risk inputs.** Elicit L_c ranges for critical assets with business owners, and set entry-node priors.

4.4 Online Risk-Update Algorithm

Algorithm 1 gives the online loop. Structural changes trigger partial recomputation; telemetry updates only evidence.

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Management
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

Algorithm 1. HyCRI online risk update

```
Input: knowledge graph G, attack graph AG, exploit model f, temperature T,
    autoencoder ensembles {enc_e, dec_e}, R-GCN g, losses {L_c}, budget B

on change to inventory, scans, or threat intelligence:
  update G;  AG_sub <- rederive rules for affected nodes        # L2
  for each vulnerability v in AG_sub:
    p_v(H) <- 1 - (1 - sigmoid(f(phi(v)) / T))^(H/30)          # Eq. (1)-(2)
  add edges with g-score > threshold, marked "inferred"         # Eq. (4)
  rebuild Bayesian network for AG_sub with step probs pi        # Eq. (5)

every telemetry window t:
  for each asset a:
    s_a <- mean_e ||x_a - dec_e(enc_e(x_a))||^2;  z_a <- standardize(s_a)
    q_a <- sigmoid(beta0 + beta1 * z_a)                         # Eq. (3)
    if q_a > q_min: add soft evidence Lambda_a = q_a / (1 - q_a) # Eq. (6)
  run belief propagation;  R(H) <- sum_c P_c(H | e) * L_c       # Eq. (7)

on schedule or when R(H) rises above alert level:
  plan <- greedy select m maximizing dR(m) / cost(m) until B spent  # Eq. (8)
  publish R(H), loss percentiles, plan, top attack paths, SHAP values
  log analyst decisions as labels for L3
```

4.5 Module Integration

Layers exchange typed records through a message bus: L1 emits normalized, timestamped facts; L2 emits attack-graph deltas; L3 emits step probabilities and evidence with model versions; L4 emits posteriors and risk; L5 emits plans and explanations. Because every probability carries its source and version, an analyst can trace any risk figure back to the facts and models that produced it, and any neural component can be replaced, for example by substituting a DeepLog-style log model [10] for the autoencoders, without changing the other layers.

5. RESULTS AND DISCUSSION

No experimental results are reported in this version. This section specifies the evaluation protocol, provides an empty results template (Table 4), and states the hypotheses to be tested. All cells marked "—" must be filled with measured values; no number in this section is a finding.

5.1 Experimental Setup

Datasets. (a) *Exploit prediction:* historical CVE records with EPSS scores [4] and KEV inclusion dates [20] as exploitation labels. (b) *Anomaly detection:* the CICIDS2017 dataset [17], which provides labeled benign and attack traffic. Its lab-generated traffic may not reflect a production network [8], so it is used for component testing only. (c) *End-to-end:* an isolated test network with a known topology, scanned vulnerabilities, and business-impact

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

values, in which authorized red-team exercises emulate ATT&CK techniques [14] at recorded times.

Leakage control. Following Arp et al. [16], all splits are temporal: models are trained only on data available before each evaluation date, and features such as EPSS scores are taken as they were published on that date, not as later revised. Every configuration is run with five random seeds, and differences from the strongest baseline are tested with paired bootstrap confidence intervals.

Baselines. B1: ranking by CVSS base score [3]. B2: ranking by EPSS [4]. B3: Isolation Forest [11] anomaly alerts without risk fusion. B4: a Bayesian attack graph with CVSS-derived probabilities [13]. B5: the HyCRI neural models without the attack graph, with risk taken as the sum over vulnerabilities on critical assets.

Ablations. A1: no R-GCN inferred edges. A2: no runtime evidence. A3: uncalibrated probabilities ($T = 1$, raw scores). A4: random instead of greedy mitigation selection.

5.2 Evaluation Metrics

Probability quality is measured by the Brier score in (9), where $\hat{p}_i$ is the predicted probability and $o_i \in \{0,1\}$ the observed outcome for N cases; lower is better.

$$\mathrm{BS} = \frac{1}{N}\sum_{i=1}^{N}(\hat{p}_i - o_i)^2$$

Table 3. Evaluation metrics

Property	Metric	Definition
Exploit prediction	Area under the precision–recall curve; Brier score, Eq. (9)	Ranking quality under class imbalance; probability accuracy
Prioritization	Effort-efficiency at k	Share of later-exploited vulnerabilities covered by the top k recommendations
Detection	True-positive rate at a fixed false-alert rate; alerts per analyst-day	Detection power at a workload analysts can absorb
Risk forecast	Calibration of $P_c(H)$; rank correlation with emulated outcomes	Whether forecast compromise probabilities match observed outcomes in the test network
Mitigation	Risk reduction per unit cost; paths broken	Benefit of the selected plan versus baselines at equal budget
Efficiency	Graph-update latency; inference time per window	Time from new data to updated risk

5.3 Results Template

Table 4. Results template. All values to be filled from experiments; mean ± s.d. over five seeds

Method	PR-AUC (exploit)	Brier	Effort-efficiency at k	TPR at fixed FPR	Risk reduction per cost	Update latency (s)

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Management
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

B1 CVSS ranking	—	—	—	n/a	—	—
B2 EPSS ranking	—	—	—	n/a	—	—
B3 Isolation Forest alerts	n/a	n/a	n/a	—	n/a	—
B4 Static Bayesian attack graph	—	—	—	—	—	—
B5 Neural models only	—	—	—	—	—	—
HyCRI (full)	—	—	—	—	—	—
A1–A4 ablations	—	—	—	—	—	—

Cells marked n/a are metrics that the method does not produce.

5.4 Hypotheses and Expected Trade-offs

The experiments test four hypotheses; none has yet been confirmed.

- **H1 (learned likelihood).** Calibrated exploit probabilities give better effort-efficiency than CVSS ranking (B1) and at least match EPSS (B2).
- **H2 (context).** Adding attack-graph context improves prioritization over neural models alone (B5), because a likely exploit on an unreachable host carries little risk.
- **H3 (evidence fusion).** Runtime evidence raises forecast risk for assets downstream of emulated intrusions earlier than static analysis (B4, A2).
- **H4 (inferred edges).** R-GCN edges recover attack paths that scanner data alone miss (A1), at an acceptable false-edge rate.

The main expected costs are the effort of building and maintaining the knowledge graph, the computational cost of attack-graph generation on large networks, and dependence on loss estimates elicited from business owners. If B2 matches HyCRI on prioritization, the architecture's value would rest on network context, evidence fusion, and explanation, and the evaluation should report that outcome plainly.

5.5 Limitations and Threats to Validity

- **Label scarcity and bias.** Exploitation and incident labels are incomplete; an exploit that was never detected appears as a negative example.
- **Dataset realism.** Public intrusion datasets differ from production traffic [8], [16]; component results may not transfer.
- **Independence assumptions.** Noisy-OR propagation and the constant-hazard conversion in (2) assume independence that real attackers violate.
- **Adversarial manipulation.** Attackers can poison open intelligence sources or shape traffic to evade detection [5].
- **Loss estimates.** Expected losses L_c are uncertain and subjective; results require sensitivity analysis over plausible ranges.

6. CONCLUSION AND FUTURE WORK

This paper addressed the gap between isolated security scores and forecasts of business risk. It proposed HyCRI, a hybrid architecture in which a security knowledge graph and rule-based attack-graph generation define which attacks are possible; a calibrated exploit-likelihood model, an autoencoder-ensemble anomaly detector, and an R-GCN estimate how likely each step is; a Bayesian attack graph fuses these into the probability of compromise of each critical

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

asset and the expected loss over a horizon; and a decision layer ranks mitigations by risk reduction per unit cost with path-based and feature-based explanations. The contributions are the layered formulation with explicit interfaces between symbolic structure and learned probabilities, the online risk-update algorithm, a reference implementation stack, and a leakage-aware evaluation protocol.

The architecture's benefits remain hypotheses until the protocol in Section 5 is executed. Its current limitations are dependence on incomplete labels, independence assumptions in propagation, sensitivity to loss estimates, and exposure to manipulated inputs.

Future work will (i) implement the reference stack and complete Table 4 on public data and an isolated test network; (ii) replace noisy-OR propagation with models that capture dependence between attack steps; (iii) map threat-intelligence text to ATT&CK techniques automatically to update technique priors; and (iv) study robustness against poisoning of open intelligence sources.

REFERENCES

[1] M. Husák, J. Komárková, E. Bou-Harb, and P. Čeleda, "Survey of attack projection, prediction, and forecasting in cyber security," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, 2019, doi: 10.1109/COMST.2018.2871866.

[2] J. Freund and J. Jones, *Measuring and Managing Information Risk: A FAIR Approach*. Oxford, U.K.: Butterworth-Heinemann, 2014, ISBN 978-0-12-420231-3.

[3] Forum of Incident Response and Security Teams (FIRST), *Common Vulnerability Scoring System Version 3.1: Specification Document*, 2019. [Online]. Available: https://www.first.org/cvss/

[4] J. Jacobs, S. Romanosky, B. Edwards, I. Adjerid, and M. Roytman, "Exploit Prediction Scoring System (EPSS)," *Digital Threats: Research and Practice*, vol. 2, no. 3, pp. 1–17, 2021, doi: 10.1145/3436242.

[5] C. Sabottke, O. Suciu, and T. Dumitraș, "Vulnerability disclosure in the age of social media: Exploiting Twitter for predicting real-world exploits," in *Proc. 24th USENIX Security Symposium*, pp. 1041–1056, 2015.

[6] Y. Liu, A. Sarabi, J. Zhang, P. Naghizadeh, M. Karir, M. Bailey, and M. Liu, "Cloudy with a chance of breach: Forecasting cyber security incidents," in *Proc. 24th USENIX Security Symposium*, pp. 1009–1024, 2015.

[7] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.

[8] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symposium on Security and Privacy*, pp. 305–316, 2010.

[9] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Network and Distributed System Security Symposium (NDSS)*, 2018, doi: 10.14722/ndss.2018.23204.

[10] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conference on*

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com

Volume-2, Issue-4, October–December 2026
ISSN: 3107-7897 | P.ID: JDEBM/01
DOI: https://doi.org/10.65713/jdebmv2i401

Computer and Communications Security (CCS), pp. 1285–1298, 2017, doi: 10.1145/3133956.3134015.

[11] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE International Conference on Data Mining (ICDM)*, pp. 413–422, 2008, doi: 10.1109/ICDM.2008.17.

[12] X. Ou, S. Govindavajhala, and A. W. Appel, "MulVAL: A logic-based network security analyzer," in *Proc. 14th USENIX Security Symposium*, 2005.

[13] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2012, doi: 10.1109/TDSC.2011.34.

[14] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK: Design and philosophy," The MITRE Corporation, Tech. Rep., revised Mar. 2020.

[15] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web: ESWC 2018*, Lecture Notes in Computer Science, vol. 10843, pp. 593–607, 2018, doi: 10.1007/978-3-319-93417-4_38.

[16] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *Proc. 31st USENIX Security Symposium*, pp. 3971–3988, 2022.

[17] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 108–116, 2018, doi: 10.5220/0006639801080116.

[18] K. K. Gajula, "AI-Augmented Web Development: A Competency Framework for the Next Generation Workforce," Journal of Science Engineering Technology and Management Science, 2025.

[19] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. 34th International Conference on Machine Learning (ICML)*, PMLR vol. 70, pp. 1321–1330, 2017.

[20] K. K. Gajula, "Advanced Stress Level Forecasting Using Quantile optimized Regression Forest Models", int. J. Eng. Res. Sci. Tech., vol. 21, no. 4(1), pp. 1–7, Dec. 2025, doi: 10.62643/ijerst.2025.v21.n4(1).pp1-7.

© 2026 Journal of Digital Engineering and Business Management | www.jdebm.com