Gemara: The GRC Architecture You Didn't Know You Built

GRC Engineering Model For Automated Risk Assessment

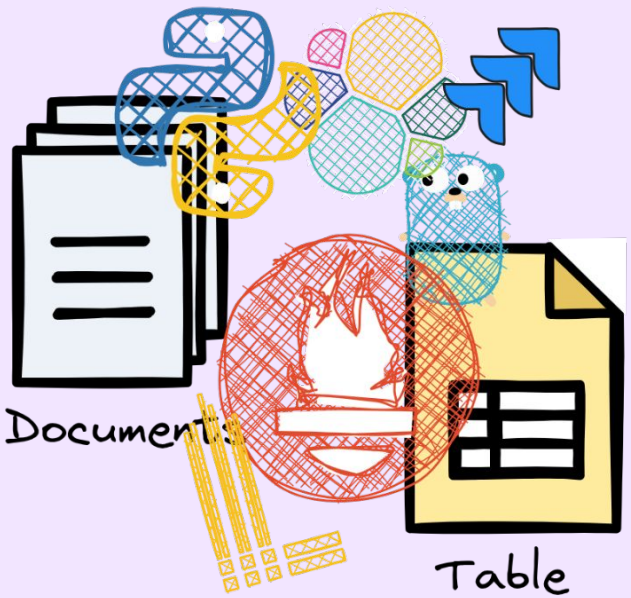
Hannah Braswell

Gemara Community Manager

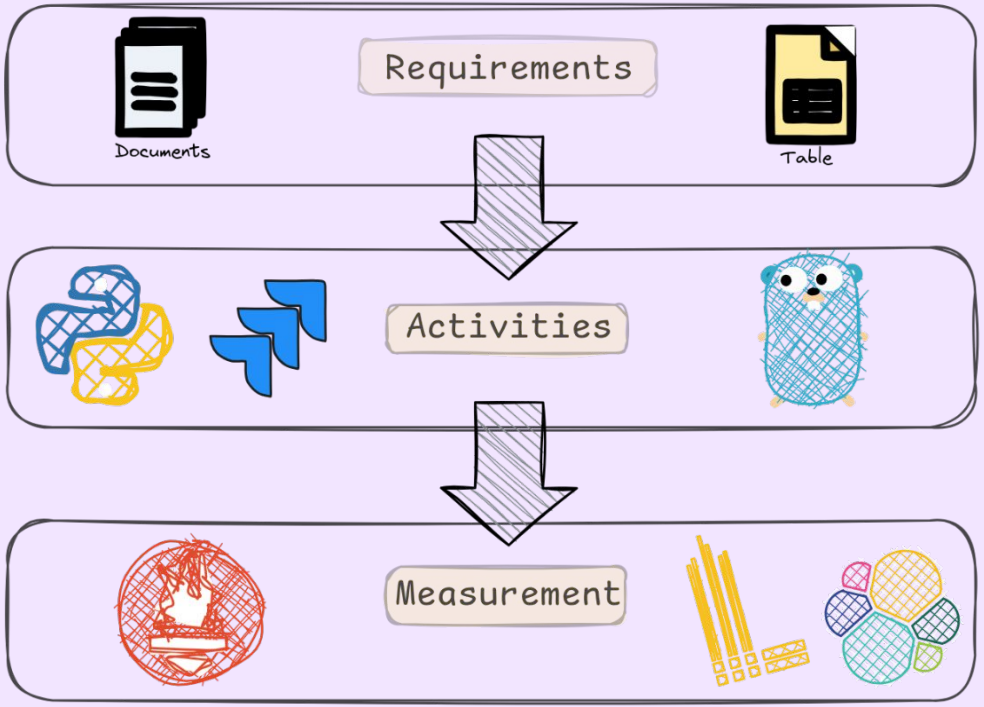
Jenn Power

Gemara Maintainer / WG ORBIT TSC

Before

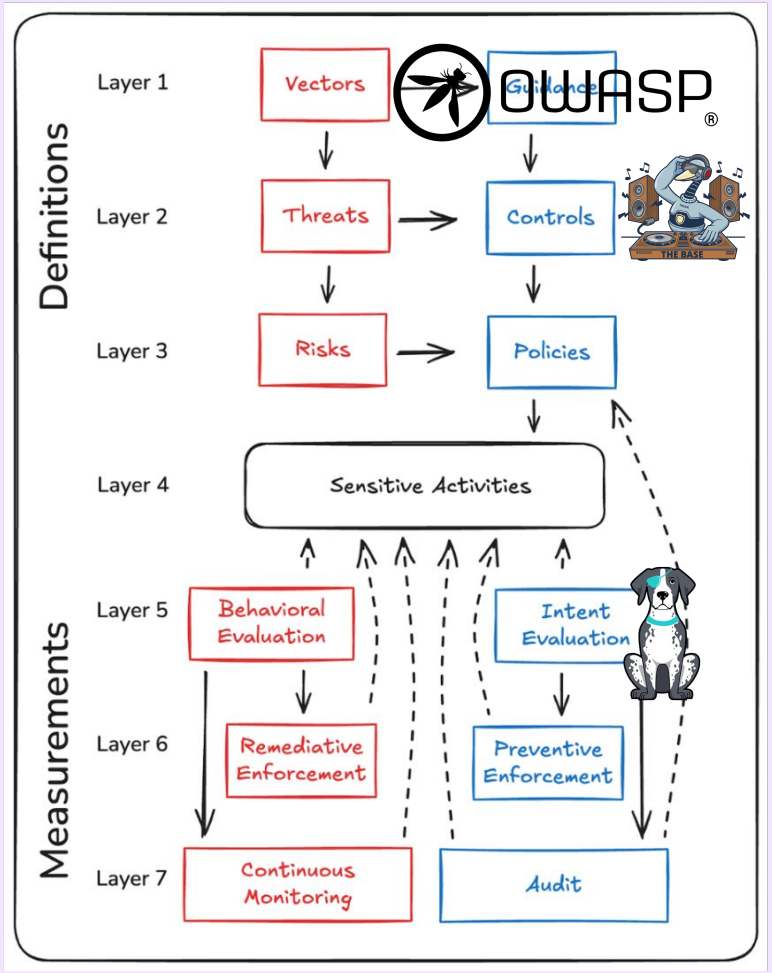


After



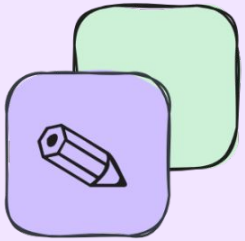
The 7 Layer Architecture

- Inspired by the OSI Model
- Evolved from the CNCF AGMM[1]
- Underpins OSPS Baseline & FINOS CCC



1: https://tag-security.cncf.io/community/resources/automated-governance-maturity-model/

Implementing the Model



The Schemas

An implementation of the model in CUE

Data fields

Semantic Constraints

```
// Catalog describes a set of topically-associated entries
#Catalog: {
    // title describes the purpose of this catalog at a glance
    title: string

    // metadata provides detailed data about this catalog
    metadata: #Metadata @go(Metadata)

    // groups contains a list of groups that can be referenced by entries in this c
    groups?: [#Group, ...#Group]

    // extends references catalogs that this catalog builds upon
    extends?: [...#ArtifactMapping] @go(Extends)

    imports?: [#MultiEntryMapping, ...#MultiEntryMapping]

    if groups != _|_ {
        _uniqueGroupIds: {for i, g in groups {(g.id): i}}
    }

    if extends != _|_ {
        metadata: "mapping-references": [#MappingReference, ...#MappingReference]
    }

    if imports != _|_ {
        metadata: "mapping-references": [#MappingReference, ...#MappingReference]
    }
}
```

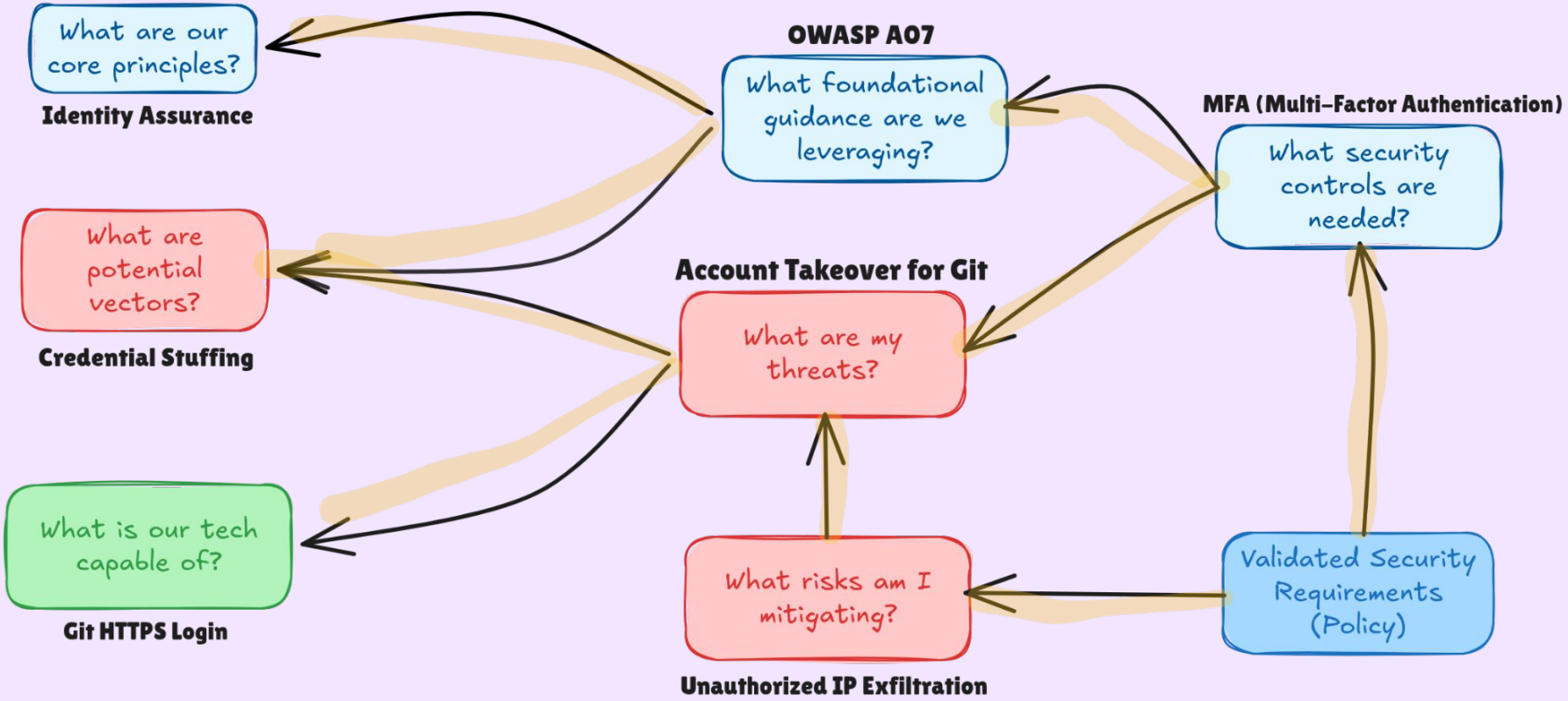

Documents


Logs


Catalogs

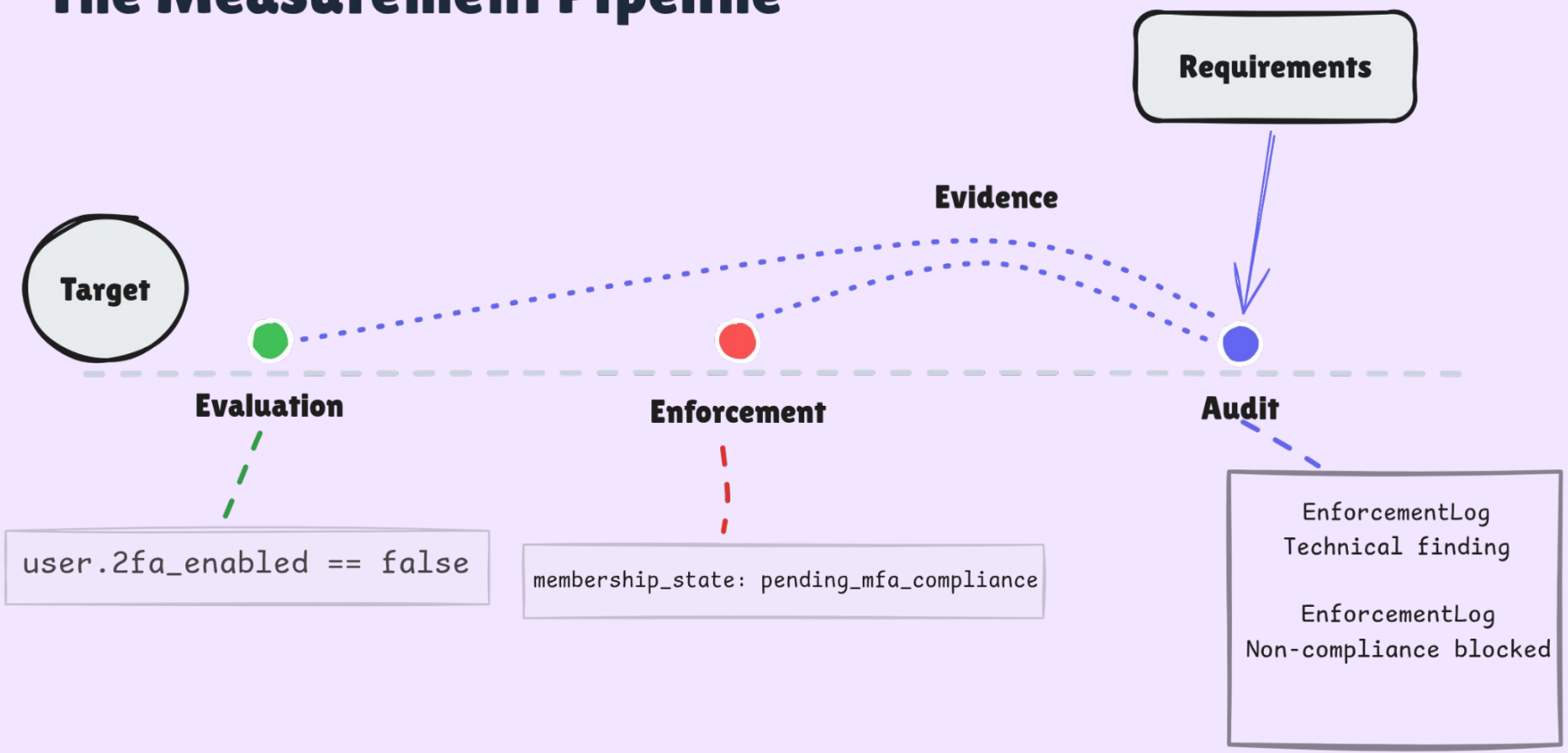
https://registry.cue.works/docs/github.com/gemaraproj/gemara@v1

Building a Better Security Requirement



The Measurement Pipeline

No MFA? No org membership.

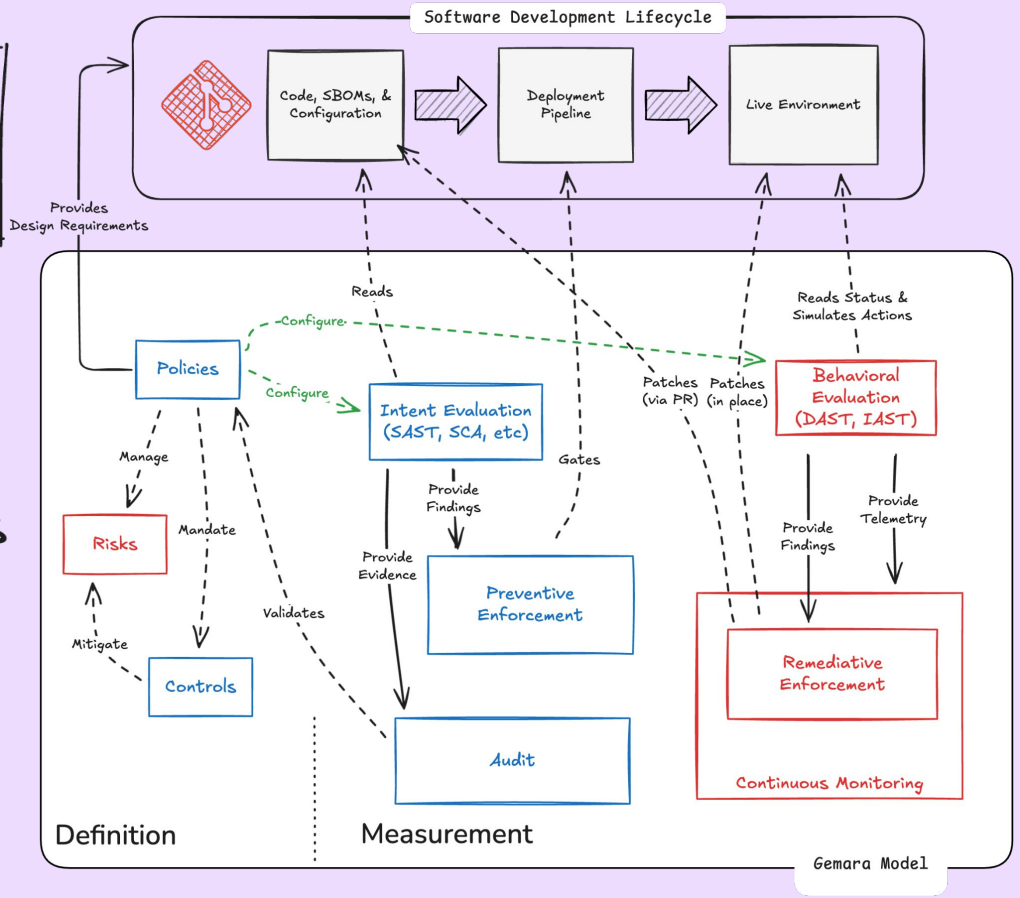


A Case Study: 2026 Security Slam

How Cloud Native PostGres put Gemara into practice

Resulting in

- New Contributors equipped to understand security considerations
- Project adopters can make informed decisions
- Strengthened security posture with self-assessment



Time for some critical thinking

01. What can your project do? - capabilities.yaml

```yaml
- id: CNPG.CP04
  title: Declarative Database Objects
  description: >-
    The operator manages declarative databases, publications, and
    subscriptions as dedicated CRDs, and managed roles within the
    Cluster spec. These are reconciled continuously to match the
    desired state.
  group: data-management
```

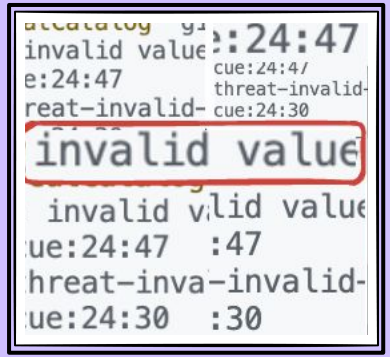
02. What could go wrong? - threats.yaml

```yaml
- id: CNPG.TH02
  title: Kubernetes Secret Exposure
  description: >-
    Database credentials, replication certificates, cloud provider
    credentials for object store access (AWS keys, Azure storage
    keys, GCP application credentials), and other sensitive material
    are stored in Kubernetes Secrets, which are base64-encoded but
    not encrypted at the application level. Compromise of etcd, RBAC
    misconfiguration, or excessive permissions on secrets in the cluster
    namespace could expose all credentials simultaneously. The operator does
    not provide built-in credential rotation. Users are responsible for
    enabling etcd encryption at rest, restricting RBAC on secrets, using
    external secret managers if needed, and implementing credential rotation
    procedures.
  group: credential-management
  capabilities:
    - reference-id: CNPG.CAPABILITIES
      entries:
        - reference-id: CNPG.CP01
        - reference-id: CNPG.CP03
        - reference-id: CNPG.CP04
```

```
cue vet -c -d '#Catalog'
github.com/gemaraproj/gemara@v1 catalog.yaml
```

Do I need to know CUE?

NO!



CUE Validation error.
Try again.

The Anatomy of a Security Control

1 Defining the control.

```yaml
- id: CNPG.CN02
  title: Protect Kubernetes Secrets
  objective: >-
    Ensure that database credentials, replication certificates, and cloud
    provider credentials stored in Kubernetes Secrets are protected from
    unauthorized access and exposure.
  group: data-protection
  state: Active
```

The "What"

2 How to satisfy the control

```yaml
assessment-requirements:
  - id: CNPG.CN02.AR01
    text: >-
      When the Kubernetes cluster stores CNPG secrets, etcd encryption
      at rest MUST be enabled to protect secret data on disk.
    applicability:
      - all-deployments
    recommendation: >-
      Enable the EncryptionConfiguration API resource in the Kubernetes
      API server to encrypt Secret resources at rest in etcd using
      AES-CBC or AES-GCM.
    state: Active
```

The "How"

mitigates

```yaml
threats:
  - reference-id: CNPG.THREATS
    entries:
      - reference-id: CNPG.TH02
  - reference-id: CCC
    entries:
      - reference-id: CCC.Core.TH01
```

The "Why"

because of

4 Back to where we started. Where we began.

```yaml
id: CNPG.CP04
title: Declarative Database Objects
```

imported

3 Why you need to implement the control and what threat(s) mitigated.

Gemara Roadmap

Whitepaper

March 2026

Gemara v1 Spec

April 2026

v1 Tooling

...Gemara MCP Server

...Ecosystem tooling

...Reusable content

Join Us

Gemara Website

- Tutorials
- The Model
- ADRs

GitHub Issues

- Start contributing
- Comment on an issue

Resources

- GRC Engineering You Didn't Know You Built Repo
- Gemara Official Publication
  - Introduction to the Gemara Model
- CUE Labs Blog on Perses and Gemara making configuration user-friendly

Ecosystem Tools

- OSPS Baseline Scanner GitHub Action
- Security-baseline repo
- ORBIT Working Group
- Go Gemara SDK
- Gemara MCP Server
- FINOS CCC: reusable catalogs for Controls, Capabilities, and Threats.