USING AI FOR PENETRATION TESTING

LINNEA GILBERTSON

PENETRATION TESTING (PENTESTING)

- Mimics real world hacker attack
- Identifies security vulnerabilities
- Pentesting tools:
  - Find or analyze vulnerabilities
  - Help perform exploitation

WHY

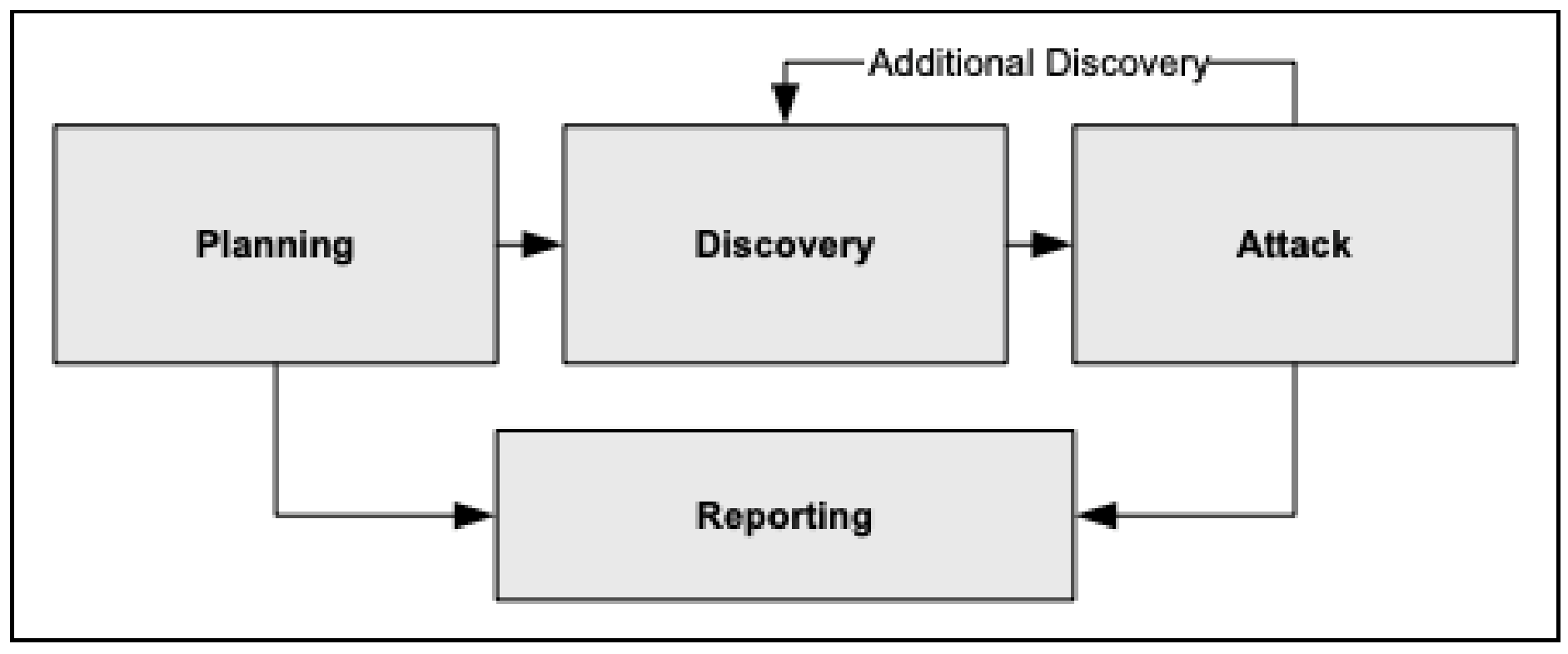
- Combines deep reinforcement learning (DRL) and a large language mode (LLM) to:
  - Reduce false positive rates
  - Save pentester time by:
    - Reducing needed database searching for vulnerabilities
    - Suggest exploitation routes for proof of concept tests

OUTLINE

- Penetration Testing Background
- BERT QA RL + RS Model Overview
- Large Language Model and BERT Background
- BERT QA
- Deep Reinforcement Learning Background
- Deep Reinforcement Learning Agent
- Results
- Conclusion

BLACK-BOX PENTESTING

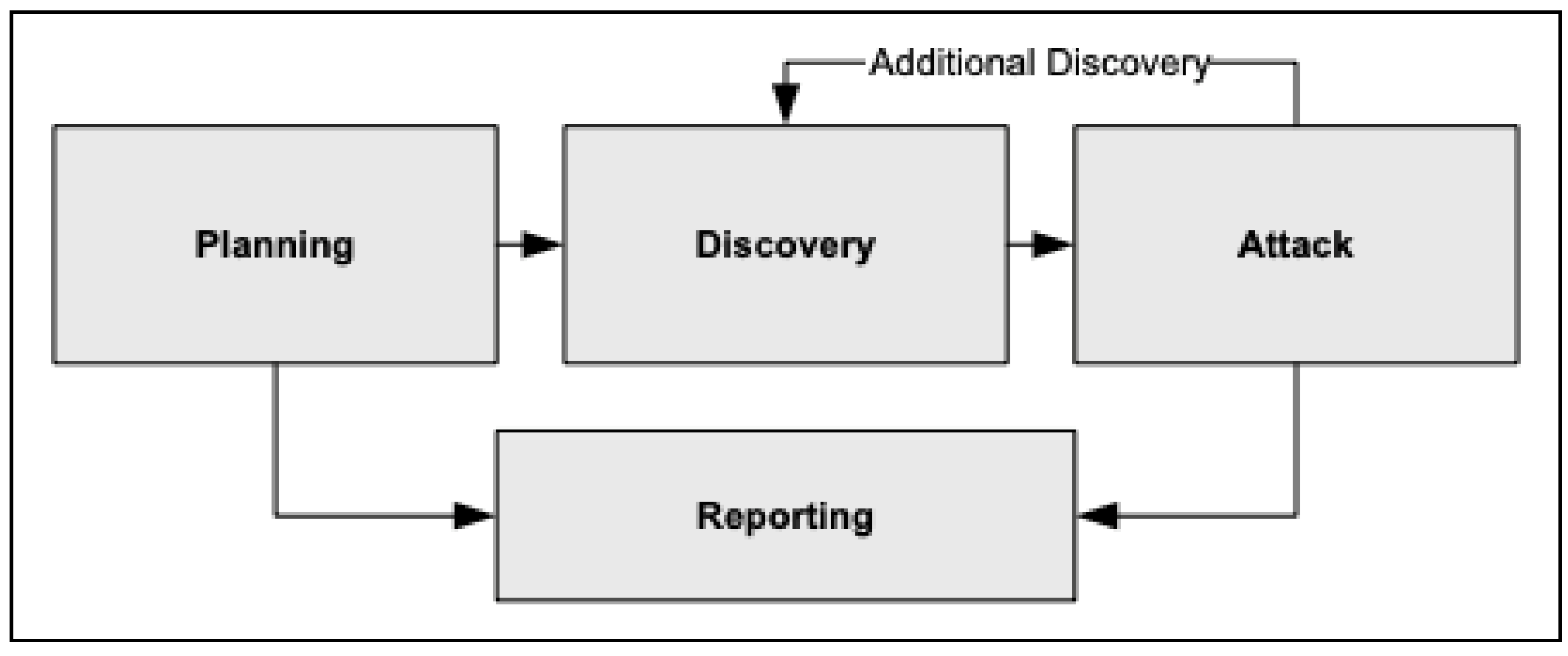
- Pentesting without prior knowledge of system structure or source code



VULNERABILITY ANALYSIS DATABASES

- Common Vulnerability and Exposures (CVE)
  - List of security vulnerabilities
- Common Weakness Enumeration (CWE)
  - List of underlying weaknesses
  - Links to CVEs

EXPLOITATION PHASE



LARGE LANGUAGE MODEL

- Trained on large amounts of data
- Generate language response to prompt and/or context
- Able to do this thanks to billions of internal parameters
- Enables them to capture patterns in language
- Create a probability distribution for likely words

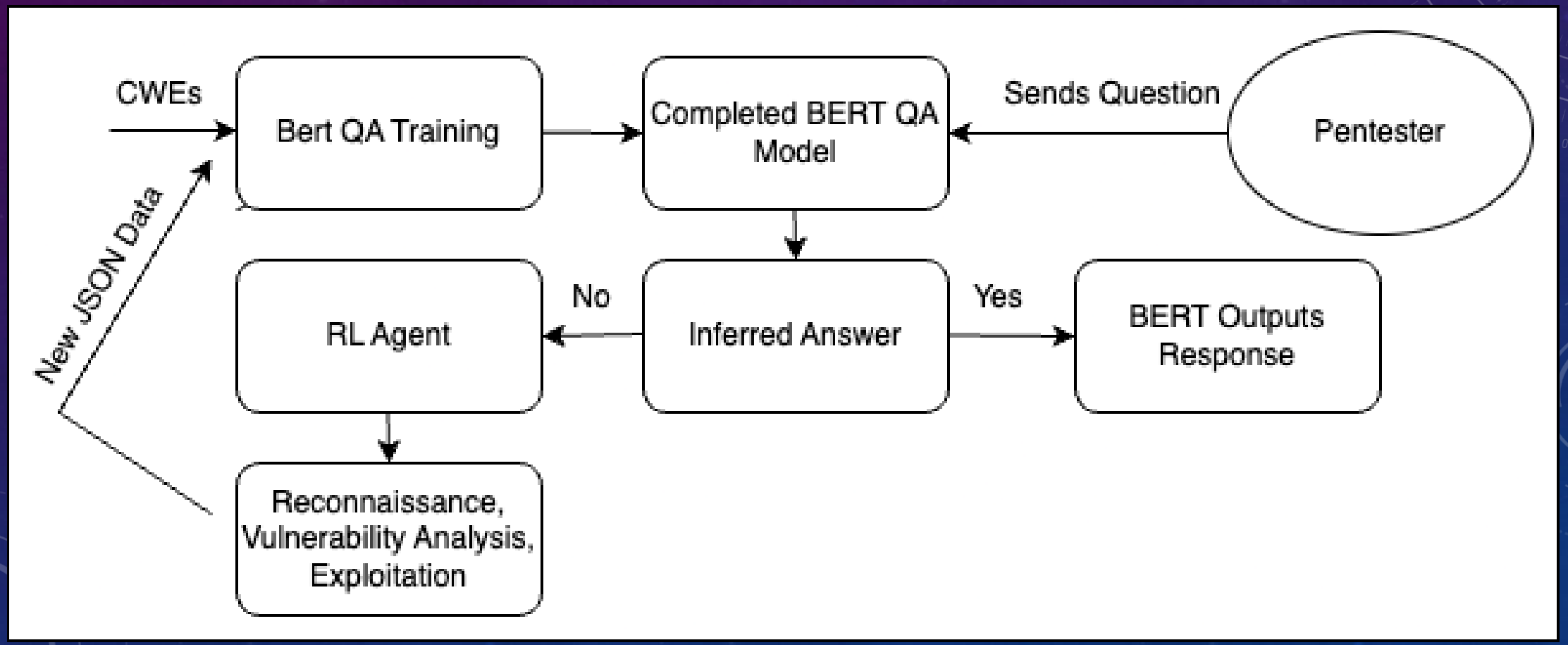
BERT BACKGROUND

- Bidirectional Encoder Representations from Transformers (BERT)
- Masked language model
- Easily fine tunable

DEEP REINFORCEMENT LEARNING BACKGROUND

- Combines reinforcement learning and deep neural networks
- Referred to as the reinforcement learning (RL) agent

MODEL OVERVIEW



BERT QA

- Query:
  - Consist of pentester goal and knowledge of target systems architecture
- Answer:
  - Response output by BERT QA, can identify vulnerabilities, give penetration test suggestions, ect.
- Context:
  - Where the information from the outputted answer came from

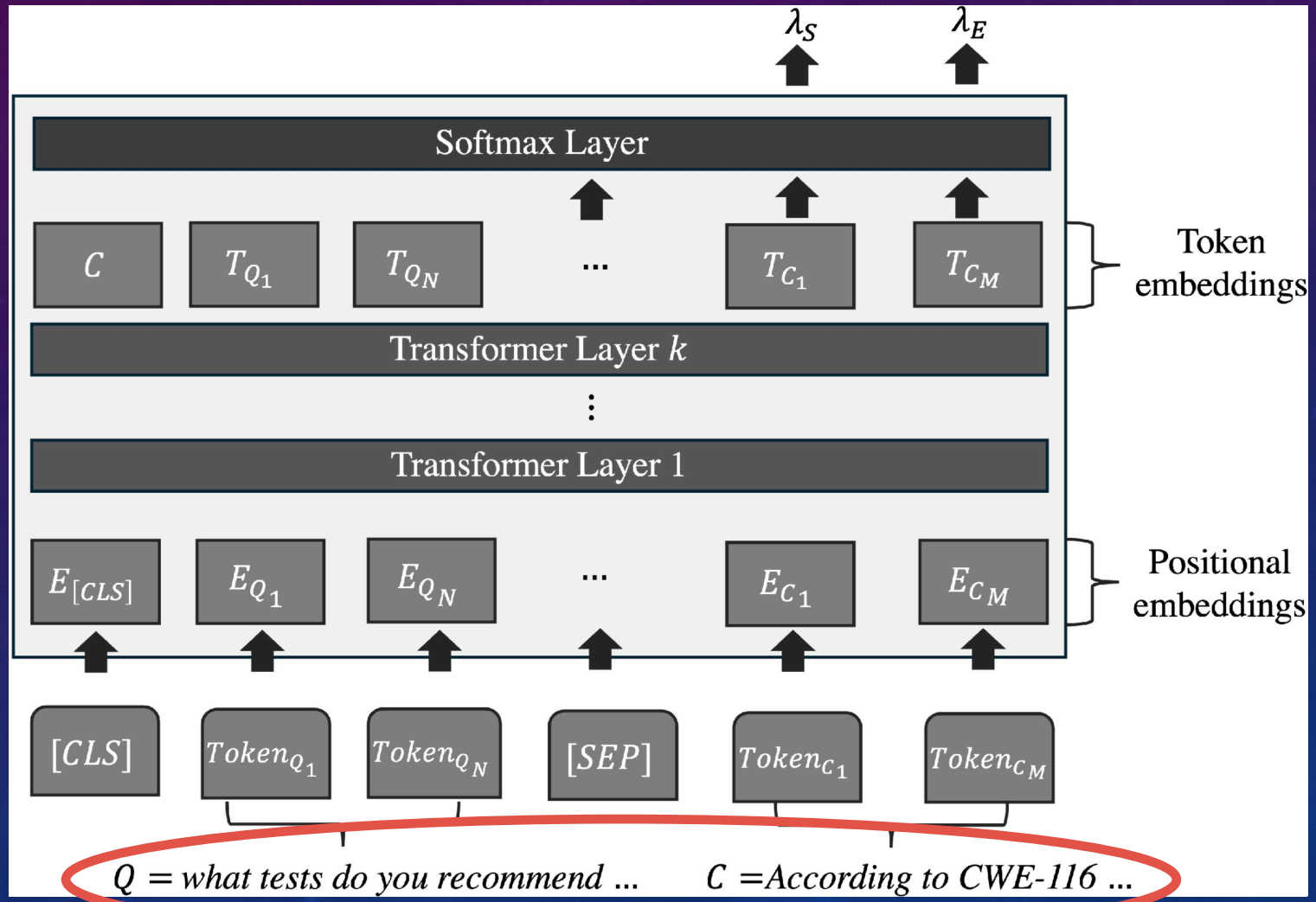
EXAMPLE QUESTION, CONTEXT, AND ANSWER

Question: What tests do you recommend for a Class C IP address, with Ubuntu 20.0 operating system, running an Apache PHP 5.2.4 server?

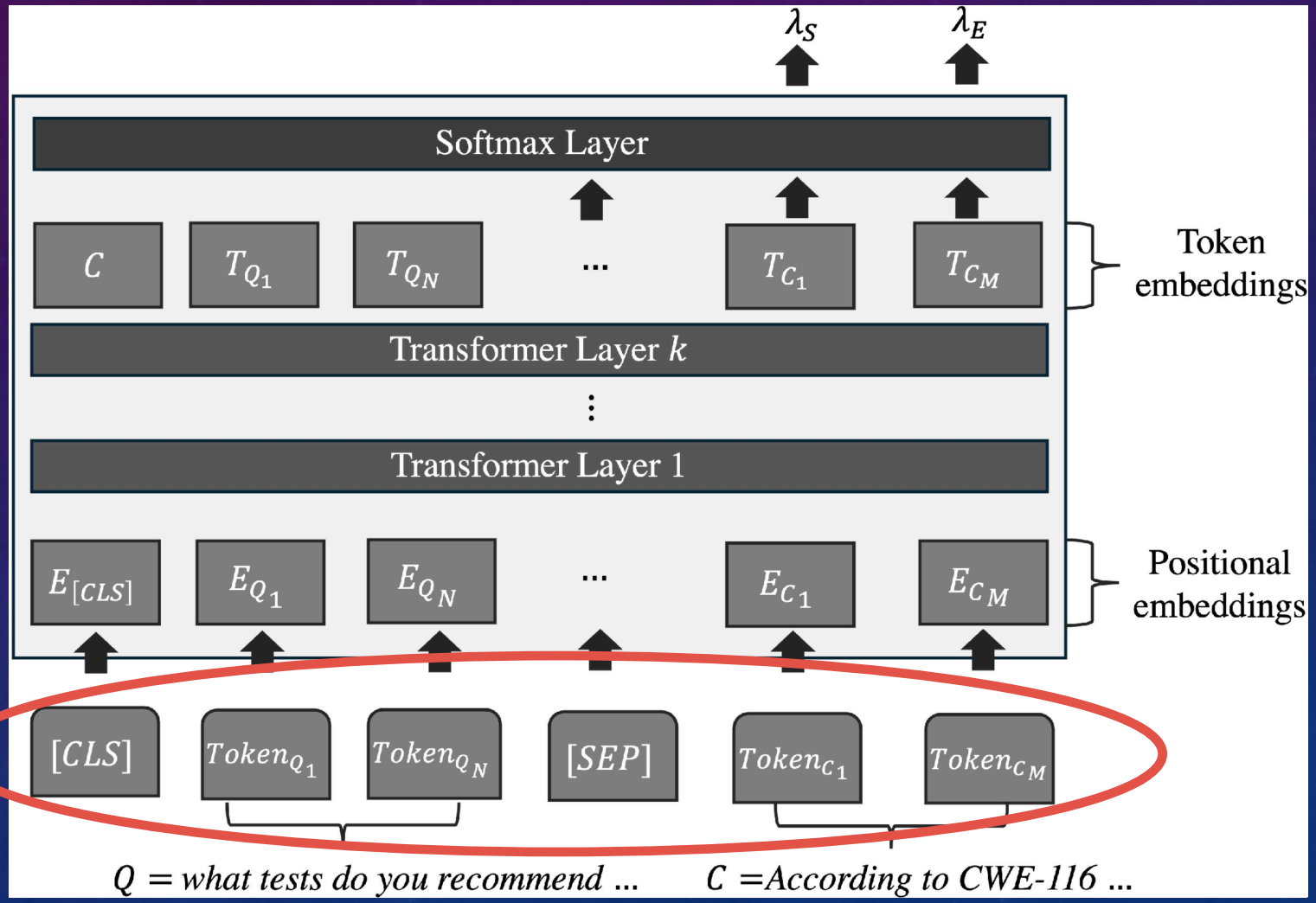
Context: According to CWE-116, CWE-79, and CWE-94, with improper neutralization of resource inputs, enabling potential remote code execution,

Answer: it is possible to use a proof of concept for XSS and then inject arbitrary code by modifying functions.lib.php.

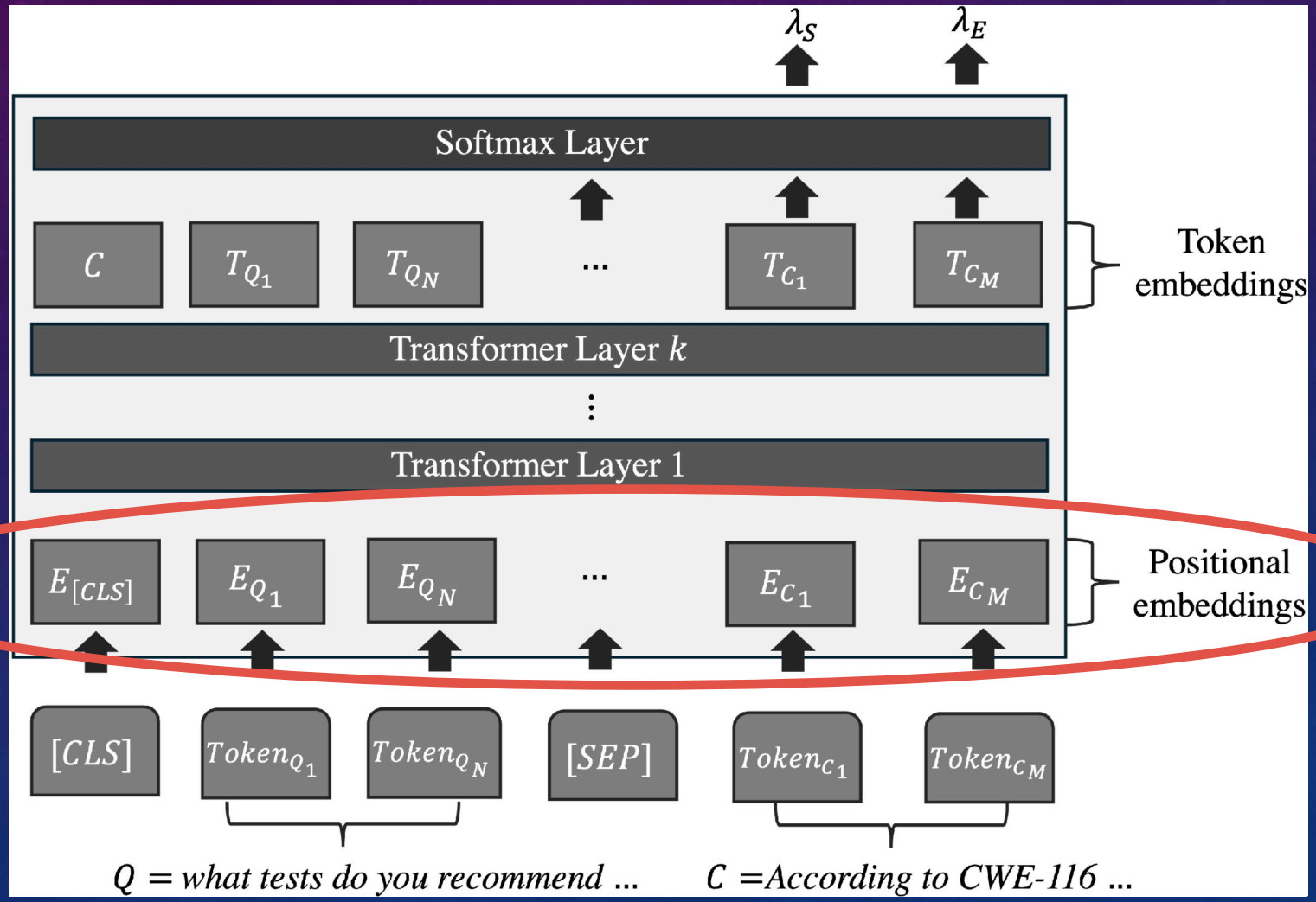
BERT QA ARCHITECTURE INPUTS



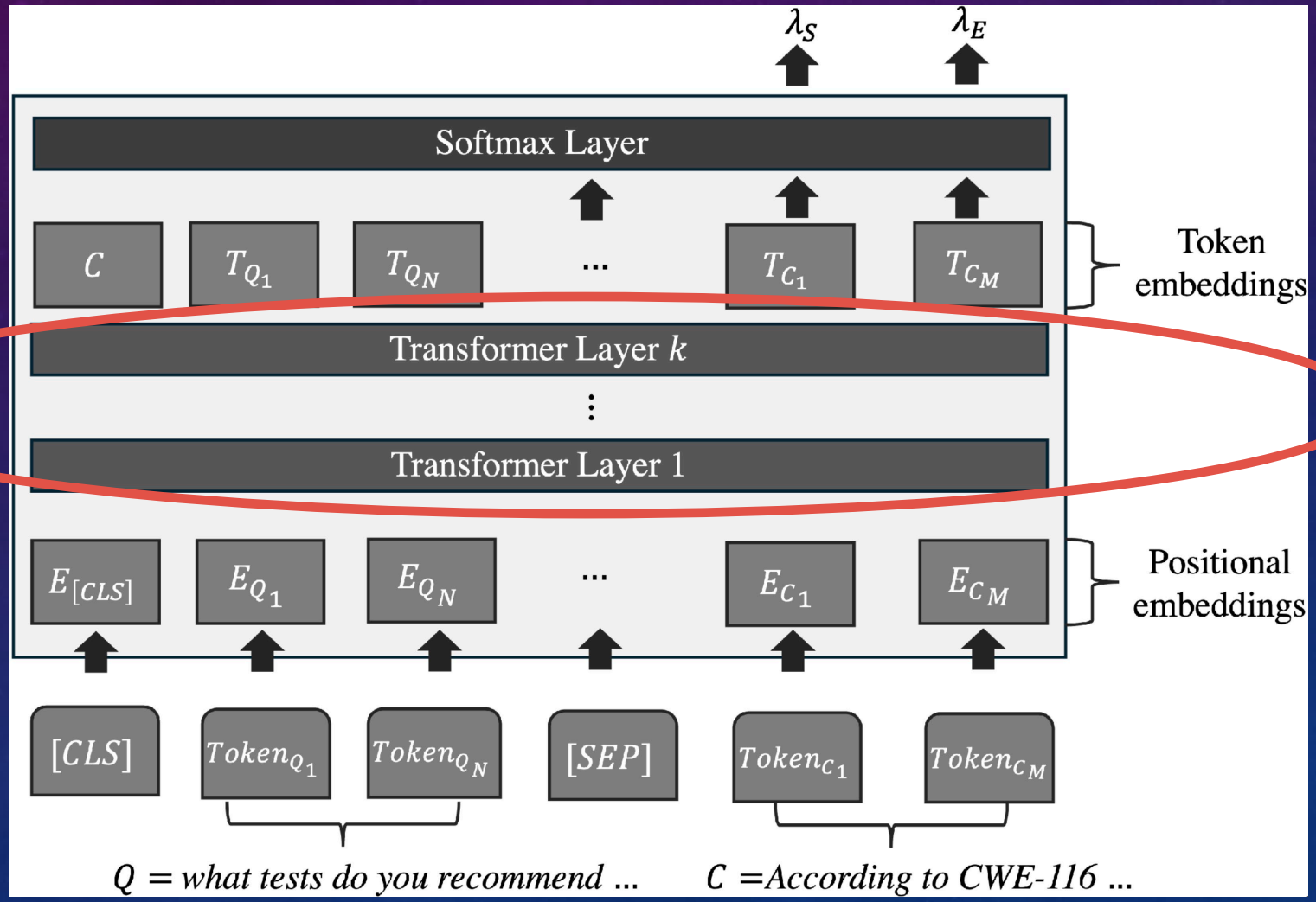
TOKENS


λ_S
λ_E
Softmax Layer
C
T_{Q_1}
T_{Q_N}
...
T_{C_1}
T_{C_M}
Token embeddings
Transformer Layer k
Transformer Layer 1
$E_{[CLS]}$
E_{Q_1}
E_{Q_N}
...
E_{C_1}
E_{C_M}
Positional embeddings
[CLS]
$Token_{Q_1}$
$Token_{Q_N}$
[SEP]
$Token_{C_1}$
$Token_{C_M}$
Q = what tests do you recommend ...
C =According to CWE-116 ...

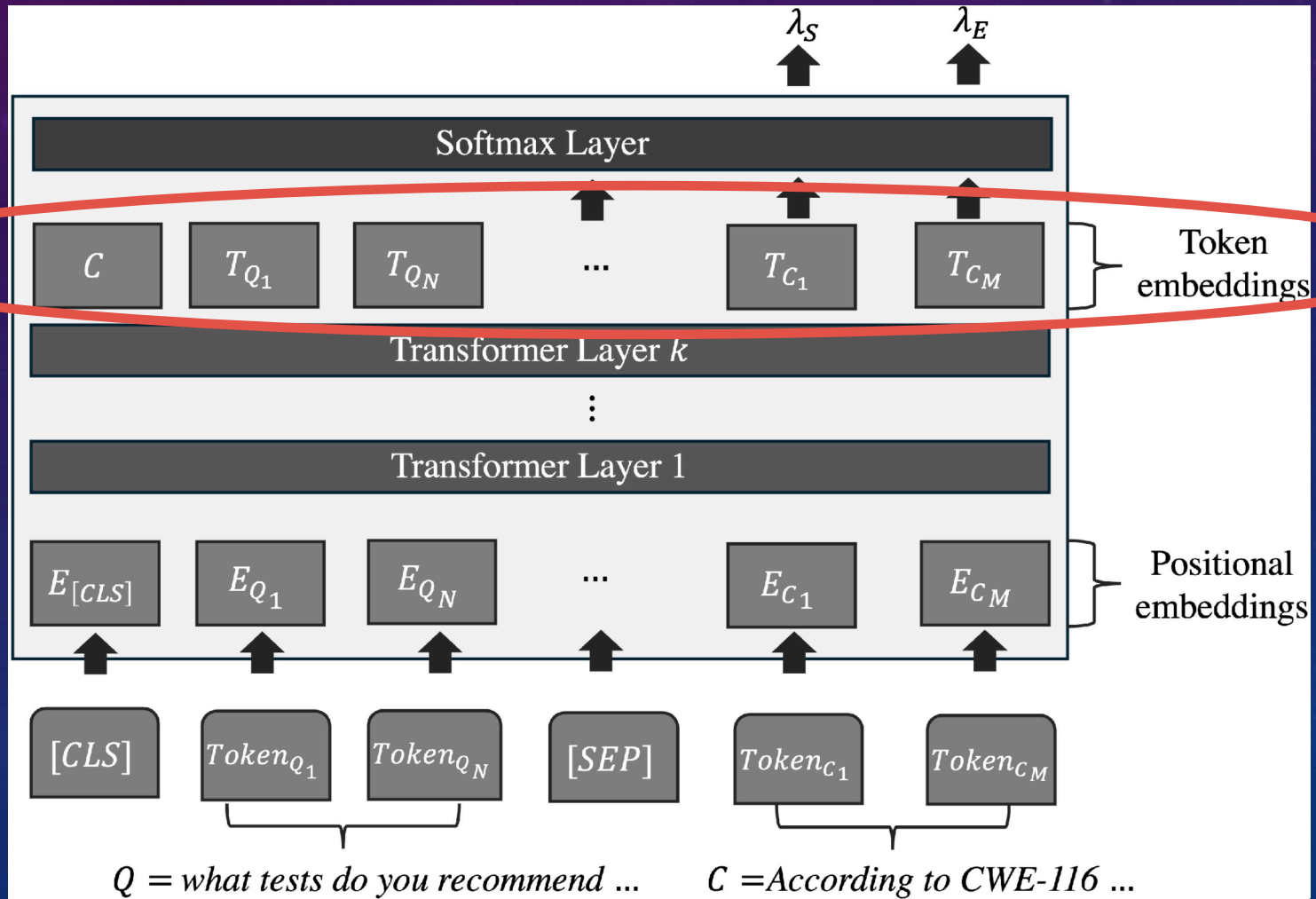
EMBEDDINGS


λ_S
λ_E
Softmax Layer
C
T_{Q_1}
T_{Q_N}
...
T_{C_1}
T_{C_M}
Token embeddings
Transformer Layer k
Transformer Layer 1
$E_{[CLS]}$
E_{Q_1}
E_{Q_N}
...
E_{C_1}
E_{C_M}
Positional embeddings
[CLS]
$Token_{Q_1}$
$Token_{Q_N}$
[SEP]
$Token_{C_1}$
$Token_{C_M}$
Q = what tests do you recommend ...
C =According to CWE-116 ...

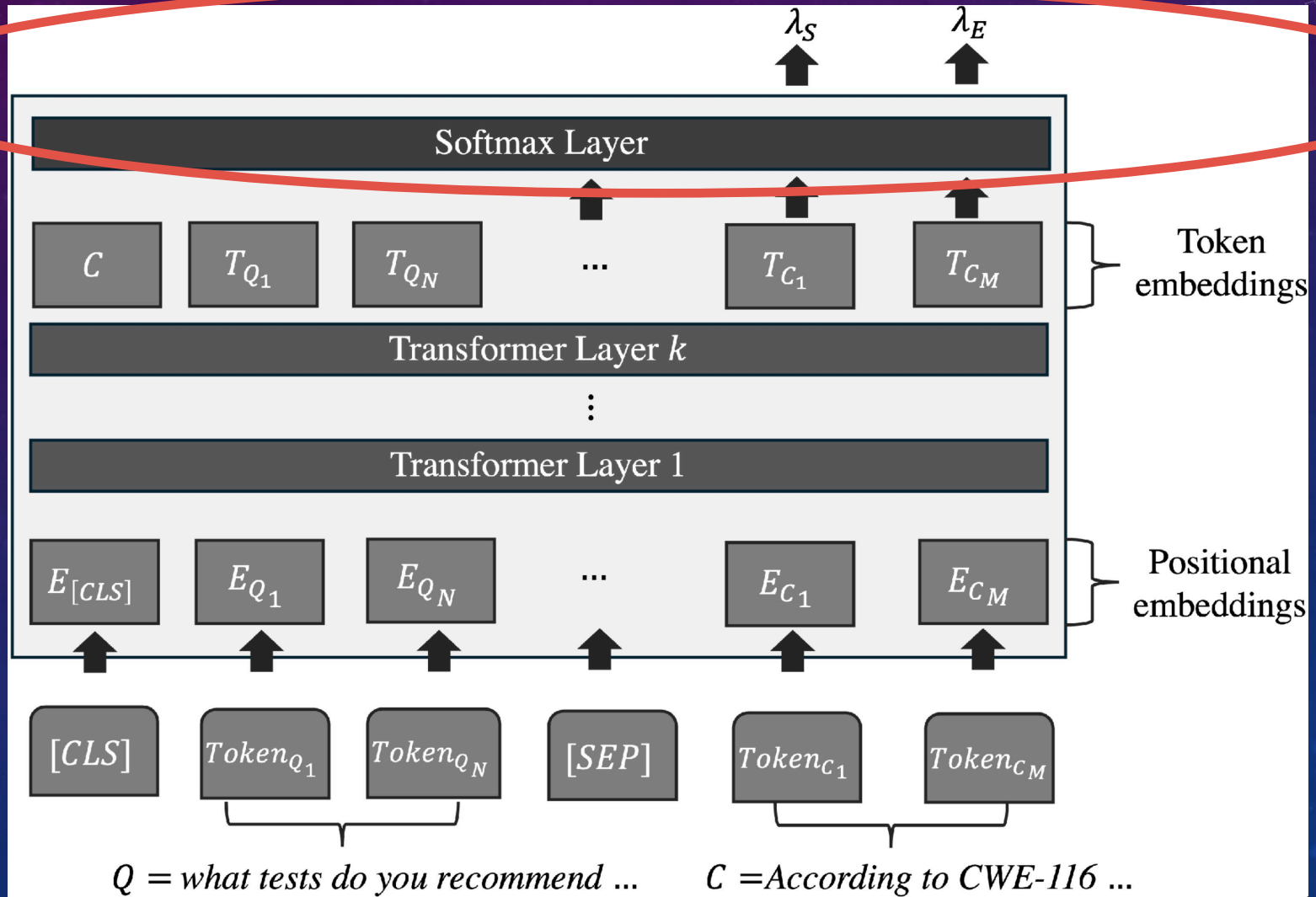
FEED-FORWARD NEURAL NETWORK AND MULTI-HEAD ATTENTION



TOKEN EMBEDDINGS



SOFTMAX


λ_S
λ_E
Softmax Layer
C
T_{Q_1}
T_{Q_N}
...
T_{C_1}
T_{C_M}
Token embeddings
Transformer Layer k
Transformer Layer 1
$E_{[CLS]}$
E_{Q_1}
E_{Q_N}
...
E_{C_1}
E_{C_M}
Positional embeddings
[CLS]
$Token_{Q_1}$
$Token_{Q_N}$
[SEP]
$Token_{C_1}$
$Token_{C_M}$
Q = what tests do you recommend ...
C =According to CWE-116 ...

REINFORCEMENT LEARNING BACKGROUND

- RL consists of:
  - An agent, the decision maker
  - An environment, the surrounding system
  - Policy, controls the agent's decisions
  - State, RL agent's location in the environment
  - Action, a possible decision for the RL agent
  - Reward, positive or negative feedback
  - Convergence, changing its policy to get the most reward possible

DEEP REINFORCEMENT LEARNING BACKGROUND

- Basic Reinforcement Learning needs environmental model
  - Not possible in pentesting as number of states is large and changes are common
- Q-learning is model free
  - Instead estimates cumulative future rewards
- Uses deep neural networks
  - Picks up on intricate input to output mappings
  - Able to use in an environment with far more states and actions

DEEP REINFORCEMENT LEARNING STEPS

1. Planning and Preparation:
    - Uses the question to define objectives and scope of the pentest
2. Reconnaissance:
    - Uses Network mapper to identify useful system architecture
3. Vulnerability Analysis:
    - Uses Nmap Vulners to identify potential vulnerabilities
4. Exploitation and Post-Exploitation:
    - Uses Metasploit modules to find attack path and develop exploit code

Q-VALUE

- Multiple iterations of prior steps
- Q value: Highest estimated reward
- Selects most effective learned action or random action
- Rewards for successful actions
- Policy changes if action fails

RL AGENT CONCLUSION

- Concludes either when:
  - Max Q-value reached for each step
  - Agent can’t make progress towards max Q-Value
- Highest reward state action pairs assembled into JSON data set
- Sent to BERT QA
- Which begins a new round of BERT QA training

BERT MODELS TESTED

- BERT Uncased:
  - First most basic BERT model
- RoBERTa:
  - Uses a larger dataset and uses slightly different masking approach
- DistilBERT:
  - Lightweight BERT with lower computational cost

TRAINING TIME AND LOSS

- Training Loss: difference between BERTs actual output and expected output for all answers

$$L = 1/N * \sum(A - \text{Expected A})$$

Model	Loss	Training (min)
BERT uncased	0.0001	1297.5
RoBERTa	0.0000	1299.8
DistilBERT	0.0043	689.1

QA ACCURACY RESULTS

- Precision: $$Precision = \frac{\text{Number of correct words in response}}{\text{total words in response}} * 100.$$
- Recall: $$Recall = \frac{\text{Number of correct words in response}}{\text{total words in expected response}} * 100.$$
- Exact Match: $$\text{Exact match} = \frac{\text{Number of correct responses}}{\text{Total responses}} * 100.$$

Model	Exact Match (%)	Precision (%)	Recall (%)
BERT uncased	97.5%	98.0904%	98.4848%
RoBERTa	99.9998%	99.9999%	99.9998%
DistilBERT	99.8763%	99.9057%	99.8763%

CONCLUSION

- BERT QA RL + RS is a highly accurate and efficient pentesting tool for all stages of black-box pentesting
- But it has only been tested on intentionally vulnerable test environments
- Both execution and training times will likely increase with real world applications
- Future research should:
  - Evaluate how it performs on real world systems
  - With real cybersecurity experts