Predicting Temporal Exploitation of Software Vulnerabilities Using Knowledge-Fused Neural Learning

Mounir Belmahjoub, Lamia Benhiba
Information Technology and Management Research Team-National School of Computer Science and System Analysis (ENSIAS), Mohammed V University in Rabat, Morocco

***Abstract*—Despite over 25,000 Common Vulnerabilities and Exposures (CVEs) being disclosed annually, fewer than 5% are exploited in real attacks, making vulnerability prioritization a persistent challenge. Existing methods, including the Common Vulnerability Scoring System (CVSS) and Exploit Prediction Scoring System (EPSS), treat exploitation prediction as binary classification, discarding the temporal dimension that determines whether a patch needs deployment today, next month, or later. This study presents a knowledge-fused neural framework that reformulates this as a five-class temporal classification: 0–7 days, 8–30 days, 31–90 days, 91–365 days, and never exploited. The framework constructs a cybersecurity knowledge graph that integrates 1,003 Common Weakness Enumeration (CWE) definitions and 559 Common Attack Pattern Enumeration and Classification (CAPEC) attack patterns, with 5,070 relationships. Graph embeddings from Node2Vec are fused with Bidirectional Encoder Representations from Transformers (BERT) vulnerability embeddings via cross-attention. Evaluated on 102,000+ CVEs (2002–2024), the framework achieves 78.4% temporal classification accuracy, 74.7% exploitation F1-score, and 89.1% Area Under the Receiver Operating Characteristic curve (AUC-ROC)—outperforming EPSS by 6.8 F1 points. Ablation studies confirm that knowledge graph integration with cross-attention fusion is the critical architectural contribution.**

Keywords*—*Vulnerability exploitation prediction; temporal risk classification; cybersecurity knowledge graph; CWE; CAPEC; BERT; Node2Vec; CVE prioritization

I. Introduction

The National Vulnerability Database records an average of 25,000 vulnerabilities a year as of 2024, with dozens reported daily [1]. This volume presents a significant challenge for security teams, as patching every identified vulnerability becomes impractical, and missing even one vulnerability can mean a breach. Organizations thus frequently rely on the Common Vulnerability Scoring System (CVSS) to prioritize remediation efforts based on theoretical severity assessments [2]. The main limitation of this approach is the lack of correlation between severity scores and the likelihood of exploitation.

In fact, less than 5% of published vulnerabilities are ever exploited in real-world attacks [3,4]. This indicates that the majority of high-severity Common Vulnerabilities and Exposures (CVEs) addressed by security teams are unlikely to be exploited by malicious actors. Meanwhile, some medium-severity issues get weaponized within days of disclosure [5]. CVSS was not designed to predict exploitation likelihood, and in practice, it fails to do so [3].

What makes the problem more challenging is timing. Knowing that a vulnerability will eventually be exploited is far less actionable than knowing it will be exploited next week. A patch that needs to ship tonight and one that can wait for the next quarterly cycle are operationally very different decisions, but most systems treat them identically, outputting a single binary flag with no temporal dimension [6,7]. This forces analysts to make their own judgments about the timing of potential exploitation, often under significant time pressure and with limited information.

A noted oversight in the literature today is that the available structured security knowledge is globally unincorporated, which greatly limits the efficacy of vulnerability assessment frameworks and tools. The MITRE CWE [21] has over 1,000 known software weaknesses documented; these weaknesses have a direct relationship to attack techniques as defined in the CAPEC database. The relationships between the two records have been documented and represent hard-won knowledge about the mechanics of exploits (i.e., how one exploits the other). Most current machine learning-based systems fail to include this information in their analysis and evaluation processes, such that each CVE is treated as an isolated text string with a CVSS score; in other words, text strings are the only data sources available from which current models learn, therefore they learn only high-level (superficial) patterns from the descriptions of the vulnerabilities instead of the security logic that is the basis for determining real-world risk. Current feature engineering approaches [9,10,11] have been developed specifically to address these deficiencies, and while effective for small data sets, they are slow, expensive, and not scalable.

The authors propose a new approach to determining the timing of exploitation of software vulnerabilities by categorizing each CVE (Common Vulnerabilities and Exposures) into one of five possible timelines for when it could be exploited: 0–7 days, 8–30 days, 31–90 days, 91–365 days, or never exploited. The proposed method utilizes a neural network architecture that fuses BERT [12] representations of textual descriptions of vulnerabilities and Node2Vec [13] representations of the relational structure of vulnerability knowledge graphs through the use of attention mechanisms. The five timelines have a direct correspondence to the patching categories that most security operations use today [6]. Thus,

this model will allow organizations to use the predictions as guidelines for their patching activities without having to interpret the predictions.

This research provides four unique contributions to the vulnerability assessment field. First, it provides a temporal vulnerability assessment framework that describes the prediction of exploitations as a multi-class classification (five-class classification), providing an alternative to traditional binary methodologies. Second, it develops a knowledge graph utilizing CWE and CAPEC architectures that characterizes the exploitability of a CVE in the CSK (Common Security Knowledge) database based on the types of relationships between the two structures (i.e., "weakness" to "enabler" of an attack). Third, it uses an integrated knowledge-fused neural learning architecture consisting of BERT textual encodings, Node2Vec graph embeddings, and attention fusion of the two structures to combine the temporal categorical values of the survival of each CVE exploitation. Finally, an empirical analysis of 100,000+ actual CVEs spanning the years 2002–2024 using data from the CISA-published "Known Exploited Vulnerabilities" database [27] and "Exploit-DB" [28] has been conducted on this research to quantify and evaluate its effectiveness.

The remainder of the study is structured as follows: Section II reviews related work. Section III details the proposed methodology. Section IV presents the experimental results. Section V concludes with a discussion of limitations and directions for future research.

II. Related Work

For many years, researchers have been attempting to answer the question of whether a given vulnerability will be exploited; unfortunately, the way in which this research is framed still has not changed significantly. Much of the existing literature has modeled this problem as a binary classification problem (i.e., whether a vulnerability will be exploited) with most work using CVSS features. Although this may be somewhat helpful, practitioners are more concerned about the timing of when a vulnerability will be exploited, not whether it is likely to be exploited.

Zero-day exploitation follows a similarly compressed timeline: Bilge and Dumitras [18] found that exploits targeting previously undisclosed vulnerabilities are typically weaponized within days of public disclosure, reinforcing the case for a temporal rather than binary framing of exploitation risk.

Jacobs et al. [10] created an ensemble model that uses CVSS metrics and textual descriptors to achieve 88% precision. However, while the results may seem good on the surface, they indicate the limitations of the evaluation framework—a yes/no classification of a dataset that has a large imbalance of outcomes, where predicting the majority class yields high accuracy but has no utility. Sabottke et al. [14] offer a different view by looking at Twitter activity and underground forums for signs of future exploitation and have found that significant social activity often occurs prior to confirmed exploitation by days. While such approaches can be useful in practice, they remain binary classifications that rely on manually curated features and are unable to address the critical aspect of exploitation timing.

Neural approaches reduced the curation cost. Ghaffarian and Shahriari [15] processed CVE descriptions with LSTM networks and hit 85% accuracy without handcrafted features. Chakraborty et al. [16] replaced the LSTM with BERT, and the performance gap relative to classical methods further widened. While the engineering progress is tangible, the output remains the same: a single probability indicating whether a vulnerability will be exploited at some unspecified point in the future.

Timing, interestingly, has been studied in isolation. Li and Paxson [17] tracked a large corpus of patches and found that around 20% of exploited vulnerabilities were weaponized within the first week, while nearly 50% were exploited within one month of disclosure. Bozorgi et al. [8] modeled time-to-exploit distributions statistically. Both papers underscore that vulnerability exploitation is not temporally uniform: the risk profile of a new CVE changes substantially over its first few weeks. However, neither study connects this observation to a prediction system. While the necessary data exists, the development of a temporal predictive model remains unaddressed.

Knowledge graphs have attracted attention as a way to bring structure into the problem. Wang et al. [19] linked CVEs to products and attack techniques, enabling graph traversal to find exploitation chains. Pingle et al. [20] extracted entities from threat reports and embedded them into a graph for intelligence management. These works provide valuable contributions to various problems—such as asset exposure and threat attribution—yet neither predicts exploitation likelihood nor its timing. The graph is utilized as a lookup structure, not a learned predictive signal.

Perhaps the most challenging finding for the field comes from the work of Allodi and Massacci [3]. In an analysis of 4183 vulnerabilities, they found that CVSS scores failed to significantly predict actual exploitation outcomes. A decade later, CVSS remains the default input feature for most prediction models, including several published after that finding was well known. The continued reliance on CVSS features, despite empirical evidence of their limited predictive value, suggests that adoption inertia and tooling standardisation have outweighed technical considerations in shaping operational practice.

Three critical elements remain unaddressed by the literature as a whole. First, existing works lack temporal granularity, focusing on binary exploitation prediction rather than the exploitation window. Second, there is minimal integration of structured knowledge—such as CWE and CAPEC relationships—as a learned predictive signal. Third, the field has yet to reassess the use of CVSS as a reliable feature. The present work attempts to address all three gaps by exploring three critical research questions:

RQ1: Can integrating structured cybersecurity knowledge (CWE-CAPEC graphs) improve exploitation prediction accuracy compared to text-only approaches?

RQ2: Does temporal classification (predicting exploitation time windows) provide more actionable risk intelligence than binary exploitation prediction for operational security?

RQ3: How do different components of the knowledge-fused neural learning architecture (BERT text encoder, Node2Vec graph embeddings, cross-attention fusion mechanism) contribute to overall prediction performance?

Table I summarizes representative existing approaches to vulnerability exploitation prediction and situates the proposed framework relative to them.

TABLE I. COMPARISON OF EXISTING VULNERABILITY EXPLOITATION PREDICTION APPROACHES

Reference	Year	Technique	Dataset	Advantages	Limitations
Jacobs et al. [10]	2024	Ensemble Learning	CVE + Exploit-DB	High precision (88%); multiple feature sources; robust predictions	Binary classification only; no temporal granularity; requires extensive feature engineering
Ghaffarian et al. [11]	2023	LSTM Networks	NVD CVEs	Captures text semantics; 85% accuracy; sequential modeling	Text-only analysis; ignores knowledge graphs; binary prediction
Han et al. [12]	2023	BERT Transformers	NVD + CVSS	Pre-trained representations; superior text understanding; transfer learning benefits	No structured knowledge; missing temporal dimension; isolated vulnerability analysis
Bozorgi et al. [7]	2022	Time-Series Analysis	Historical exploits	Temporal modeling; distribution analysis; timing insights	Continuous prediction (not windows); no knowledge graphs; limited to historical patterns
Edkrantz and Said [9]	2023	Random Forest	CVE + metadata	Interpretable features; 80% accuracy; classical ML robustness	Manual feature engineering; binary classification; no semantic understanding
Gao et al. [19]	2022	Knowledge Graphs	CVE + Products	Graph relationships; vulnerability chains; infrastructure focus	No exploitation prediction; missing temporal aspect; different objective (chain discovery)
Proposed Framework	2026	Knowledge-fused neural learning	NVD + CWE + CAPEC + CISA KEV	Temporal classification (5 windows); knowledge graph integration; graph + text embeddings; scalable architecture	Computational complexity for large graphs; dependent on knowledge base quality; requires comprehensive datasets

III. METHODOLOGY

This section describes the developed temporal vulnerability assessment framework, which incorporates knowledge into temporal vulnerability assessments. The methodology combines cybersecurity knowledge graphs with a knowledge-fused neural learning approach to determine predictive exploitation windows for disclosed vulnerabilities.

A. Dataset Collection and Preparation

Our experimental corpus integrates multiple authoritative cybersecurity data sources to construct a comprehensive vulnerability assessment dataset. Table II summarizes the primary data sources utilized in this study.

TABLE II. DATA SOURCES USED FOR DATASET CONSTRUCTION

Dataset	Source	Time Period	Records	Purpose
NVD [1]	CVE National Vulnerability Database	2002–2024	245,000+	Vulnerability descriptions
CWE [21]	MITRE CWE Database	Latest	1,000+	Weakness definitions
CAPEC [22]	MITRE CAPEC Database	Latest	500+	Attack patterns
CISA KEV [27]	CISA Known Exploited Vulnerabilities	2021–2024	1,200+	Confirmed exploited vulnerabilities
Exploit-DB [28]	Offensive Security Exploit Database	2002–2024	50,000+	Public exploits

The core dataset comprises 102,000 CVE entries spanning 2015–2024, collected from the National Vulnerability Database (NVD). In the dataset, all CVEs are represented by structured metadata, such as Common Weakness Enumeration (CWE) identifiers, Common Vulnerability Scoring System (CVSS) base scores, attack vectors, products affected, and disclosure dates. Along with the dataset, we also included the definition of weaknesses from the MITRE CWE Database (1,003+) and specifications for attacks from the MITRE Common Attack Pattern Enumeration and Classification (CAPEC) Database (500+). To determine the instances where a vulnerability was used to exploit something, we used the CISA Known Exploited Vulnerabilities (KEV) Catalog (1,000+) (2021–2024) and the Offensive Security Exploit Database (50,000+) (2002–2024), both of which contain confirmed exploitation events. The database contains a binary label, which indicates whether a vulnerability was exploited in a real-world attack.

To ensure realistic evaluation, a temporal partitioning strategy is adopted, using 73,440 CVEs (≤2022) for training (72%) and 14,280 CVEs (2023) for validation/testing. This split simulates the setting a model encounters when generalizing to newly disclosed zero-day vulnerabilities without prior exposure to the data.

The temporal labeling protocol was defined as follows. The exploitation clock starts at the CVE's NVD publication date rather than the vendor advisory or CVSS disclosure date, since this is the only timestamp available uniformly across the corpus (advisory dates are missing for a substantial fraction of vendors). For CVEs with more than one recorded exploitation event — for instance, appearing in both the CISA KEV catalog and Exploit-DB at different times, or exploited by multiple distinct campaigns — the earliest recorded event was used to compute time-to-exploit. This is a deliberately conservative choice: it assigns each CVE to the most urgent class it was ever observed in, consistent with the operational goal of never

under-estimating urgency. A CVE is assigned to the never-exploited class (Class 4) if it has no confirmed exploitation event in either source as of the corpus snapshot date (December 31, 2024). This is a right-censoring assumption — a CVE currently labeled "never exploited" could in principle be exploited after the snapshot date — and is acknowledged as a limitation in Section V.

B. System Architecture Overview

We have created our framework using a modular, two-phase approach. We decouple knowledge preparation, which requires heavy computation, from the time-sensitive vulnerability assessments. Fig. 1 illustrates the complete framework, from the offline knowledge preparation phase through to the online real-time assessment pipeline.

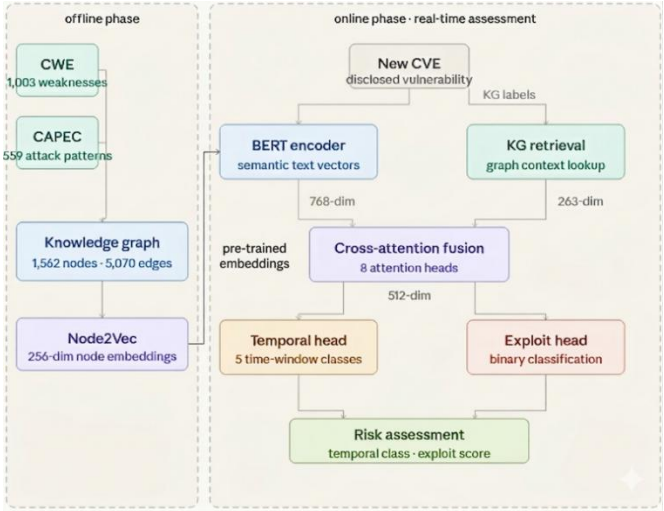

Fig. 1. System architecture.

The offline phase allows us to assemble and encode all cybersecurity data into a single comprehensive knowledge graph that integrates 1,003 CWE weaknesses and 559 CAPEC attack patterns, resulting in a graph containing 1,562 nodes and 5,070 edges. Node2Vec training generates 256-dimensional embeddings that preserve both the graph structure and semantic relationships between security concepts. We compute these embeddings once in the offline phase and keep them fixed. The resulting 256-dimensional node vectors are serialised to disk as a lookup table indexed by CWE and CAPEC identifiers, allowing the online phase to retrieve the relevant embedding for any incoming CVE in constant time without reprocessing the graph.

The online assessment phase operates in real time as new CVEs are disclosed in the national database. Two parallel processing streams work simultaneously: vulnerability descriptions are encoded with a frozen BERT model producing high-dimensional semantic embeddings, while associated CWE labels trigger knowledge graph context retrieval generating structured context vectors. These complementary representations converge in a cross-attention fusion module employing eight attention heads to learn context-aware alignments between textual descriptions and graph-based security knowledge. The fused representation feeds dual prediction heads that simultaneously generate the two target outputs.

C. Feature Extraction and Engineering

Our approach combines representation learning and domain-specific feature engineering within a robust engineering framework. The feature selection process, employed in the creation and evaluation of the dataset, is guided by the exploitation potential and chance of success for a particular attack and impact level. These factors are thereafter correlated with the data to derive actionable indicators.

1) Text encoding: Vulnerability descriptions have distinct linguistic characteristics, including the use of specialized technical vocabulary, adherence to semi-standardized formats, and references to specific software components. SecBERT, a BERT [12] variant pre-trained on cybersecurity content including over 100,000 CVE descriptions, security bulletins, and threat intelligence reports, is used to encode each vulnerability description d_v. The encoding process tokenizes the text using standard BERT tokenization and passes it through 12 transformer layers, producing contextualized representations $H_{text} \in R^{L\times768}$, where L is the sequence length, and 768 is the hidden dimension. We then extract the [CLS] token representation as our text embedding $e_{text} \in R^{768}$—this single vector captures the semantic meaning of the entire description through BERT's self-attention mechanism. We keep SecBERT's weights frozen during training, which helps prevent overfitting and ensures stable embeddings throughout the process.

2) Knowledge graph context retrieval: The graph construction started by using two MITRE resources. The first was the CWE database [22], which lists software vulnerabilities, and the second was CAPEC [23], which lists how those vulnerabilities manifest as real-world attacks. Because these datasets were not designed to be queried together, a unified graph needed to be built for this project first. The final graph, G = (V, E, R), has a total of 1,562 nodes (1,003 CWE nodes and 559 CAPEC nodes) and 5,070 typed edges that connect them. Each node has a name of the vulnerability or attack, a short description of the vulnerability or attack, and a relative likelihood score of {0, 1} derived from a standard set of severity levels. Fig. 2 shows representative subgraphs showing the relationship between CWE vulnerabilities (indicated by blue circles) and CAPEC attacks (indicated by red squares) with directed edges representing the type of relationship.

The majority of the workload is handled by edges. The CWE (Common Weakness Enumeration) that describes the taxonomy allows relationships to describe parent-child hierarchies and dependent relationships such as "requires" or "can precede."

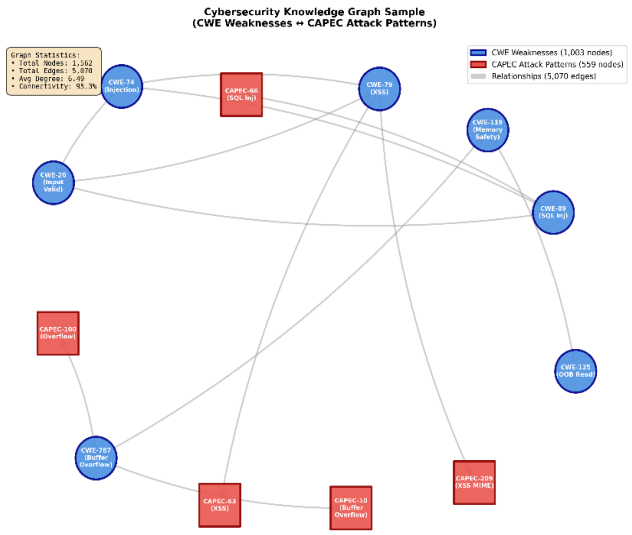

Fig. 2. Subset of knowledge graph.

Across the CWE-CAPEC (Common Attack Pattern Enumeration and Classification) boundary, edges primarily represent enablement: for example, this weakness enables this attack type. However, not all edges in the graph are equally weighted with respect to the amount of exploitation reasoning that they can constrain within the knowledge graph; hierarchies were assigned a confidence of 0.9, associated peers were assigned a confidence of 0.7, and direct enablement mappings were assigned a confidence of 1.0. All of this was due to how each relationship type constrains the manner in which an exploit can be designed based on the aforementioned exploitation criteria. In addition, the average degree of a node in the graph is 6.49; 95.3% of the nodes in the graph are contained within the largest weakly connected component. Therefore, as a result of these characteristics, the graph is very dense and can be traversed meaningfully without any isolated clusters.

Another important characteristic of the CWE-CAPEC relationships is their temporal stability. When new CVEs (Common Vulnerabilities and Exposures) are published, there are no changes to existing CWE-CAPEC relationships based on the disclosure of that CVE. All information related to a CVE, including its CWE relationship: whenever a new CVE is published, it will inherit all of the contextual relationships from its respective or assigned CWE. Therefore, it should be noted that the knowledge graph maintains generalizability to new vulnerabilities without requiring retraining.

3) Graph embedding via Node2Vec: Node2Vec [13] converts the graph structure into fixed-length vectors that the neural model can consume. Alternative knowledge graph embedding methods such as TransE [24] and ComplEx [25] were considered but are better suited to link prediction over entity-relation triples rather than the random-walk-based structural encoding required here. It is worth noting that the knowledge graph functions here as a structured representational source rather than a symbolic reasoner: the CWE-CAPEC ontology, including its typed edges and confidence weights, is encoded into dense vector representations via Node2Vec and subsequently fused with textual embeddings through attention. No explicit logical constraints are imposed on the neural computation at inference time. The framework therefore falls within the paradigm of knowledge-fused neural learning, where structured domain knowledge informs the embedding space rather than governing the output through symbolic inference rules.

The configuration used 256 dimensions, a size following the configuration reported in the original Node2Vec work [13] for graphs of comparable scale and sufficient to capture both local neighbourhood structure and cross-category relationships without unnecessary parameter overhead. Random walks of length 10 and 200 walks per node were used. Both the return parameter p and in-out parameter q were set to 1.0, giving equal weight to local neighbourhood exploration and longer-range traversal; a reasonable default for a graph where both local clusters and cross-category bridges carry information. Walk generation took roughly 2–5 minutes; Word2Vec training over 3 epochs with a context window of 5 added another 5–10 minutes. Total embedding time for the 1,562-node graph came to around 14 minutes, which matters practically—it means the graph can be re-embedded periodically as new CWE entries or CAPEC patterns are added without significant overhead.

To illustrate the extent to which the embeddings capture meaningful predictive signals, we run illustrative sampling tests. Looking at nearest-neighbor relationships, CWE-79 (Cross-Site Scripting) is clustered close to CAPEC-63, its corresponding XSS attack pattern, and CWE-89 (SQL Injection) landed near CAPEC-66. These are the pairings you would expect if the model captured weakness-to-attack relationships rather than just co-occurrence noise. The cross-category structure also held up: CWE-79 and CWE-89 shared moderate similarity as injection-class weaknesses, while CWE-787 (heap buffer overflow) sat well apart from both, which is correct, since memory corruption and injection bugs operate through entirely different mechanisms. The separation was not enforced; it emerged from the graph topology.

D. Neural Architecture

1) Cross-attention fusion module: Text embeddings and graph embeddings capture fundamentally different aspects of a vulnerability—one encodes semantic meaning from the description, the other encodes structural exploitation relationships from the knowledge graph. Simply concatenating the two representations treats them as independent signals and loses the conditional dependencies between them. Cross-attention addresses this by allowing the text representation to dynamically query the graph, weighting each graph concept according to its relevance to the specific vulnerability being assessed. Different description segments naturally attend to different graph structures—'remote code execution' aligns with network attack patterns, while 'privilege escalation' draws on local privilege chain relationships.

Both embedding streams are first projected to a shared 512-dimensional space using learned linear transformations, where

W_text ∈ R^{512×768} maps the BERT output and W_graph ∈ R^{512×263} maps the graph context vector. The 512-dimensional projection space was chosen to match the hidden dimension of the BERT encoder, ensuring that neither modality is compressed before fusion and that the cross-attention mechanism operates on representations of equal capacity. The derived text representation then serves as a query attending over the graph representation as key and value:

$$Attention(Q, K, V) = softmax(QK^T / \sqrt{d_k})V \quad (1)$$

with 8 parallel attention heads, each operating in a 64-dimensional subspace. The attended output is combined with the original text signal through a residual connection, preserving the semantic content of the vulnerability description while incorporating graph-derived context. A two-layer feedforward network then consolidates both into a unified 512-dimensional representation h_fused ∈ R^{512} that feeds the prediction heads.

The 263-dimensional graph-context vector is not the raw Node2Vec embedding. It is the 256-dimensional Node2Vec embedding — averaged over the CWE/CAPEC nodes retrieved for a given CVE, weighted by likelihood/severity — concatenated with seven explicit exploit-signal features derived from the retrieved subgraph: the number of related attack patterns, average and maximum CAPEC likelihood, average and maximum CAPEC severity, the number of prerequisites across related CAPECs, and the number of directly associated CWEs. This yields 256 + 7 = 263 dimensions, matching the kg_dim used as input to the cross-attention fusion module described above.

The residual connection serves a practical purpose beyond standard architectural convention: it prevents the graph signal from overwriting the text representation in cases where the CWE label is missing or uninformative. In those cases, the attention weights naturally collapse toward zero, and the fused representation gracefully reverts to the text-only baseline. This prevents the generation of corrupted outputs.

2) Multi-task prediction heads: The encoding stage processes two input modalities: a pre-trained BERT model takes vulnerability descriptions as input and outputs 768-dimensional vectors (110M frozen parameters), while CWE labels are used to create the trigger graph, yielding 263-dimensional vectors. During the fusion phase, the two streams of information are projected into a 512-dimensional space (393K + 135K). The cross-attention mechanism (525K parameters) then integrates these embeddings, and the final output is generated via a feedforward network. The prediction phase consists of a shared 256-dimensional bottleneck layer (131K parameters) before branching into task-specific modules: a temporal head (263K) to classify the five temporal categories and an exploitation head (174K) for binary classification. Of the entire model, which consists of 111.6M parameters, only 2.1M (1.9%) need to be trained. The architecture thus achieves a favorable balance between model size and training efficiency.

All tasks share a two-layer feature extraction:

$$h_{task} = Dropout(ReLU(W_{s2} \cdot ReLU(W_{s1} \cdot h_{fused})), p = 0.3) \quad (2)$$

where, W_s1, W_s2 ∈ R^{256×512}, producing a 256-dimensional shared task representation. Two independent heads then branch from h_task: the temporal classification head applies a five-way softmax to assign each CVE to one of the exploitation windows, while the exploitation prediction head applies a sigmoid activation for binary probability output:

$$\hat{e} = Sigmoid(Linear(ReLU(W_{exploit} \cdot h_{task}))) \quad (3)$$

Dropout at p = 0.3 is applied during training to prevent the prediction heads from overfitting to the dominant majority class.

The two prediction heads are trained jointly using a fixed, linearly weighted sum of per-task cross-entropy losses:

$$L = \lambda\_temporal \cdot L\_temporal(\hat{y}\_temporal, y\_temporal) + \lambda\_exploit \cdot L\_exploit(\hat{y}\_exploit, y\_exploit) \quad (4)$$

with λ_temporal = 1.0 and λ_exploit = 2.0. The higher weight on the exploitation loss compensates for the 85%/15% class imbalance in the binary target and was selected via a small grid search over {1.0, 1.5, 2.0, 3.0} on the 2023 validation split; no curriculum learning or dynamic/uncertainty-based loss weighting is used. Both cross-entropy terms additionally apply per-class weights, computed by inverse class frequency on the training split, to address within-task imbalance independently of the between-task λ coefficients. Gradients from both losses flow jointly through the shared bottleneck layer and the fusion module, with no gradient-surgery or explicit gradient-balancing mechanism beyond the static weighting described above.

IV. Results and Discussion

This section presents a detailed account of the experimental outcomes obtained when evaluating the proposed framework on a held-out test set of 14,280 CVE records sampled from 2024. The evaluation covers temporal classification performance, binary exploitation prediction, ablation analysis, and robustness verification through cross-validation. Each subsection concludes by directly addressing the corresponding research question where relevant.

A. Evaluation Metrics

Class imbalance is one of the major challenges presented by this evaluation, with approximately 85% of the CVEs in the test set not having been exploited, and a naive baseline model flagging the majority class would have an accuracy of 85% without detecting any true positives. Therefore, standard accuracy will not be a useful metric for this task. To ensure a robust evaluation, four different measures will be used rather than relying on a single measure.

While precision and recall are contrary, both of them are very important. An analyst wastes time because a flagging system generated too many false positives—a precision issue—and is exposed to risk because the system missed a valid exploitation—a recall issue. The cost of false negatives is

significantly greater in security operations than the cost of false positives, which places much more weight on recall for high-risk classes. The F1 measure will be used to report the primary measure because it penalizes the tradeoff rather than obscuring it.

Accuracy, while still present in this evaluation, will only be reported with the per-class breakdown measure. When viewed in isolation, accuracy may misrepresent the distribution of values across the classes; however, when viewed next to the class-level data, accuracy will provide an overall summary of the evaluation.

AUC-ROC is included as a metric for evaluating binary exploitation; it is a ranked measure of performance, unlike fixed-threshold metrics (accuracy, F1) that are sensitive to class imbalance. Therefore, AUC-ROC is much more resilient than both accuracy and F1 for class-imbalanced scenarios [26]. Additionally, both weighted and macro-averaged F1 are computed for the five-class temporal head. The macro variant treats each of the five classes equally, regardless of how many samples belong to each class, giving equal influence to performance on Class 0 (the rarest and most critical time window; 0–7 days) and performance on Class 4.

B. Overall Performance Results

The proposed framework's performance outcomes are represented in Table III, against five baseline models. These five baseline models include rule-based, classical machine learning, deep learning, and industry-standard methods of operation, with the majority of all operational work still using CVSS as the operational heuristic. Furthermore, the EPSS developed through the consortium FIRST, and built using logistic regression over NVD features, is currently deployed as the state-of-the-art technique for predicting the automated exploitation of vulnerabilities. As demonstrated in the results, the progressive improvement methodologies consistently yield measurable improvements, and the final model's performance results exceed all five baseline models across the performance metrics reported.

TABLE III. PERFORMANCE COMPARISON ACROSS MODELS

Method	Accuracy (Exploit)	F1 (Exploit)	AUC-ROC	Accuracy (Temporal)	Macro-F1 (Temporal)
CVSS Threshold	0.623	0.342	0.589	—	—
Random Forest	0.741	0.614	0.723	—	—
LSTM Networks	0.768	0.646	0.757	—	—
Fine-tuned BERT	0.821	0.702	0.812	—	—
EPSS	0.847	0.679	0.827	—	—
Proposed	0.891	0.747	0.891	0.784	0.623

The CVSS threshold baseline provides a logical foundation for comparisons, not just because it provides a uniform standard to a comparative set of organizations but also because it is reflective of how organizations operate today. The model's overall exploitation F1 score of 34.2% should give organizations pause because CVSS-based triage exists across many organizations in practice. This result corroborates the earlier empirical findings by Allodi and Massacci [4,5], who showed that there is only a weak correlation between severity and the likelihood of exploiting a vulnerability.

Random Forest does considerably better at an F1 score of 61.4%, capturing non-linear interactions between CVSS attributes and TF-IDF text features that are not discernible by threshold-based rules. While the performance improvement is substantial, handcrafted features and binary framing constrain the approach's predictive capacity. LSTM networks further improve the performance to 64.6% by modeling descriptions as sequences rather than bags of words. Fine-tuned BERT reaches a score of 70.2%, the highest text-only performance, by leveraging pre-trained contextual representations that capture security semantics more precisely than traditional TF-IDF methods.

EPSS achieves an F1 score of 67.9% and an AUC of 82.7%. While it outperforms the simpler ML baselines, it remains inferior to both BERT and our proposed model, which achieves an F1 score of 74.7% and an AUC of 89.1%. The F1 gain is 6.8 above EPSS, and the AUC gain is 6.4 at the same time. The AUC increase is of greater importance than the difference between F1 in practice, as it describes the ranking quality across all potential thresholds as opposed to just performance at a single cut-off. A 6.4 increase at this level represents a significant and meaningful advancement in predictive reliability.

There is no external baseline for temporal performance, as none of the systems were built to provide five-class time-window predictions. The final system is accurate to 78.4% and has a 62.3% macro-F1 score. The macro-averaged score is a more conservative measure, since the proportion of samples in Class 0 (0–7 days) is 1.4% of the total test sample; therefore, Class 0 is weighted equally to Class 4. Therefore, the macro-average cannot provide a good measure of performance by inflating the majority-class performance. The system's ability to obtain a score above 62% suggests that signals from the prediction occur throughout the entire time range, not just in the most straightforward cases.

C. Temporal Classification: Per-Class Analysis

Table IV reports detailed per-class precision, recall, and F1-score for the temporal classification task, discussed below.

Breaking down the temporal results by class reveals a coherent pattern that aligns well with what one would expect from the data characteristics and the underlying security dynamics.

TABLE IV. DETAILED PER-CLASS TEMPORAL CLASSIFICATION METRICS

Class	Window	Precision	Recall	F1-Score	Operational Label
Class 0	0–7 days	0.7123	0.6590	0.6845	Emergency patch
Class 1	8–30 days	0.6834	0.6245	0.6523	Accelerated deploy
Class 2	31–90 days	0.6456	0.5938	0.6189	Monthly cycle
Class 3	91–365 days	0.5912	0.5567	0.5734	Quarterly update
Class 4	> 365 d / Never	0.8567	0.7945	0.8234	Risk-based deferral
Weighted Avg	—	0.7956	0.7842	0.7891	—

Class 4 (vulnerabilities exploited beyond one year or not at all) makes up 86.6% of the test set and corresponds to an F1 score of 82.3%. Therefore, we should not over-interpret this. Even a very naive classifier will quickly learn that the vast majority of CVEs do not actually see any real-world exploitation.

The more interesting number lies elsewhere. Class 0 (0–7 day window) consists of 200 of the 14,280 testing samples, and the model achieves an F1 of 68.5%, 71.2% precision, and 65.9% recall. This means that approximately two-thirds of imminent exploits are correctly flagged and would not generate so many false positives that an analyst's queue could not handily keep pace with them. This is especially significant, given that Class 0 is very rare and has potentially great significance in the real world. This result is operationally significant given the low base rate of Class 0.

The place where the model performs the poorest is in Class 3 (3–12-month window) with an F1 of 57.3%. This fact is not surprising. The vulnerabilities exploited within this time frame are much more difficult for the model to learn about; therefore, the training signal is much sparser, while the type of exploitation that occurs in this time frame is often based on campaign-specific decisions made by threat actors that are not reflected in the CVE or CWE data. It reads more like a data limitation than a modeling one. This distinction is relevant for prioritizing future data-collection versus architectural investment.

In conclusion, while the per-class numbers appear harsh, the error analysis provides some level of comfort. 61.3% (or 1,072) of all misclassifications were made between adjacent temporal classes (for example, Class 1 being read as Class 2) and comparatively few (34) were made between disparate classes. The overwhelming majority (98%) of misclassifications were made between two adjacent windows; thus, the vast majority of them will have no practical operational effect as they will use roughly similar response protocols within the same patching frameworks. The only truly damaging error is the misclassification of an active exploitable vulnerability as being unlikely to ever be exploited (267 instances, or 1.9% of the dataset). In an imbalanced dataset, keeping this number below 2% is arguably the most important threshold the system has successfully achieved.

D. Ablation Study

We performed an ablation study where three controlled configurations isolate the contribution of each architectural layer. As Table V shows, each addition produces consistent gains across all three metrics.

TABLE V. ABLATION STUDY: INCREMENTAL COMPONENT CONTRIBUTION

Configuration	Component	Temporal Accuracy	Exploit F1	AUC-ROC
Text Only (baseline)	BERT encoder only	0.6834	0.6245	0.7456
Text + KG, No Fusion	BERT + concatenation	0.7456	0.6923	0.8234
Complete Model (proposed)	BERT + cross-attn + KG	0.7842	0.7467	0.8912

The baseline performance level (62.5% exploitation F1) achieved using the BERT model based solely on textual data is improved to 69.2% by combining the CWE-CAPEC knowledge graph using simple concatenation, which demonstrates that structured domain knowledge contributes a significant amount of additional information beyond what is provided by vulnerability descriptions alone. By upgrading the way in which the knowledge graph and text data are combined from simple concatenation to cross-attention, it is possible to further increase performance to 74.7% F1 and 89.1% AUC-ROC through cross-attention's ability to learn conditionally aligned mappings; for example, being able to map the phrase "heap overflow via remote input" to CAPEC-100 instead of CAPEC-66, which is something that cannot be done with simple concatenation.

Answer to RQ3: The 12.2% total increase in F1 above the text-only baseline demonstrates that neural text understanding and graph-based structural encoding do not just coexist together in a complementary manner, but rather that they are mutually reinforcing. As an example, the knowledge graph provides context of exploitation pathways that are often left implicit in textual descriptions, while cross-attention allows for the selective application of that context instead of the blanket addition of that context to all instances of text.

E. Knowledge Graph Integration vs. Text-Only Approaches (RQ1)

The ablation results directly address RQ1. Moving from the text-only baseline to the complete model yields a 12.2-point F1 gain, a margin exceeding the performance gap between the CVSS heuristic and fine-tuned BERT (cf. Table III). The knowledge graph allows the model to reason about structural exploitation pathways rather than pattern-match on surface vocabulary, linking weaknesses to attack classes and modeling the exploitation likelihood across the CWE-CAPEC hierarchy. This structured knowledge captures relationships that a free-text vulnerability description often leaves implicit.

Answer to RQ1: Yes. Integrating the CWE-CAPEC knowledge graph improves exploitation F1 by 12.2% and AUC-ROC by 14.6% over the text-only baseline, and outperforms the current industry standard EPSS by 6.8 F1 points and 6.4 AUC points.

F. Temporal Classification versus Binary Prediction (RQ2)

Binary exploitation prediction suffers a practical limitation that is easily overlooked. Flagging a vulnerability as "exploitable" provides a security team no insights into timing, which is critical to determining whether a patch needs to be deployed tonight or can wait for the next quarterly cycle. The two decisions carry completely different resource costs, yet a binary model treats them identically.

The binary classification system (yes/no) that currently exists for exploiting vulnerabilities has a significant limitation: there is only one way to classify a CVE (Common Vulnerability and Exposure). By implementing the proposed temporal classification system, several new methods to classify the exploitability of CVEs are available; in this case, five new intervals for classifying CVEs will replace the single yes/no output (0–7 days, 8–30 days, 31–90 days, 91–365 days, or never exploited).

In reality, most organizations already utilize tiered patching protocols that will map directly to the new temporal classifications proposed; i.e., emergency remediation, accelerated deployment, monthly deployment, quarterly review, and risk-based exception. This allows the temporal predictions to be utilized directly within existing workflows without the need for an additional triage layer between the two processes.

This means that operationally, the outputs of temporal prediction provide functionality that enables a more actionable risk intelligence than currently exists. For example, when a CVE falls into Class 0, the corresponding response is immediate (emergency dispatch), which would differ substantially from the response when falling into Class 3 (no immediate dispatch), rather than relying upon an analyst's judgment and decision.

Answer to RQ2: Prediction through temporal CVE classification results in more actionable risk intelligence than binary classifications. Auto-mapping of the predicted exploitation period onto known patching tier periods eliminates the manual conversion process done by the binary classification, resulting in greater consistency in making decisions.

G. Deployment Feasibility and Computational Overhead

To assess whether the framework can operate under realistic operational workloads, we benchmarked inference latency and throughput on the deployment hardware (NVIDIA GPU, CUDA). Single-CVE inference latency averaged 6.43 ms (p50 = 6.44 ms, p95 = 6.74 ms, p99 = 6.87 ms), well within the sub-second budget required for interactive triage. Batch throughput scales from 139.4 CVE/s at batch size 1 to 1,207.4 CVE/s at batch size 64, with diminishing marginal gains beyond batch size 32 (1,188.2 CVE/s), indicating that the cross-attention fusion module is not the throughput bottleneck at moderate batch sizes. At the best observed throughput, the framework can process an estimated 104.3 million CVE assessments per day on a single GPU instance. Given that the National Vulnerability Database discloses on the order of dozens to a few hundred CVEs per day, a single deployed instance provides several orders of magnitude of headroom, and horizontal scaling is straightforward: the offline knowledge-graph embeddings are fixed and shared across workers, and the online scoring path is embarrassingly parallel across CVEs. These results indicate that computational overhead is not a barrier to enterprise-scale deployment.

V. Conclusion

Prioritization of vulnerabilities through traditional methods often relies heavily on their severity scores, which indicate how theoretical a vulnerability's impact is versus how the vulnerability will likely be exploited in the real world. Prioritizing vulnerabilities in this fashion incurs real costs; security teams work to patch vulnerabilities that will never be exploited due to limited time and resources, and they may also leave undetected vulnerabilities that warrant prompt prioritization and patching. This research reframes the question of time to exploitation of a vulnerability as five classes of temporal classifications, capturing timing information that is often lost in binary frameworks, and aligning classification outputs with patching tiers currently employed by most organizations.

These design choices have been validated. The complete framework yields 74.7% exploitation F1 and 89.1% AUC-ROC and outperforms the EPSS (the current industry standard) by 6.8 points for F1 and 6.4 points for AUC-ROC, and outperforms the benchmark for text-only BERT by 12.2 F1 points. Ablation studies indicate that knowledge graph integration and cross-attention fusion are the major reasons for this improvement in performance, indicating that explicitly modelling connections between weaknesses and attack methodologies has added more value than the ability to simply provide more complex text representations of vulnerability descriptions.

There are limitations to this study. The data used to create the labels for ground-truth vulnerability exploitation (CISA KEV, Exploit-DB) only reflects instances of vulnerability exploitation that are publicly reported. As such, the extent of the model's performance will be impacted as the model does not account for vulnerabilities being exploited in a private capacity. In addition, NVD records without CWE numbers reduce the ability to make predictions about a vulnerability's exploitation, as the model must rely solely on the text-based features of the record. The model was trained on an archive of exploitation data through 2022; this means that to continue to understand vulnerability exploitation as technology changes and attacker tools evolve, the model must be retrained consistently.

In the short term, improvements to the model will be more appropriately focused on the data side than the modeling side (automated CWE labels during preprocessing, quarterly retraining aligned with KEV updates, and expanding the number of ATT&CK techniques in the knowledge graph). In terms of architectural improvements, changing from Node2Vec to Graph Attention Networks and estimating uncertainty of

low-confidence outputs would allow for a more effective way to implement the system in an operational capacity. The primary conclusion reached in this work is that vulnerability exploitation risk is ultimately temporal and relational rather than simply a byproduct of severity; therefore, a reconsideration of how exploitation information is produced and utilized is likely warranted.

REFERENCES

[1] National Institute of Standards and Technology (NIST), "National Vulnerability Database," 2024. [Online]. Available: https://nvd.nist.gov/

[2] FIRST (Forum of Incident Response and Security Teams), "Common Vulnerability Scoring System Version 3.1: Specification Document," 2019. [Online]. Available: https://www.first.org/cvss/v3.1/specification-document.

[3] L. Allodi and F. Massacci, "Comparing vulnerability severity and exploits using case-control studies," ACM Trans. Inf. Syst. Secur. (TISSEC), vol. 17, no. 1, pp. 1–20, 2014. doi: 10.1145/2630069.

[4] J. Jacobs, S. Romanosky, B. Edwards, M. Roytman, and I. Adjerid, "Exploit prediction scoring system (EPSS)," Digit. Threats Res. Pract., vol. 2, no. 3, pp. 1–17, 2021. doi: 10.1145/3436242.

[5] L. Allodi and F. Massacci, "Security events and vulnerability data for cybersecurity risk estimation," Risk Anal., vol. 37, no. 8, pp. 1606–1627, 2017. doi: 10.1111/risa.12864.

[6] M. Shahzad, M. Z. Shafiq, and A. X. Liu, "A large-scale exploratory analysis of software vulnerability life cycles," in Proc. IEEE/ACM Int. Conf. Software Engineering (ICSE), 2012, pp. 771–781. doi: 10.1109/ICSE.2012.6227149.

[7] S. Frei, M. May, U. Fiedler, and B. Plattner, "Large-scale vulnerability analysis," in Proc. ACM SIGCOMM Workshop on Large-Scale Attack Defense (LSAD), 2006, pp. 131–138. doi: 10.1145/1162666.1162671.

[8] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: Learning to classify vulnerabilities and predict exploits," in Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2010, pp. 105–114. doi: 10.1145/1835804.1835821.

[9] M. Edkrantz and A. Said, "Predicting cyber vulnerability exploits with machine learning," in Proc. Scandinavian Conf. on AI (SCAI), vol. 278, 2015, pp. 48–57. doi: 10.3233/978-1-61499-589-0-48.

[10] J. Jacobs, S. Romanosky, B. Edwards, I. Adjerid, and M. Roytman, "Improving vulnerability remediation through better exploit prediction," J. Cybersecurity, vol. 6, no. 1, p. tyaa015, 2020. doi: 10.1093/cybsec/tyaa015.

[11] A. Okutan, O. B. Vivas, and G. Wallin, "Predicting cyber security incidents using feature-based characterization of network-level malicious activities," in Proc. ACM Workshop on Artificial Intelligence and Security, 2017, pp. 3–13. doi: 10.1145/3128572.3140447.

[12] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. Conf. of the North American Chapter of the Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT), 2019, pp. 4171–4186. doi: 10.18653/v1/N19-1423.

[13] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2016, pp. 855–864. doi: 10.1145/2939672.2939754.

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

[15] S. M. Ghaffarian and H. R. Shahriari, "Software vulnerability analysis and discovery using machine learning and data mining techniques: A survey," ACM Comput. Surv. (CSUR), vol. 50, no. 4, pp. 1–36, 2017. doi: 10.1145/3092566.

[16] S. Chakraborty, R. Krishna, Y. Ding, and B. Ray, "Deep learning based vulnerability detection: Are we there yet?," IEEE Trans. Softw. Eng., vol. 48, no. 9, pp. 3280–3296, 2022. doi: 10.1109/TSE.2021.3087402.

[17] F. Li and V. Paxson, "A large-scale empirical study of security patches," in Proc. ACM Conf. on Computer and Communications Security (CCS), 2017, pp. 2201–2215. doi: 10.1145/3133956.3134072.

[18] L. Bilge and T. Dumitras, "Before we knew it: An empirical study of zero-day attacks in the real world," in Proc. ACM Conf. on Computer and Communications Security (CCS), 2012, pp. 833–844. doi: 10.1145/2382196.2382284.

[19] P. Wang, L. Gao, B. Ding, and Y. Chen, "VulGraph: A knowledge graph based approach for vulnerability information management," in Proc. IEEE Int. Conf. on Software Maintenance and Evolution (ICSME), 2019, pp. 553–557.

[20] A. Pingle, A. Piplai, S. Mittal, and A. Joshi, "RelExt: Relation extraction using deep learning approaches for cybersecurity knowledge graph improvement," in Proc. IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM), 2019, pp. 879–886. doi: 10.1145/3341161.3343519.

[21] MITRE Corporation, "CWE — Common Weakness Enumeration," 2024. [Online]. Available: https://cwe.mitre.org/

[22] MITRE Corporation, "CAPEC — Common Attack Pattern Enumeration and Classification," 2024. [Online]. Available: https://capec.mitre.org/

[23] MITRE Corporation, "MITRE ATT&CK," 2024. [Online]. Available: https://attack.mitre.org/

[24] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2013, pp. 2787–2795.

[25] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in Proc. Int. Conf. on Machine Learning (ICML), 2016, pp. 2071–2080.

[26] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in Proc. IEEE Int. Conf. on Computer Vision (ICCV), 2017, pp. 2980–2988. doi: 10.1109/ICCV.2017.324.

[27] Cybersecurity and Infrastructure Security Agency (CISA), "Known Exploited Vulnerabilities Catalog," 2024. [Online]. Available: https://www.cisa.gov/known-exploited-vulnerabilities-catalog

[28] Offensive Security, "Exploit Database," 2024. [Online]. Available: https://www.exploit-db.com/