



IdentityShield  
Summit '26

# Deep Dive into Building Next-Gen Local AI Security Reviewers

Hardik Mehta (hardw00t)  
Rajanish Pathak (h4ckologic)

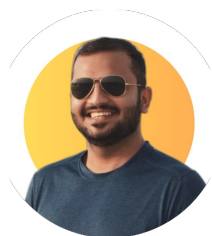




IdentityShield  
Summit '26

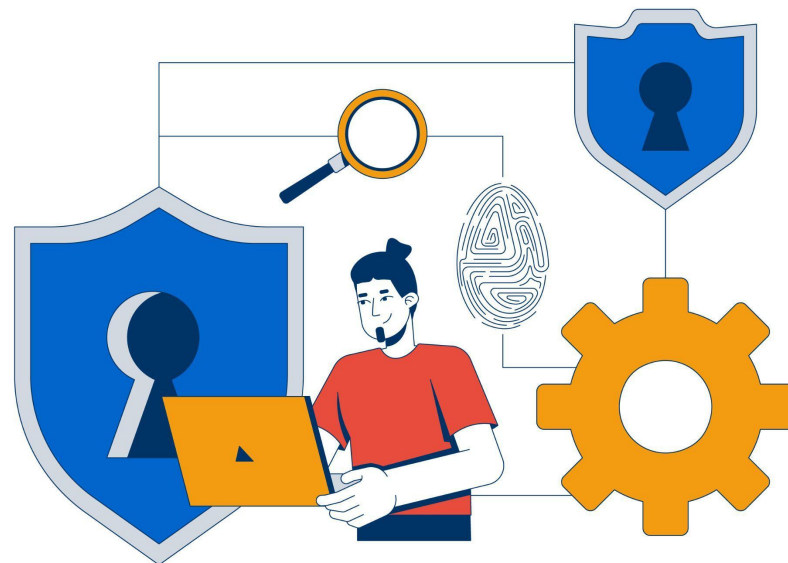
# About me

I (@h4ckologic) am a security researcher passionate about uncovering and addressing critical vulnerabilities in complex technology implementations. My work includes identifying and reporting issues to top tech companies like Apple, Google, Microsoft and many others, some of my CVEs identified are Apple, PhantomJS, and NPM html-pdf. I've had the privilege of sharing my research at leading conferences, including HackLu, BlackHat MEA, NoNameCon, Ekoparty, Hacktivity, Hack in the Box and Romhack. With a focus on practical solutions and deep technical insights, I'm dedicated to advancing security practices and contributing to the global infosec community.



**RAJANISH PATHAK**  
Senior Manager  
Security Services

#Product #Security #Research





# Today's Journey

1. The Problem → Why traditional SAST tools falls short
2. Understanding Code → AST and semantic analysis foundations
3. Human Expert Reasoning → How security experts actually think
4. Teaching AI → Combining LLMs + RAG for code review
5. Privacy First → Why local models change everything
6. Implementation → Building a production-ready system
7. Economics → Free security reviews for life
8. Open Source → FalconEYE demonstration





IdentityShield  
Summit '26

# THE PROBLEM

Why Traditional SAST Falls Short



# The 85% False Positive Problem

## Pattern Matching Is Fundamentally Limited

- Regex and signatures only find what they're programmed to find
- No understanding of context, intent, or data flow
- Cannot distinguish real vulnerabilities from safe patterns



## The Result: Alert Fatigue

- 70-90% of alerts are false positives
- Developers learn to ignore security tools
- Real vulnerabilities get buried in noise



"We stopped using our SAST tool because every scan produced 500+ alerts that were 95% noise." — Senior Security Engineer



# The Missing Context Problem

Traditional tools see: `user_input` in query

They flag: "SQL Injection!"

But they miss:

- Is `user_input` actually user-controlled?
- Is it sanitized upstream?
- Is the query even reachable from external input?
- What's the business context?

Without context, every potential pattern is a "vulnerability"

This is why we need semantic understanding





IdentityShield  
Summit '26

# UNDERSTANDING CODE

AST and Semantic Analysis



# Abstract Syntax Trees: The Foundation

## What is an AST?

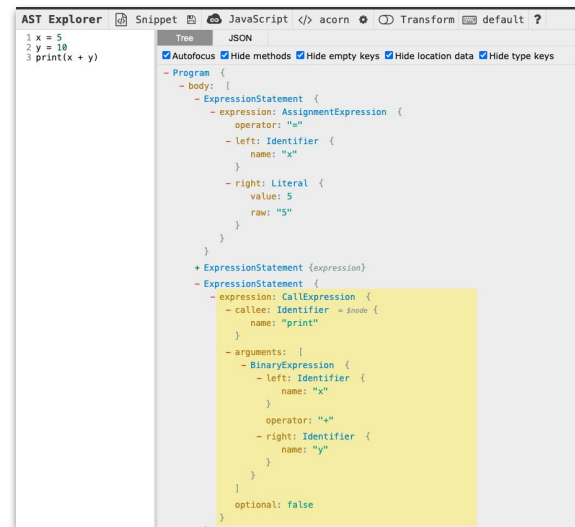
Source code transformed into a structured tree representing actual logic

## Why AST Matters for Security:

- Semantic Boundaries — Know where functions, classes, scopes begin/end
- Language Agnostic — Same analysis approach across Python, JS, Go, Rust
- Data Flow Ready — Follow variables through assignments and returns
- No Regex Needed — Work with code structure, not text patterns

## Tree-sitter: The Engine

- Powers VS Code, GitHub syntax highlighting
- Incremental parsing — fast on large codebases
- Supports 40+ languages with consistent API





# From Text to Structure

This is the foundation for semantic analysis

## Source Code:

```
1 def process(user_input):  
2     query = "SELECT * FROM users WHERE id=" + user_input  
3     return db.execute(query)
```

## AST Representation:

```
1 FunctionDef "process"  
2   | args: ["user_input"]  
3   | body:  
4     | Assign "query" ← BinOp(string + user_input)  
5     | Return: Call db.execute(query)  
6
```

Now we can trace: `user_input` → `query` → `db.execute`



IdentityShield  
Summit '26

# HUMAN EXPERT REASONING

How Security Experts Actually Think



# The Expert Security Review Process

## Step 1: Context Gathering

"What does this app do? What's the threat model?"

## Step 2: Data Flow Tracing

"Where does this input come from? Where does it go?"

## Step 3: Pattern Recognition (Experience)

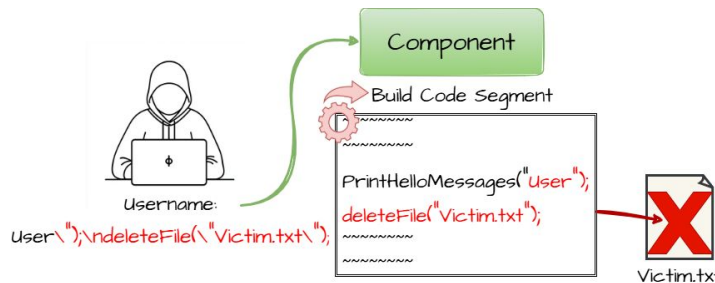
"I've seen this anti-pattern cause CVEs before"

## Step 4: Risk Assessment

"What's the blast radius? Is it reachable externally?"

## Key Insight: Experts don't pattern-match

They understand intent, trace flows, and reason about impact





# Expert vs Traditional SAST

## Traditional SAST asks:

"Does line X match regex Y?"

No context. No reasoning. Binary yes/no.

## Human Expert asks:

"This user input flows through function X, gets partially sanitized in Y, but the encoding in Z doesn't handle this edge case..."

Full context. Deep reasoning. Nuanced assessment.

## The Question:

Can we teach AI to think like a human security expert?

Answer: Yes. With the right architecture.





IdentityShield  
Summit '26

# TEACHING AI

LLMs + RAG for Security Review



# Why Large Language Models?

## LLMs Understand Code Semantics

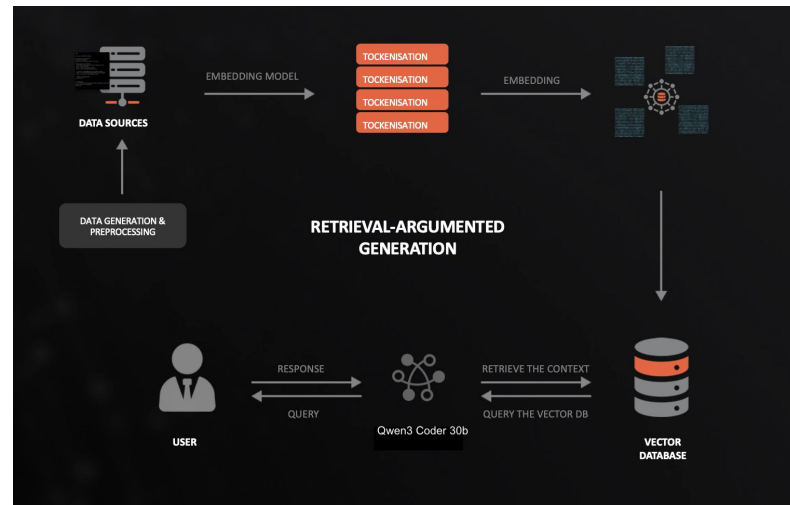
- Trained on millions of repositories
- Learned patterns, idioms, and anti-patterns
- Can reason about code intent, not just syntax

## Beyond Pattern Matching:

- Understand what code is trying to do
- Recognize security-relevant patterns from training
- Generate contextual explanations
- Suggest appropriate fixes

## The Gap: LLMs need context

They can't read your entire codebase in one prompt





# The Complete Analysis Process

- Stage 1:  
Code Ingestion**  
Parse source files with Tree-sitter into AST
- Stage 2:  
Intelligent Chunking**  
Break code at semantic boundaries (functions, classes)
- Stage 3:  
Vector Indexing**  
Embed chunks and store in vector database
- Stage 4:  
Context Assembly (RAG)**  
For each file, retrieve architecturally relevant code
- Stage 5:  
AI Analysis & Reporting**  
LLM reviews with full context + validation pass,  
Findings with confidence scores and fix suggestions





IdentityShield  
Summit '26

# PRIVACY FIRST

Why Local Models Change Everything



# Why Local Large Language Models?

The Cloud Security Paradox

To analyze your code for security, you send it to someone else's servers?

Local LLMs Solve This:  **Ollama**

- Zero data transmission — code stays on your hardware
- Complete IP protection — proprietary code remains private
- Regulatory compliance — meet data residency requirements
- Airgapped environments — works without internet

**Modern Hardware Makes It Possible:**

- Consumer GPUs can run capable code models
- 32GB RAM runs production-quality analysis
- Apple Silicon M-series excellent for local inference

```
Desktop ollama ls
NAME                                ID                                SIZE  MODIFIED
qwen3-embedding:8b                  64b923495768                     4.7 GB 2 days ago
nomic-embed-text:latest             0a109f422b47                     274 MB 2 days ago
qwen3-coder:30b                     06c1097efce0                     18 GB 2 days ago
qwen2.5-coder:7b                    d8e161d27b9e                     4.7 GB 4 days ago
embeddinggemma:30b                 85462619ee72                     421 MB 8 weeks ago
embeddinggemma:latest               85462619ee72                     421 MB 3 months ago
qwen3-coder:latest                  06c1097efce0                     18 GB 3 months ago

>>> write a python code to add 2 numbers
Here's a simple Python code to add two numbers:

'''python
# Method 1: Using input() to get numbers from user
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
result = num1 + num2
print(f"The sum of {num1} and {num2} is: {result}")

# Method 2: Direct assignment
a = 5
b = 3
sum_result = a + b
print(f"The sum of {a} and {b} is: {sum_result}")

# Method 3: Using a function
def add_numbers(x, y):
    return x + y

number1 = 10
number2 = 15
total = add_numbers(number1, number2)
print(f"The sum of {number1} and {number2} is: {total}")
'''
```





# The Economics of Local AI

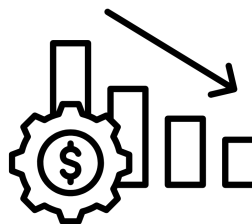
## Traditional Cloud SAST:

- Per-developer licensing: \$50-200/month
- Per-scan API costs: \$0.01-0.10 per scan
- Enterprise tier: \$50K-500K/year
- 10-person team annual cost: \$6,000-\$24,000



## Local AI Approach:

- Software: \$0 (open source)
- Per-scan cost: \$0 (local compute)
- Hardware: Existing developer machines
- ANY team size annual cost: \$0



## The Future Gets Better:

Models improve while cost stays zero



# The Future Gets Better, Not More Expensive

Unlike SaaS That Gets More Expensive:

## Models Keep Improving

- New open-source models release monthly
- Better quality at same compute cost

## Hardware Gets Cheaper

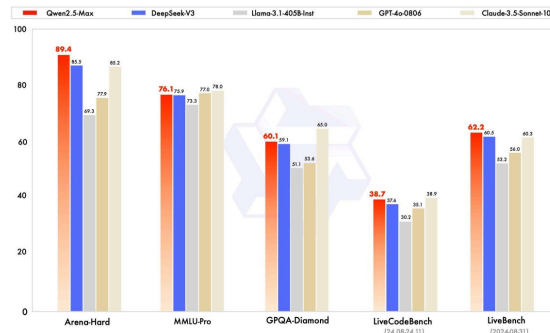
- GPUs that cost \$10K today → \$2K in 3 years
- Same capability, lower barrier to entry

## Inference Gets Faster

- Quantization, speculative decoding, flash attention
- Each generation is 2-3x faster

## Unlimited Scans Forever

Scan every commit, every branch, every PR — no limits





IdentityShield  
Summit '26

# IMPLEMENTATION

Building the System



# Architecture: Hexagonal Design

## Ports & Adapters Pattern

### Domain Core (Pure Business Logic):

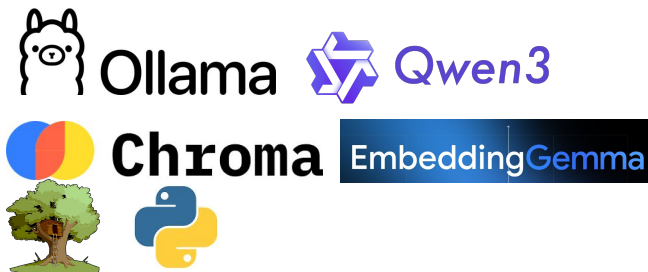
- Security Analyzer — vulnerability detection logic
- Context Assembler — RAG pipeline orchestration
- LLM Service — model interaction abstraction

### Infrastructure (Swappable):

- LLM Providers: Ollama, OpenAI, Anthropic
- Vector Stores: ChromaDB, Pinecone, Weaviate
- AST Parsing: Tree-sitter for all languages

### Benefits:

Testable • Extensible • Maintainable





# Technology Stack

## Local LLM Runtime: Ollama

- Easy model management, optimized for local inference
- Recommended: qwen3-coder:30b or deepseek-coder

## Vector Database: ChromaDB

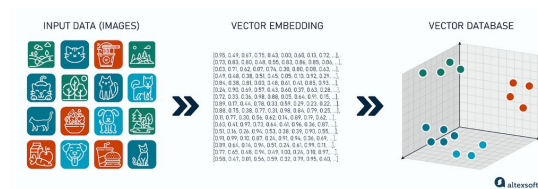
- Embedded mode — no server needed
- Persistent storage, fast semantic search

## AST Parsing: Tree-sitter

- Battle-tested (VS Code, GitHub)
- Incremental parsing, 40+ language support

Language: Python 3.12+

- Async/await for parallel processing
- Rich ecosystem for ML/AI



Ollama



Qwen3



Chroma



EmbeddingGemma



IdentityShield  
Summit '26

# FalconEYE: Open Source Implementation

We've Built This: <https://github.com/falconEYE-ai/FalconEYE>

## Features:

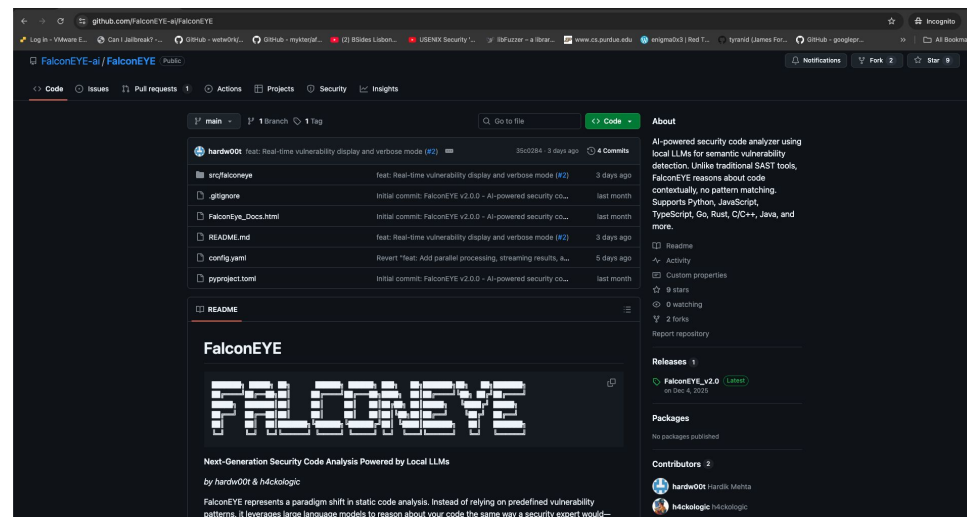
- All techniques discussed today — implemented and tested
- 9+ language support out of the box
- 4 output formats: Console, JSON, HTML, SARIF
- Production-ready with CI/CD integration

## Quick Start:

```
pip install falconeye
```

```
falconeye index ./your-project
```

```
falconeye scan ./your-project
```





IdentityShield  
Summit '26

# Live Demo

Let's See It In Action

## Setup (one-time):

```
$ ollama pull qwen3-coder:30b
```

```
$ pip install -e .
```

## Index Your Codebase:

```
$ falconeye index ./project
```

## Run Security Scan:

```
$ falconeye scan ./project
```

## What We'll See:

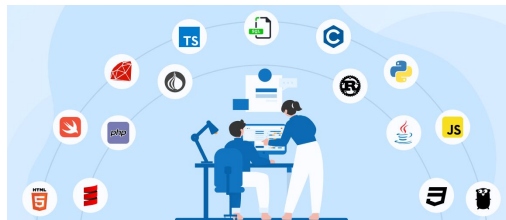
- Semantic vulnerability detection
- Context-aware analysis
- Actionable fix suggestions



# Roadmap & Future Work

## Near Term:

- IDE plugins
- Pre-commit hooks integration
- More language support



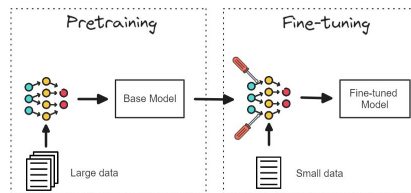
## Medium Term:

- Custom model fine-tuning on your codebase
- Team knowledge base integration
- Compliance report generation

## Research:

- Multi vulnerability chains
- Automated fix generation
- Security regression testing

### Large Language Model





IdentityShield  
Summit '26

# Thank You

Questions? | [github.com/falconEYE-ai/FalconEYE](https://github.com/falconEYE-ai/FalconEYE)



IdentityShield  
Summit '26

# Any Feedback?



[github.com/falconEYE-ai/FalconEYE](https://github.com/falconEYE-ai/FalconEYE)