Continuous Auditing Implemented

itpedia.nl/2021/06/14/continuous-auditing-implemented/

CA community June 14, 2021

Dit artikel beschrijft een prototype voor een Continuous Auditing (CA)-tool op basis van het CA-concept en CA-ontwerp. In het eerste artikel is het concept van CA uiteengezet waarna we in artikel 2 een ontwerp voor implementatie hebben gemaakt die we in dit artikel concreet uitwerken in een werkende oplossing / Minimal Viable Product (MVP): Continuous Auditing Implemented.

Dit derde artikel heeft een meer technische insteek en is bedoeld voor architecten, ontwikkelaars / engineers en technical auditors. Er wordt een voorbeeld uitgewerkt van een control die continue wordt gemeten en het resultaat van de business rule (toetsen van de control met de norm) wordt gepresenteerd door middel van een dashboard welke real-time wordt bijgehouden.

Deze MVP is een eerste aanzet (werkend prototype) tot het ontwikkelen van een volledige CA-tool en biedt een goede basis voor verdere uitbreiding en ontwikkeling. Voor deze MVP is gebruik gemaakt van technologieën / platformen zoals Flask, Python, ProgreSQL en Azure.

MVP: Continuous Auditing Implemented

Een MVP voor een CA-tool wil zeggen dat er een tool is ontwikkeld om een constante informatiestroom te creëren ten einde de effectiviteit van controls vast te stellen. De gekozen functionaliteit is voor deze MVP conform het CA-ontwerp, maar dan alleen voor wat betreft één managed object (SQL Server), één risk (capaciteit tekort), één control (bewaking capaciteit) en een aantal capaciteitsmetingen (evidence).

MVP intro

Om ervoor te zorgen dat alle managed objects op dezelfde manier behandeld worden, wordt gebruik gemaakt van een auditinstrument dat zorgt dat alles op dezelfde manier getoetst wordt. De bedoeling van deze MVP is om te komen tot een werkend concept (Continuous Auditing Implemented) dat door meer bedrijven kan worden toegepast als open source.

Wanneer meer bedrijven de controls op dezelfde manier toetsen wordt een beter inzicht verkregen in de diverse risico's van en de effectiviteit van de controls. Deze informatie kan weer gedeeld worden onder de CA-tool community.

- Een voorbeeld risico is ongeautoriseerde toegang (R101)
- Voorbeelden van controls voor dit risico zijn:
    C101-1. De password control die bij alle users afdwingt dat een password policy niet langer dan 90 dagen ongewijzigd blijft.
    C101-2. MFA is enabled voor elke gebruiker.
    C101-3. Intruders van het netwerk (SSH verkeer) worden gemonitord.

De controls voor de risico's kunnen het beste worden gedefinieerd in een standaard format als de Gherkin language (given – when – then) zoals besproken in ons eerste artikel.

#Control: C101 – 1 (ISO 27001:2013 Annex Axyz)

GIVEN the users need to change their passwords at least once per 90 days
WHEN a time period of 90 days has elapsed
THEN the user gets an popup screen to reset the password

Deze control kan worden gemonitord met de CA-tool door evidence op behalen die aantoont dat C101-1 inderdaad van kracht is voor alle gebruikers. De implementatie is dan een REST API call to Azure which return the expiration date of a password.

MVP uitgelegd

De MVP is opgebouwd uit de volgende componenten:

Tool/Techniek	Functie	Objecten
Flask	Gebruikersinterface	index.html
Python	Applicatie	app.py
ProgreSQL	Database	ca_schema.py ca_populate.py
Azure	Managed objecten	Rest API

Tabel 1. CA-tool functies

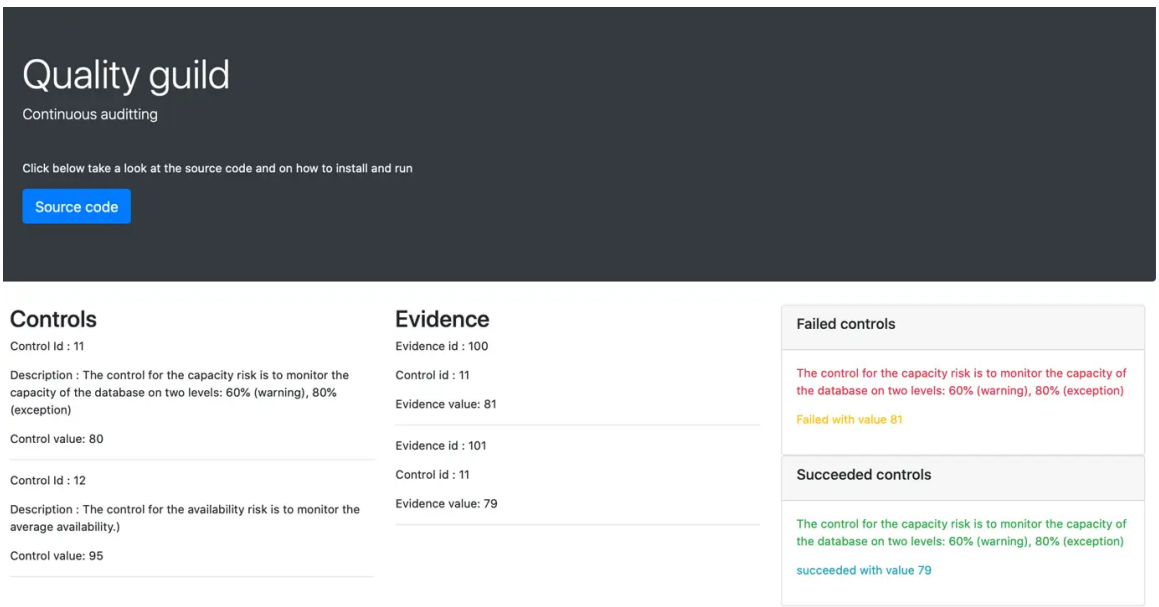
Hieronder volgt een beschrijving van elke functie uit bovenstaande tabel.

Gebruikersinterface

Voor de gebruikersinterface wordt de functionaliteit, de roadmap en de technology besproken en geëvalueerd.

GUI Functionaliteit

De gebruikersinterface is tot de minimale functionaliteit beperkt. Links staan de controls met een control ID, control definitie en control threshold value. In het midden staan de evidence value die verkregen zijn uit de monitoring middels de Rest API en rechts staan de resultaten van de vergelijking van de control threshold value en de evidence value.



Figuur 1. Gebruikersinterface CA-tool.

GUI Roadmap

De userinterface is een eerste opzet en alleen voor demo doeleinden geschikt. Voor een volwaardige CA-tool moet de CRUD (Create-Read-Update-Delete) functionaliteit voor risicobeheer, Managed Objects, Controls en Evidence worden ingericht. Hierbij moeten de objecten aan elkaar gerelateerd kunnen worden en moet er een rapportage functionaliteit komen.

Ook moet er een dashboard komen met een drill down interface van value stream en use cases naar managed objects inclusief de cascade van verkeerslichten. Een use case is pas groen als alle managed objects die gerelateerd zijn ook groen zijn.

De presentatie van de informatie moet vooral gericht zijn op het in één oogopslag overzien van de mate waarin de value stream in control is. Tegelijkertijd is een interface bedoeld om een relatie te leggen tussen de controls en een normenkader zoals ISO 27001:2013 alsmede de mate waarin die control beheerst is. Hiertoe dient een goed navigatiepad te worden gedefinieerd zodat per Annex A control direct gekeken kan worden welke risico's in welke mate beheerst zijn op basis van de verkregen evidence van die control.

GUI Technology

De Flask GUI biedt een zeer beknopte maar krachtige web interface die snel en simpel is op te bouwen. Er zijn voorgedefinieerde componenten die het mogelijk maken om snel een goed ogende GUI te ontwikkelen.

Applicatie

Voor de applicatie wordt de functionaliteit, de roadmap en de technology besproken en geëvalueerd.

Applicatie functionaliteit

De applicatie app.py is gebruikt om de content in de index.html te publiceren. De content is verkregen door informatie via de SQLAlchemy interface van Flask uit de postgressql database (ca_db) te halen. De pseudocode van app.py is:

- Definieer control models voor de control tables en evidence tables.
- Definieer de functies voor het ophalen van de success en failed control.
- Retourneer de render_template index.html.

Applicatie roadmap

Dit is een zeer rudimentaire implementatie van de CA-tool en toch toont deze aan dat er snel een dashboard verkregen kan worden. Om te spreken over de CA-tool die snel is in te zetten zijn er een aantal zaken die toegevoegd moeten worden. Dit zijn de volgende aspecten:

Rules engine

De evaluatie van een threshold en een evidence waarde moet worden uitgebreid tot een expressie in de control waarde niet alleen de threshold wordt gedefinieerd maar ook een expressie als “<=”, “==”, “>=” moet kunnen worden opgegeven. De rules engine moet dan de expressies en threshold gebruiken om de evidence waarde te evalueren. De expressies zullen rijk genoeg moeten zijn om tal van controls te kunnen definiëren. Dit betekent dat er veel operatoren mogelijk moeten zijn en dat er ook meer waarden tegelijkertijd in één rule gedefinieerd kunnen worden. Nieuwe rules moeten op een eenvoudige wijze via een REST API of GUI op te voeren zijn in de app.py.

Evidence

De evidence is nu niet geformatteerd en bestaat alleen uit een threshold. Voor de evaluatie van complexe rules moet een evidence structure gedefinieerd worden, bijvoorbeeld in de vorm van een JSON format. De waarden uit het JSON format moeten overeenkomen met de informatiebehoefte uit de rule. Per rule kan een unieke JSON structuur worden bedacht.

Patterns

Een rule is gebaseerd op een managed object en een risico. Om het aantal rules beperkt te houden is het nodig om patterns te definiëren die van toepassing zijn op meer managed objects voor eenzelfde risico. Een voorbeeld is een rule voor zowel SQL Server als Postgresql managed objecten. Ook kan één rule worden gebruikt voor meerdere risico’s.

Applicatie Technology

Flask geeft niet alleen een web interface maar biedt ook een Object Relation Model (ORM) genaamd SQLAlchemy. Deze interface legt de relatie tussen een tabel in ProgreSQL en een class in Python. Hiertoe moet een ControlModel worden aangemaakt voor elke tabel, alhoewel SQLAlchemy ook de optie biedt om dit automatisch te doen met de module ‘automap_base’. De db.session.query functie staat het toe om een lijst te vullen met de inhoud van de query, waarin overigens ook joins worden

toegestaan. Zoals bekend is Python een krachtige een eenvoudige programmeertaal en dat blijkt ook te gelden voor de programmering van de CA-tool code.

Database

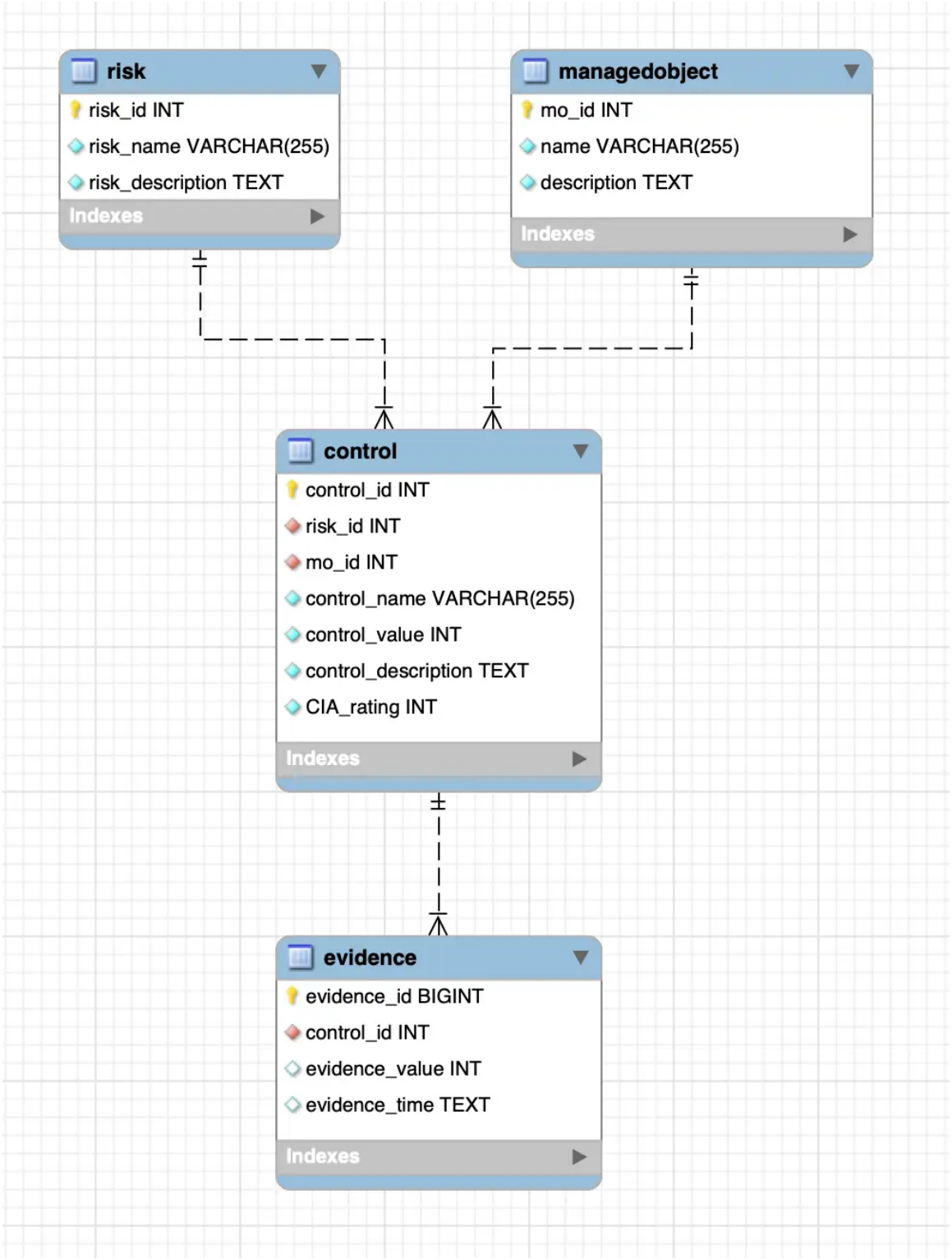
Voor de database wordt de functionaliteit, de roadmap en de technology besproken en geëvalueerd.

Database functionaliteit

De database is beperkt gehouden qua functionaliteit. Er zijn twee python scripts geschreven te weten:

- het script CA_schema.py met de creatie van de tabellen alsmede de gerelateerde testcases
- het script CA_populatie.py die de basisvulling geeft voor het datamodel zoals de risico's, de managed objects en de controls.

De vulling van de evidence is tot stand gekomen door de aanroep van een REST API naar Azure.



Figuur 2. CA-tool Entity Relationship Diagram.

Database roadmap

De database moet voorzien worden van standaard migration scripts om het beheer van het datamodel mogelijk te maken. Er is expliciet gekozen om geen business rules in het datamodel te definiëren, omwille van de migreerbaarheidseis voor de database tool. De referentiële integriteit en de indexen moeten echter nog wel aangemaakt worden.

Database technology
PostgreSQL voldoet vooralsnog uitstekend aan de eisen en wensen van de CA-tool. Voor enterprise organisaties is het denkbeeldig dat er gekozen moet worden voor een clustering technology met uitwijk mogelijkheden. Het Entity Relation Diagram is gecreëerd in MySQL workbench. Deze tool is echter alleen met een commerciële ODBC adaptor koppelbaar met postgreSQL. Het bleek echter heel eenvoudig om de CA-tool te migreren naar MySQL. Alleen de connection is verschillend geprogrammeerd in Python.

Bevindingen en conclusie Continuous Auditing Implemented

Belangrijkste lessons learned

- Het is best lastig om de juiste REST API te vinden voor een control. Zowel Azure als AWS bieden deze mogelijkheid wel maar je hebt specifieke kennis nodig van deze omgevingen om je weg snel te vinden.
- De bouw van een eigen tool kan redelijk snel met open source tools, maar er dient goed gelet te worden op de compatibiliteit van de systeem software, zeker in geval de hardware gedateerd is.
- De CA-tool vereist meer functionaliteit dan een REST API collector. Zo is een rules engine die evidence kan vergelijken met een norm en must. Tevens dienen de basis objecten flexibel gedefinieerd te kunnen worden te weten de managed objecten, risico's, controls, rules, en evidence collectie. Wellicht is hiervoor ook een JSON interface te bouwen die middels een Rest API is aan te roepen.
- Een connectie naar een (S)-CMDB tool is ook een must om de compleetheid te borgen van de managed objects.
- Om de hoeveelheid controls in te perken moet er ook een conventie worden gekozen om hergebruik mogelijk te maken (control patterns).

Een goede oplossing zou een community zijn die een open source CA-tool op de markt zet.

De Continuous Auditing (CA) Guild is empowered door Jan-Willem Hordijk, CTO van Plint AB, een Zweedse localisatie-organisatie. Deze publicatie is mogelijk gemaakt door Bart de Best (www.dbmetrics.nl), Dennis Boersen (Argis IT Consultants), Freek de Cloet (smartdocuments), Jan-Willem Hordijk (Plint AB), Louis van Hemmen (www.bitall.nl) Niels Talens – www.nielstalens.nl en Willem Kok (Argis IT Consultants).

Discussieer mee op LinkedIn.