PROJECT: NIST VULNERABILITY EXTRACTOR

NIST Vulnerability Information Extractor

> A Transformer-based Named Entity Recognition (NER) system for automated cybersecurity intelligence extraction using DeBERTa architecture.

December 2025 | CyNER2.0 Augmented | DeBERTa v3 Base

Project Team

- **Ishaan Singh** LCI2022025
- **Aun Abbas** LCI2022014
- **Shashank Agarwal** LCI2022045
- **Aaditya Mishra** LCI2022001
- **Disha Yadav** LCI2022021

SECTION 1.0

Introduction: Automating CVE Extraction

PROJECT STATUS
DEPLOYED

The Challenge: Unstructured Data

Cybersecurity ecosystems rely on CVE entries, but descriptions are unstructured text. Manual analysis is slow, error-prone, and cannot scale with the volume of modern threats.

The Goal: Automated Extraction

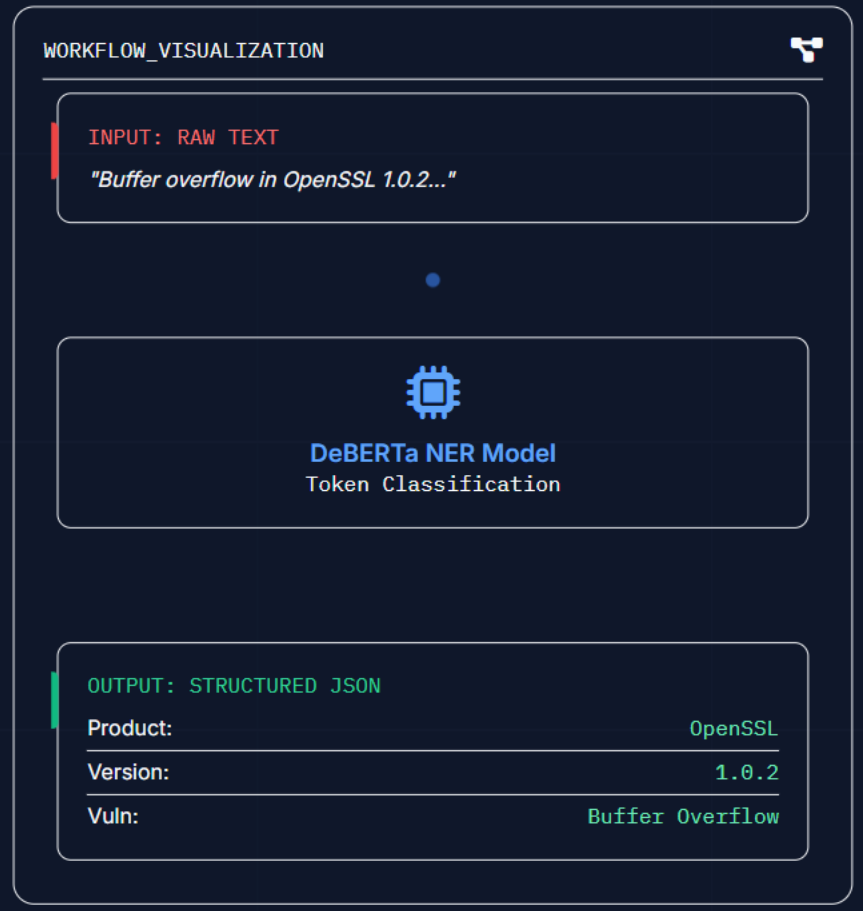
Develop a system to automatically identify and classify critical entities: `Products`, `OS`, `Attack Vectors`, and `Impact Effects`.

The Approach: Transformer NER

We built a Named Entity Recognition (NER) system using **DeBERTa v3**, fine-tuned on an augmented cybersecurity dataset (CyNER2.0) to handle technical jargon.

The Outcome: Scalable Intelligence

A robust pipeline that transforms raw text into structured JSON, enabling immediate integration with SIEM tools, SOC dashboards, and vulnerability scanners.



Objectives & Use Cases

STRATEGIC GOALS

Core Objectives

- Achieve **high-accuracy Named Entity Recognition (NER)** for specialized cybersecurity entities within unstructured text.
- Develop robustness to handle **rare technical terms** and product names often missed by general-purpose models.
- Ensure scalability to process **massive volumes** of CVE descriptions and security advisories in real-time.

Key Deliverables

- Fine-tuned DeBERTa Model
- End-to-End Inference Pipeline
- Streamlit & Gradio Apps
- Comprehensive Technical Report

Application Ecosystem

SIEM Enrichment

Automatically tag incoming logs with affected products and vulnerabilities to prioritize alerts in Security Information and Event Management systems.

Vulnerability Scanners

Parse scan reports to extract standardized entity data for cross-referencing against internal asset inventories.

SOC Dashboards

Visualize trends in attack vectors and affected operating systems extracted from daily threat feeds.

Threat Intelligence

Build knowledge graphs linking threat actors to specific product versions and vulnerability types from unstructured reports.

SECTION 3.0

Dataset: CyNER2.0 Augmented

DATA SOURCE

Dataset Source

PranavaKailash/CyNER2.0_augmented_dataset

Hugging Face · High Quality · NER Task

An improved, enlarged version of the original CyNER 2.0 dataset specifically designed for training robust cybersecurity entity recognition models.

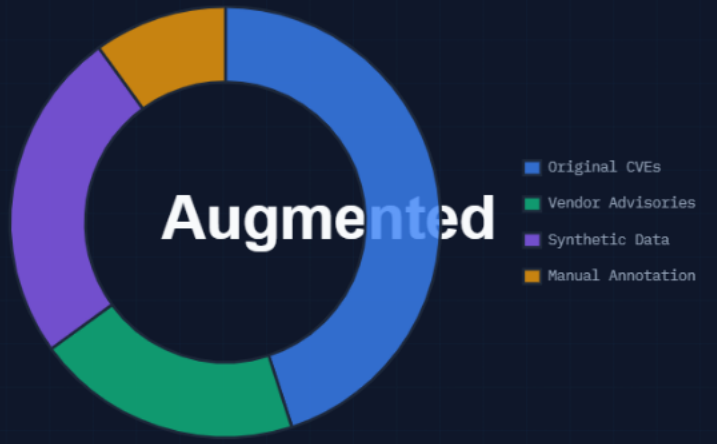
Dataset Composition

CVE Descriptions
Sourced directly from MITRE CVE and NIST NVD databases covering wide range of vulnerabilities.

Vendor Advisories
Security bulletins from major software vendors (Microsoft, Cisco, RedHat) providing diverse technical context.

Manual Annotations
Expert-verified labels ensuring high ground-truth quality for training and evaluation.

Synthetic Augmentations



AUGMENTATION TECHNIQUES

Synonym Replacement — VOCABULARY
"Execute code" → "Run arbitrary commands"

Version Swapping — ROBUSTNESS
"Apache 2.4.49" → "Apache 2.4.50-rc1"

Template Generation — STRUCTURE
Synthesizing new sentence structures for unseen formats

Product-OS Combos — DIVERSITY
Randomizing "PostgreSQL on Windows" vs "PostgreSQL on Fedora"

SECTION 2.2

Entity Labels: BIO Schema Classes

8 TOTAL CLASSES

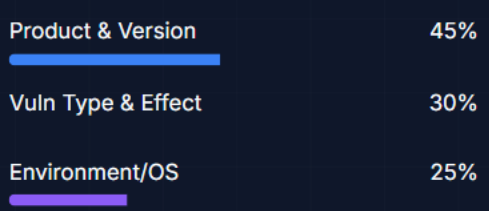
ENTITY LABEL	DESCRIPTION & MEANING	REAL-WORLD EXAMPLES
PRODUCT	Software, hardware, or firmware names affected by the vulnerability.	`OpenSSL`, `Apache HTTPD`, Cisco ASA, Zoom Client
OS	Operating systems mentioned as the platform or environment.	`Ubuntu 18.04`, `Windows Server 2016`, macOS Big Sur
ENVIRONMENT	Runtime environments, containers, or virtualization contexts.	Docker `Container`, Kubernetes `pod`, VMware `vSphere`
VERSION / PATCH	Specific version numbers, builds, or patch identifiers.	prior to `1.0.2g`, `7.5.2`, build `2021-04`, KB `5001330`
ATTACK_VECTOR	The method or pathway used to deliver the exploit.	`Remote` attacker, via `network`, `physically present` user
PREREQUISITE	Conditions required before exploitation is possible.	`Requires login`, `authenticated user`, with `admin privileges`
EFFECT	The consequence or impact of successful exploitation.	`Code execution`, `privilege escalation`, denial of service
CWE/VULN_TYPE	The category or technical nature of the weakness.	`Buffer overflow`, `SQL injection`, Cross-site scripting (XSS)

Annotation Format

Data follows the **CoNLL** token-tagging format using BIO (Beginning, Inside, Outside) notation.

OpenSSL	B-PRODUCT
1.0.2	B-VERSION
on	O
Linux	B-OS

LABEL DISTRIBUTION



Dataset: Format & Augmentation

DATA STRUCTURE
CONLL + BIO

CoNLL Token-Tagging Format

Standard BIO (Begin-Inside-Outside) schema used for token-level annotation. Each token is aligned with a specific label, ensuring granular entity boundaries even for complex technical terms.

`OpenSSL B-PRODUCT` `1.0.2g B-VERSION` `buffer B-VULN` `overflow I-VULN`

Token-Level Annotation Alignment

Annotations are strictly aligned to subword tokens generated by the tokenizer (WordPiece/BPE). This prevents misalignment issues common in cybersecurity texts containing special characters (e.g., CVE-2023-1234, mod_ssl).

Purpose of Augmentation

Cybersecurity vocabulary is niche and rapidly evolving. Augmentation addresses:

- Handling rare product names (e.g., legacy or custom software)
- Improving recognition of long entity spans (e.g., "arbitrary code execution via buffer overflow")
- Increasing robustness for unseen CVE formats and varied phrasing

Robustness to Phrasing

By training on augmented sentences (synonyms, version swaps, templates), the model generalizes better to real-world reports that differ from standard NVD descriptions.

DATA_FORMAT_VISUALIZATION

Token	BIO Label
Apache	B-PRODUCT
Tomcat	I-PRODUCT
9	B-VERSION
.	I-VERSION
0	I-VERSION
allows	O
RCE	B-EFFECT

AUGMENTATION_STRATEGY

ORIGINAL	AUGMENTED
"Ubuntu 20.04"	"Debian 11"
"Code Execution"	"Arbitrary command run"

SECTION 4.0

Model Architecture

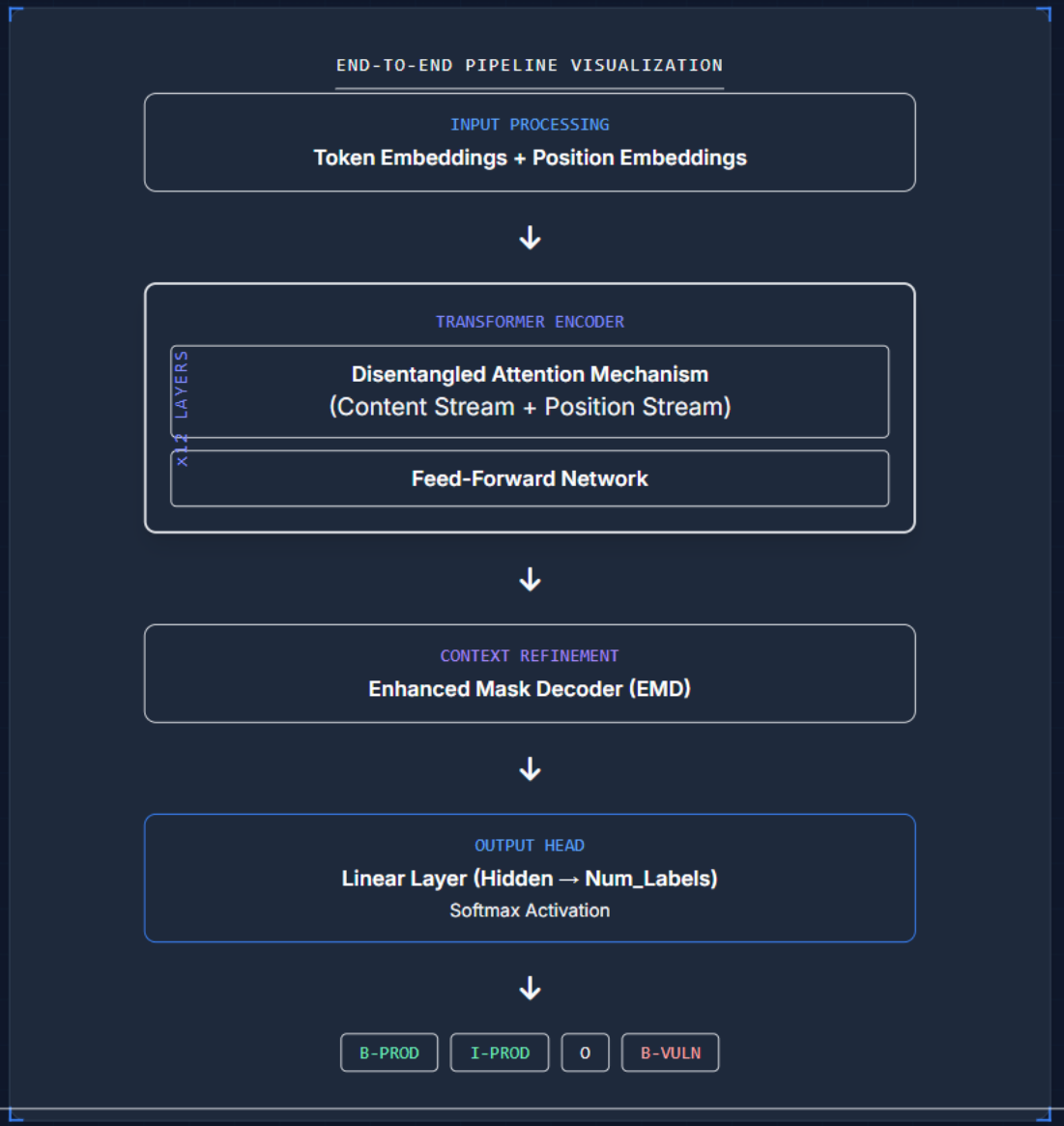
DEBERTA V3 BASE

Core Specifications

Base Model	DeBERTa-v3-base
Fine-Tuned Ver.	cyner_ner_model
Task Type	Token Classification
Output Head	Linear + Softmax

Why DeBERTa?

- **Disentangled Attention:** Unlike BERT, it separates content and position embeddings, allowing better modeling of relative positions crucial for technical syntax.
- **Enhanced Mask Decoder:** Incorporates absolute positions in the decoding layer to improve context awareness for masked token prediction.
- **Domain Adaptation:** Superior handling of "context-dependent" cybersecurity terms (e.g., distinguishing "remote" as vector vs. location).



DeBERTa Innovations for Technical Language

Disentangled Attention Mechanism

Unlike BERT which sums embeddings, DeBERTa separates **content embeddings** from **position embeddings**. Two distinct attention scores are computed: content-to-content and content-to-position.

`Content Vector` `Position Vector`

Explicit Relative Positioning

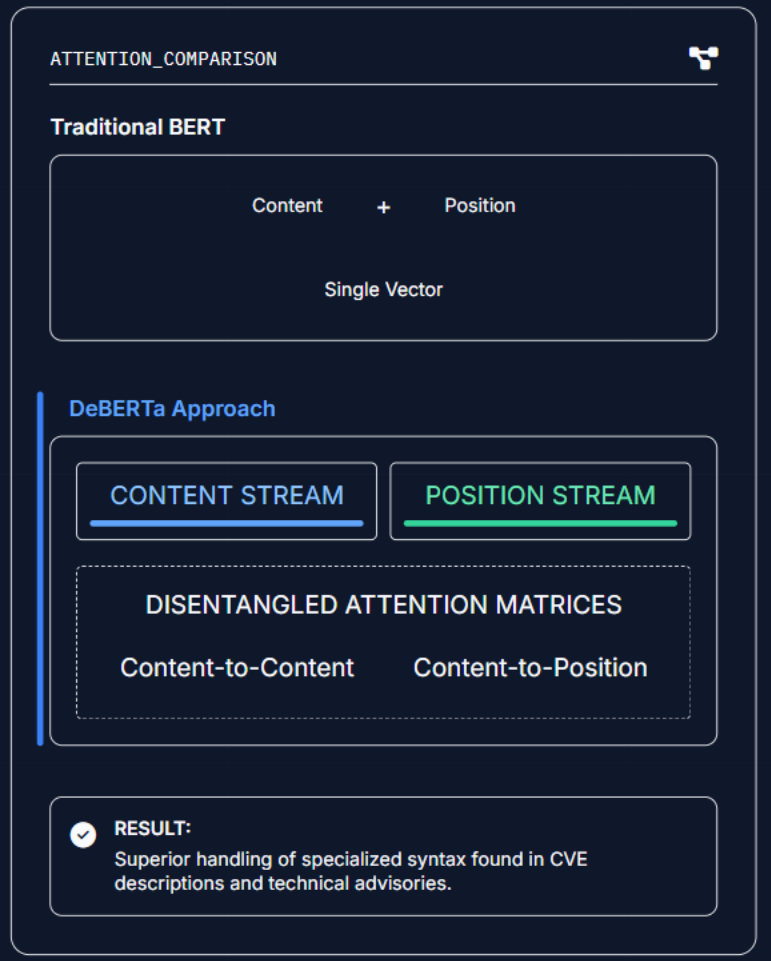
The model understands the *relative distance* between tokens rather than just absolute positions. This is critical for parsing complex technical syntax where word order defines relationships (e.g., "Apache 2.4" vs "2.4 Apache").

Enhanced Mask Decoder (EMD)

Incorporates absolute position information right before the softmax layer during pre-training. This significantly improves the model's ability to predict masked tokens based on both context and exact location.

Domain Context Preservation

These architectural choices help preserve the specific meaning of polysemous words in cybersecurity. Terms like "remote" (attack vector) vs. "remote" (adjective) are distinguished with higher accuracy.



SECTION 4.2

Tokenization & Embedding Pipeline

PREPROCESSING

Tokenizer Mechanics

The tokenizer transforms raw text into numeric representations essential for the model. It produces three key outputs:

`token_ids`	Numerical ID for each token in vocab
`attn_mask`	Binary mask (1=content, 0=padding)
`type_ids`	Segment ID (usually 0 for single seq)

Subword Handling (WordPiece/BPE)

Cybersecurity texts contain rare technical terms. Subword tokenization breaks these down into meaningful components.

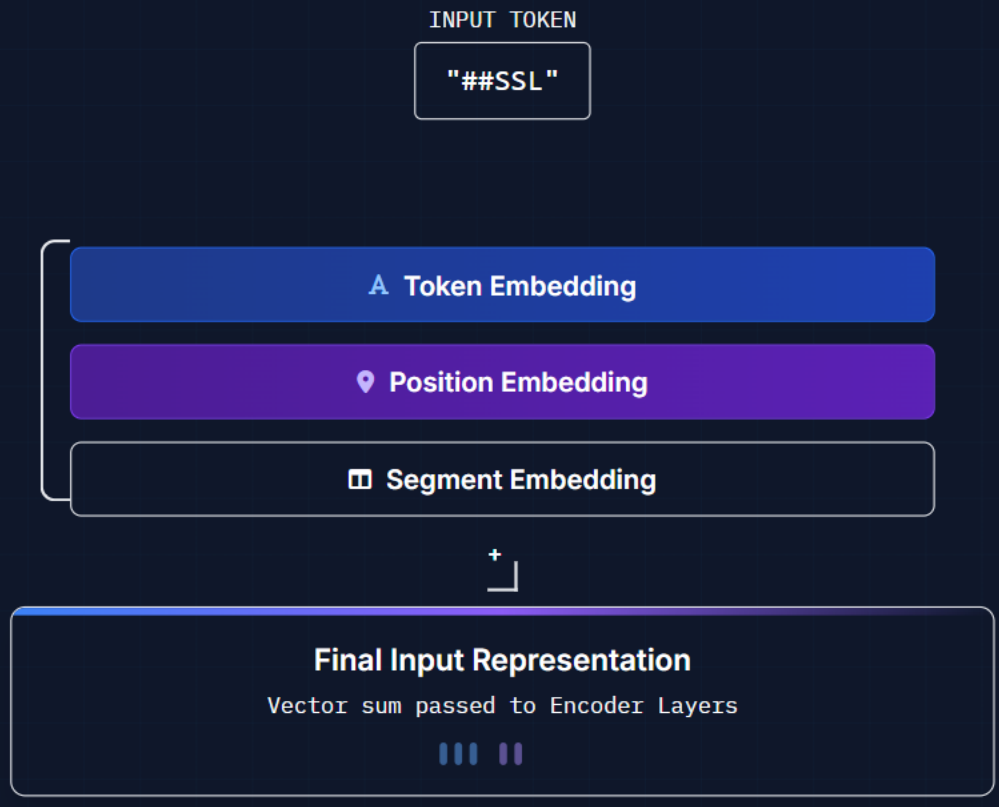
EXAMPLE: "OPENSSL VULNERABILITY"

`Open` `##SSL` `vulnerability`

Token 1 Token 2 Token 3

Label Alignment: BIO tags are expanded to cover all sub-tokens (e.g., B-PRODUCT, I-PRODUCT).

Embedding Architecture



Encoder & Classification Head

MODEL ARCHITECTURE

Transformer Encoder Layers

The core processing unit consists of stacked transformer layers designed to capture deep contextual relationships within technical cybersecurity text.

LAYER COMPONENTS

- ✓ Multi-head Self-Attention
- ✓ Feed-Forward Network
- ✓ Residual Connections
- ✓ Layer Normalization

- **Contextual Mapping:** Self-attention mechanisms enable the model to relate distant tokens, such as linking a product name "OpenSSL" at the start of a sentence to a version number "1.0.2g" at the end.
- **Domain Adaptation:** Crucial for disambiguating terms like "exploit" (noun vs. verb) or recognizing "buffer overflow" as a specific vulnerability type (CWE).

Token Classification Head

The final stage projects the high-dimensional hidden states into specific entity class probabilities for every token in the sequence.

Linear Projection

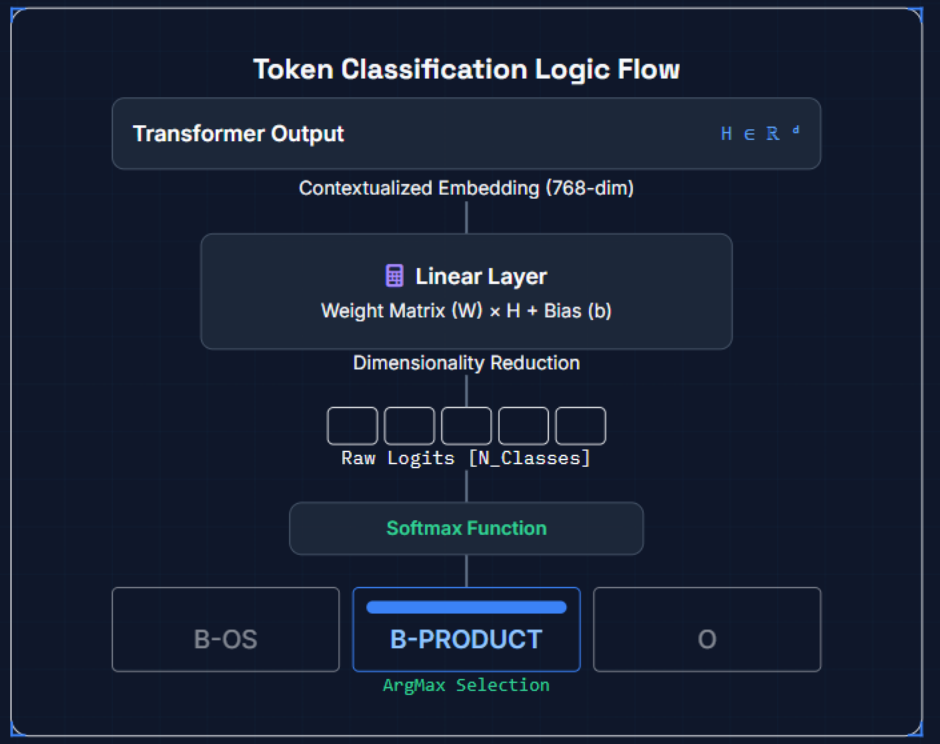
`Logits = Linear(H_token)`

Projects 768-dim hidden state → num_labels dimension.

Softmax Activation

`P(class | token) = Softmax(Logits)`

Outputs probability distribution across all BIO tags (B-OS, I-PRODUCT, O, etc.).



SECTION 5.0

Training Pipeline Overview

END-TO-END FLOW

STEP 01

Preprocessing

- Tokenize using WordPiece
- Align BIO labels to sub-tokens
- Pad/truncate to `max_len=256`

STEP 02

Batching

- Random shuffle dataset
- Create mini-batches
- Batch size: 8–16

STEP 03

Optimization

- Optimizer: `AdamW`
- LR: `2e-5` with warmup
- Epochs: 3–6

STEP 04

Loss Function

- Token-level `Cross Entropy`
- Mask padding tokens
- Weighted for imbalance

RAW DATA ------→ [TENSOR TRANSFORMATION] ------→ UPDATED WEIGHTS

SECTION 5.2

Training Hyperparameters

MODEL CONFIGURATION

PARAMETER	VALUE / SETTING	DESCRIPTION & CONTEXT
Base Model	`DeBERTa-v3-base`	Pre-trained transformer foundation optimized for NLU tasks.
Max Length	`256 tokens`	Padding/truncation limit covering most CVE descriptions.
Batch Size	`8 - 16`	Dependent on GPU VRAM availability (typically 16GB+).
Optimizer	`AdamW`	Adam with weight decay fix; standard for transformers.
Learning Rate	`2e-5`	Low rate to prevent catastrophic forgetting during fine-tuning.
Epochs	`3 - 6`	Early stopping applied if validation loss plateaus.
Scheduler	`Linear Warmup`	Gradual LR increase for first 10% steps, then linear decay.
Loss Function	`Cross Entropy`	Token-level loss calculation with padding masks applied.

Compute Profile

Training was performed on single-GPU instances. The `DeBERTa-v3-base` model (184M params) fits comfortably in standard 16GB VRAM GPUs with batch size 16.

VRAM USAGE ~11.2 GB

Optimization Strategy

The Linear Warmup scheduler is critical for stability. It prevents large gradient updates at the start when weights are still adjusting from pre-training distribution to the NER task.

Warmup Steps ~100-200

Class Imbalance & Metrics

The Imbalance Challenge

Cybersecurity entities are naturally skewed. Common entities like **OS** (e.g., Windows) appear frequently, while critical but specific entities like **EFFECT** (e.g., privilege escalation) are much rarer.

`HIGH: PRODUCT` `LOW: EFFECT`

Mitigation Strategies

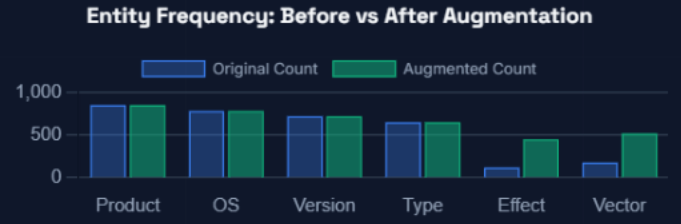
- **Weighted Loss:** Assign higher penalty weights to scarcity classes during backpropagation.
- **Oversampling:** Artificially increase frequency of rare entities in training batches.
- **Augmentation:** Generate synthetic sentences specifically for underrepresented classes.

Evaluation Metrics

Standard accuracy is misleading for imbalanced data. We focus on entity-level performance:

PRECISION Correctness of positive predictions	**RECALL** Coverage of actual positive cases
F1-SCORE Harmonic mean of Precision & Recall	**SPAN ACCURACY** Exact match of full entity boundaries

DISTRIBUTION_VISUALIZATION



WEIGHT ADJUSTMENT LOGIC

Loss(Product)	x 1.0
Loss(Effect)	x 2.5
Loss(Attack_Vec)	x 1.8

SECTION 6.0

Inference Pipeline

REAL-TIME EXTRACTION

STEP 01

Tokenization

- Convert raw CVE text to tokens
- Handle subwords (Open ##SSL)
- Add [CLS], [SEP] specials

STEP 02

Forward Pass

- DeBERTa transformer layers
- Apply disentangled attention
- Output logits [N × Labels]

STEP 03

Softmax

- Calculate class probabilities
- Select highest prob label
- Generate raw BIO tags

STEP 04

Grouping

- Merge consecutive labels
- Combine B-Tag with I-Tags
- Reconstruct word spans

STEP 05

Post-Process

- Remove invalid overlaps
- Filter low-confidence
- Format to final JSON

RAW TEXT → [TOKENS] → [LOGITS] → { ENTITIES } → JSON OUTPUT

SECTION 6.2

Inference Example

LIVE TRANSFORMATION

1 Raw Input Text

"A buffer overflow in OpenSSL on Ubuntu 18.04 allows attackers to execute arbitrary code."

2 Token Classification (Model Output)

A	buffer B-VULN	overflow I-VULN	in	Open B-PROD	##SSL I-PROD	on	Ubuntu B-OS
18 I-OS	. I-OS	04 I-OS	allows	attackers	to	execute B-EFF	
arbitrary I-EFF	code I-EFF	.					

PRODUCT OS VULN_TYPE EFFECT

3 Structured JSON Output

JSON Format

```
[
  {
    "entity": "PRODUCT",
    "value": "OpenSSL",
    "confidence": 0.9942
  },
  {
    "entity": "OS",
    "value": "Ubuntu 18.04",
    "confidence": 0.9876
  },
  {
    "entity": "VULN_TYPE",
    "value": "buffer overflow",
    "confidence": 0.9631
  },
  {
    "entity": "EFFECT",
    "value": "execute arbitrary code",
    "confidence": 0.9205
  }
]
```

Post-Processing Applied:
Sub-tokens like "Open" + "##SSL" merged. "Ubuntu" + "18" + "." + "04" grouped into single OS entity.

Frontend Applications

Streamlit Implementation

LOCAL DEV

Designed for rapid local execution and development testing. Provides a lightweight interface for direct model interaction.

- Simple text box input mechanism
- Clear entities table visualization
- Automatic entity grouping display
- Easy local deployment

Gradio Interface

PRODUCTION

Polished web interface suitable for public demonstrations and HuggingFace Spaces deployment. Features enhanced responsiveness.

- Live inference updates
- Fast prototyping capabilities
- Built-in API endpoint generation
- Seamless HuggingFace integration

UI_PREVIEW.exe

DEPLOYED

NIST Vulnerability Extractor

INPUT CVE DESCRIPTION

A buffer overflow vulnerability in OpenSSL 1.0.2 allows remote attackers to execute arbitrary code via...

EXTRACT ENTITIES

EXTRACTED ENTITIES

Entity Type	Value
PRODUCT	OpenSSL
VERSION	1.0.2
VULN_TYPE	Buffer Overflow
EFFECT	Execute Code

SECTION 8.0

Challenges & Solutions

ISSUE RESOLUTION

Pipeline Import Failure

Runtime error when initializing the Hugging Face pipeline, causing `ImportError: cannot import name 'pipeline'`.

```
ModuleNotFoundError: No module named 'transformers.pipelines'
```

Version Compatibility Fix

Identified version mismatch in the environment. Downgrading to a stable compatible version resolved the dependency conflict.

```
pip install transformers==4.53
```

Meta Tensor Error

Model weights failed to load correctly when initializing from cache, resulting in tensor shape mismatch or empty weights.

```
RuntimeError: Expected all tensors to be on the same device
```

Local Weight Loading

Forced loading from local directory to ensure complete weight initialization and correct device placement.

```
from_pretrained(local_dir, local_files_only=True)
```

Niche Product Detection

Low recall on obscure security tools (e.g., "OpenVAS", "Webmin") due to limited representation in training data.

Recall < 60%

Subword Augmentation

Implemented data augmentation with synthetic CVEs and leveraged DeBERTa's subword tokenization to generalize unseen terms.

F1 Score > 85%

Current System Limitations

Uncommon Vendor Products

The model occasionally misses niche or legacy vendor products not present in the augmented training set. While common vendors (Microsoft, Adobe, Cisco) are well-recognized, specialized industrial or obscure software may be overlooked.

Effect Wording Variance

Impact effects ("execute arbitrary code", "gain privilege") have high linguistic variance. Normalizing these extracted phrases into standard taxonomy remains challenging due to the diverse ways vulnerability impacts are described by different analysts.

Multi-Sentence Complexity

Performance degrades with long, multi-sentence CVE descriptions where context is separated by significant distance. Cross-sentence dependencies (referring to "the software" in the second sentence) can lead to entity resolution gaps.

Dataset Distribution

Although augmented, CyNER2.0 represents only a fraction of the total real-world CVE distribution. The model's generalization to completely novel attack vectors or emerging technologies (like new blockchain protocols) is limited by current training data.

PERFORMANCE_BOTTLENECKS

ERROR FREQUENCY DISTRIBUTION

Rare Products	12%
Complex Syntax	8%
Context Loss	5%

CONSTRAINT IMPACT MATRIX

High RECALL ON NICHE SOFTWARE	**Med** LONG-TEXT COHERENCE
Low STANDARD OS DETECTION	**None** INFERENCE SPEED

Current limitations primarily affect edge cases. Core functionality remains robust for >90% of standard NVD entries.

SECTION 10.0 & 11.0

Future Roadmap & Conclusion

PROJECT COMPLETE

FUTURE ENHANCEMENTS

Relation Extraction

Link extracted entities to understand context (e.g., explicitly connecting "OpenSSL" to "Ubuntu 18.04").

CVSS Score Prediction

Add a classification head to predict vulnerability severity scores based on the description text.

Larger Corpora Integration

Expand training data with ExploitDB, Microsoft Security Advisories, and vendor disclosures.

Production Deployment

Containerize inference pipeline using Docker and expose via high-performance FastAPI endpoints.

PROJECT SUMMARY

NIST Vulnerability Extractor

FULL-FLEDGED NLP SYSTEM

This project successfully demonstrates the application of modern Transformer-based NLP to the high-stakes domain of cybersecurity.

- **MODEL**
  DeBERTa-v3 Transformer Architecture
- **DATASET**
  CyNER2.0 Augmented Custom Dataset
- **INTERFACE**
  Streamlit & Gradio Frontends

READY FOR INTEGRATION

SIEM / SOC / THREAT INTEL