RAGs to reqs

Irene Michlin

AppSec Lead, Neo4j

DEEPSEC

Making ASVS more accessible through graphs and LLMs

DEEPSEC

Agenda

1. Architecture and concepts
2. Demo
3. How we use it
4. Ideas for future development

DEEPSEC

What is ASVS?

The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls and also provides developers with a list of requirements for secure development.

Many ways to use it

- Secure Coding
- Security Archit
- Guide for auto
  integration tes
- Secure Develo
- Framework for
  secure softwar
- Agile Applicati

DEEPSEC

Example

V3.1 Fundamental Session Management Security

#	Description	L1	L2	L3	CWE	NIST §
3.1.1	Verify the application never reveals session tokens in URL parameters.	✓	✓	✓	598	

V3.2 Session Binding

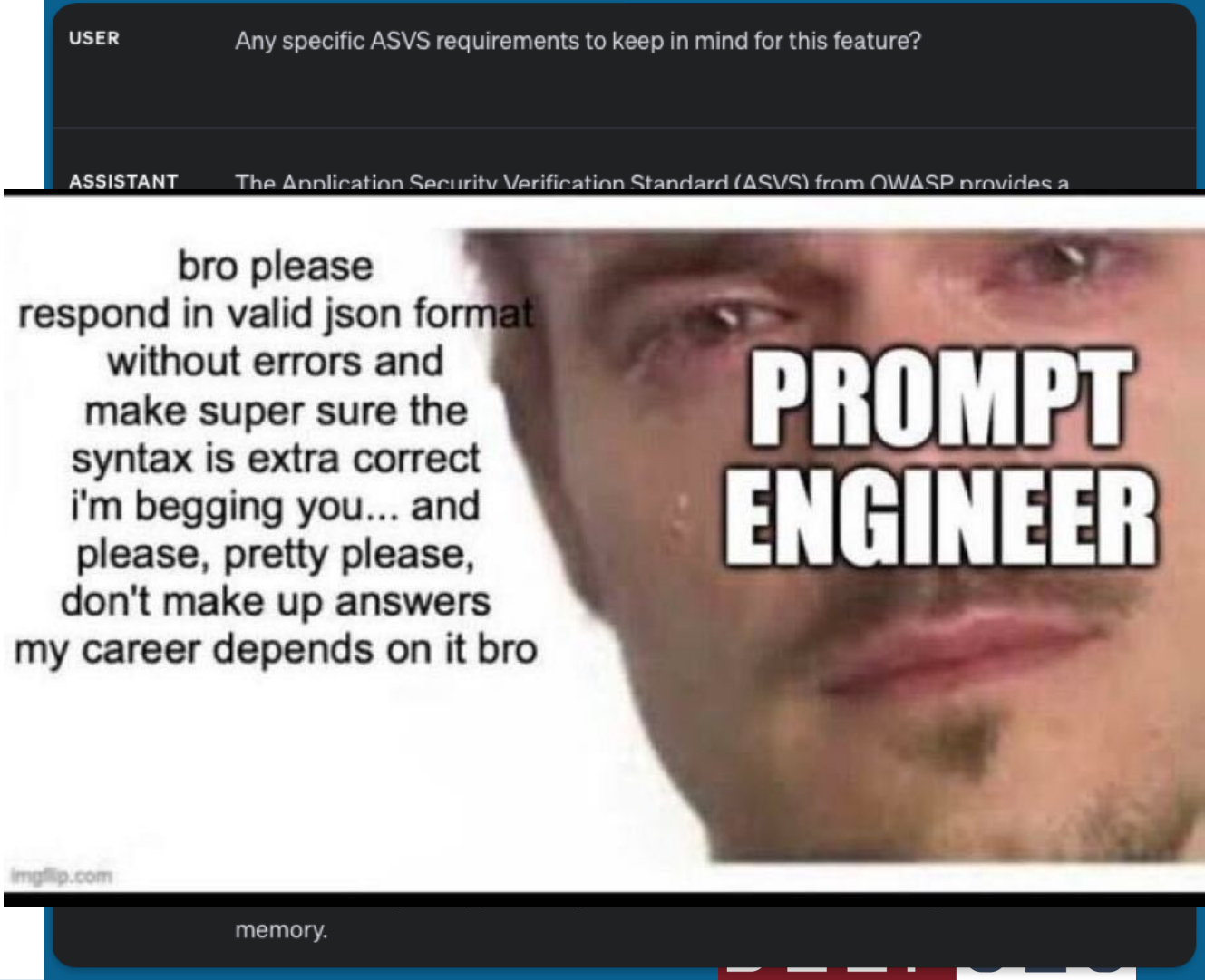
#	Description	L1	L2	L3	CWE	NIST §
3.2.1	Verify the application generates a new session token on user authentication. (C6)	✓	✓	✓	384	7.1
3.2.2	Verify that session tokens possess at least 64 bits of entropy. (C6)	✓	✓	✓	331	7.1
3.2.3	Verify the application only stores session tokens in the browser using secure methods such as appropriately secured cookies (see section 3.4) or HTML 5 session storage.	✓	✓	✓	539	7.1
3.2.4	Verify that session tokens are generated using approved cryptographic algorithms. (C6)		✓	✓	331	7.1

Tailor it!

https://www.infoq.com/presentations/sustainable-security-requirements-asvs/

DEEPSEC

Lies, big lies, and LLMs

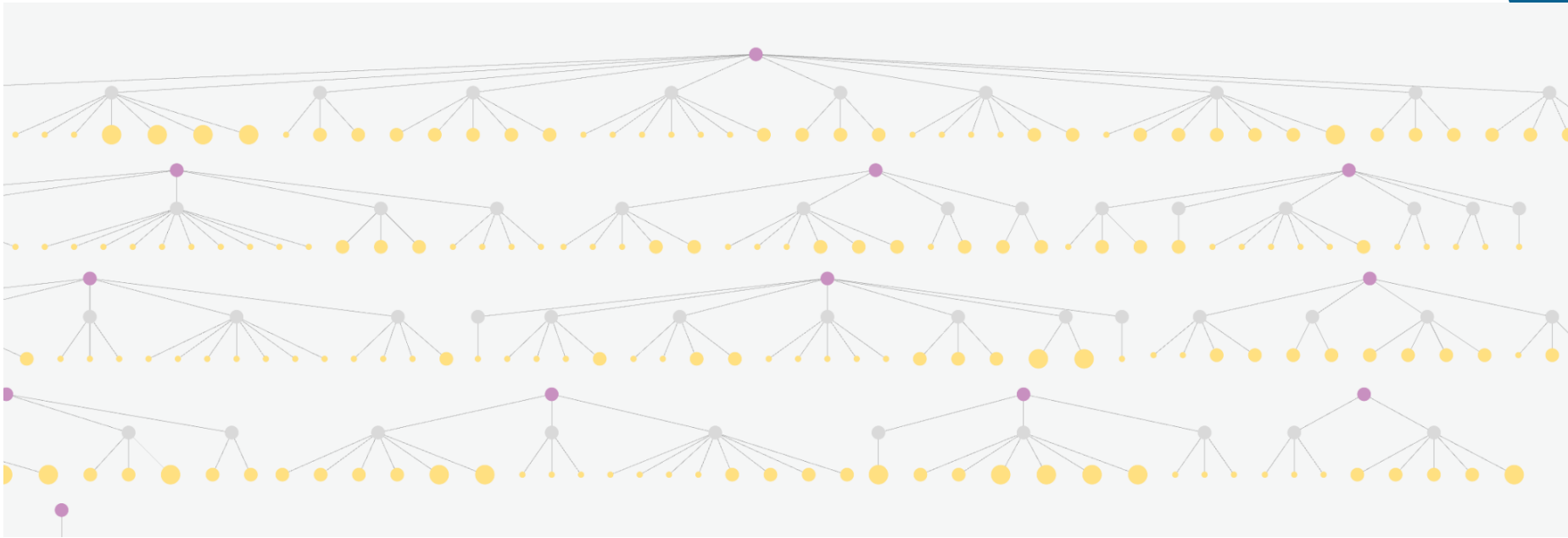


ASVS is “graphy”

Chapter->Section->Requirement

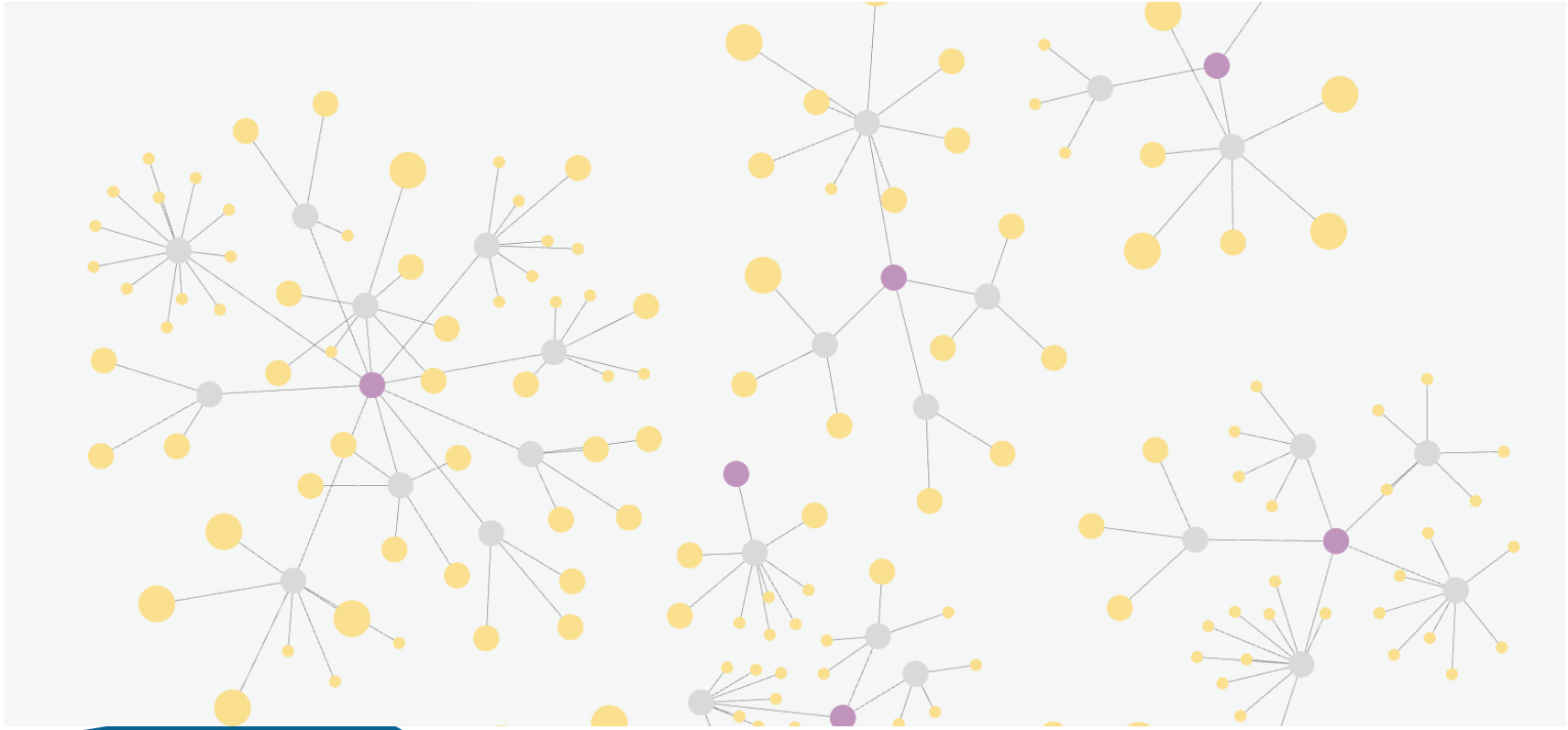
V4 -> V4.1 -> V4.1.1

Hierarchical view



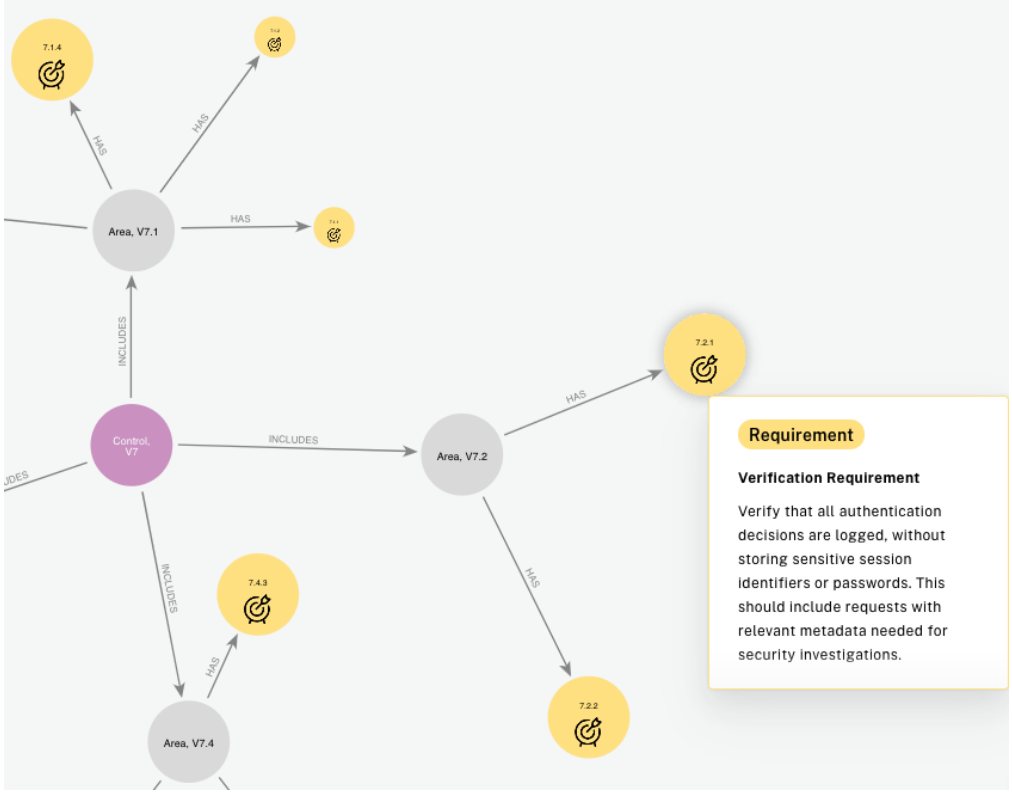
DEEPSEC

Force-based layout



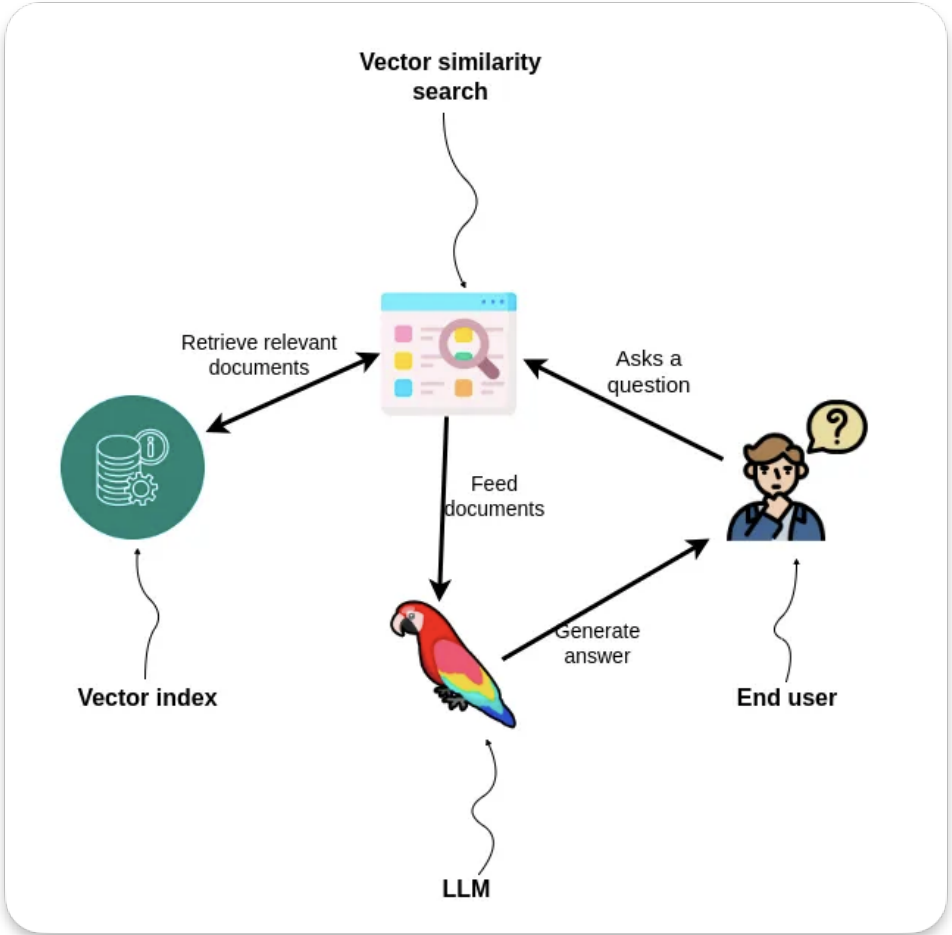
DEEPSEC

Zoom-in



DEEPSEC

Architecture



DEEPSEC

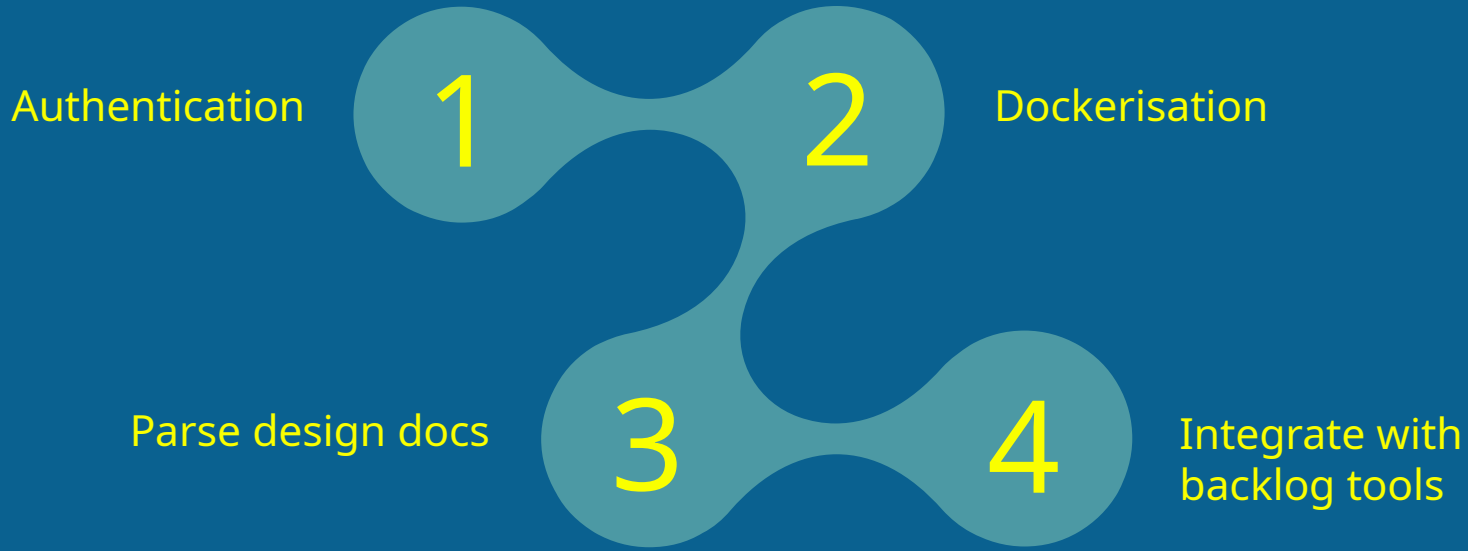
Demo

All the tools you need for this are free: https://github.com/neo4j-examples/appsec-asvs-bot

(except for OpenAI subscription)

DEEPSEC

Ideas for further development



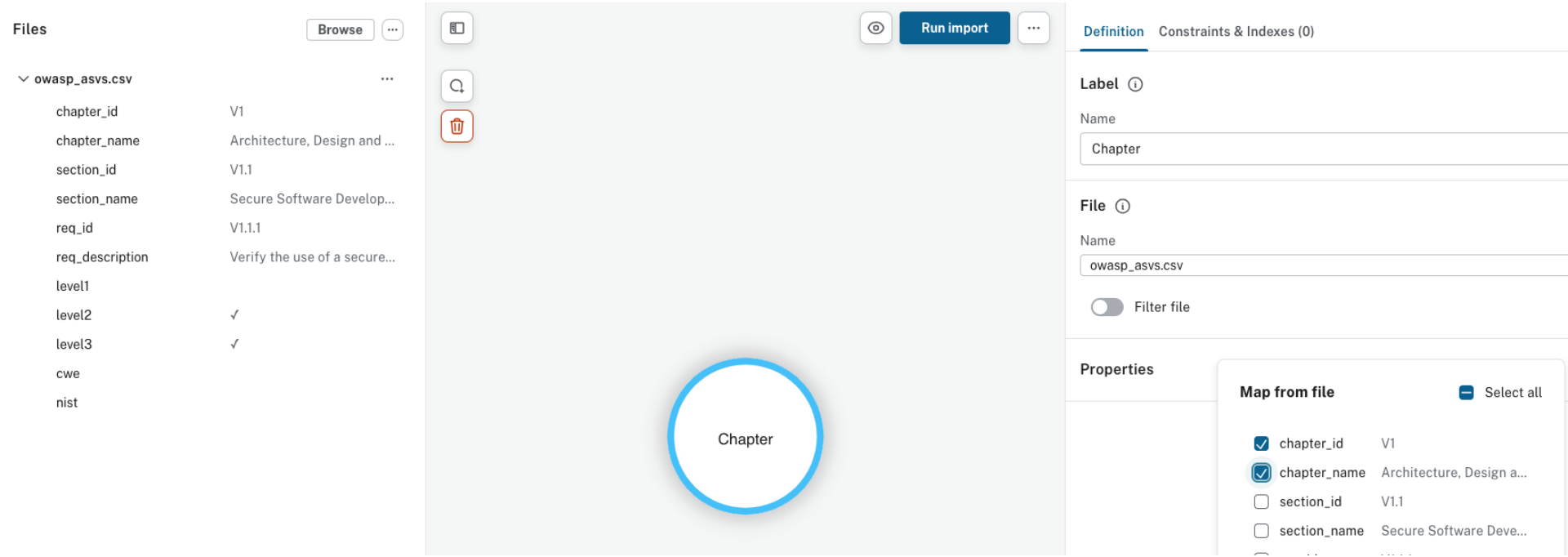
DEEPSEC

How to update my graph?

V5 is coming

Maybe 4.0.4 is coming?

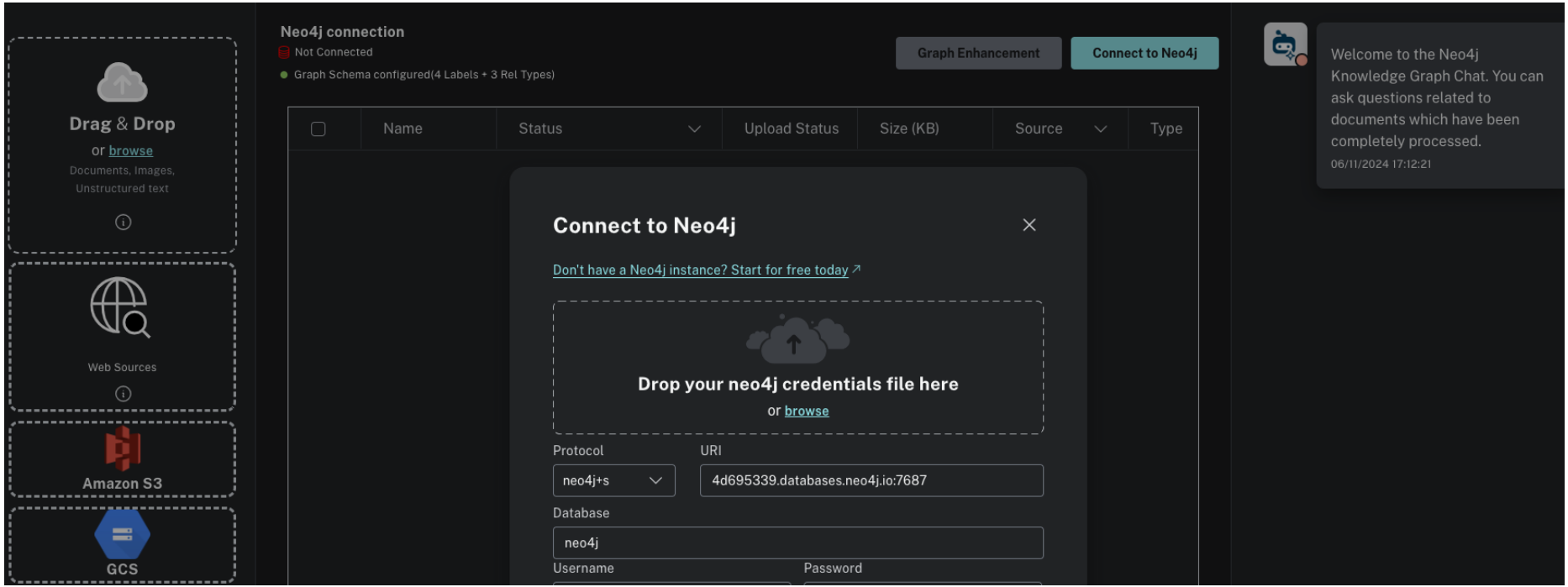
Use Data Importer



https://workspace-preview.neo4j.io/workspace/import

DEEPSEC

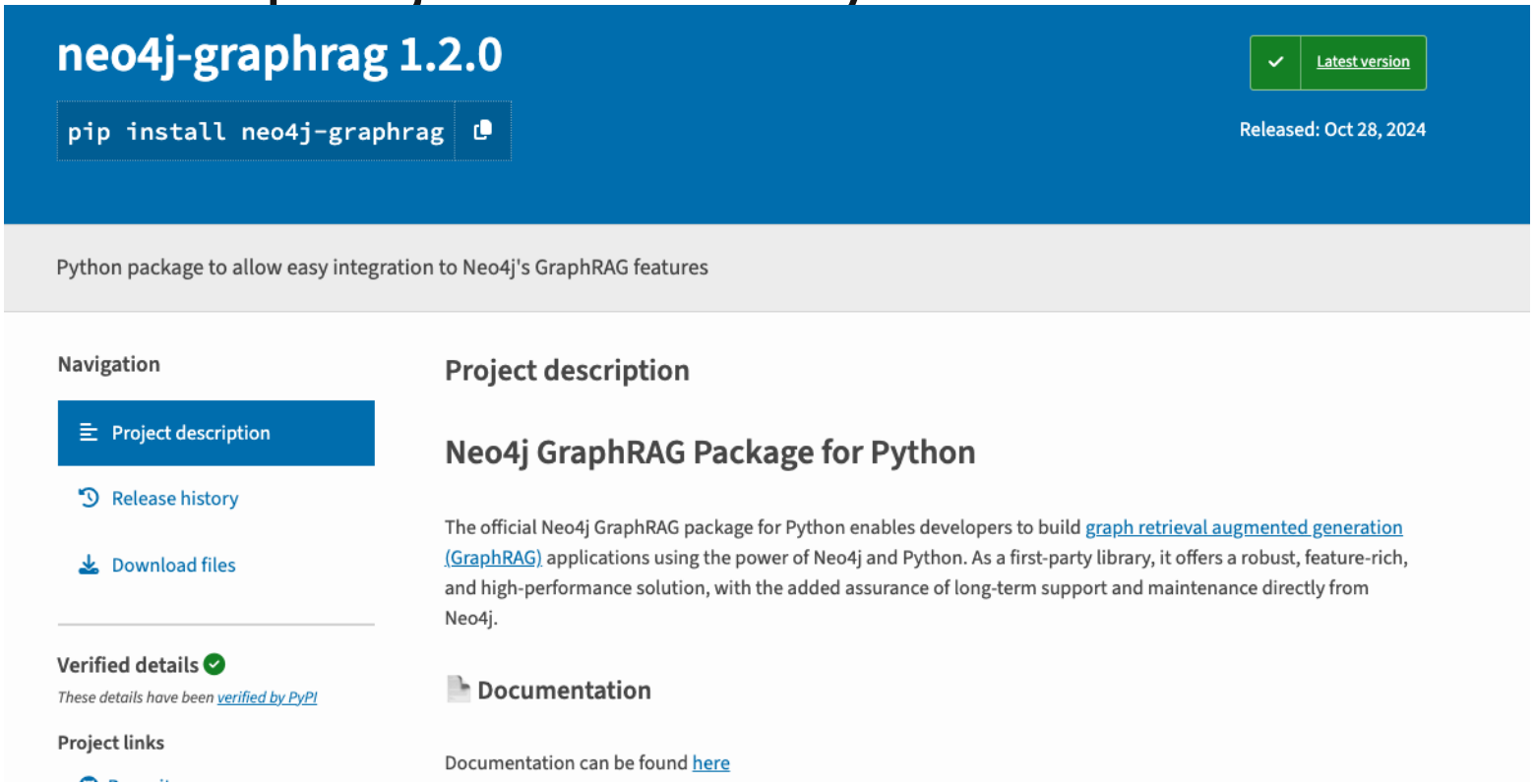
Use LLM Knowledge Graph Builder



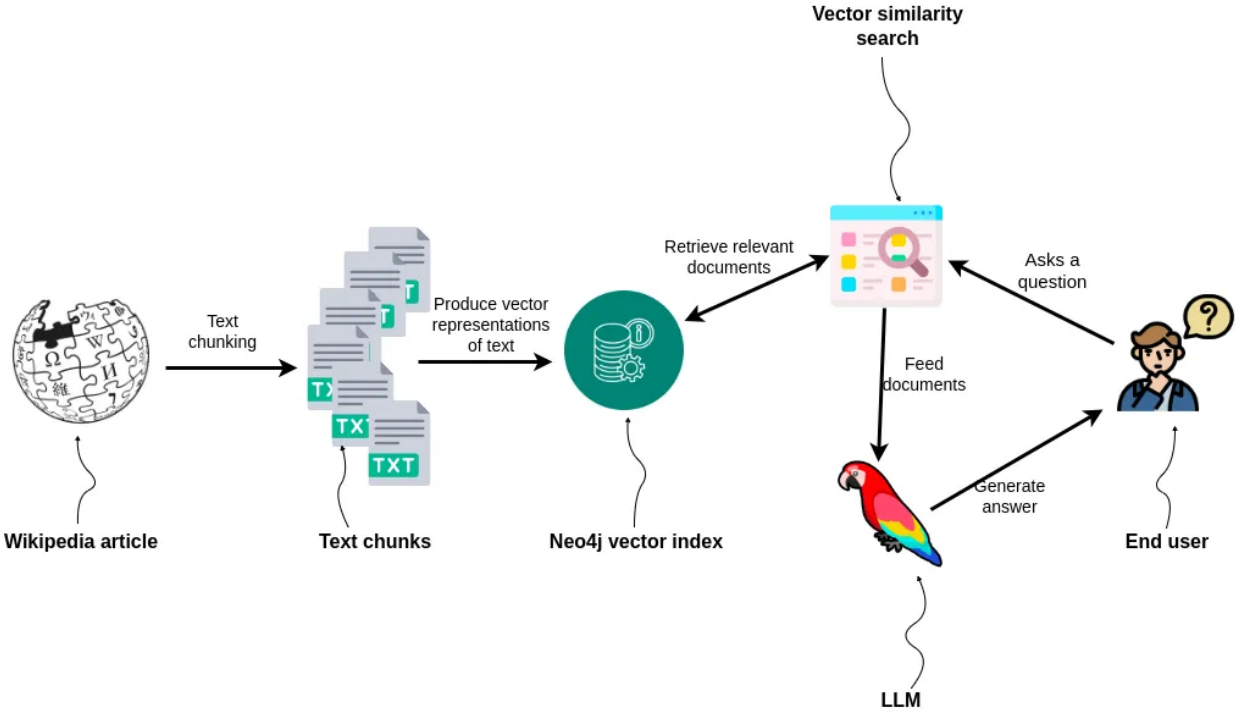
https://llm-graph-builder.neo4jlabs.com/

DEEPSEC

Build it programmatically



First thought - chunks



DEEPSEC

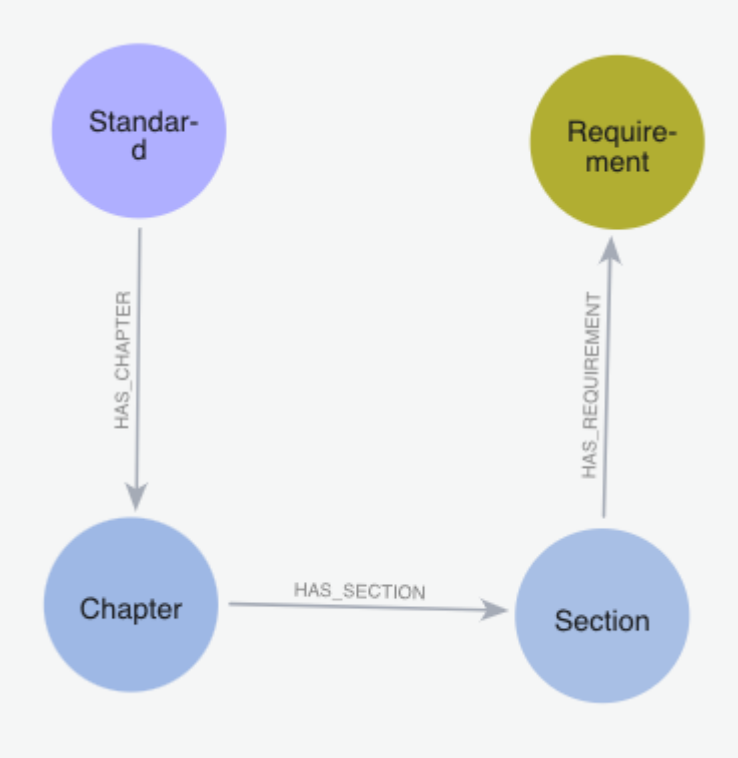
Taking advantage of the structure

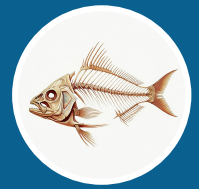
- https://medium.com/@irina.karkkanen/rag-on-graph-db-using-fixed-entity-architecture-make-you-retrieval-work-for-you-f4bfcac5277f
- https://medium.com/@irina.karkkanen/three-layer-fixed-entity-architecture-for-efficient-rag-on-graphs-787c70e3151a

- Ontological fishbone!

DEEPSEC

Ontological fishbone

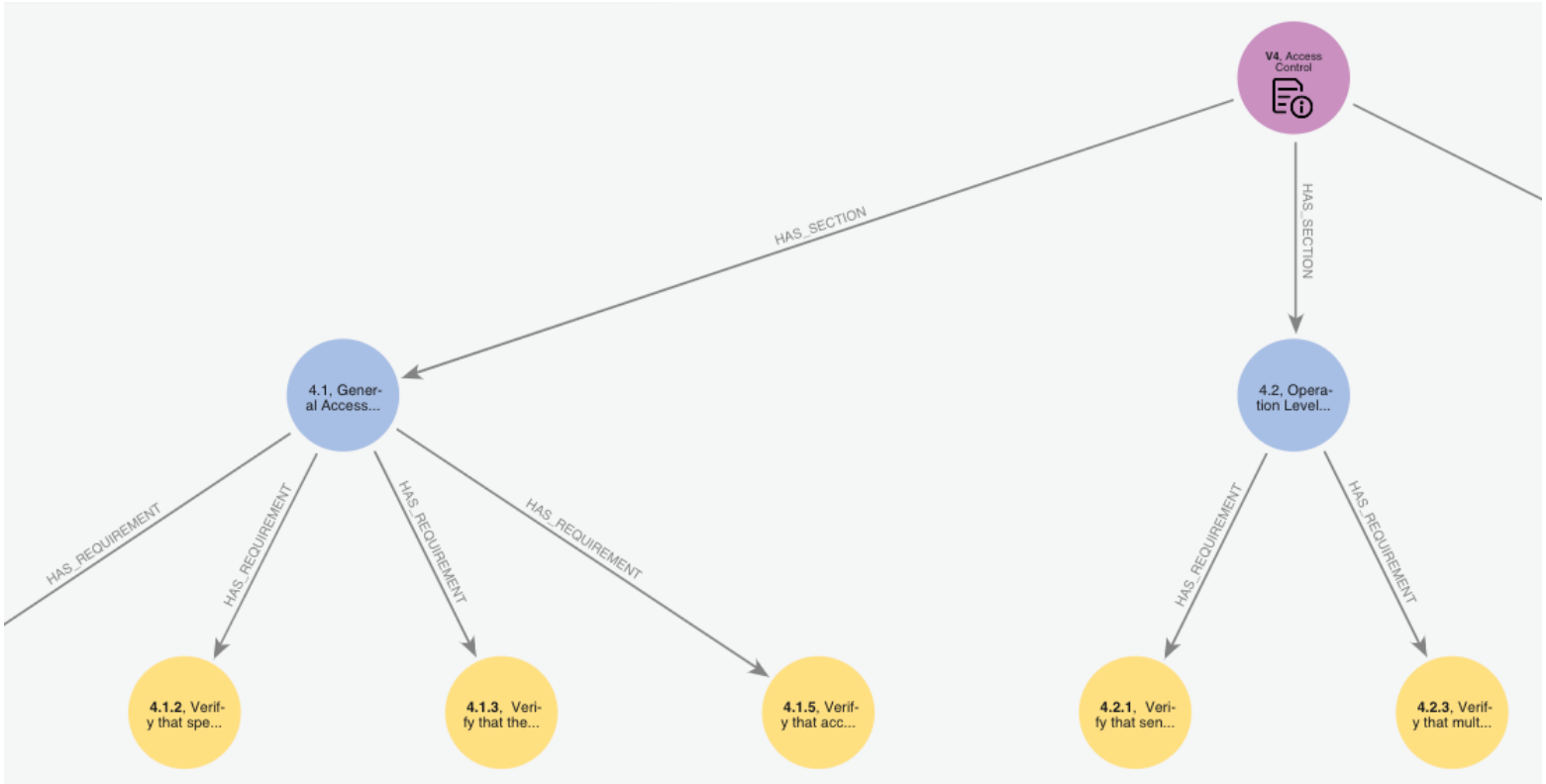




In many organizations, subject matter experts can easily identify critical entities and relationships within well-defined domains. This foundational knowledge is essential for building effective knowledge graphs. By leveraging this expertise, a basic “fishbone” of key entities and their relationships can be established.

DEEPSEC

Next version



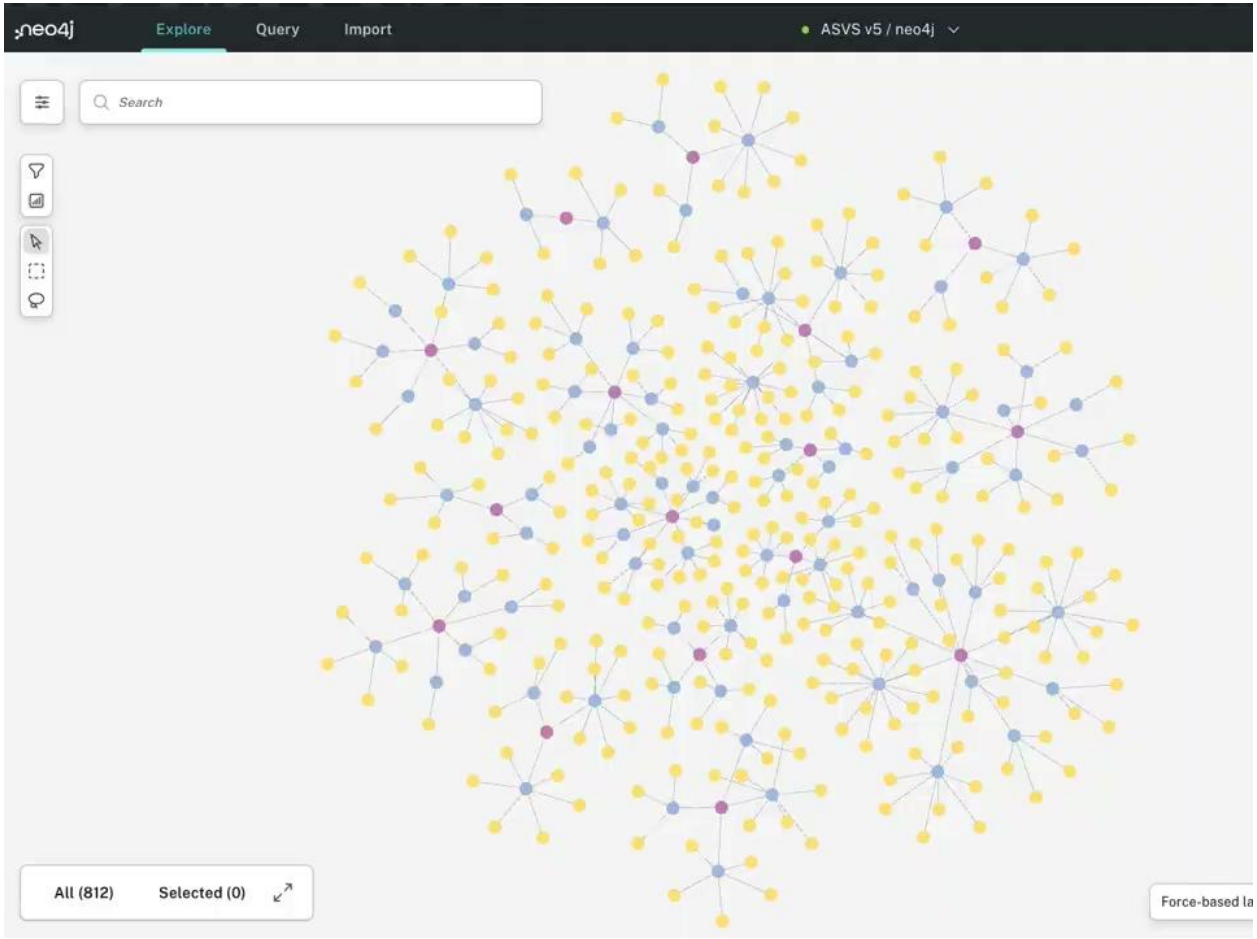
DEEPSEC

What’s different?

- Uses ASVS V5
- Embedding is a property rather than a node
- Chapters, sections, requirements all have embeddings
- Uses EMBEDDING_MODEL = “text-embedding-3-small”
- Comes with full text indexes in addition to the vector index

https://github.com/neo4j-examples/appsec-asvs-bot /tree/ASVS-v5

DEEPSEC



DEEPSEC

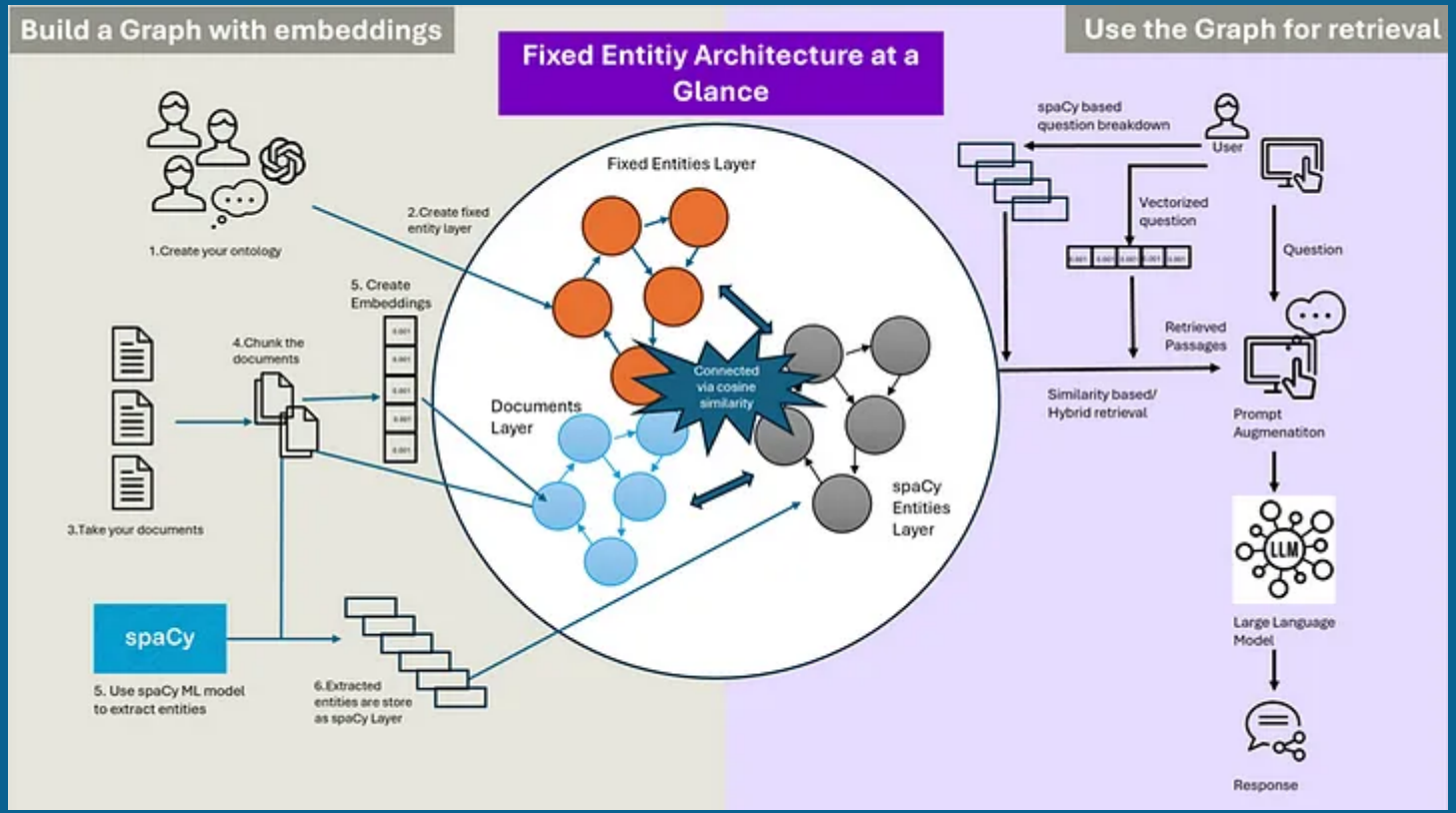
```
MATCH (m1:Requirement{ID:'2.9.1'})
        WITH m1, m1.embedding as e

        // 2. Find other requirements which have high semantic similarity on description
        CALL db.index.vector.queryNodes("embeddingIndex", 10, e) YIELD node, score
        WITH m1, node AS m2, score
        WHERE m2:Requirement and score < 1.0  and score > 0.80 // exclude self and low-scoring matches

        // 3. For returned Requirement nodes, check they are in the same section and/or chapter
        // Counting hops between the two requirements. 2 means same section, 4 means same chapter

        WITH m1, m2, score
        MATCH (m1)-[r*..6]-(m2)

        // 4. Use structure to calculate weights and apply to similarity score
        WITH m1, m2, score,

           CASE size(r)
              WHEN 2 THEN 2
              WHEN 4 THEN 1+log(2)
              ELSE 1
           END AS proximity

        RETURN  m2.ID AS reqNumber, m2.`Description` AS requirement, score,
         proximity * score AS rank
        ORDER BY rank DESC
```

Taking advantage of the structure

- https://medium.com/@irina.karkkanen/rag-on-graph-db-using-fixed-entity-architecture-make-you-retrieval-work-for-you-f4bfcac5277f
- https://medium.com/@irina.karkkanen/three-layer-fixed-entity-architecture-for-efficient-rag-on-graphs-787c70e3151a

DEEPSEC

What's next?



DEEPSEC

Thank you!

https://github.com/neo4j-examples/appsec-asvs-bot

irene.michlin@neo4j.com

