WordCamp Asia 2025

Future

Decoding Vulnerabilities: Elevating WordPress Security with LLMs

Rashmi Nagpal

Philippine International Convention Center
Manila, Philippines
February 20-22, 2025

Why is WordPress security important?

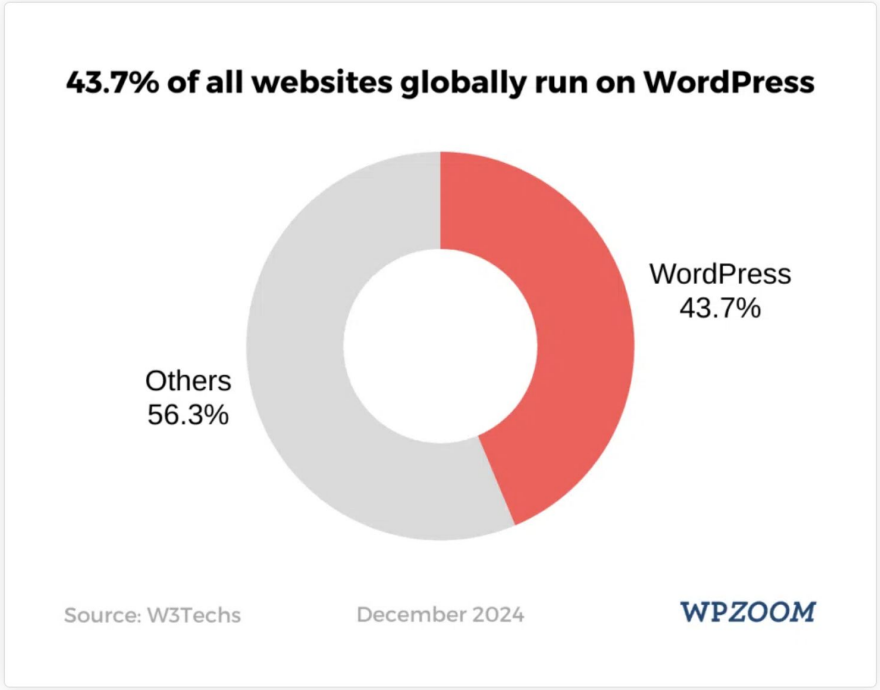


Image source: https://www.wpzoom.com/blog/wordpress-statistics/

60000+

Plugins on the WordPress plugin repository

~4579

Vulnerabilities reported on WordPress ecosystem in 2024

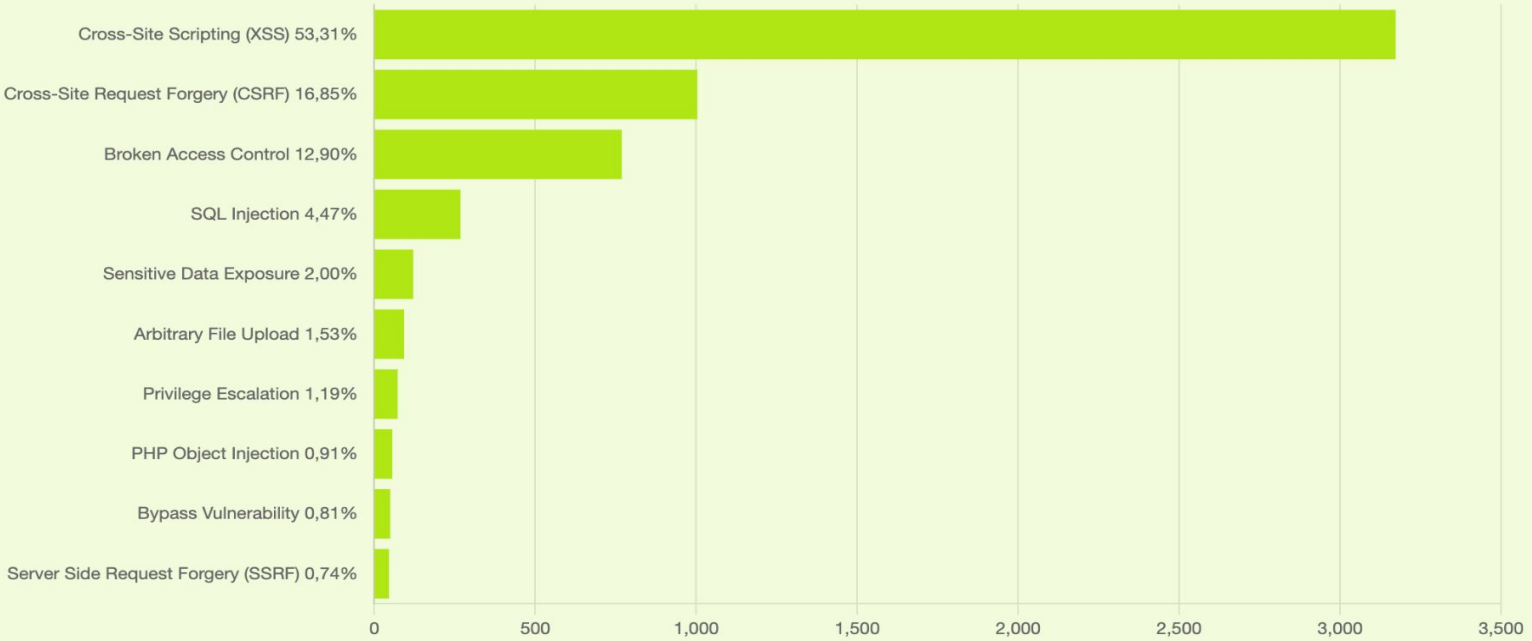
~1600+

Plugins and themes closed (temporary or permanently) in 2024 due to security issues

Cross-Site Scripting (XSS) was the most common vulnerability in WordPress in 2023

patchstack

Image



Source: https://patchstack.com/whitepaper/state-of-wordpress-security-in-2024/

In 2023, plugins were responsible for 96.77% of all new WordPress vulnerabilities

patchstack

Image

Source: https://patchstack.com/whitepaper/state-of-wordpress-security-in-2024/

42,9% of all new WordPress vulnerabilities in 2023 had a high or critical CVSS severity

patchstack

Image

Source: https://patchstack.com/whitepaper/state-of-wordpress-security-in-2024/

Introduction

ARTIFICIAL INTELLIGENCE

Any technique that enables computers to mimic human behavior

MACHINE LEARNING

Ability to learn without explicitly being programmed

DEEP LEARNING

Extract patterns from data using neural networks

3 1 3 5 6 7

1 4 5 9 2 3

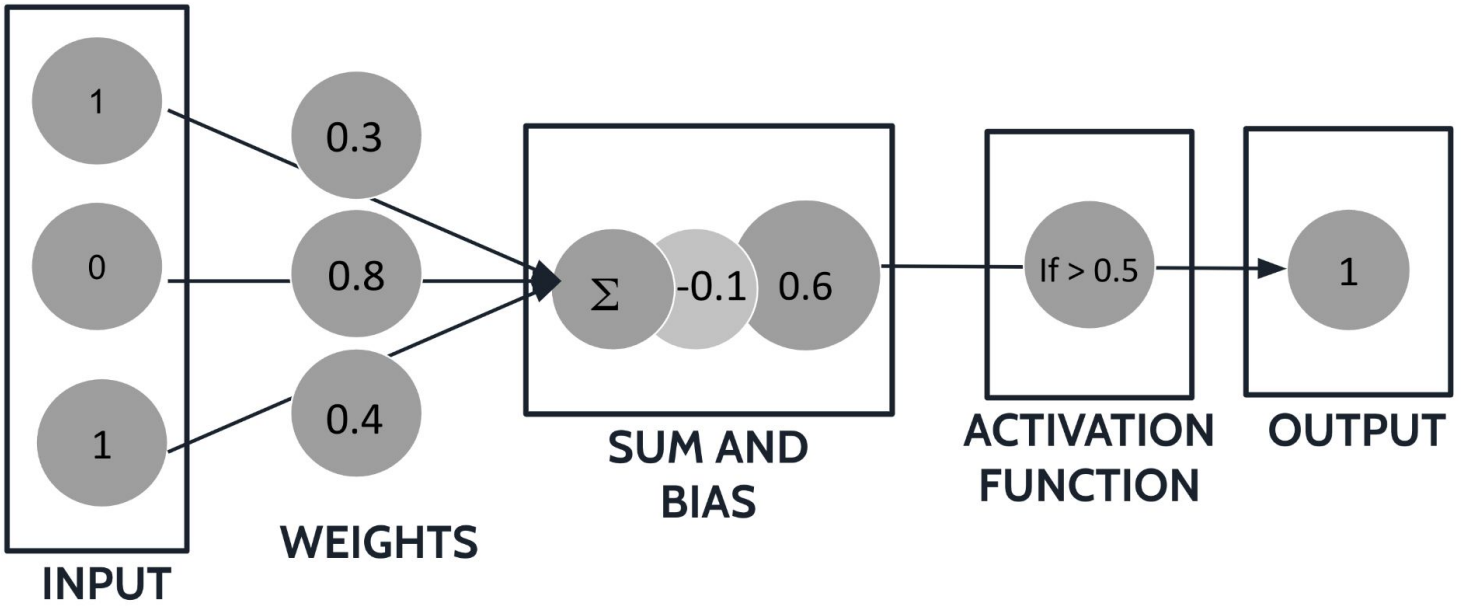
Prompt: which of the following is the response generated by an LLM?

A

B

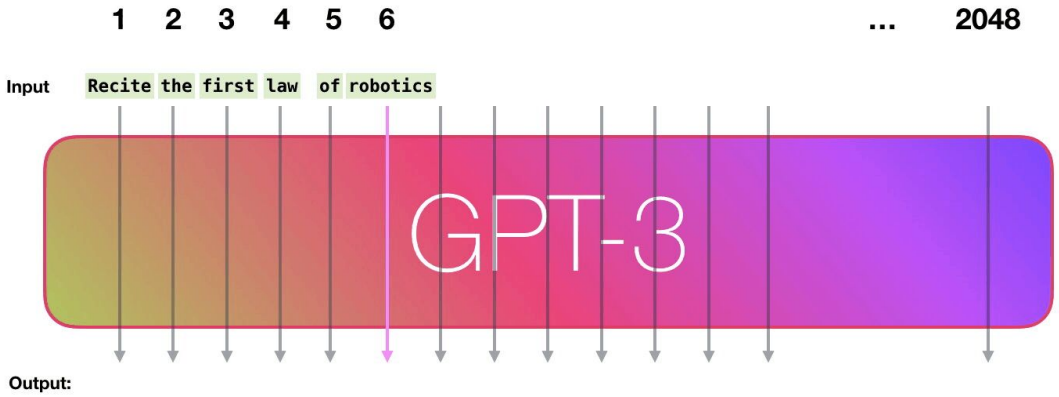
C

Basic Neural Network



Transformer Architecture

Transformer architecture has the ability to scale effectively, allowing the models to train on the massive datasets!



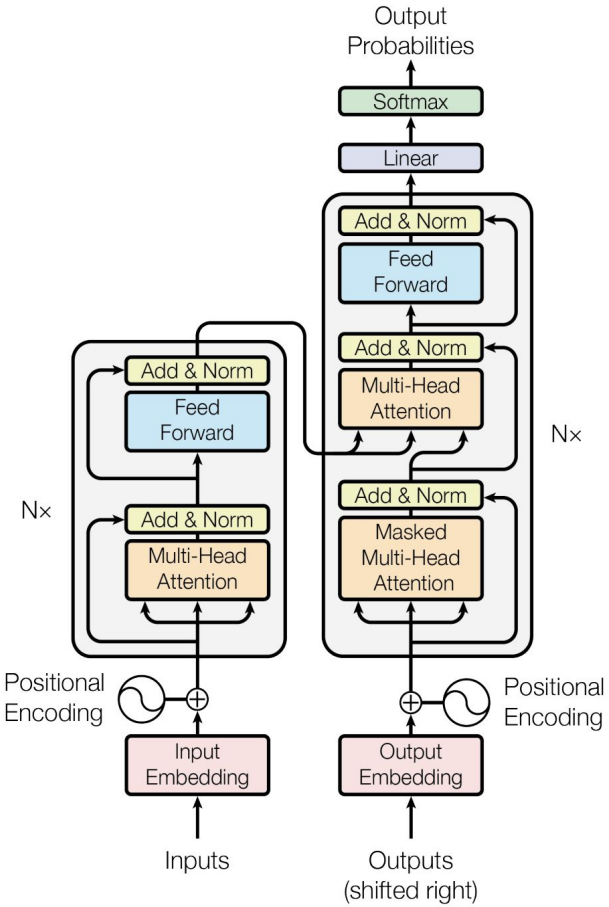
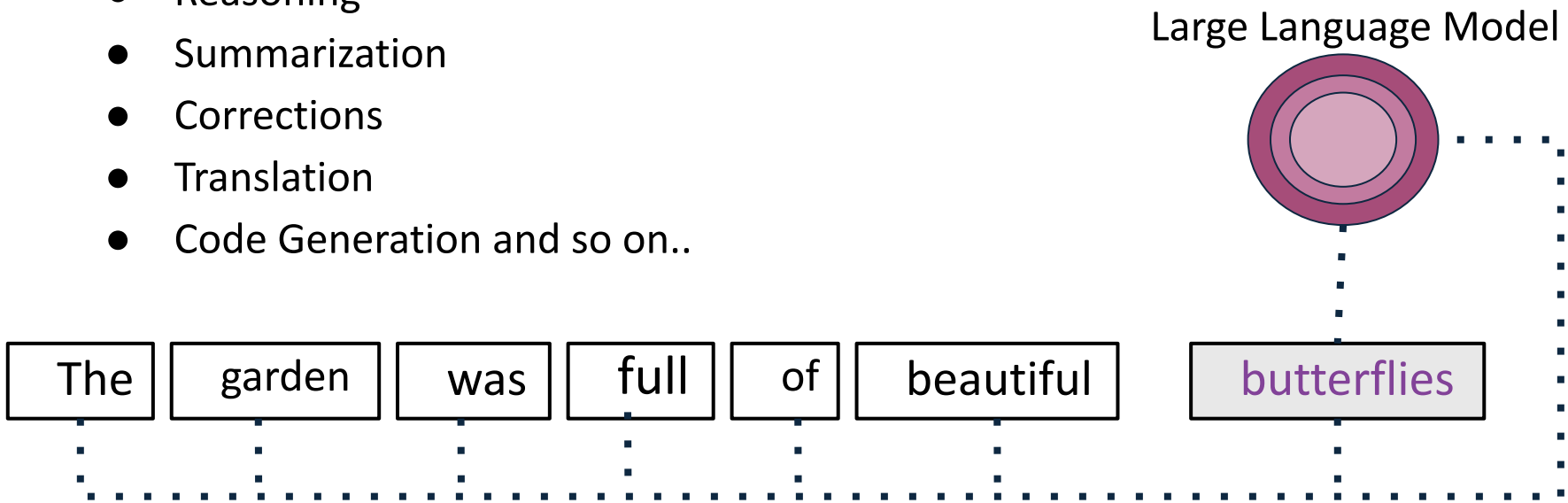


Figure 1: The Transformer - model architecture.

Source: https://jalammar.github.io/illustrated-transformer/

Large Language Models can be used for

- Reasoning
- Summarization
- Corrections
- Translation
- Code Generation and so on..

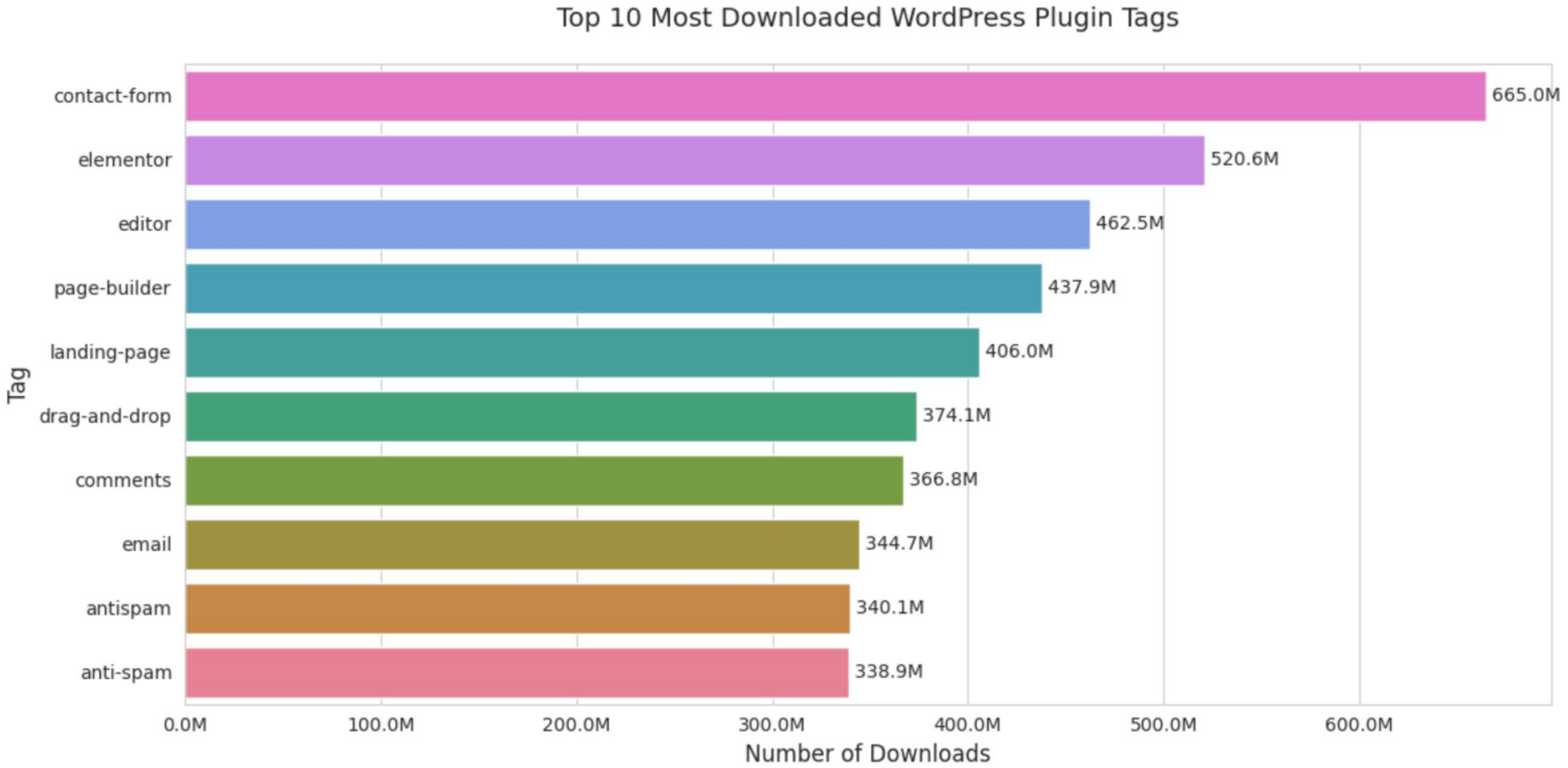


LLM Use-Case: Plugin Quality Assessment

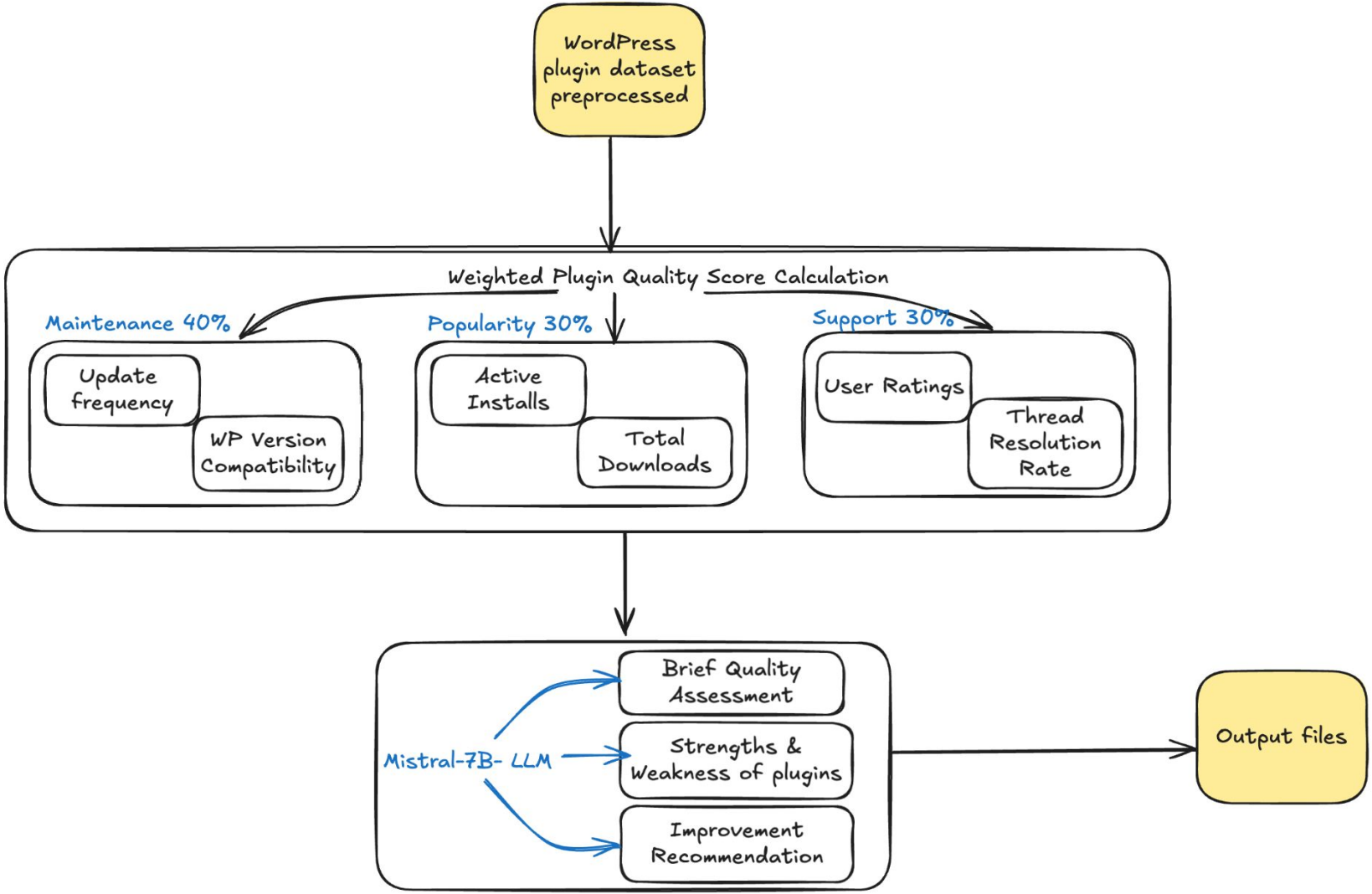
Plugins Dataset (couple of the sample rows) - open source dataset retrieved from Kaggle

A	B	C	D	E	F	G	H	I	J	K	L
slug	tag	version	requires	tested	requires_	rating	num_rati	support_t	support_t	active_installs	downloaded
wp-fullcalendar	ajax-caler	1.5	3.6	6.2.3	False	94	93	1	0	10000	185407
wp-fullcalendar	calendar	1.5	3.6	6.2.3	False	94	93	1	0	10000	185407
wp-fullcalendar	calendars	1.5	3.6	6.2.3	False	94	93	1	0	10000	185407
wp-fullcalendar	event-cale	1.5	3.6	6.2.3	False	94	93	1	0	10000	185407
wp-fullcalendar	jquery-cal	1.5	3.6	6.2.3	False	94	93	1	0	10000	185407
wp-full-screen-	ajax-searc	1.0.0	4.0	5.8.8	5.2	60	1	1	0	70	1197
wp-full-screen-	custom-se	1.0.0	4.0	5.8.8	5.2	60	1	1	0	70	1197
wp-full-screen-	search	1.0.0	4.0	5.8.8	5.2	60	1	1	0	70	1197
wp-full-screen-	search-for	1.0.0	4.0	5.8.8	5.2	60	1	1	0	70	1197
wp-full-screen-	wordpres	1.0.0	4.0	5.8.8	5.2	60	1	1	0	70	1197
wp-fullscreen-p	fulscreen	0.2.0	3.2.0	3.2.1	False	0	0	0	0	0	3865
wp-fullscreen-p	preview	0.2.0	3.2.0	3.2.1	False	0	0	0	0	0	3865
wp-fun	fun	1.0.0	4.3	4.5.30	False	0	0	0	0	0	593

EDA - Identifying most downloaded plugins



Use-case: Identify plugin quality using open-source LLM


WordPress plugin dataset preprocessed
Weighted Plugin Quality Score Calculation
Maintenance 40%
Update frequency
WP Version Compatibility
Popularity 30%
Active Installs
Total Downloads
Support 30%
User Ratings
Thread Resolution Rate
Mistral-7B- LLM
Brief Quality Assessment
Strengths & Weakness of plugins
Improvement Recommendation
Output files

```python
def assess_plugin_quality(df):
    """Perform comprehensive plugin quality assessment."""
    print("Calculating quality scores...")

    df['maintenance_score'] = df.apply(calculate_maintenance_score, axis=1)
    df['popularity_score'] = df.apply(calculate_popularity_score, axis=1)
    df['support_score'] = df.apply(calculate_support_score, axis=1)

    weights = {
        'maintenance_score': 0.4,
        'popularity_score': 0.3,
        'support_score': 0.3
    }

    df['quality_score'] = (
        df['maintenance_score'] * weights['maintenance_score'] +
        df['popularity_score'] * weights['popularity_score'] +
        df['support_score'] * weights['support_score']
    )

    print("Quality assessment completed!")
    return df
```

```python
def setup_mistral():
    """Setup Mistral model and tokenizer."""
    print("Setting up Mistral model...")

    from transformers import AutoTokenizer, AutoModelForCausalLM


    model_name = "mistralai/Mistral-7B-v0.1"
    tokenizer = AutoTokenizer.from_pretrained("mistralai/Mistral-7B-v0.1")
    model = AutoModelForCausalLM.from_pretrained(
        model_name,
        device_map="auto",
        torch_dtype=torch.float16
    )
    return model, tokenizer
```

Plugin Quality Identification using Mistral-7B LLM's Output

```
==================================================
Analysis for astra-sites
==================================================
[INST] Analyze this WordPress plugin based on its quality metrics:

Plugin: astra-sites

Quality Scores:
- Overall Quality: 1.32/5.0
- Maintenance: 1.00/5.0
- Popularity: 5.00/5.0
- Support: 51.39/5.0

Usage Statistics:
- Active Installations: 1000000
- Total Downloads: 46442209
- Support Thread Resolution: 5/9
- Rating: 1/5.0
```

Please provide:
1. A brief quality assessment
2. Key strengths and weaknesses
3. Recommendations for improvement [/INST]

Quality assessment

The plugin has a high number of active installations and a low rating, which suggests that it is popular. However, the plugin has a low maintenance score, which indicates that it may not be actively maintained and updated regularly. It also has a low support score, which means that users may experience difficulties getting support for the plugin.

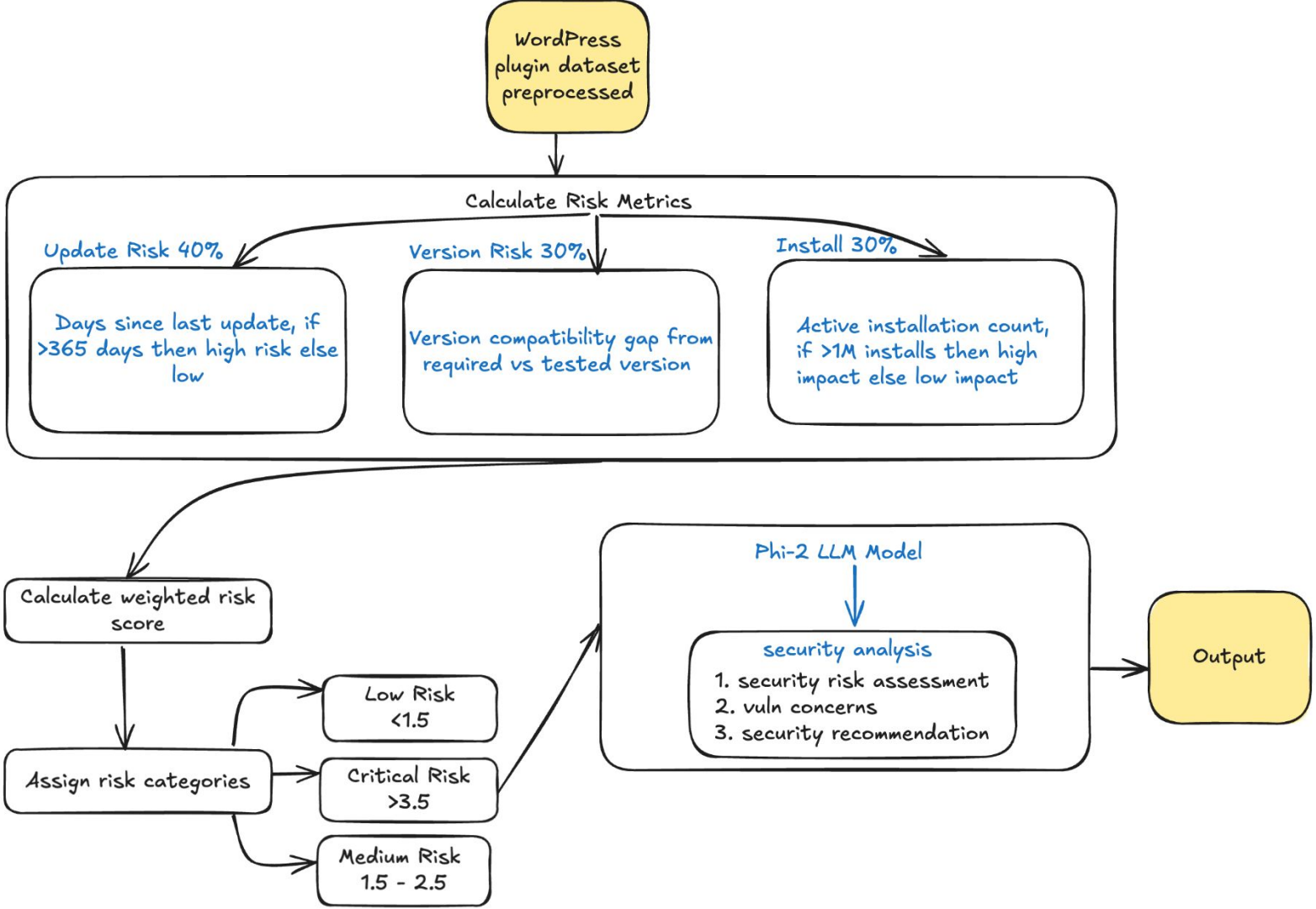
Key strengths

- The plugin has a high number of active installs, which shows that it is widely used and popular.

Weaknesses

- The plugin has a low maintenance score, which suggests that it may not be actively maintained and updated regularly.
- The plugin has a low support score, which means that users may experience difficulties getting support for the plugin.

Use-case: Identify plugin quality using open-source LLM


WordPress plugin dataset preprocessed
Calculate Risk Metrics
Update Risk 40%
Days since last update, if >365 days then high risk else low
Version Risk 30%
Version compatibility gap from required vs tested version
Install 30%
Active installation count, if >1M installs then high impact else low impact
Calculate weighted risk score
Assign risk categories
Low Risk <1.5
Critical Risk >3.5
Medium Risk 1.5 - 2.5
Phi-2 LLM Model
security analysis
1. security risk assessment
2. vuln concerns
3. security recommendation
Output

```python
def setup_phi2():
    """Setup Phi-2 model for risk analysis."""
    model_name = "microsoft/phi-2"
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForCausalLM.from_pretrained(
        model_name,
        device_map="auto",
        torch_dtype=torch.float16,
        load_in_4bit=True
    )
    return model, tokenizer
```

```
def analyze_plugin_risk(plugin_data, model, tokenizer):
    """Generate security risk analysis using Phi-2."""
    prompt = f"""Analyze the security risk for this WordPress plugin:

Plugin: {plugin_data['slug']}

Risk Metrics:
- Risk Score: {plugin_data['risk_score']:.2f}/5.0 ({plugin_data['risk_category']})
- Days Since Last Update: {(datetime.now() - plugin_data['last_updated']).days}
- WordPress Version Required: {plugin_data['requires']}
- WordPress Version Tested: {plugin_data['tested']}
- Active Installations: {plugin_data['active_installs']}

Based on these metrics, provide:
1. Security risk assessment
2. Specific vulnerability concerns
3. Recommended security actions for administrators

Analysis:"""

    inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
    outputs = model.generate(
        inputs.input_ids,
        max_length=512,
        temperature=0.7,
        top_p=0.95,
        do_sample=True
    )

    return tokenizer.decode(outputs[0], skip_special_tokens=True)
```

Plugin Update Risk Assessment using Phi-2's LLM Output

```
==================================================
Security Analysis for 012-ps-multi-languages
Risk Category: Critical
==================================================

Plugin: 012-ps-multi-languages

Risk Metrics:
- Risk Score: 3.80/5.0 (Critical)
- Days Since Last Update: 4380
- WordPress Version Required: 3.3.x
- WordPress Version Tested: 3.4.2

Based on these metrics, provide:
1. Security risk assessment
2. Specific vulnerability concerns
3. Recommended security actions for administrator
```

Plugin Update Risk Assessment using Phi-2's LLM Output

Analysis:

This WordPress plugin, 012-ps-multi-languages, is currently at a critical security risk. The risk score of 3.80/5.0 indicates that the plugin has significant vulnerabilities and requires immediate attention. The plugin has not been updated in 4380 days, suggesting that the development team may not be actively monitoring and addressing security concerns.

The vulnerability concerns include:

1. The plugin does not have input validation and does not properly sanitise user input, allowing malicious code to be injected into the WordPress admin interface. This can lead to the compromise of user credentials and can lead to unauthorised access to the database and potential data breaches.

2. The plugin does not provide protection against CSRF attacks,
which can allow attackers to perform unauthorised actions on behalf of a logged-in user.

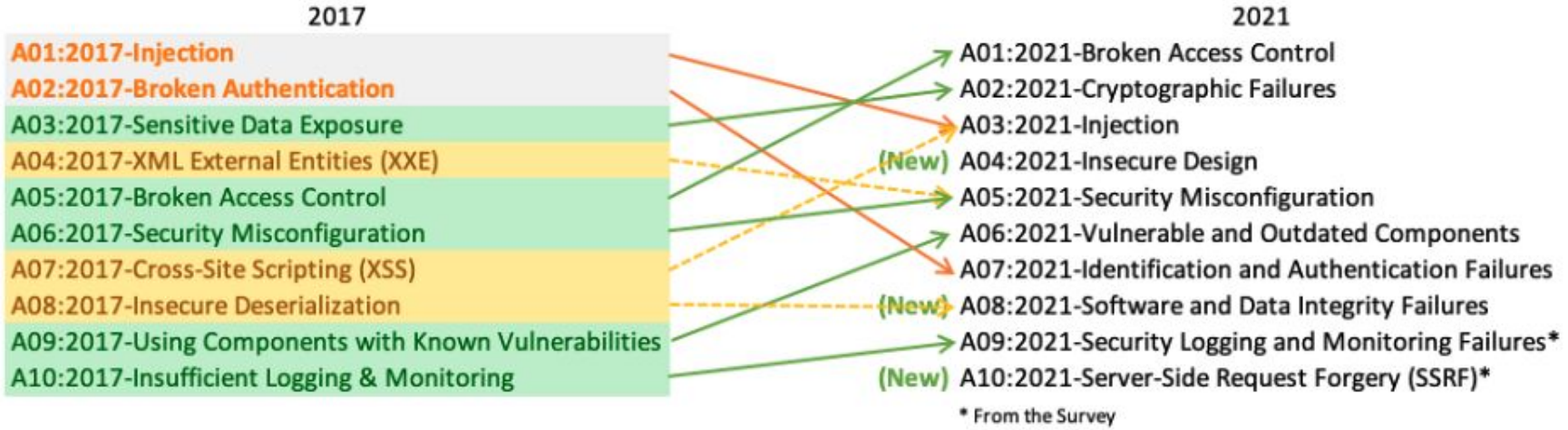
Plugin Update Risk Assessment using Phi-2's LLM Output

```
Recommended Security Actions:

1. Update the plugin: Administrators should update the 012-ps-multi-languages plugin to the
latest version available. This will address known security vulnerabilities and provide
necessary patches.

2. Enable SSL/TLS: Administrators should enable SSL/TLS encryption for the admin
interface to secure communication between the server and the admin interface.

3. Implement input validation: Administrators should implement strict input validation
```

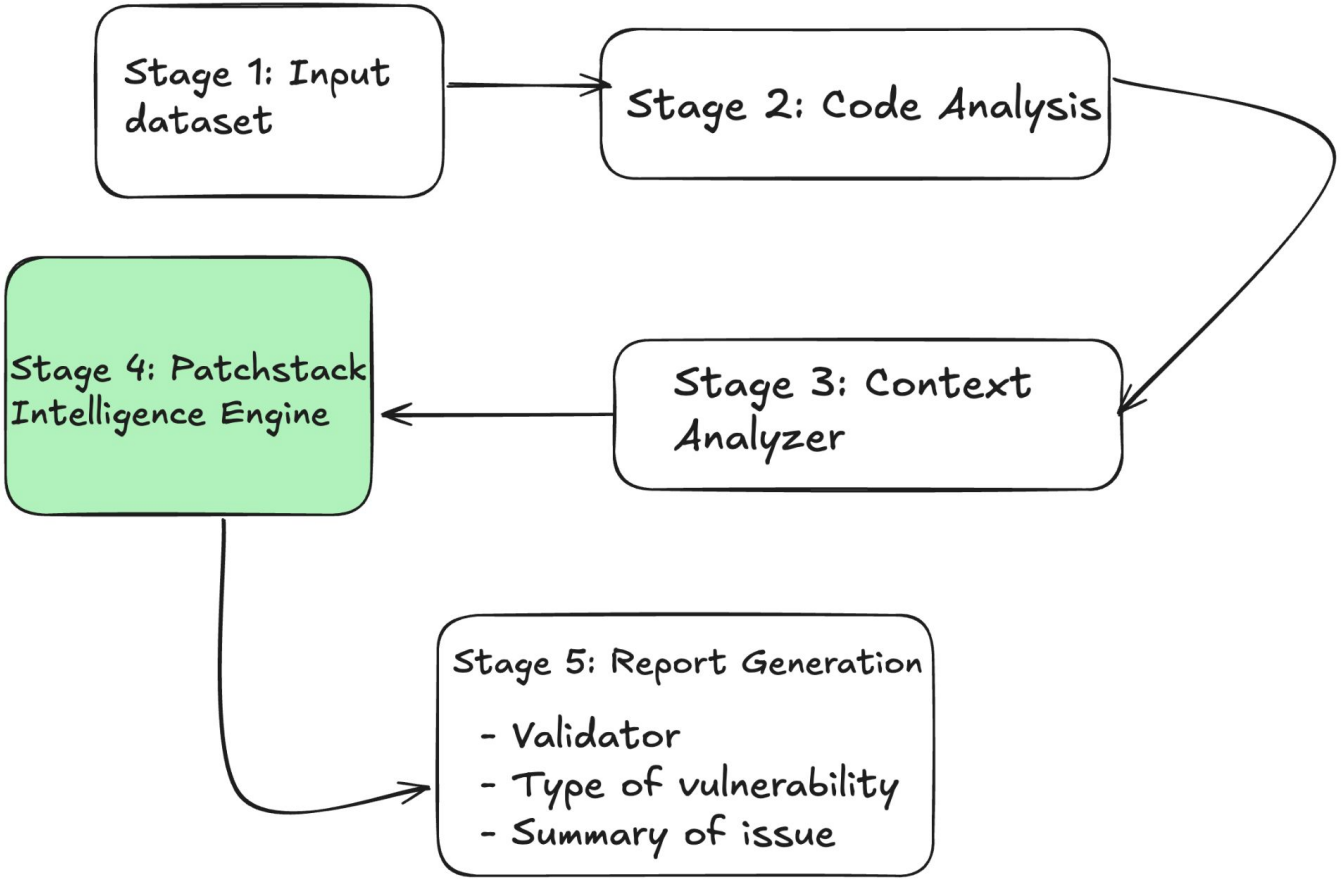
Use-case: Identify code vulnerabilities using LLMs

What is CODE vulnerability?

Vulnerability is a weakness or a hole in the application which could be a design flaw or the implementation bug which allows an attacker to harm an application. - OWASP



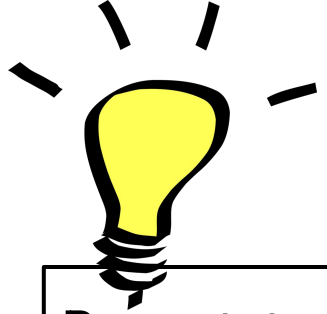
Vulnerability Scanner Tool Architecture (High Level)



Vulnerability Scanner Output

```
Plugin: WPContact-Form7
Filename: settings.php
Vulnerability Type: XSS
Vulnerable Code:
public function settings_field_input_text($args)
{
    $field = $args['field'];
    $value = get_option($field);

    echo sprintf('<input type="text" name="%s" id "%s" value "%s" style "%s" placeholder "%s" />' ,
$field, $field, $value, $style, $placeholder);

    if ($helptext) {
        echo sprintf('<br> <small>%s</small>', $helptext);
    }
}
Summary:
The code lacks essential security controls such as input validation and sanitization, making it
vulnerable to untrusted user submissions and exposure of sensitive data in raw form.
```

Challenges of using LLMs for building security use-cases



Prompt: translate this from english to french: “What is the capital of Italy”?

A. Quelle est la capitale de l’Italie?

B. La capitale de l’Italie est Rome.

translate this english to french - "What is the capital of Italy?"

La capitale de l'Italie est Rome.

translate this question from english to french - "What is the capital of Italy?"

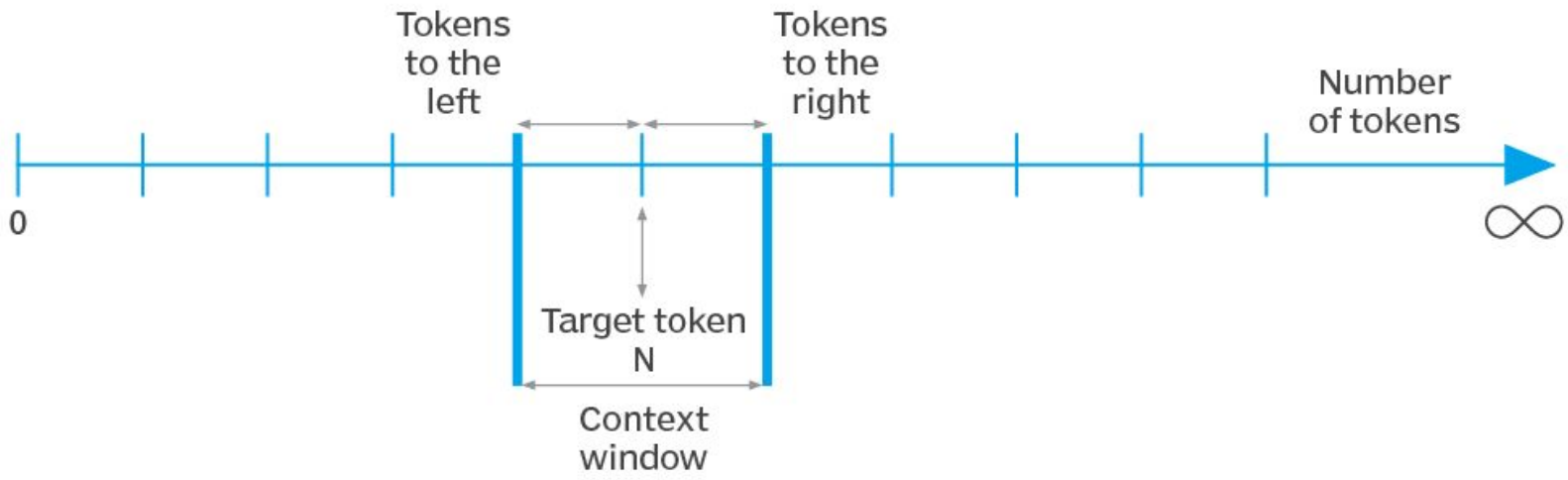
La réponse à votre question est : la capitale de l'Italie est Rome.

Search web

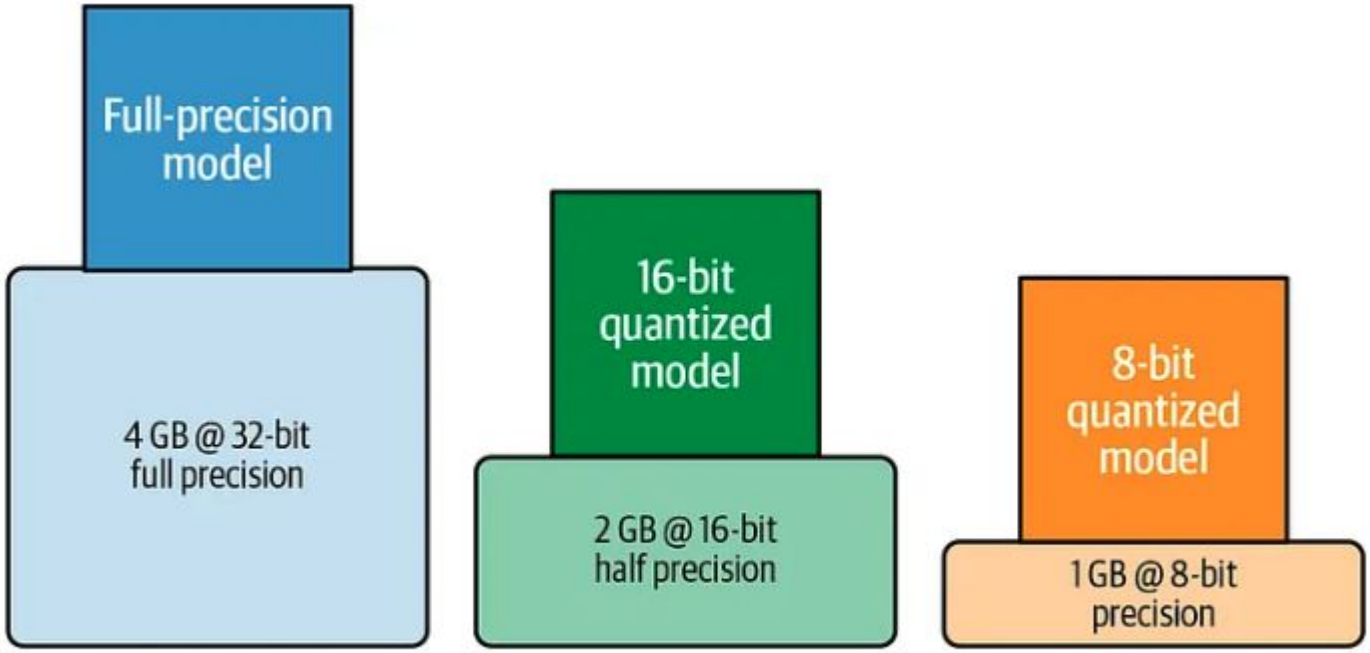
Ask anything

Model: mistralai/Mistral-7B-Instruct-v0.3 · Generated content may be inaccurate or false.

LLMs Confabulations


Tokens to the left
Tokens to the right
Number of tokens
0
∞
Target token N
Context window

Limited Context Window for LLMs



Approximate GPU RAM needed to load a 1-billion-parameter model at 32-bit, 16-bit, and 8-bit precision [5]

Resource constraints to utilize billions of params LLMs

Ways to mitigate challenges in LLMs?

Implement ground-truth mechanisms so that LLMs has reliable external knowledge source.

Utilize techniques like summarization, hierarchical attention mechanisms, and batch process the dataset by dividing into chunks.

Integrate model quantization, distillation, pruning techniques.

In summary,

- The integration of LLMs in WordPress security augments our security practices with intelligent, context-aware systems.

- LLMs will improve WordPress security by combining traditional methods with AI.

- LLMs are not replacing security experts, in fact they're empowering them to work more effectively and efficiently.

Thank you!

In the WordPress’s vast digital sphere
Where plugins and code far and near,
LLMs could be our guiding light,
Making its security clear and bright!

Yours truly

Happy to take any questions!

Reach out on X

@imrashminagpal

Sign up for Patchstack's newsletter

https://patchstack.com/newsletter/