AI Code Security: A Challenge to be Solved by... AI

Ofri Ouzan

Agenda

- AI to GenAI
- GenAI Types
- Code GenAI Security Issues
- Code Snippet Examples
- NextGen Demo

Whoami?

Ofri Ouzan

Security Researcher currently at JFrog.

Specializes in researching vulnerabilities and exploits,

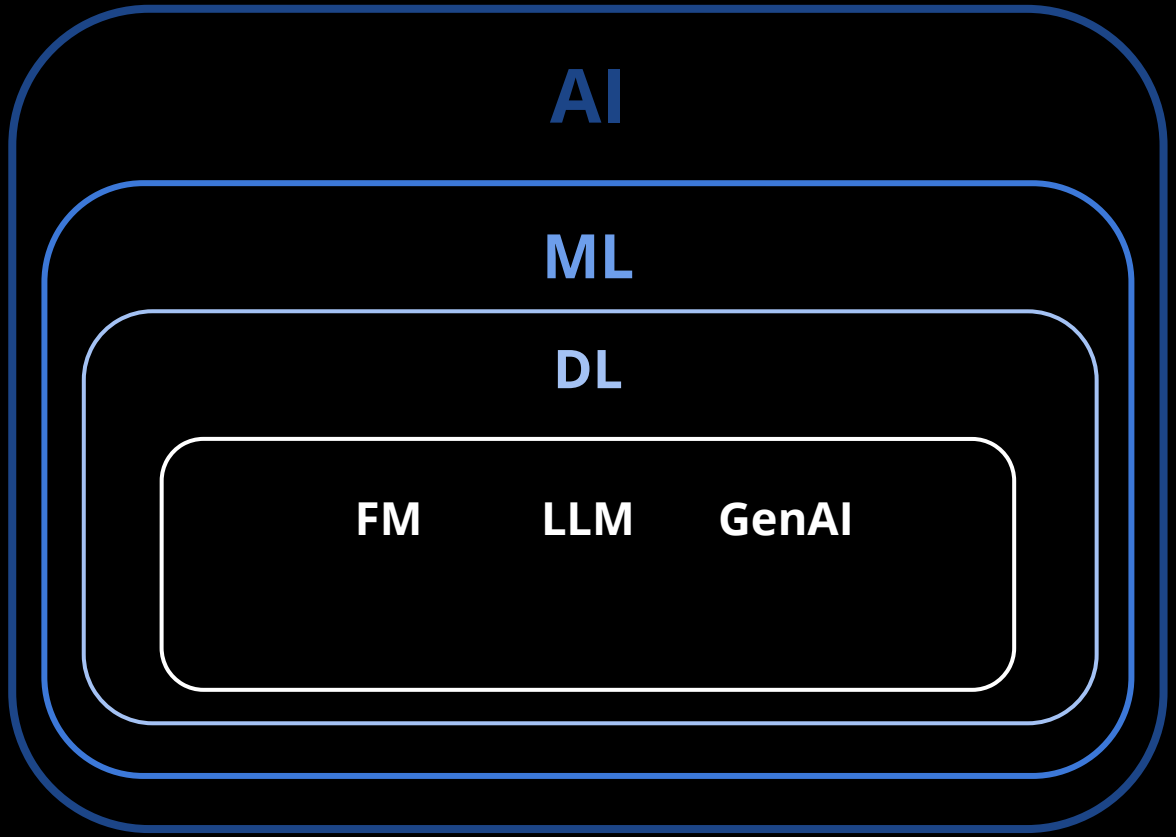
developing open source tools.

Researching AI-based solutions for vulnerability fixes.

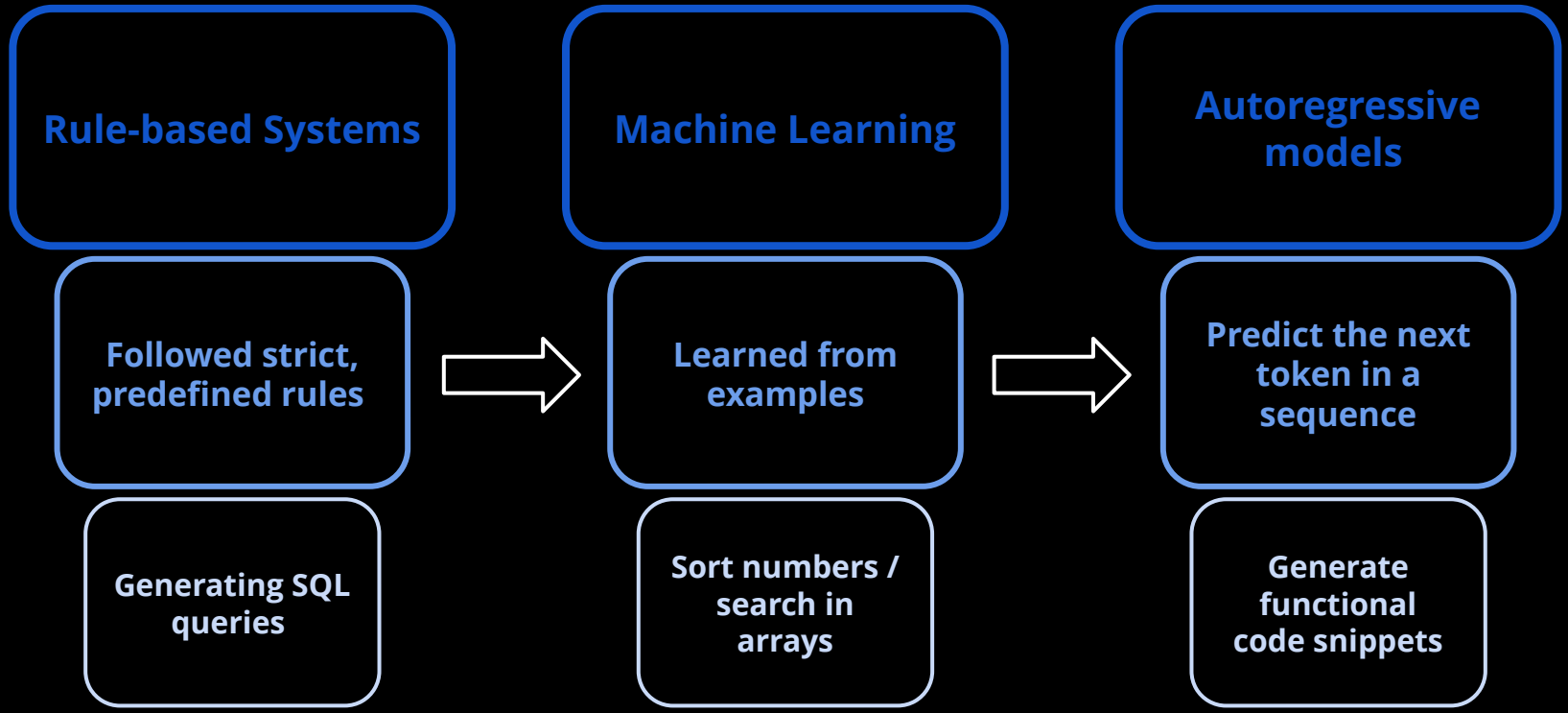
Enjoys sharing findings and insights with the community.

- https://medium.com/@ofriouzan

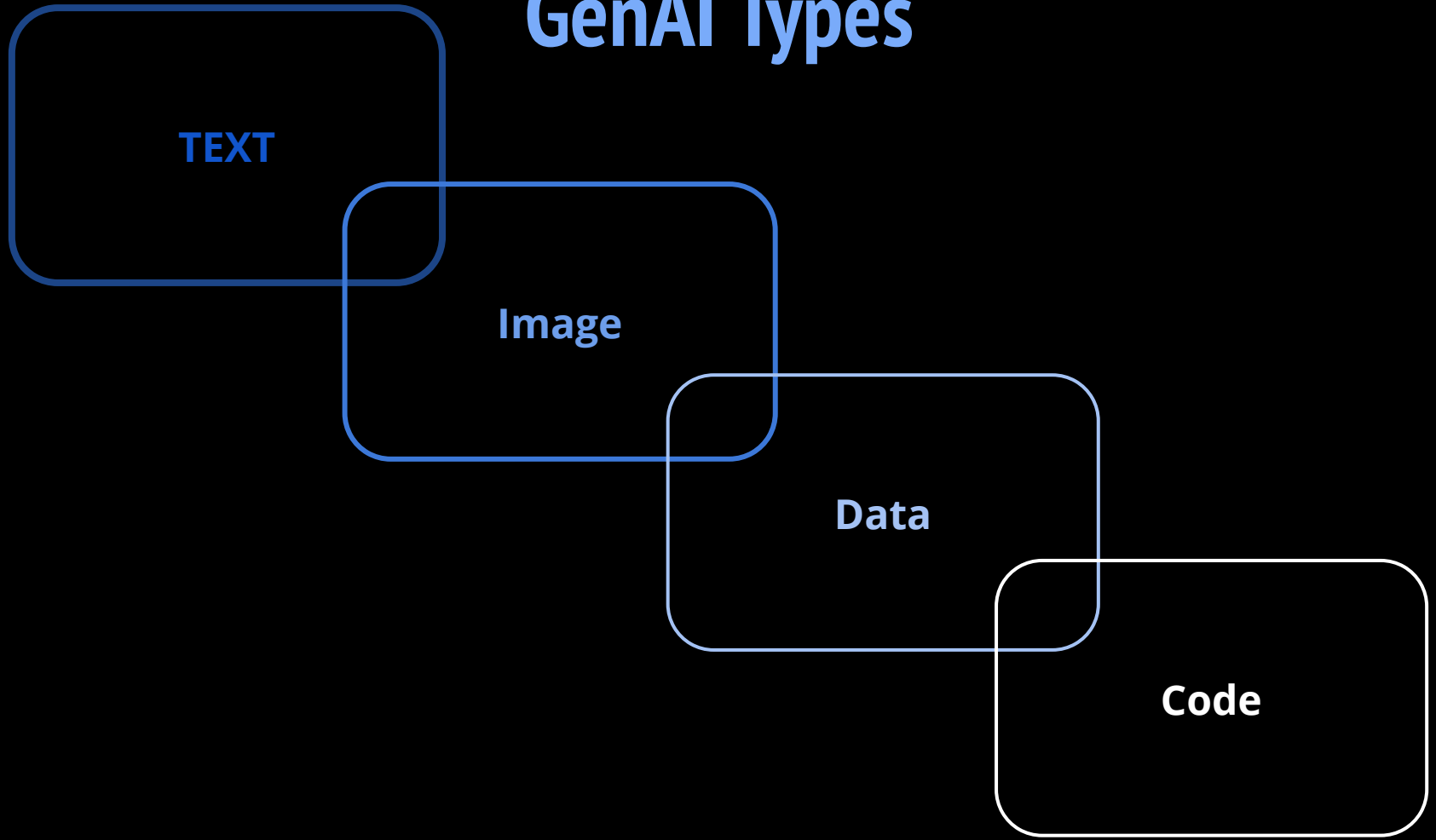
The Evolution of Gen AI



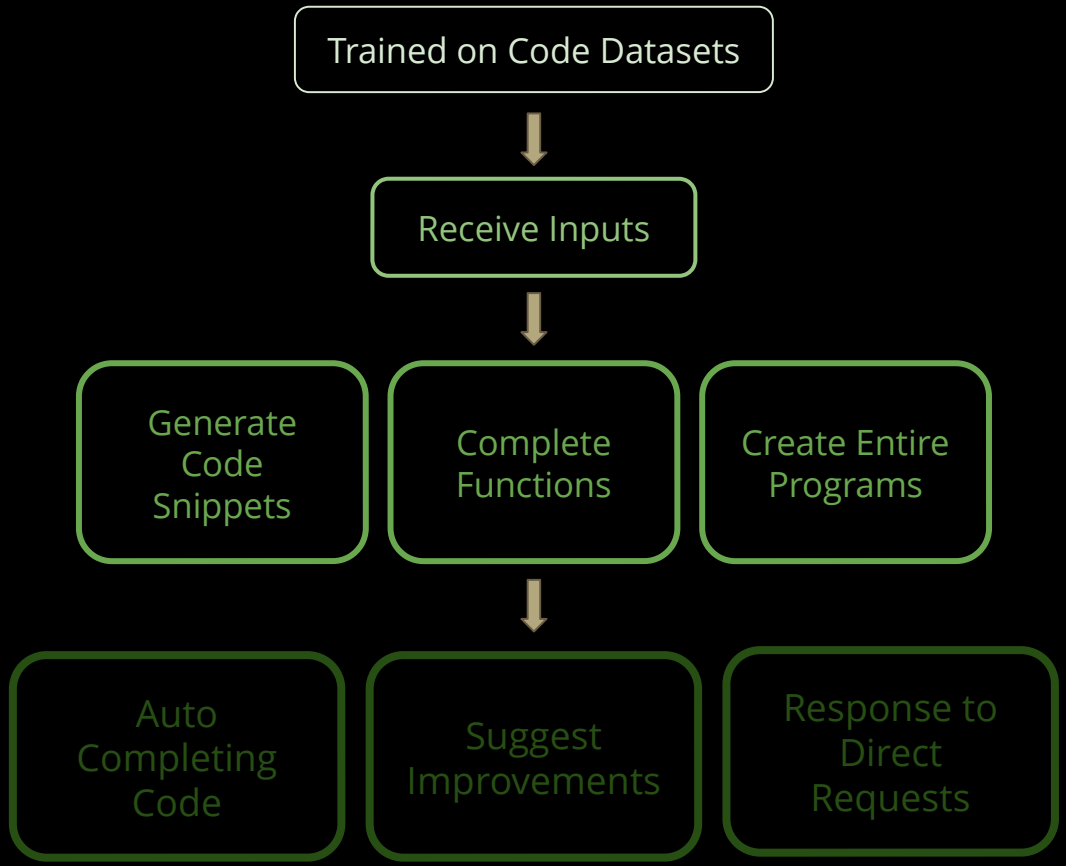
The Evolution of Code Gen AI



GenAI Types



Code GenAI


Trained on Code Datasets
Receive Inputs
Generate Code Snippets
Complete Functions
Create Entire Programs
Auto Completing Code
Suggest Improvements
Response to Direct Requests

AI Code Security Challenge

- Improper Input Validation
- Insecure Configuration
- Hallucinations Package Risk
- Dependency Vulnerabilities
- Inadequate Authentication
- Insecure Data Storage
- Insecure Functions
- Lack of Security Testing

Fetching Files From User Input

```
1  import flask
2
3  app = flask.Flask(__name__)
4
5  # Send file to the user based on filename provided by the user input
6  @app.route('/download/<filename>')
7  def download(filename):
       return flask.send_file(filename)
```

Arbitrary File Read

Attackers can insert a path traversal input to access files in the fs

"../../../etc/shadow"

Flask's send_file documentation

> Please never pass filenames to this function from user sources; you should use `send_from_directory()` instead.

Flask's send_file documentation: https://tedboy.github.io/flask/generated/flask.send_file.html

Safe use

send_from_directory()

Safe Version

```python
from flask import Flask, send_from_directory, abort, safe_join
import os

app = Flask(__name__)

# Base directory you want to serve files from
BASE_DIRECTORY = '/path/to/serve/files/from'

@app.route('/files/<path:filename>')
def download_file(filename):
    # Preventing directory traversal with safe_join
    safe_path = safe_join(BASE_DIRECTORY, filename)

    # Check if path is safe and file exists
    if safe_path and os.path.exists(safe_path):
        # Serve the file located in the safe path
        return send_from_directory(BASE_DIRECTORY, filename)
    else:
        # In case of invalid path or file not found, respond with 404 Not Found
        abort(404)


if __name__ == "__main__":
    # Disable debug mode when deploying
    app.run(debug=False)
```

Generating Configuration Files

kub.yaml

Click here to ask Blackbox to help you code faster

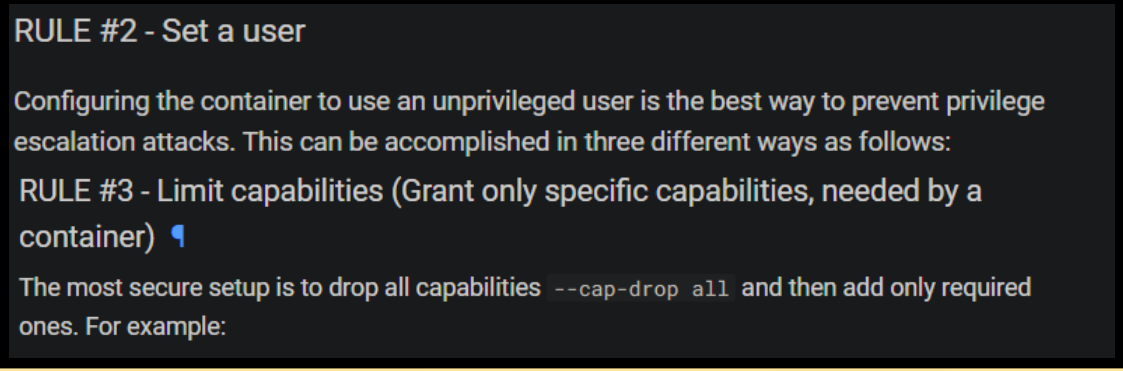
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
    name: test-pod
    image: busybox
    securityContext:
      runAsUser: 0
        capabilities:
          add: ["SYS_ADMIN
```

Privilege Escalation Risk

The setting makes the container vulnerable to:

1. Privilege escalation attacks
2. Potential breaches into the host system

Docker Security Cheat Sheet (OWASP)



RULE #2 - Set a user

Configuring the container to use an unprivileged user is the best way to prevent privilege escalation attacks. This can be accomplished in three different ways as follows:

RULE #3 - Limit capabilities (Grant only specific capabilities, needed by a container) ¶

The most secure setup is to drop all capabilities `--cap-drop all` and then add only required ones. For example:

Safe use

- Set unprivileged user
- Drop all capabilities
- Set necessary capabilities (if needed)

Safe Version

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: secure-title
  labels:
    app: secure-title
spec:
  containers:
    - name: secure-title
      image: ubuntu
      imagePullPolicy: Always
      securityContext:
        runAsUser: 1000
        runAsNonRoot: true
        readOnlyRootFilesystem: true
        allowPrivilegeEscalation: false
```

Upload Files From User Input

```
1   import flask
2
3   # Write a function that uploads a file provided by the user
4   def upload_file():
5       if flask.request.method == 'POST':
            file = flask.request.files['file']
            file.save('uploads/' + file.filename)
            return 'File uploaded successfully'
```

Arbitrary File Write

Introduce many risks:

1. Malware Upload and Execution
2. Cross-Site Scripting (XSS)
3. Overwrite Sensitive Files

Flask's fileuploads documentation

So first we need a couple of imports. Most should be straightforward, the **`werkzeug.secure_filename()`** is explained a little bit later. The `UPLOAD_FOLDER` is where we will store the uploaded files and the `ALLOWED_EXTENSIONS` is the set of allowed file extensions.

Safe use

- werkzeug.secure_filename()
- UPLOAD_FOLDER
- ALLOWED_EXTENSIONS

Safe Version

```python
from flask import Flask, request
from werkzeug.utils import secure_filename
import os

app = Flask(__name__)

# Configure upload folder and allowed extensions
UPLOAD_FOLDER = 'path/to/the/uploads'
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

1 usage
def allowed_file(filename):
    return '.' in filename and \
        filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route( rule: '/upload', methods=['POST'])
def upload_file():
    # Check if the post request has the file part
    if 'file' not in request.files:
        return "No file part"
    file = request.files['file']
    # If the user does not select a file, the browser submits an
    # empty file without a filename.
    if file.filename == '':
        return "No selected file"
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        return "File uploaded successfully"
    else:
        return "File type is not allowed"

if __name__ == "__main__":
    app.run()  # Do not use debug=True in production
```

ALLOWED_EXTENSIONS

ALLOWED_EXTENSOINS is not bulletproof

Attackers can change the extension of a malware and successfully upload

```
13  def allowed_file(filename):
14      return '.' in filename and \
15          filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
16
```

Safe use

Use python-magic to inspect the real file's type

```
import magic

def is_valid_filetype(file_path, expected_mime_type):
    mime = magic.Magic(mime=True)
    file_mime_type = mime.from_file(file_path)
    return file_mime_type == expected_mime_type
```

Copying Strings

```
1   // Write a function that copies one string to another
2
3   #include <stdio.h>
4   #include <string.h>
5
6   void copy(char *to, char *from) {
        while (*from) {
            *to++ = *from++;
        }
        *to = '\0';
    }
```

Copying strings without checking the size

Leads to stack-based overflow attacks

Safe Version

```c
#include <stdio.h>
#include <string.h>

void safe_copy(char *to, const char *from, size_t max_len)
{
    if (max_len == 0) {
        return; // No space to copy anything
    }

    // Copy up to max_len -1 characters and make sure to always null-terminate
    size_t i;
    for (i = 0; i < max_len - 1 && from[i] != '\0'; ++i) {
        to[i] = from[i];
    }
    to[i] = '\0'; // Ensure the string is null-terminated
}

int main()
{
    char from[] = "Copy this safe string.";
    char to[20]; // Example destination buffer

    safe_copy(to, from, sizeof(to));

    printf("Copied string: %s\n", to);
    return 0;
}
```

Comparing Secrets

```php
<?php
# compare two strings from user input
$data = json_decode(file_get_contents('php://input'), true);
$secret_token = "super secret"
if ($data['token'] == $secret_token) {
    if ($data['string1'] == $data['string2']) {
        echo "The strings are the same";
    } else {
        echo "The strings are different";
    }
} else {
    echo "Invalid token";
}
```

Type Juggling/Confusion Vulnerability

Exploit Type Juggling/Confusion Vulnerability

By inserting True or 0

PHP

```php
<?php
  $user_input = true;
  if ($user_input == "good_password") {
      print("Authentication success!\n");
  }
  else {
      print("Authentication fail!\n");
  }
?>
```

```
Authentication success!
```

Safe use

- Type Casting
- Triple Equal Signs
- hash_equals()

Safe Version

```php
<?php

require 'vendor/autoload.php';
$dotenv = Dotenv\Dotenv::createImmutable(__DIR__);
$dotenv->load();

// Getting JSON input securely and decoding it
$data = json_decode(file_get_contents("php://input"), true);

// Checking if the expected token is set in the input
if (isset($data["secret_token"])) {
    // Using hash_equals for secure string comparison
    if (hash_equals($_ENV['SECRET_TOKEN'], $data["secret_token"])) {
        echo "You are authorized";
    } else {
        echo "You are not authorized";
    }
} else {
    echo "You are not authorized";
}
?>
```

Using Code GenAI Be Like...

Avoid Blindly Trusting

Examine the Output

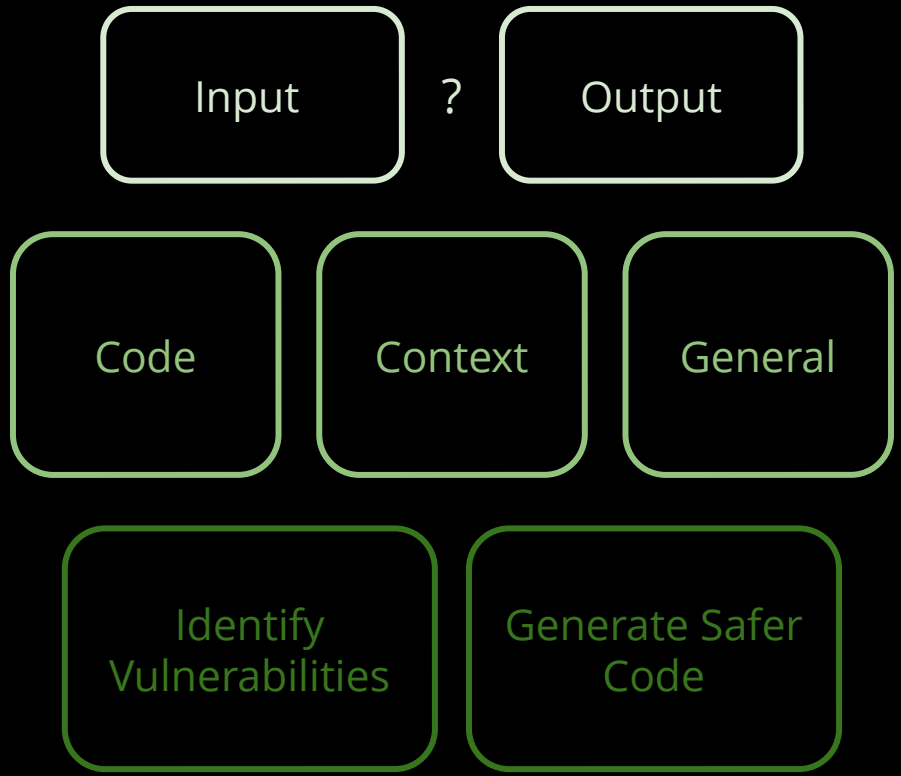
Pay Attention to Security Guidelines

Who Knows The Best?

AI !!!

Well OpenAI...

Prompt Engineering



Better Than I Thought

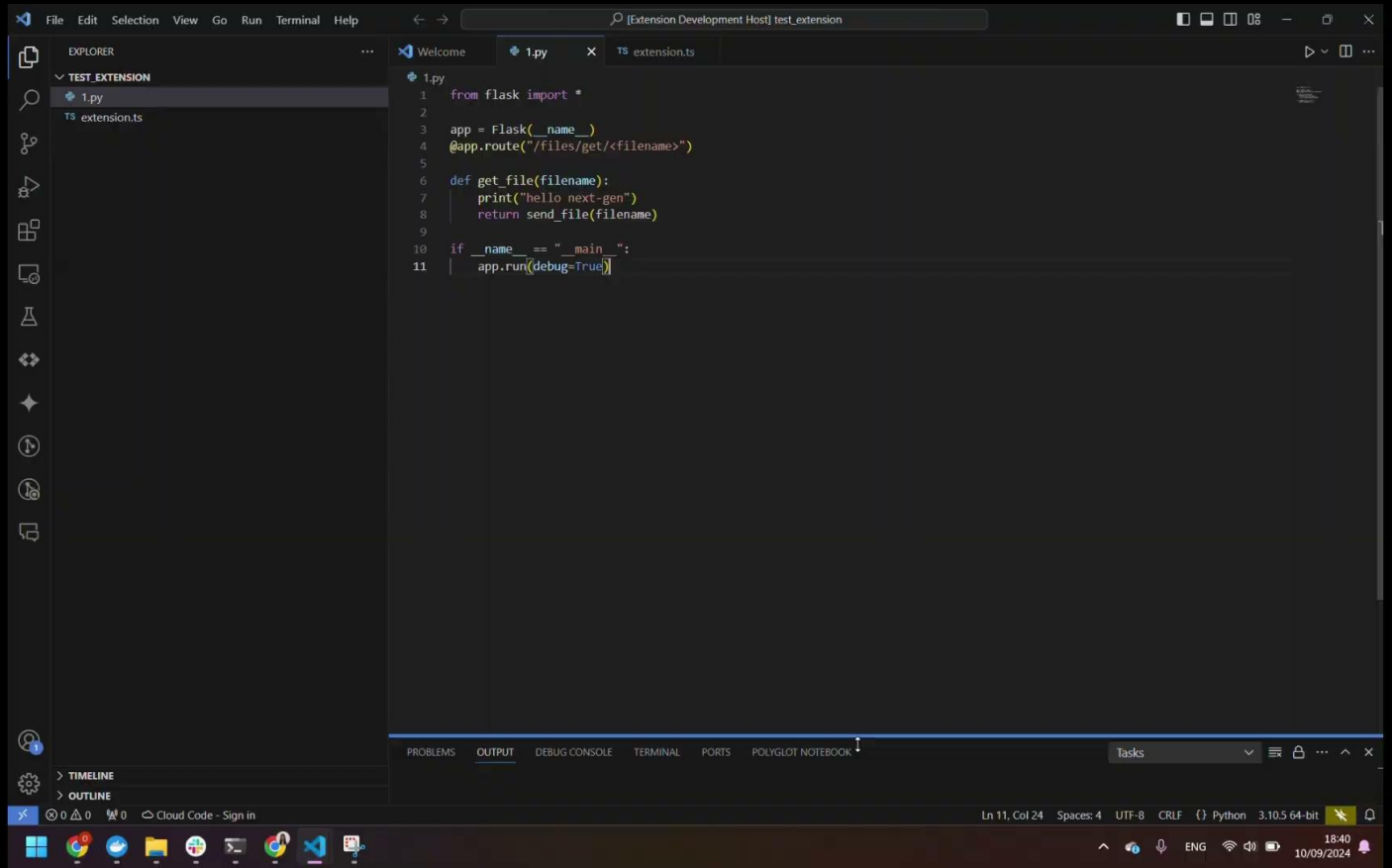
```php
<?php

require 'vendor/autoload.php';
$dotenv = Dotenv\Dotenv::createImmutable(__DIR__);
$dotenv->load();

// Getting JSON input securely and decoding it
$data = json_decode(file_get_contents("php://input"), true);

// Checking if the expected token is set in the input
if (isset($data["secret_token"])) {
    // Using hash_equals for secure string comparison
    if (hash_equals($_ENV['SECRET_TOKEN'], $data["secret_token"])) {
        echo "You are authorized";
    } else {
        echo "You are not authorized";
    }
} else {
    echo "You are not authorized";
}
?>
```

How To Automate?

Code
GenAI Tool

OpenAI
Model

NextGen


File Edit Selection View Go Run Terminal Help
[Extension Development Host] test_extension
EXPLORER
TEST_EXTENSION
1.py
extension.ts
Welcome
1.py
extension.ts
1.py
from flask import *
app = Flask(__name__)
@app.route("/files/get/<filename>")
def get_file(filename):
print("hello next-gen")
return send_file(filename)
if __name__ == "__main__":
app.run(debug=True)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POLYGLOT NOTEBOOK
Tasks
TIMELINE
OUTLINE
Cloud Code - Sign in
Ln 11, Col 24 Spaces: 4 UTF-8 CRLF Python 3.10.5 64-bit
ENG
18:40
10/09/2024

Conclusions

Adopt AI but avoid blindly trusting it

Sometimes the easiest solutions are the best

NextGen

ofriouzan@gmail.com

https://medium.com/@ofriouzan