

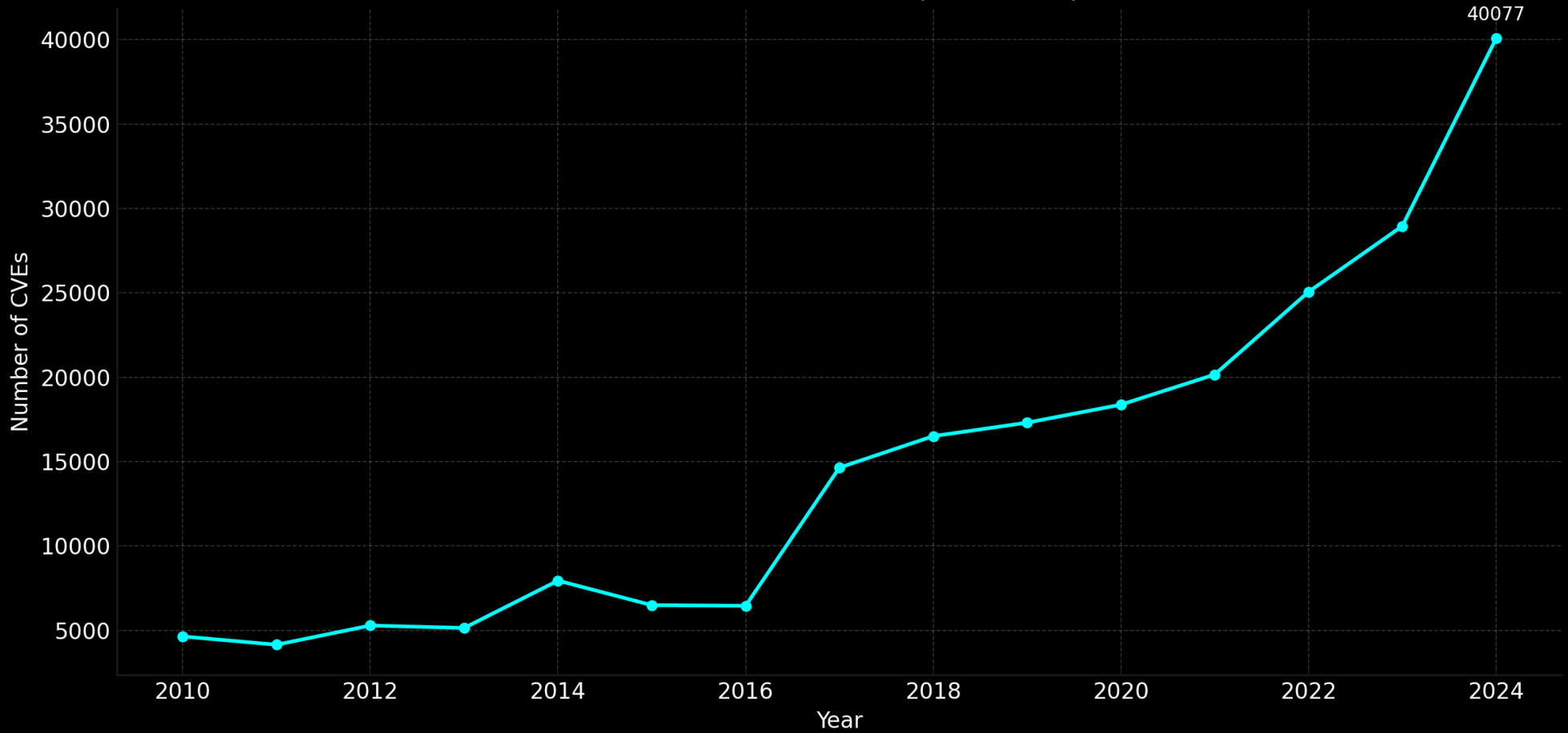
CVE/FIRST VulnCon 2025

Resolution Revolution: Turbocharging Security Ticketing Timelines

Joseph Seasly
Shruti Datta Gupta

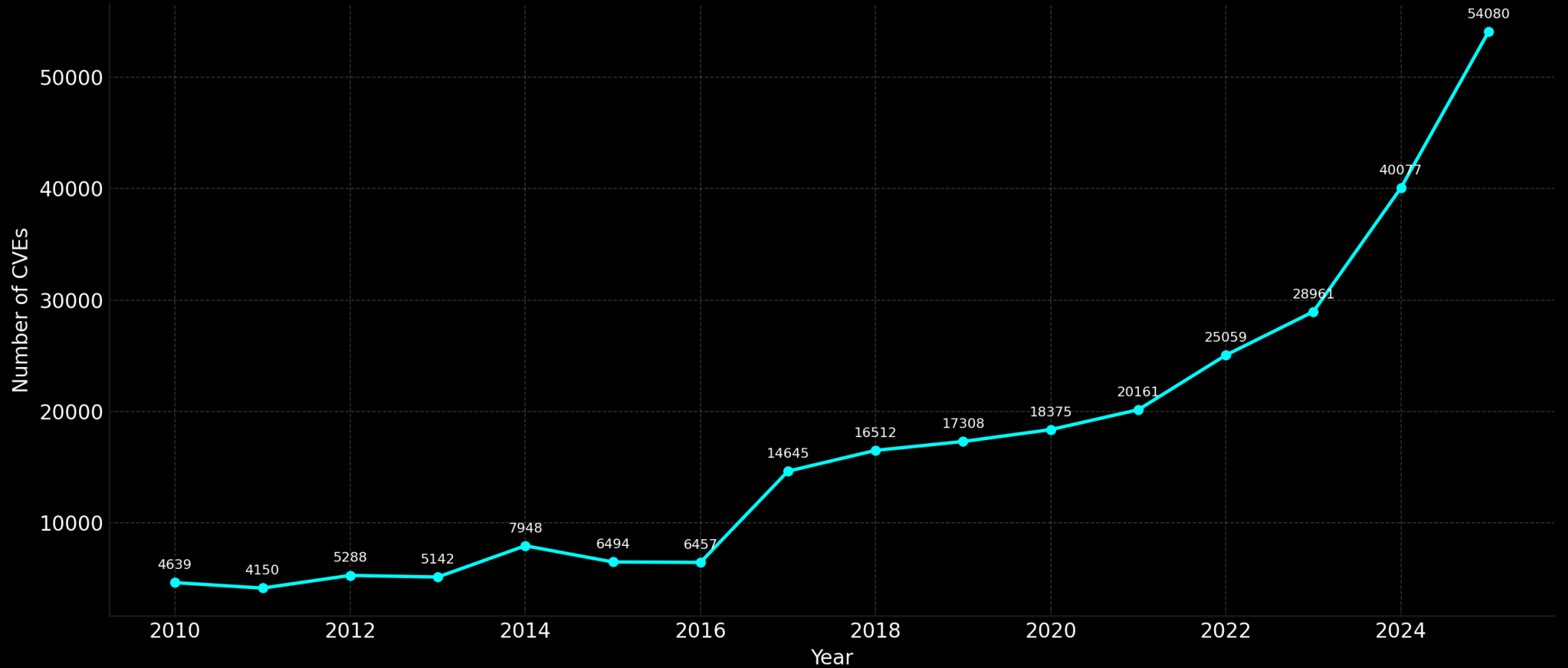
Skyrocketing vulnerabilities (through 2024)

CVEs Published Per Year (2010-2024)



Vulnerabilities: 2025 on pace for a record year!

CVEs Published Per Year (2010-2025*)



*2025 projected based on the current pace as of March 31st, 2025

Reality: Limited Security Resources

Security ticket overload

Unmanageable deadlines

High developer-to-security ratio

Staffing levels not keeping pace

Reality: Knowledge Challenges

Scattered communication

Fragmented documentation

Gaps in knowledge

Why We Care

Impact of vulnerabilities

Systemic risk

Redistribute bandwidth



Security engineer

Vulnerability expert
exploit, risk, CVE, CPE, CVSS ...



Developer

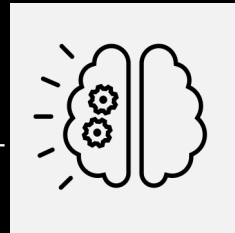
Product expert
code flow, structure, how it works ...



Fix the vulnerability



Security engineer



Artificial Intelligence



Developer

Background: AI Code Fix

Motivation: Pre-empt vulnerabilities

Goal: automate pull-request level fixes

AI Code Fix? Reality Check

Can we do an exact code fix?

Context missing

Complex logic

Code alone not enough

Human insight vital

AI Code Fix: What worked?

Advanced prompting techniques

Quality measurements and logging

Context, context, context

Shifting the Approach

Knowledge + AI

Tailored processes

Proactive solutions

Unified Knowledge Base

How to access: Jira & Slack

Contextual AI

Domain expertise

Capability 1: Code/Config Fixes

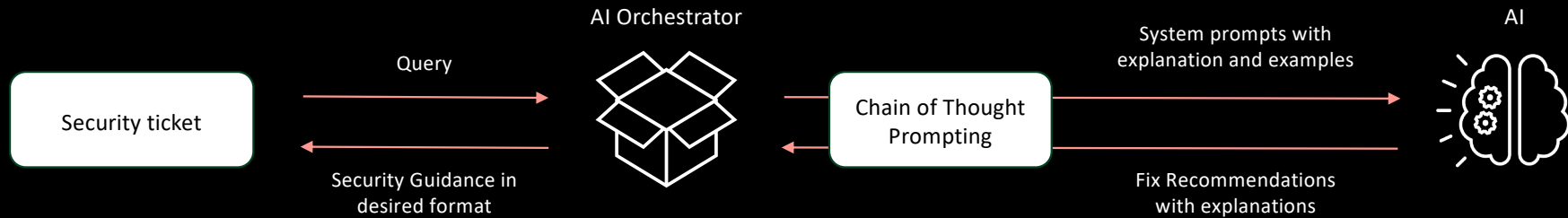
Expectation alignment

Varying ticket quality + details

Deliver a representative example

Capability 1: Code/Config Fixes

You are a cyber security expert with specialized knowledge in fixing vulnerabilities. You will be provided with a vulnerability description. Generate a secure fix for the issue. |



Chain of Thought Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✓

Example code fix

1. Reason for Vulnerable Code

The vulnerability is due to the improper handling and sanitization of user input in the JavaScript file [REDACTED]. This allows attackers to inject malicious scripts that can be executed in the context of the user's browser, leading to Cross-Site Scripting (XSS) attacks.

2. Vulnerable Code

The specific vulnerable code snippet in [REDACTED] is not provided, but a typical example of such vulnerability might look like this:

```
// Vulnerable code example
var userInput = getUserInput(); // Function to get user input
document.getElementById('output').innerHTML = userInput; // Directly inserting user input into the DOM
```

3. Fixed Code

To fix the vulnerability, user input should be properly sanitized before being inserted into the DOM. Using a library like [REDACTED] can help mitigate this issue. Here is an example of how to fix the code using [REDACTED]:

```
// Fixed code example
var xssAPI = CO.shared.XSS.[REDACTED]();
var userInput = getUserInput(); // Function to get user input
var sanitizedInput = xssAPI.isValidHref(userInput); // Sanitize user input
document.getElementById('output').innerHTML = sanitizedInput; // Insert sanitized input into the DOM
```

4. Reason why the suggested fix should work

The suggested fix works by sanitizing the user input using the [REDACTED] library before inserting it into the DOM. This ensures that any potentially malicious scripts are neutralized, preventing them from being executed in the user's browser. By using a trusted library for sanitization, the risk of XSS attacks is significantly reduced, enhancing the security of the application.

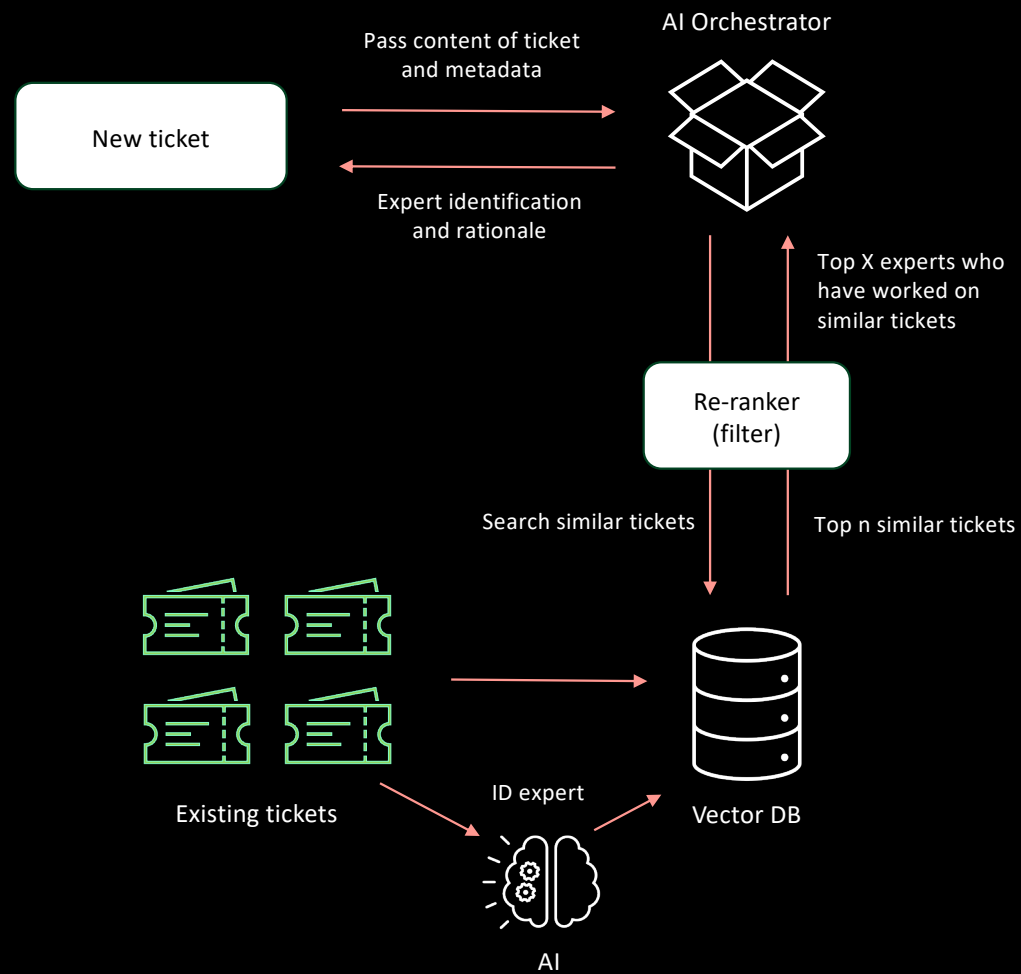
Capability 2: Expert Finder

Who solved it before?

Reduce wait time

Collaboration boost

Capability 2: Expert Finder



Example: Expert Finder

SME: [REDACTED]

Reasoning: The error message indicates a problem with the JSON structure, which aligns with issues resolved in tickets like [REDACTED] and [REDACTED] where incorrect payloads were a common cause of errors. [REDACTED] has been involved in resolving similar issues related to audience lifecycle state updates and JSON payload errors, making him a suitable SME for this case. The suggested resolution is based on the successful strategies employed in these past tickets, focusing on correcting the JSON payload and ensuring all necessary fields are included.

Edit · Delete · Pin · 🗨

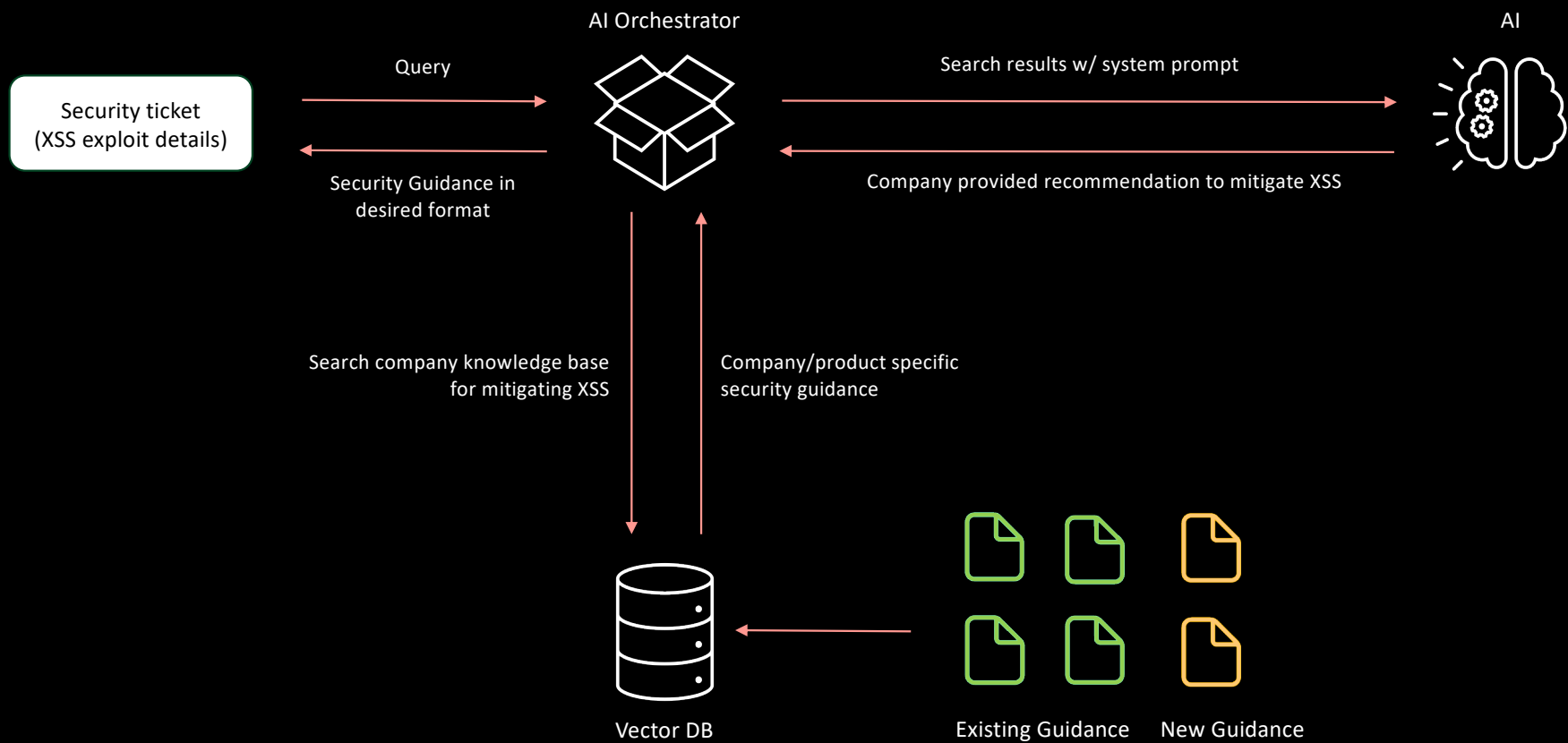
Capability 3: Best Practices

Scale security guidance

Company/product specific recs

RAG-based approach

Capability 3: Best Practices



Example best practice guidance

Summary: Stored Cross-site Scripting (XSS) vulnerability in Adobe [REDACTED].

Vulnerability: Cross-site Scripting (Stored XSS) (CWE-79).

Adobe Specific Recommendations: Since the product is [REDACTED], use Adobe Approved Libraries for sanitization such as XSSAPI. More details can be found in the product's tech stack.

Short Term Fix:

- Implement input validation to ensure that all data received from users is sanitized before being processed.
- Use the Adobe approved library XSSAPI to encode or escape user inputs that are output in HTML or JavaScript contexts.
- Ensure that all dynamic data inserted into HTML or JavaScript is properly sanitized using the XSSFilter library to prevent malicious code execution.

Long Term Fix:

- Conduct regular security audits and code reviews to identify and mitigate XSS vulnerabilities, focusing on areas where user input is handled.
- Implement Content Security Policy (CSP) headers to reduce the risk of XSS attacks by specifying trusted sources for executable scripts.

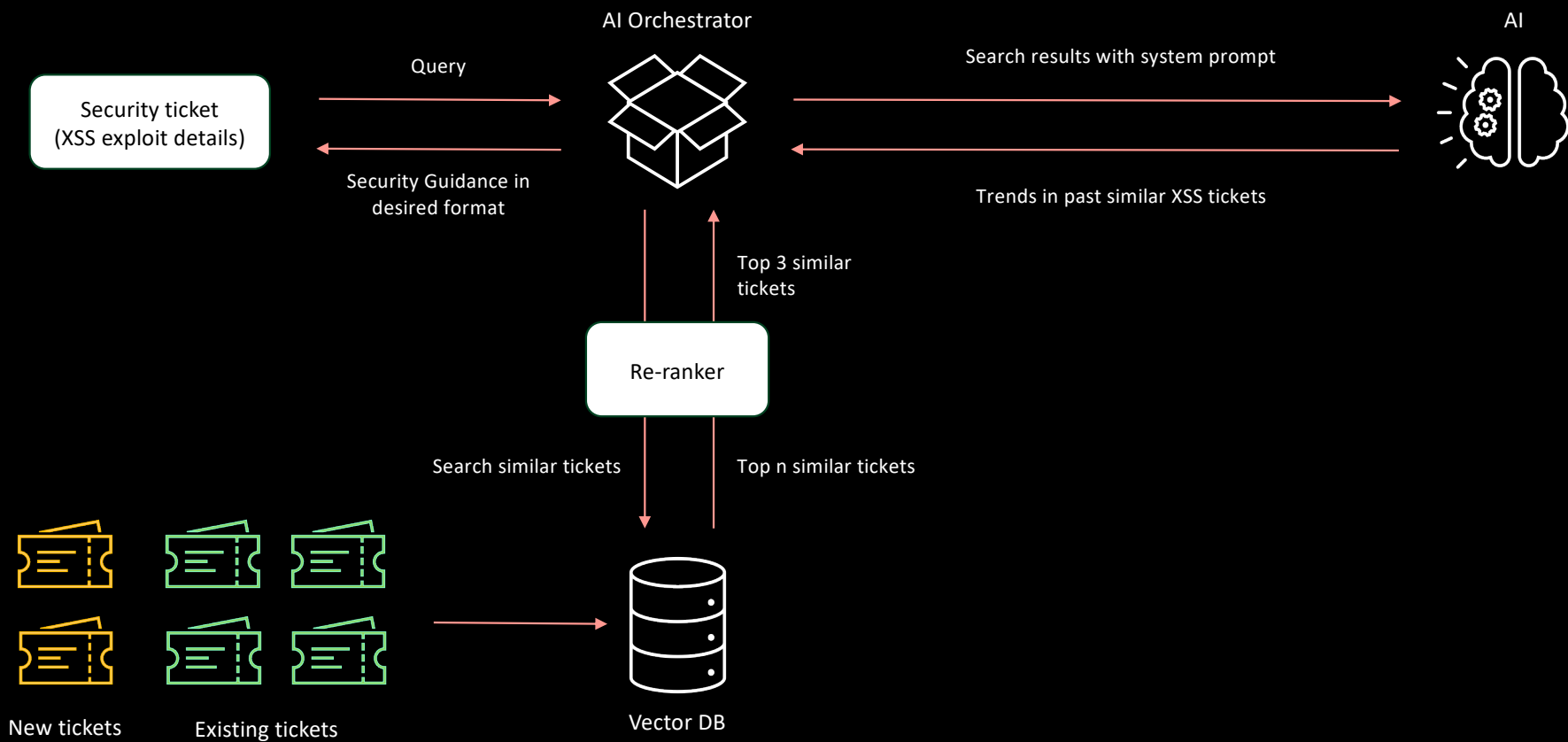
Capability 4: Similar Tickets

Find references

Discover patterns

Spot systemic issues

Capability 4: Similar Tickets



Example similar tickets

Ticket Summary:

- Ticket ID: SEC-123
- Summary: XSS issue in product X code repo

Similar Tickets: We have found 97 tickets in the last 90 days that are similar to this ticket.

Top 3 Similar Tickets:

1. Ticket ID: SEC-400
 - Summary: XSS in product X
 - Similarity Score: 0.911387503
2. Ticket ID: SEC-436
 - Summary: Cross-site scripting bug detected in X
 - Similarity Score: 0.908890724
3. Ticket ID: SEC-629
 - Summary: Injection found - XSS in product X
 - Similarity Score: 0.907649279

Status Trends: This data shows a count of the statuses of the 97 similar tickets.

- In Progress: 31
- Open: 64
- Blocked: 2

Severity Trends: This data shows a count of the severity of similar tickets that are UNRESOLVED.

- Number of Unresolved Similar Tickets: 97
- Critical: 9
- High: 28
- Medium: 36
- Low: 24

Feedback & Evaluations

Human feedback mechanisms

Model output evals

Continuous improvement

Learnings

Assess knowledge gaps

Why did this work for us?

Be where developers are

Iterate & scale

Key Takeaways

AI can help reduce human toil

Context enriches AI

Empower teams

You can build this too!

Thank you!

Q&A

Connect with us :)



Joseph Seasley



Shruti Datta Gupta