THOMAS MORE

Uitbreiding van het External Attack Surface Management voor Baloise

Realisatiedocument

Tuur Hulselmans
Student Bachelor in de Elektronica-ICT – Cloud & Cyber Security

2024 - 2025

Inhoudsopgave

1. Inleiding

Publieke webapplicaties vormen een aantrekkelijk doelwit voor cyberaanvallen. Omdat deze applicaties rechtstreeks toegankelijk zijn via het internet, lopen organisaties voortdurend risico op misbruik van kwetsbaarheden, zoals verouderde software, onveilige configuraties of slechte invoervalidatie. Om deze risico's te beperken, wordt steeds vaker gebruikgemaakt van technologieën zoals External Attack Surface Management (EASM), die gericht zijn op het in kaart brengen, monitoren en beveiligen van de externe digitale voetafdruk van een organisatie.

In het kader van deze stage werd onderzocht hoe een geautomatiseerd, open-source EASM-systeem kan bijdragen aan snellere en betrouwbaardere kwetsbaarheidsdetectie binnen Baloise. Het project focust op het combineren van meerdere kwetsbaarheidsscanners en monitoringtools om zo de zichtbaarheid van externe assets te vergroten en het detectieproces te automatiseren.

De uitvoering van het project verloopt in drie opeenvolgende fasen. In de eerste fase worden de bestaande publieke assets van Baloise geïnventariseerd en gemonitord, met bijzondere aandacht voor vergeten of foutief geregistreerde domeinen. In de tweede fase worden de gedetecteerde assets gescand met behulp van open-source scanners, om zo kwetsbaarheden te detecteren en te analyseren. De derde fase, die optioneel is en enkel wordt uitgevoerd indien de tijd het toelaat, voorziet in de verwerking van de scanresultaten in een centraal dashboard. Deze visualisatie laag wordt beschouwd als een "nice to have", die toelaat om de resultaten op een overzichtelijke en visuele manier op te volgen.

Het uiteindelijke doel is om binnen een periode van dertien weken een robuust en herhaalbaar EASM-systeem op te leveren dat de verschillende processen integreert, automatiseert en zo veel mogelijk aansluit bij de bestaande infrastructuur van Baloise. Dit document beschrijft de gehanteerde werkwijze, de gemaakte keuzes, de realisatie en de concrete resultaten van dit project.

2. Plan van aanpak

Dit hoofdstuk schetst de context, doelstellingen en aanpak van de stageopdracht binnen Baloise. De stage maakt deel uit van een breder initiatief om de beveiliging van publieke webapplicaties te versterken door middel van geautomatiseerde kwetsbaarheidsdetectie. Daarbij wordt niet alleen gefocust op het identificeren van bestaande risico's, maar ook op het opzetten van een schaalbaar monitoring- en detectiesysteem. Verder worden de betrokken stakeholders, de probleemstelling, en de afbakening van het project toegelicht.

2.1. Organisatiecontext: Baloise Belgium

Baloise Belgium nv is een toonaangevende verzekeringsmaatschappij die deel uitmaakt van de Zwitserse Baloise Group, een duurzaam en innovatieve speler in de Europese verzekeringsmarkt. De Belgische tak is ontstaan uit de fusie van drie gerenommeerde verzekeraars: Mercator, Avéro en Nateus, en combineert zo jarenlange ervaring met een sterke lokale verankering.

De Baloise Group is actief in 4 Europese landen: Zwitserland, België, Duitsland en Luxemburg. Vanuit dez landen werkt Baloise aan het opbouwen van duurzame samenwerking en aan de toekomst door het aanbieden van passende verzekeringsoplossingen. In totaal stelt de Baloise Group ongeveer 8.900 medewerkers tewerk, verspreid over vier landen.

2.2. Probleemstelling

Publieke webapplicaties vormen een aantrekkelijk doelwit voor aanvallers, omdat ze direct toegankelijk zijn via het internet. Automatische programma's scannen dagelijks deze applicaties op bekende kwetsbaarheden, zoals verouderde software, misconfiguraties of onveilige invoervalidatie. Een potentiële aanval kan leiden tot ernstige gevolgen zoals diefstal van data, ongeautoriseerde toegang tot interne systemen of het misbruiken van de applicatie als springplank voor bredere aanvallen.

Binnen Baloise worden momenteel al vulnerability scans uitgevoerd met een bestaande scanner. Toch is er behoefte aan meer variëteit in de gebruikte scanners. Elke scanner heeft zijn eigen sterktes en zwaktes, en kan unieke kwetsbaarheden opsporen. Door meerdere scanners te combineren, ontstaat een completer en betrouwbaarder beveiligingsbeeld. Daarnaast kunnen de resultaten van verschillende tools met elkaar worden vergeleken, wat de kwaliteit van de beveiliging ten goede komt.

2.3. Doelstellingen van het project

Zoals eerder aangehaald, wordt er binnen de huidige omgeving al gebruikgemaakt van een enkele vulnerability scanner. Daarnaast wordt er samengewerkt met een derde partij die aanbevelingen geeft over de huidige infrastructuur. Er is echter nood aan een systeem dat verschillende scanners integreert en centraal beheert. Dit project richt zich op het ontwikkelen en implementeren van zo'n geïntegreerde oplossing.

Een vorige stagiair heeft reeds een inventarisatie gemaakt van de verschillende webapplicaties, waaronder de bijbehorende domeinnamen. Op basis hiervan wordt nu gezocht naar geschikte vulnerability scanners die de applicaties periodiek kunnen scannen en hierover kunnen rapporteren. Daarnaast is het belangrijk om de evolutie van kwetsbaarheden in de tijd in kaart te brengen, omdat dit waardevolle inzichten oplevert over hoe lang een kwetsbaarheid heeft bestaan voordat ze werd opgelost.

De implementatie moet volledig geautomatiseerd verlopen, zodat installatie en integratie zo efficiënt mogelijk kunnen worden uitgevoerd. Indien de planning dit toelaat, zal er aanvullend een koppeling worden voorzien met het bestaande vulnerability dashboard, zodat de resultaten centraal beschikbaar worden en eenvoudig kunnen worden geraadpleegd.

2.4. Aanpak en methodologie

Dit project wordt uitgevoerd binnen een periode van 13 weken en volgt een Agile werkwijze, ondersteund door Azure DevOps. De verschillende onderdelen van het project worden opgesplitst in sprints, met regelmatige sprint planning en backlogbeheer om de voortgang te bewaken en flexibel in te spelen op nieuwe inzichten of vereisten. Binnen het securityteam van Baloise vindt tweewekelijks een teamvergadering plaats. Deze overlegmomenten worden als referentiepunt genomen voor de sprintcycli, zodat de projectactiviteiten afgestemd blijven op de interne planning en tijdig besproken kunnen worden met relevante stakeholders.

In de eerste week wordt een onderzoek uitgevoerd naar mogelijke oplossingen, inclusief het identificeren van relevante risico's. Daarna start de implementatie van de monitoringfunctionaliteit, gevolgd door het opzetten en configureren van de vulnerability scanners. In de laatste fase wordt het volledige proces geautomatiseerd en, indien de planning het toelaat, geïntegreerd met het bestaande vulnerability dashboard. Hierdoor kunnen de scanresultaten centraal beschikbaar worden gesteld en op een gestructureerde manier worden geanalyseerd.

2.5. Verwachte resultaat

Het beoogde eindresultaat van dit project is een geautomatiseerd External Attack Surface Management systeem voor het monitoren en beveiligen van publieke webapplicaties van Baloise. In een eerste fase worden bestaande applicaties geïnventariseerd. Daarna wordt een passende monitoringoplossing opgezet, inclusief de integratie en configuratie van een geschikte vulnerability scanner.

Het volledige proces wordt geautomatiseerd via een CI/CD-pipeline, zodat scans periodiek en zonder handmatige interventie kunnen plaatsvinden. De oplossing wordt grondig gedocumenteerd en geoptimaliseerd met het oog op herbruikbaarheid en toekomstig onderhoud.

Indien de projectplanning dit toelaat, wordt tevens een koppeling met het bestaande vulnerability dashboard gerealiseerd. Activiteiten zoals manuele red teaming of het vervangen van bestaande securityprocessen vallen buiten de scope van dit project.

3. Analyse

In dit hoofdstuk wordt toegelicht welke technologieën, tools en methodes zijn onderzocht en geselecteerd voor de ontwikkeling van een geautomatiseerd External Attack Surface Management systeem binnen Baloise. De keuzes zijn gebaseerd op technische geschiktheid, integratiemogelijkheden met bestaande infrastructuur en hun vermogen om te voldoen aan de functionele eisen van het project.

Elk onderdeel van de oplossing, van infrastructuurautomatisatie tot kwetsbaarhedenscanning en data-visualisatie, werd geëvalueerd op basis van detectiediepte, automatisering, false positives, ondersteuning en snelheid. Hierbij werd gestreefd naar een optimale balans tussen de verschillende criteria zodat de belangrijkste criteria meer invloed hebben. In de volgende paragrafen worden de gemaakte keuzes toegelicht, onderbouwd met objectieve argumentatie en waar relevant vergeleken met alternatieve oplossingen.

3.1. Selectiecriteria en evaluatiemethodiek

Om tot een doordachte keuze van tools en technologieën te komen, werd in dit project gewerkt met een gestructureerde evaluatiemethodiek. De geselecteerde oplossingen moesten niet alleen technisch geschikt zijn, maar ook aansluiten bij de infrastructuur en werkmethodes van Baloise. Daarom werden alle potentiële tools beoordeeld op basis van vooraf bepaalde selectiecriteria, waaronder automatiseringsgraad, schaalbaarheid, gebruiksvriendelijkheid, ondersteuning voor CI/CD-integratie en ondersteuning.
Aan de hand van deze criteria kon elke oplossing objectief geëvalueerd worden op haar geschiktheid binnen de context van Baloise, wat heeft geleid tot een onderbouwde en transparante keuze.

Voor het afwegen van alternatieven werd gebruikgemaakt van de Weighted Ranking Methode (WRM). Hierbij kregen de belangrijkste criteria een gewogen score toegekend, zodat de meest relevante eigenschappen zwaarder meewegen in de uiteindelijke beslissing. Deze aanpak maakte het mogelijk om objectief verschillende infrastructuur- en scanningoplossingen met elkaar te vergelijken en om op transparante wijze te motiveren waarom gekozen werd voor onder andere Terraform, Azure en specifieke vulnerability scanners zoals Nikto, Nuclei en OWASP ZAP.

In de volgende secties wordt per technologie of tool de gemaakte keuze verder toegelicht en onderbouwd, telkens aan de hand van de hierboven vermelde evaluatiefactoren.

3.2. Infrastructure-as-Code en deploymentstrategie

Terraform is een tool ontwikkeld door HashiCorp die het mogelijk maakt om infrastructuur op een declaratieve manier te definiëren en beheren. Binnen dit project wordt Terraform gebruikt om de volledige omgeving automatisch uit te rollen, waaronder:

- De rekenkracht voor de scanners
- Logische applicaties die de triggers aansturen op basis van tijdsintervallen
- De opslag- en netwerkconfiguraties die vereist zijn voor de scanners

Een belangrijk aspect van de gekozen oplossing is dat Terraform niet handmatig hoeft te worden uitgevoerd. Het opzetten van infrastructuur gebeurt volledig geautomatiseerd via een Azure DevOps pipeline. Bij het uitvoeren van een commit of een wijziging in de infrastructuurdefinities wordt de pipeline automatisch getriggerd. Deze pipeline voert de standaard Terraform-stappen uit (init, plan en apply), waardoor elke wijziging in de infrastructuur direct en gecontroleerd wordt uitgerold naar de Azure-omgeving.

Deze CI/CD-aanpak garandeert consistentie, herhaalbaarheid en vermindert de kans op menselijke fouten. Bovendien kunnen meerdere teamleden samenwerken aan de infrastructuur, met volledige versiecontrole via Git.

3.3. Cloudplatform: Microsoft Azure

Voor de uitvoering van dit project is gekozen om gebruik te maken van Microsoft Azure als cloudplatform. Azure vormt de basisinfrastructuur waarop alle componenten van het EASM-systeem draaien. De keuze voor Azure is geen toeval: Baloise werkt nauw samen met Microsoft en maakt al breed gebruik van het Azure-ecosysteem.

3.3.1. Redenen voor de keuze van Azure

Het Azure-platform werd gekozen op basis van meerdere overwegingen die zowel technische als organisatorische voordelen bieden. Een van de belangrijkste redenen is de naadloze integratie met Azure DevOps. Doordat beide diensten onderdeel zijn van hetzelfde ecosysteem, is het mogelijk om de volledige CI/CD-pipeline van code tot infrastructuur uitrol centraal te beheren. Binnen dit project wordt Terraform bijvoorbeeld automatisch getriggerd vanuit een Azure DevOps pipeline. Zodra een wijziging wordt doorgevoerd in de infrastructuurcode, wordt deze pipeline gestart en worden de benodigde resources aangemaakt. Dit zorgt voor een efficiënte, geautoriseerde en volledig geautomatiseerde implementatie.

Een ander groot voordeel van Azure is de mogelijkheid om zogenaamde sandbox-omgevingen op te zetten door Baloise. Deze tijdelijke omgevingen worden gebruikt voor testdoeleinden, zodat nieuwe configuraties en kwetsbaarheidsscanners grondig geëvalueerd kunnen worden zonder dat dit invloed heeft op de productieomgeving. Zeker in de context van cybersecurityonderzoek, waarbij veel geëxperimenteerd wordt met tools en instellingen, bieden deze sandboxes een veilige en flexibele uitkomst.

3.3.2. Vergelijking van compute-opties

Voor de uitvoering van open-source kwetsbaarheidsscanners binnen dit project werden verschillende compute-opties binnen Azure geëvalueerd: Azure Virtual Machines (VMs), Azure Kubernetes Service (AKS) en Azure Container Instances (ACI).

Virtual Machines bieden volledige controle over het besturingssysteem en configuratie, maar brengen aanzienlijke beheerlast met zich mee. Ze zijn minder geschikt voor kortdurende, event-gedreven taken zoals het ad-hoc uitvoeren van scans. Bovendien is de provisioning trager en minder eenvoudig te automatiseren (Virtual machines in Azure, 2024).

Azure Kubernetes Service is krachtig en schaalbaar, maar introduceert extra complexiteit. De setup en het beheer van een Kubernetes-cluster vergen diepgaande kennis en zijn niet proportioneel aan de vereisten van dit project, waarin taken meestal kort en niet parallel verlopen (What is Azure Kubernetes Service (AKS)?, 2024).

Azure Container Instances daarentegen bieden een serverless, lichtgewicht oplossing waarmee containers on-demand en zonder onderliggende infrastructuurbeheertaken kunnen worden uitgerold. Dit maakt ACI uitermate geschikt voor taken zoals scheduled scanning, waarbij snelheid, eenvoud en tijdelijke resourcegebruik centraal staan. Op basis van deze evaluatie werd ACI gekozen als compute-oplossing, vanwege de minimale beheerlast, snelle provisioning en naadloze integratie met Logic Apps en CI/CD-pipelines (What is Azure Container Instances?, 2024).

3.4. Monitoring van publieke applicaties

Naast het uitvoeren van kwetsbaarheidsscans is ook continue monitoring van publieke webapplicaties van belang. Monitoring biedt inzicht in de beschikbaarheid en responstijd van applicaties en helpt om potentiële problemen zoals downtime, connectiviteitsfouten of aanvallen snel te detecteren. Deze passieve vorm van bewaking vormt een waardevolle aanvulling op actieve scans en verhoogt de algehele betrouwbaarheid van het External Attack Surface Management-systeem.

Voor dit project is gekozen voor Uptime Kuma, een open-source monitoringtool die ondersteuning biedt voor HTTP(S), TCP, DNS en ICMP-controles. Uptime Kuma onderscheidt zich door zijn gebruiksvriendelijke webinterface, realtime meldingsmogelijkheden (zoals via e-mail of webhooks), en eenvoudige containergebaseerde installatie. De tool maakt het mogelijk om publieke endpoints voortdurend te controleren en automatisch een melding te sturen wanneer een dienst niet meer bereikbaar is. Dit biedt operationeel voordeel in situaties waarin snelheid van reactie belangrijk is.

Er werden nog andere monitoringtools bekeken, zoals commerciële alternatieven StatusGator of Pulsetic. In een vergelijkende studie van Max Shash kan men zien dat deze commerciële tools al snel honderden euro's per jaar kosten voor vergelijkbare functionaliteit. Als dan gekeken wordt naar ander open source tools zoals Upptime of Monitoror, biedt Uptime Kuma meer functionaliteiten en betere integratiemogelijkheden doormiddel van een API (Shash, 2024).

De toepassing werd opgezet als een container binnen de Azure-omgeving en geconfigureerd om de belangrijkste webapplicaties van Baloise periodiek te controleren. Alle meldingen worden gelogd en zijn zichtbaar in een overzichtelijk dashboard. Dankzij deze integratie wordt de beschikbaarheid van publieke applicaties continu bewaakt, wat het reactievermogen bij incidenten vergroot en bijdraagt aan het algemene beveiligingsbeeld.

3.5. Scanners voor kwetsbaarheidsdetectie

Er bestaan verschillende kwetsbaarheidsscanners, zowel commerciële als ook open-source varianten. Commerciële tools bieden vaak professionele ondersteuning en uitgebreide functionaliteit, maar hier zit natuurlijk een stevig prijskaartje aan vast. Open source scanners daarentegen zijn vrij beschikbaar en profiteren doorgaans van actieve community-ondersteuning.

Omdat Baloise al gebruik maakt van een commerciële scanner namelijk Nessus waren ze opzoek naar een open source scanner. Dit geeft ook meer input van de community en heeft soms een groter bereik voor het aanleveren van detectie regels voor nieuwe kwetsbaarheden.

3.6. Analyse van de scanners

Om een gegronde keuze te maken tussen verschillende kwetsbaarheidsscanners werd gebruikgemaakt van de Weighted Ranking Methode (WRM). Deze methode kent aan elke tool scores toe op basis van vooraf vastgelegde selectiecriteria, gewogen naar hun relevantie voor het project. De volgende tools werden geëvalueerd: Nikto, OWASP ZAP, Nuclei, Nessus en OpenVAS.

3.6.1. Definities en gewichtsbepaling van criteria

De gewichten van de selectiecriteria zijn vastgesteld in overleg met het CISO-team van Baloise en zijn gebaseerd op de belangrijkste vereisten die voortkomen uit de specifieke noden van de organisatie. Tijdens een gezamenlijke evaluatie werden de verschillende criteria besproken en gevalideerd. Mijn rol bestond uit het aanreiken van technische input en praktische inzichten met betrekking tot de toepasbaarheid van de tools, waarna het team tot een definitieve weging is gekomen.

De hoogste prioriteit werd gegeven aan detectiediepte en automatisering. Voor Baloise is het cruciaal om een zo volledig mogelijk beeld te krijgen van potentiële kwetsbaarheden, verspreid over verschillende technologieën en platformen. Tegelijk is het belangrijk dat deze detectieprocessen zoveel mogelijk geautomatiseerd verlopen en eenvoudig kunnen worden geïntegreerd in de bestaande infrastructuur.

Ondersteuning werd eveneens als een kritieke factor aangemerkt. Bij technische problemen of het uitblijven van updates is het noodzakelijk dat er snel gereageerd kan worden, hetzij via een actieve community, hetzij via commerciële ondersteuning. Bovendien moet de scanner in staat zijn om snel nieuwe kwetsbaarheden op te nemen en actueel te blijven in een snel veranderend dreigingslandschap.

Gebruiksvriendelijkheid en snelheid kregen een lager gewicht. Binnen Baloise zijn voldoende technische profielen aanwezig om met complexere tools te werken indien die betere resultaten opleveren. Bovendien worden de scans doorgaans op de achtergrond uitgevoerd, waardoor de tijdsduur van een scan minder impact heeft op de dagelijkse werking.

False positives daarentegen vormen wel een belangrijke aandachtspunt. Een te hoge foutmarge zou leiden tot overmatige meldingen en nodeloze opvolging, wat de efficiëntie van het securityteam onder druk zet. Accuratesse van de resultaten is daarom essentieel binnen het gekozen toollandschap.

Criteria	Definitie
Detectiediepte (30%)	Mate waarin de scanner verschillende kwetsbaarheidstypen detecteert
Automatisering (25%)	Mogelijkheid tot integratie in CI/CD-pipelines en automatische scans
False Positives (20%)	Nauwkeurigheid van resultaten, minder valse alarmen
Ondersteuning (15%)	Mate van ondersteuning door community of commercieel
Snelheid (5%)	Tijdsduur tot voltooiing van een gemiddelde scan
Gebruiksvriendelijkheid (5%)	Gebruiksgemak en toegankelijkheid van de tool

Tabel 1: Criteria met definitie

3.6.2. Vergelijkingstabel

De onderstaande tabel geeft een overzicht van de scores die aan elke kwetsbaarheidsscanner zijn toegekend op basis van vooraf bepaalde selectiecriteria. De scores werden toegekend op basis van documentatieonderzoek, community-ervaringen en praktijkervaringen binnen de cybersecuritysector. Elk criterium werd gewogen volgens de relatieve belangrijkheid voor het project, zoals eerder toegelicht. De totaalscores geven een indicatie van de geschiktheid van elke tool binnen de context van de noden van Baloise.

Criteria	Weight	Nikto	Owasp ZAP	Nuclei	Nessus	openvas
Detectiediepte	30%	3	4	4	4	3
Automatisering	25%	4	4	3	2	4
False Positives	20%	3	4	4	3	3
Ondersteuning	15%	3	5	5	4	2
Snelheid	5%	3	4	5	4	3
Gebruiksvriendelijkheid	5%	3	3	5	4	2
Totaal	100%	3,25	4,1	4	3,3	3,05

Tabel 2: Weighted Ranking Methode *(WRM)*

Op basis van de uitgevoerde vergelijkende analyse en de Weighted Ranking Methode werd besloten om verder te werken met Nikto, OWASP ZAP en Nuclei. Elk van deze scanners heeft specifieke sterktes en zwaktes, maar juist door ze te combineren ontstaat een aanvullend geheel dat de kwetsbaarheidsdetectie aanzienlijk versterkt. Nuclei biedt hoge snelheid en uitgebreide automatiseringsmogelijkheden, terwijl OWASP ZAP bijzonder sterk is in interactieve tests en community-ondersteuning. Nikto vult dit aan met snelle detectie van configuratiefouten en eenvoudige inzetbaarheid. Door deze tools parallel in te zetten ontstaat een breder en betrouwbaarder beeld van potentiële kwetsbaarheden, wat aansluit bij de doelstellingen van dit project en de behoeften van Baloise.

3.6.3. Onderbouwing van de scores

Nuclei behaalde hoge scores op meerdere vlakken van de WRM. De tool heeft een groot detectie bereik doormiddel van het makkelijk toevoegen van nieuwe kwetsbaarheden. Doormiddel van YAML bestanden kunnen nieuwe regels gemaakt worden op nieuwe kwetsbaarheden (Introduction to Nuclei Templates, 2025). Door een vergelijkende studie van “Pentest.com” ziet men dat de “false positive rate” laag is in vergelijking met andere scanners, dus er zijn weinig foute analyses dat gedaan worden (Network vulnerability scanners benchmark 2024, 2024). Daarnaast is er een grote community die extra support biedt en deze tool kan verschillende templates tegelijk runnen op een target domein.

OWASP ZAP is vooral sterk in ondersteuning en detectiediepte. De scanner behaalde een hoge score voor automatisering dankzij het bestaande Automation Framework en integratieopties voor CI/CD-pijplijnen (Automation Guide, 2024). Detectiediepte werd met 4/5 beoordeeld, omdat ZAP zich vooral richt op bekende webapplicatiefouten (bijvoorbeeld OWASP Top 10), maar minder breed inzetbaar is voor andere kwetsbaarheidstypen (Alazmi & De Leon, 2022).

Nikto richt zich voornamelijk op het opsporen van webserverconfiguratiefouten. De tool behaalde een score van 3/5 op detectiediepte binnen zijn specifieke domein, maar is minder geschikt voor diepgaande applicatieanalyse (Sullo & Lodge, 2024). Automatisering is mogelijk via eenvoudige CLI-commando's, wat resulteert in een score van 4/5.
Snelheid is matig, omdat Nikto lineair test en relatief traag werkt bij grote sets van servers. False positives komen regelmatig voor door de brede zoekstrategie (Nikto: An Overview, 2023). Ondersteuning is redelijk door communitybijdragen, maar beperkte updates. Gebruiksgemak is goed dankzij eenvoudige bediening en goede documentatie.

Nessus is de scanner die momenteel al actief wordt ingezet binnen Baloise. De tool wordt commercieel aangeboden door Tenable en staat bekend om zijn brede detectiemogelijkheden op zowel netwerk- als applicatieniveau, wat een hoge detectiediepte rechtvaardigt (Network vulnerability scanners benchmark 2024, 2024). De ondersteuning wordt beoordeeld met 3/5: professionele support is beschikbaar, maar gekoppeld aan de betaalde versie en de communityondersteuning is beperkter dan bij open-source alternatieven. Op automatisering scoort Nessus laag, omdat integratie in CI/CD-pijplijnen vaak afhankelijk is van commerciële plugins of aanvullende licenties (Automation of Nessus Scan via API, 2022). De snelheid van de scans is degelijk, zeker bij geplande, periodieke scans, en het aantal false positives blijft relatief beperkt dankzij de volwassen scanengine (Awati, 2025). Gebruiksvriendelijkheid wordt gewaardeerd op 4/5, dankzij een intuïtieve interface en uitgebreide documentatie, wat de tool toegankelijk maakt voor een breed gebruikerspubliek.

OpenVAS is een open-source alternatief dat zich richt op netwerk- en applicatiescans. Detectiediepte is matig vanwege beperkte webapplicatiefocus (Network vulnerability scanners benchmark 2024, 2024). Automatisering wordt goed ondersteund door API-integraties.
Snelheid en false positives worden als gemiddeld beoordeeld. Ondersteuning is zwakker vanwege tragere community-updates. Gebruiksgemak wordt beoordeeld als redelijk omdat configuratie soms omslachtig is en de documentatie niet uitgebreid is (Greenbone Community Edition – Documentation, 2024).

3.7. Beperkingen en randvoorwaarden

Het automatisch uitvoeren van kwetsbaarheidsscans brengt enkele risico's met zich mee. Zo kan een onzorgvuldig ingestelde scan leiden tot een overbelasting van de webserver door een te hoge hoeveelheid verzoeken, het onbedoeld crawlen van volledige directorystructuren, of zelfs het verstoren van normale werking van de applicatie. Ook bestaat het risico op een overvloed aan meldingen, waaronder false positives, die de efficiëntie van het opvolgproces kunnen ondermijnen. Om deze risico's te beperken, is het essentieel om scans zorgvuldig te plannen, configuraties grondig af te stemmen op de doelomgeving, en alle testactiviteiten goed te documenteren. Bovendien moeten kwetsbaarheidsscanners met de nodige voorzichtigheid worden ingezet op productieomgevingen, en bij voorkeur eerst getest worden in sandbox-omgevingen.

3.8. Kostenanalyse

Voor het uitvoeren van het External Attack Surface Management-systeem werd gebruikgemaakt van diverse Azure-diensten, waaronder Azure Container Instances, Logic Apps, Azure Files en de Azure Container Registry. Om het financiële plaatje van deze infrastructuur in kaart te brengen, werd een schatting opgesteld met behulp van de Microsoft Azure Pricing Calculator. De berekening is gebaseerd op het verwachte gebruikspatroon over een periode van één maand en betreft uitsluitend de resources die door het project automatisch worden uitgerold via Terraform.

De totale geschatte maandelijkse kost bedraagt €176,33. Deze kost omvat onder meer:

Azure Container Instances (€17,74 + €33,81):
Deze worden gebruikt voor het tijdelijk uitvoeren van kwetsbaarheidsscanners zoals Nuclei, Nikto en OWASP ZAP. De kosten zijn gebaseerd op het rekenverbruik en geheugengebruik van meerdere containers in de regio West-Europa. Daarnaast zal er ook een container de Uptime Kuma monitoring systeem hosten.

Azure Files (€1,61 + €67,51):
Voor de opslag van configuratiebestanden, scanresultaten en tijdelijke logs. Er wordt gebruikgemaakt van zowel provisioned als pay-as-you-go opslagmodellen, met transaction optimized access voor performantieverbetering. Voor de persistente opslag van de Uptime Kuma is er ook een

Logic Apps (€0,01):
Deze dienen als triggermechanisme om scans op geplande tijdstippen uit te voeren. Door het lage volume aan acties en de beperkte dataverwerking blijven deze kosten verwaarloosbaar.

Azure Container Registry (€4,39):
Voor het hosten en beheren van containerimages die de scanners bevatten. Er wordt gebruikgemaakt van de Basic Tier met een beperkte hoeveelheid opslag en datatransfer.

De belangrijkste kostenpost binnen deze raming is de opslag via Azure Files met transaction optimized access (€67,51), gevolgd door de uitvoering van scans via Azure Container Instances. Logic Apps en Container Registry hebben een zeer beperkte financiële impact. Er werd in deze raming geen kost voorzien voor ondersteuning of licenties, aangezien enkel gebruik wordt gemaakt van gratis en open-source tools.

Het opgestelde budget is voorlopig gericht op een test- en ontwikkelomgeving. Indien het systeem later op grotere schaal in productie zou worden genomen, kan dit kostenplaatje variëren naargelang het scanvolume, de retentie van bestanden en het aantal simultane containers. Desondanks toont deze raming aan dat het gebruik van serverless en schaalbare componenten een kostenefficiënte oplossing vormt voor externe kwetsbaarheidsmonitoring.

Microsoft Azure Estimate				
Your Estimate				
Service category	**Service type**	**Region**	**Description**	**Estimated monthly cost**
Compute	Azure Container Instances	West Europe	3 Container group(s) x 446,400 Second(s), Linux OS, Pay as you go, 1.5 GB Memory, 1 vCPU(s)	€17,74
Compute	Azure Container Instances	West Europe	1 Container group(s) x 2,678,400 Second(s), Linux OS, Pay as you go, 1 GB Memory, 1 vCPU(s)	€33,81
Internet of Things	Logic Apps	West US	Workloads: Consumption plan, 3 Action Executions per day x 31 days, 0 GB Data Retention, 3 Standard	€0,01
Containers	Azure Container Registry	West Europe	Basic Tier, 1 registry x 30 days, 0 GB Extra Storage, Container Build - 1 CPUs x 1 Seconds - Inter Region transfer type, 5 GB	€4,39
Storage	Azure Files	West Europe	HDD (Standard) Media tier, LRS Redundancy, Pay-as-you-go Billing model, Transaction Optimized	€1,61
Storage	Azure Files	West Europe	HDD (Standard) Media tier, LRS Redundancy, Pay-as-you-go Billing model, Transaction Optimized	€67,51
Support		**Support**		€0,00
		Licensing Program	**Microsoft Customer Agreement (MCA)**	
		Billing Account		
		Billing Profile		
		Total		€125,07

Figuur 1: (Pricing calculator, 2025)

4. Realisatie

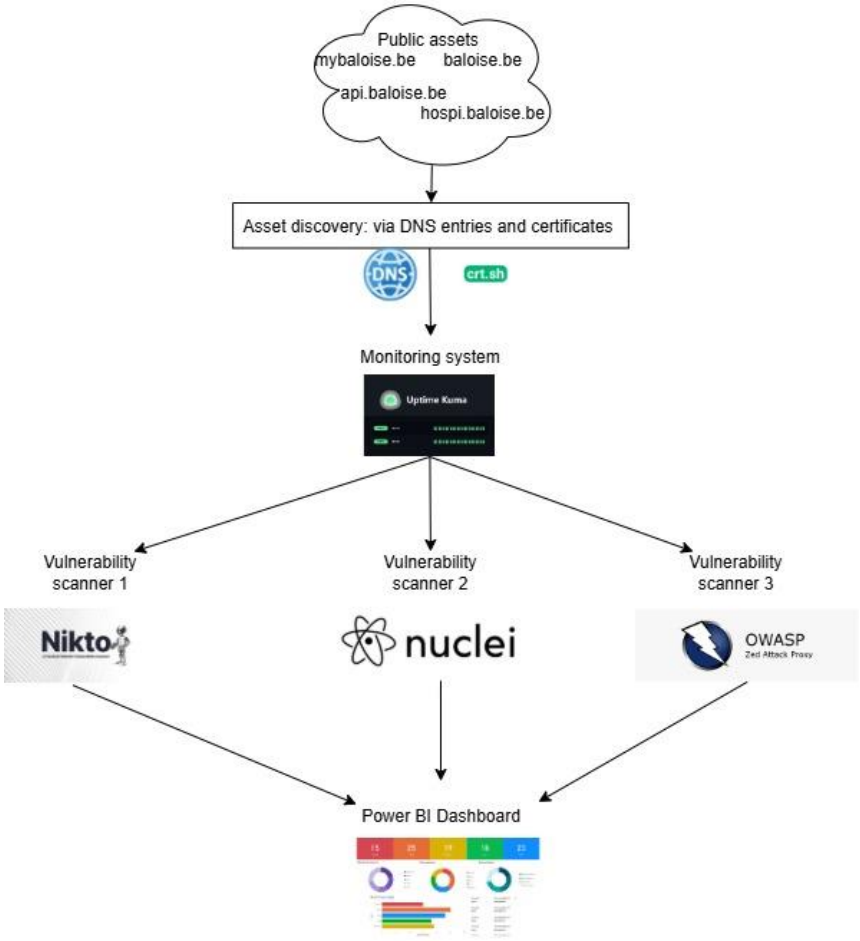
In dit hoofdstuk wordt de gerealiseerde oplossing voor het External Attack Surface Management-systeem in detail beschreven. Er wordt toegelicht welke componenten zijn opgebouwd, hoe deze onderling samenwerken, en op welke manier de gekozen tools zijn geïntegreerd in een geautomatiseerde workflow. Zowel de technische infrastructuur als de functionele werking van de verschillende onderdelen, zoals de scanners, monitoringtool en dashboardintegratie, komen aan bod. Het doel van dit hoofdstuk is om inzicht te geven in hoe de theoretische keuzes uit het vorige hoofdstuk zijn omgezet in een concrete, werkende oplossing binnen de Azure-omgeving.

4.1. Overzicht van de oplossing

De uiteindelijke oplossing bestaat uit een systeem dat publieke webapplicaties van Baloise permanent monitort en periodiek onderwerpt aan kwetsbaarheidsscans. De drie belangrijkste componenten zijn het monitoringsysteem (Uptime Kuma), een set van open source kwetsbaarheidsscanners (Nikto, Nuclei en OWASP ZAP) en een toekomstige visualisatielaag via Power BI.

In de bovenste laag worden verschillende publieke webapplicaties van Baloise, zoals mybaloise.be, baloise.be en api.baloise.be, continu opgevolgd via Uptime Kuma. Bij detectie van beschikbaarheidsproblemen of op vooraf ingestelde tijdstippen worden geautomatiseerde scans uitgevoerd. Elke scanner heeft een eigen focusgebied: Nikto controleert webserverconfiguraties, Nuclei voert snelle template-gebaseerde scans uit, en OWASP ZAP richt zich op diepgaande webapplicatieanalyse.

De resultaten van deze scanners worden centraal verzameld en in de toekomst verwerkt in een Power BI-dashboard, dat de gedetecteerde kwetsbaarheden visualiseert op basis van prioriteit, type en betrokken applicatie. Deze centrale rapportering zal Baloise in staat stellen om in één oogopslag inzicht te krijgen in de actuele status van hun externe aanvalsoppervlak.



Figuur 2: Overzicht EASM

4.2. Architectuur

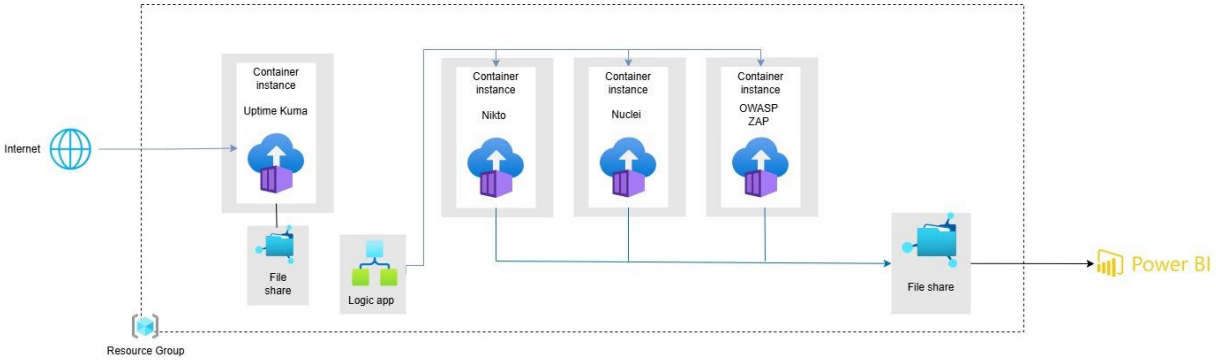
De technische architectuur van het totaal plaatje is opgebouwd rond een Azure en containergeoriënteerde infrastructuur. Figuur 3 toont een overzicht van de gebruikte componenten en hun onderlinge relaties.

Uptime Kuma draait in een aparte Azure Container Instance en monitort de publieke webapplicaties op beschikbaarheid. De data wordt opgeslagen in een Azure File Share, zodat de dat persistent is. Hierdoor kan het systeem worden herstart zonder informatie te verliezen.

Het uitvoeren van kwetsbaarheidsscans gebeurt eveneens in afzonderlijke container instances, elk verantwoordelijk voor een specifieke tool: Nikto, Nuclei en OWASP ZAP. Deze containers worden periodiek opgestart via een Azure Logic App, die fungeert als geautomatiseerde trigger op basis van vooraf ingestelde schema's. Iedere scancontainer voert zijn scan uit op de opgegeven targets en schrijft de resultaten weg naar een gedeelde Azure File Share.

Deze centrale opslag vormt de schakel naar de rapportageomgeving: de scanresultaten worden via een connectie ingelezen in een Power BI-dashboard, waar ze worden gevisualiseerd per kwetsbaarheidstype, ernst en betrokken applicatie. Hierdoor krijgt Baloise in realtime inzicht in de status van hun externe aanvalsoppervlak.

Alle componenten draaien binnen één resource group, wat het beheer vereenvoudigt en toelaat om de volledige omgeving automatisch te provisioneren via Terraform. Deze modulaire en schaalbare architectuur maakt het mogelijk om snel nieuwe scanners toe te voegen, monitoringcriteria aan te passen of uit te breiden naar extra applicaties.



Figuur 3: Technisch diagram

4.3. Opzet van de infrastructuur

Voor de inrichting van de infrastructuur werd gebruikgemaakt van Terraform, een open-source Infrastructure as Code-tool van HashiCorp. Hiermee wordt de volledige cloudomgeving declaratief gedefinieerd en automatisch uitgerold. Binnen dit project worden met Terraform onder andere de Azure Container Instances uitrol die de kwetsbaarheidsscanners uitvoeren, Azure Logic Apps voor het aansturen van geplande scans, en de nodige opslag- en netwerkcomponenten voor het beheren van resultaten en configuraties.

De aanmaak verloopt volledig geautomatiseerd via een CI/CD-pipeline in Azure DevOps. Bij elke wijziging in de infrastructuurcode (commit) wordt de pipeline automatisch getriggerd, die vervolgens de standaard Terraform-stappen (init, plan, apply) uitvoert. Hierdoor wordt elke infrastructuuraanpassing gecontroleerd, herhaalbaar en volledig traceerbaar uitgerold naar de Azure-omgeving.

Deze aanpak zorgt niet alleen voor consistentie en schaalbaarheid, maar vermindert ook het risico op menselijke fouten. Bovendien wordt samenwerking binnen het team vereenvoudigd door versiebeheer in Git en duidelijke change-tracking. Door infrastructuur als code te beheren, wordt het mogelijk om snel aanpassingen door te voeren en nieuwe omgevingen op te zetten op een uniforme manier.

4.4. Workflow van de scanners

De workflow van de scanners is geautomatiseerd en wordt periodiek uitgevoerd via Azure Logic Apps, die fungeren als triggers op basis van vooraf ingestelde tijdsintervallen. Zodra een trigger actief wordt, roept deze een vooraf gedefinieerde Azure Container Instance aan waarin de gewenste kwetsbaarheidsscanner, zoals Nikto, OWASP ZAP of Nuclei, wordt uitgevoerd.

Binnen de container voert de scanner de scan uit op de opgegeven domeinen. Elke scanner wordt afzonderlijk geconfigureerd op basis van het type kwetsbaarheden dat gedetecteerd moet worden. Na voltooiing van de scan worden de resultaten in een vooraf gedefinieerd formaat (bijv. CSV, TXT of XML) opgeslagen op een Azure File Share of doorgestuurd naar een centrale webapplicatie of dashboard.

Deze geautomatiseerde workflow zorgt ervoor dat scans onafhankelijk en herhaalbaar kunnen plaatsvinden, zonder menselijke tussenkomst. Door het gebruik van containers is elke uitvoering gescheiden van de vorige, wat de veiligheid en betrouwbaarheid verhoogt. Bovendien maakt deze opzet het eenvoudig om meerdere scanners parallel te laten werken, wat bijdraagt aan een snellere en bredere kwetsbaarheidsdetectie.

De workflow van de scanners wordt volledig geautomatiseerd en op vaste tijden uitgevoerd via Azure Logic Apps. Deze functioneren op basis van triggers zoals tijdsintervallen. Zodra een ‘trigger’ actief wordt, wordt er een Azure Container Instance aangeroepen. In deze ACI kan de gewenste scanner worden uitgevoerd, zoals Nikto, OWASP ZAP of Nuclei.

In de container voert de scanner de scan uit op de doelapplicaties. Elke scanner wordt apart geconfigureerd op basis van het type kwetsbaarheden dat gedetecteerd moet worden. Na de scan slaat de scanner de resultaten in een gewenst formaat op, dit is HTML of JSON afhankelijk van de scanner, op een Azure File Share of kan doorgestuurd worden naar een centrale webapplicatie om daar in een overzicht getoond te worden.

Door deze geautomatiseerde manier van werken kunnen de scans zonder menselijke tussenkomst blijven lopen en herhaalbaar uitgevoerd worden. Omdat containers van elkaar gescheiden zijn, is iedere uitvoering ook gescheiden van een andere, waardoor veiligheid en betrouwbaarheid toeneemt. Bovendien kan op deze manier eenvoudig ook met meerdere scanners parallel worden gewerkt, hetgeen positief bijdraagt aan de snelheid en het aantal te detecteren kwetsbaarheden.

4.5. Asset discovery

Voor het in kaart brengen van de publieke domeinen en webservices van Baloise werd voortgebouwd op het werk van een vorige stagiair. Deze had reeds een basisinventaris opgesteld met behulp van Subfinder, een open-source tool die automatisch subdomeinen opspoort via passieve en actieve methodes.

Om deze inventarisatie verder aan te vullen en actualiseren, werd gebruikgemaakt van crt.sh, een openbare Certificate Transparency-logservice. Deze tool maakt het mogelijk om domeinen te detecteren op basis van SSL/TLS-certificaten waarbij de naam Baloise bevat. Hierdoor kunnen ook minder zichtbare subdomeinen aan het licht komen die niet direct in DNS of web crawling-resultaten voorkomen.

Tot slot werd een manuele controle uitgevoerd op de DNS-records van de officiële domeinen van Baloise. Dit liet toe om bijkomende subdomeinen en diensten te identificeren die rechtstreeks beheerd worden door de organisatie. Door deze gecombineerde aanpak werd een zo volledig mogelijk overzicht opgesteld van alle externe digitale aanraakpunten die in aanmerking komen voor monitoring en kwetsbaarheidsanalyse.

4.6. Uptime Kuma

De monitoringtool Uptime Kuma is geïntegreerd in de infrastructuur van het project en wordt automatisch uitgerold via de Terraform-code binnen de Azure DevOps-pipeline. Bij elke wijziging of nieuwe uitrol van de infrastructuur wordt ook Uptime Kuma als onderdeel van het geheel geprovisioned, waardoor handmatige installatie niet nodig is. De toepassing wordt gehost in een Azure container instance wat configuratie makkelijk maakt.

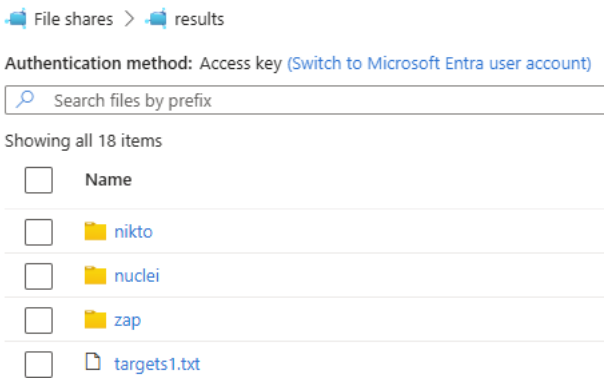
Binnen het project wordt Uptime Kuma gebruikt voor het continu monitoren van de beschikbaarheid van publieke webapplicaties en endpoints. De tool voert ook HTTP(S)-checks uit en controleert of de applicaties correct reageren binnen een vooraf bepaalde responstijd. Bij afwijkingen, downtime of foutmeldingen kan Uptime Kuma direct een notificatie uitsturen via meerdere kanalen, zoals e-mail of webhook-integraties. Ook SSL-vervaldata worden gecontroleerd, wat helpt bij het tijdig vernieuwen van certificaten en het voorkomen van ongewenste onderbrekingen.

Om Uptime Kuma centraal te beheren, werd een Python-script ontwikkeld dat het toevoegen en verwijderen van publieke webapplicaties automatiseert. Hierdoor kunnen wijzigingen in de applicaties, zoals het toevoegen van nieuwe domeinen of het verwijderen van oude domeinen, snel en gestructureerd worden doorgevoerd, zonder handmatige interactie met de webinterface van Uptime Kuma.
Het script maakt gebruik van de Uptime Kuma API en leest de verschillende publieke domeinen van een tekst bestand. Naast het beheren van checks, is het script ook in staat om de actuele uptimestatus **van** domeinen op te vragen. Deze gegevens kunnen vervolgens automatisch worden gelogd of geëxporteerd naar een apart document (zoals een CSV- of JSON-bestand), waardoor de monitoringresultaten eenvoudig extern kunnen worden bewaard, gedeeld of verwerkt.

Door deze integratie wordt beschikbaarheidsmonitoring een vast onderdeel van het bredere External Attack Surface Management-systeem. Dit zorgt voor een continue passieve bewaking van publieke applicaties, wat toelaat om sneller te reageren bij storingen of aanvallen en verhoogt zo de algehele weerbaarheid van de externe infrastructuur.

4.7. Resultaten

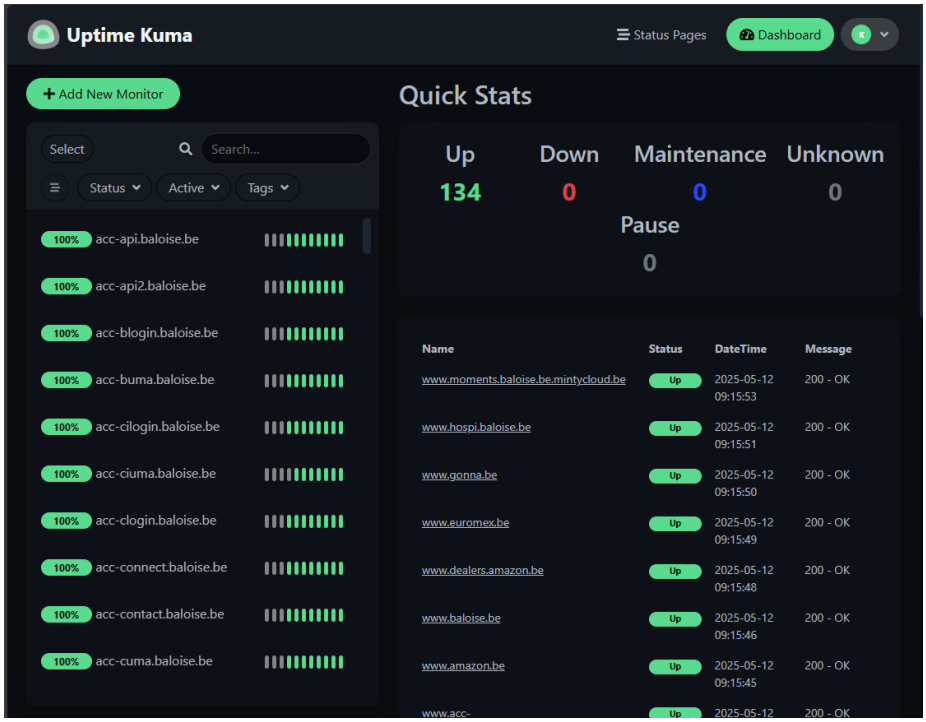
In dit onderdeel worden de gerealiseerde resultaten per component toegelicht. Elk onderdeel van het systeem werd afzonderlijk getest en geverifieerd op werking, automatisering en integratie. De combinatie van deze modules vormt een volledig functioneel en schaalbaar External Attack Surface Management-systeem.



Figuur 4: mappenstructuur van de file share

4.7.1. Uptime Kuma

Het monitoringsysteem Uptime Kuma werd succesvol uitgerold als een Azure Container Instance en geconfigureerd voor het periodiek controleren van de publieke domeinen van Baloise. Via een intuïtieve webinterface worden uptimepercentages, statuscodes en responstijden inzichtelijk gemaakt.



Figuur 5: Uptime Kuma applicatie

Daarnaast werd een eigen Python-script ontwikkeld om het beheer van de monitors te automatiseren. Het script maakt gebruik van de Uptime Kuma API en stelt de gebruiker in staat om eenvoudig nieuwe domeinen toe te voegen of te verwijderen op basis van een tekstbestand. Verder kan het script ook de actuele uptimegegevens opvragen en deze exporteren naar een tekstbestand, zodat de monitoringresultaten ook buiten de webinterface beschikbaar zijn voor analyse of rapportering.

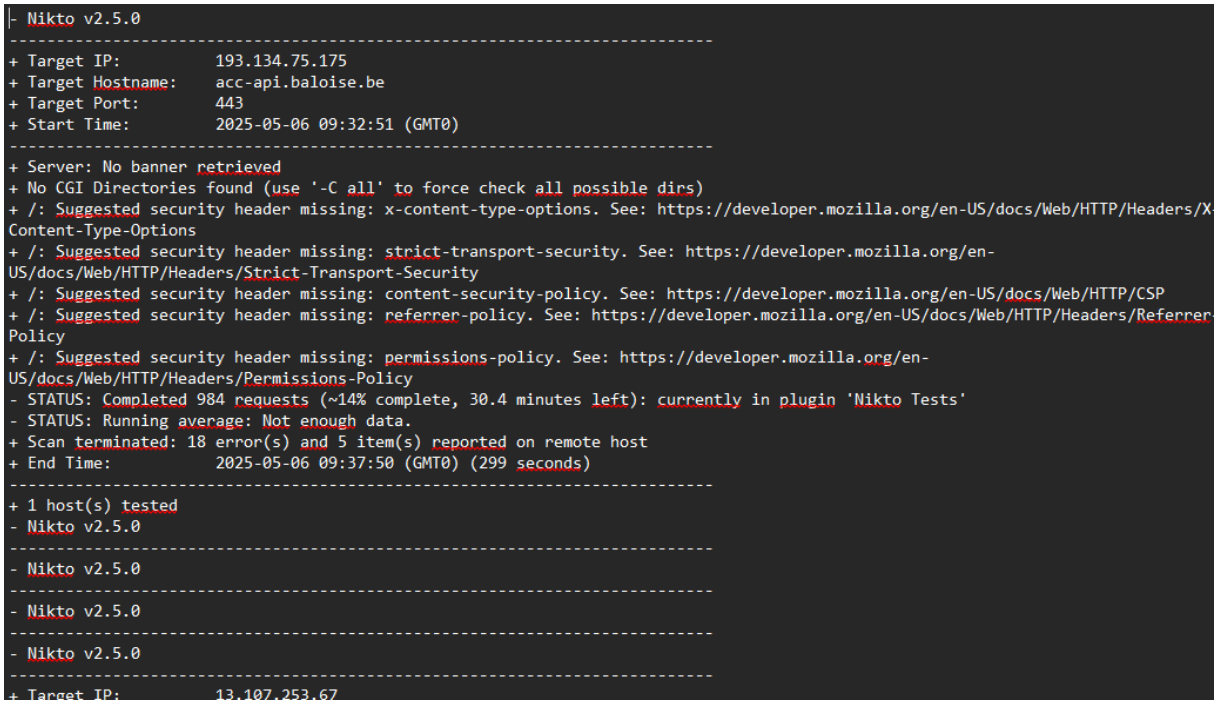
```
(venv) kat@Ubuntu-setup:~/kuma$ python3 kumascript.py manage
 Added monitor: https://berekenjebafamilialepremie.be
 Added monitor: https://berekenjeongevallenpremie.be
 Added monitor: https://berekenmijnpremie.be
 Added monitor: https://calculermaprime.be
 Added monitor: https://calculezvotreprimeaccidents.be
```

Figuur 6: Python script voor het beheer van Uptime Kuma

4.7.2. Nikto

Nikto werd geïntegreerd in de infrastructuur via een containergebaseerde uitvoering, aangestuurd door Logic Apps. Bij uitvoering scant Nikto de doelwebserver op een breed scala aan configuratiefouten, verouderde softwareversies en bekende kwetsbare scripts.

De resultaten worden automatisch gegenereerd in CSV formaat, en opgeslagen in een centrale Azure File Share. In tests detecteerde Nikto meerdere testcases zoals toegankelijke admin-pagina's en verouderde HTTP-headers, wat bevestigt dat de integratie correct functioneert. Hoewel de scanner bekendstaat om een hoger aantal false positives, levert hij waardevolle inzichten voor snelle, brede detectie.



Figuur 7: Voorbeeld Nikto

Voor het uitvoeren van vulnerabilityscans met Nikto werd gekozen voor een directe command-line-aansturing binnen de containeromgeving. Het gebruikte commando is als volgt:

/opt/nikto/program/nikto.pl -h "$url" -nossl -no404 -Display 3 -Format csv -output "$OUTPUT_FILE"

In deze opdracht wordt Nikto aangeroepen via het Perl-script nikto.pl, met als doeladres de variabele $url, die tijdens de containeruitvoering dynamisch wordt ingevuld. De optie -nossl schakelt SSL-gerelateerde controles uit. Deze keuze werd bewust gemaakt om de scanduur en false-positves te beperken, aangezien SSL-checks in Nikto vaak aanzienlijk vertragen en voornamelijk gericht zijn op certificaatfouten.

De optie -no404 zorgt ervoor dat Nikto pagina's die een 404-foutmelding genereren, uitsluit van analyse. Dit voorkomt ruis in de resultaten en verkleint het aantal false-positves. De parameter -Display 3 bepaalt het detailniveau van de output, waarbij niveau 3 alle gevonden kwetsbaarheden en serverfouten toont zonder overbodige headers of metadata.

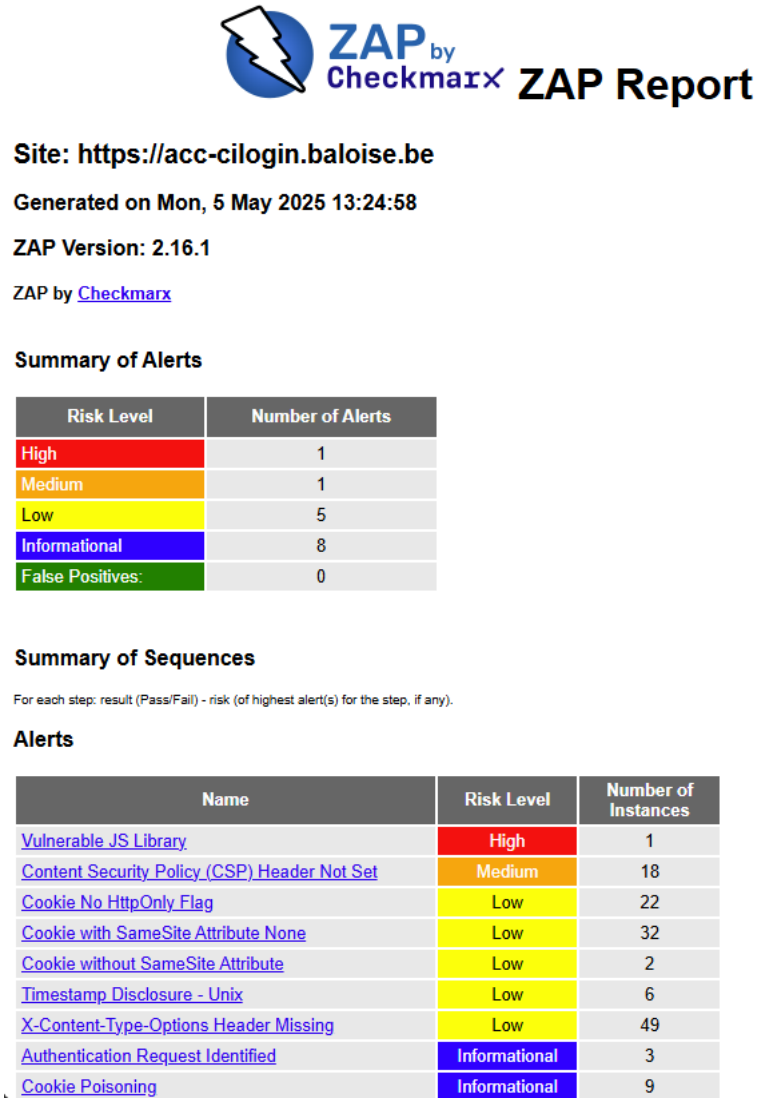
De uitvoer wordt via -Format csv opgeslagen in een CSV-bestand, op het pad dat is gedefinieerd in $OUTPUT_FILE. CSV werd gekozen als bestandsformaat vanwege de eenvoud van verwerking en compatibiliteit met verdere verwerkingstools, zoals Power BI of Excel.

Deze configuratie levert gestructureerde, beknopte en verwerkbare scanresultaten op, en maakt het mogelijk om snel zicht te krijgen op veelvoorkomende serverfouten en misconfiguraties, zoals verouderde softwareversies, toegankelijke adminpanelen of directory listing.

4.7.3. Owasp Zap

OWASP ZAP werd eveneens gecontaineriseerd en automatisch aangestuurd via een Logic App. De tool werd geconfigureerd met een vooraf ingestelde scan policy, gericht op het detecteren van OWASP Top 10-kwetsbaarheden.

Tijdens testuitvoeringen werden succesvolle detecties verricht van onder andere XSS (Cross-Site Scripting) en insecure cookies. De scanresultaten worden gegenereerd in XML-formaat en klaargezet voor import in Power BI. ZAP is met name nuttig gebleken voor diepere applicatiescans waarbij interactie vereist is, zoals bij loginpagina's of formulierverwerking.



ZAP by Checkmarx ZAP Report

Site: https://acc-cilogin.baloise.be

Generated on Mon, 5 May 2025 13:24:58

ZAP Version: 2.16.1

ZAP by Checkmarx

Summary of Alerts

Risk Level	Number of Alerts
High	1
Medium	1
Low	5
Informational	8
False Positives:	0

Summary of Sequences

For each step: result (Pass/Fail) - risk (of highest alert(s) for the step, if any).

Alerts

Name	Risk Level	Number of Instances
Vulnerable JS Library	High	1
Content Security Policy (CSP) Header Not Set	Medium	18
Cookie No HttpOnly Flag	Low	22
Cookie with SameSite Attribute None	Low	32
Cookie without SameSite Attribute	Low	2
Timestamp Disclosure - Unix	Low	6
X-Content-Type-Options Header Missing	Low	49
Authentication Request Identified	Informational	3
Cookie Poisoning	Informational	9

Figuur 8: Voorbeeld Owasp Zap

Voor het uitvoeren van vulnerabilityscans met **OWASP ZAP**, werd gebruikgemaakt van het volgende commando:

> *zap.sh -cmd -quickurl "$url" -quickout "$OUTPUT_DIR/zap_$${SANITIZED_NAME}.xml" -quickprogress -config spider.maxDuration=20 -config spider.maxDepth=2 -config*

In deze configuratie wordt ZAP opgestart in command-line mode (-cmd), wat geschikt is voor headless uitvoering in een CI/CD-pijplijn of container. Met -quickurl "$url" wordt een specifieke doel-URL meegegeven die moet worden gescand. De optie -quickout bepaalt waar de scanresultaten worden weggeschreven. In dit geval gebeurt dat in XML-formaat naar een bestandsnaam die gebaseerd is op het gesaneerde domein, wat het verwerken van meerdere domeinen overzichtelijk houdt.

De vlag -quickprogress zorgt ervoor dat de voortgang van de scan actief wordt weergegeven, wat nuttig is voor logging of foutopvolging. Vervolgens worden er twee configuratieparameters ingesteld via -config:

- spider.maxDuration=20: beperkt de duur van de spiderfase tot 20 minuten, zodat scans beheersbaar blijven.

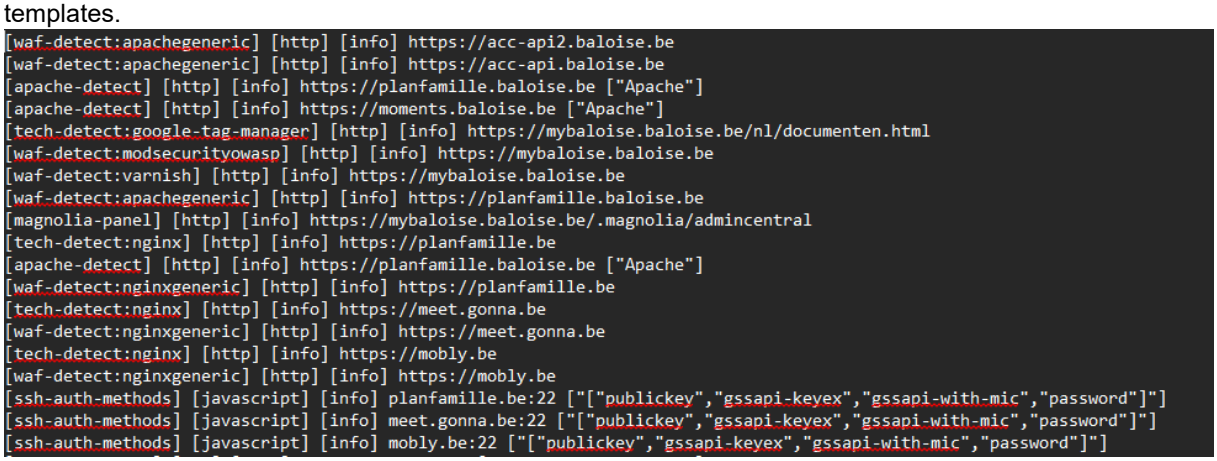
- spider.maxDepth=2: beperkt de diepte van het crawlen, zodat de scan zich niet onbeperkt verdiept in alle subnavigatiestructuren van de webapplicatie.

Deze instellingen zorgen ervoor dat OWASP ZAP op een gecontroleerde manier draait, met een duidelijke balans tussen diepgang en efficiëntie, en zonder het risico op langdurige of overmatige belasting van de doelapplicatie. De gegenereerde XML-output kan vervolgens worden verwerkt voor verdere analyse of rapportage in het dashboard.

4.7.4. Nuclei

Nuclei is geconfigureerd als een high-performance scanner die YAML-templates gebruikt om doelgerichte kwetsbaarheden op te sporen. De integratie werd opgezet via een container die meerdere templates tegelijk kan draaien op vooraf ingestelde domeinen.

Tijdens de testfase werden templates ingezet voor CVE-detectie, misconfiguraties en headerscans. De uitvoer werd opgeslagen in JSON-formaat en geïntegreerd in het centrale Power BI-dashboard. Dankzij de snelheid van Nuclei kon het volledige domeinportfolio frequent gescand worden, zonder merkbare vertraging. Nuclei leverde consistente, gedetailleerde output en was eenvoudig uitbreidbaar met eigen templates.



Figuur 9: Voorbeeld Nuclei

Om de vulnerabilityscans met Nuclei automatisch uit te voeren, werd gebruikgemaakt van een commandoregel die afgestemd is op de behoeften van het project en de structuur van de containeromgeving. Het gebruikte commando is als volgt:

nuclei -l "$file" -o /mnt/results/nuclei/temp_results.txt -as -pd

Hierbij staat -l "$file" voor de invoer van een lijst met domeinnamen. De parameter -o specificeert het pad waar de resultaten worden opgeslagen, in dit geval is dat een tijdelijk bestand in een gemounte map die wordt gedeeld met de rest van het systeem.

De vlag -as activeert automatic scan strategy, waarmee Nuclei automatisch templates kiest die het meest relevant zijn voor het doelwit. De optie -pd (passive detection) schakelt actieve benaderingen uit, zodat Nuclei alleen informatie verzamelt zonder interactie die als aanvallend kan worden geïnterpreteerd. Dit is bijzonder nuttig in omgevingen waar voorzichtigheid geboden is bij scans op productieomgevingen.

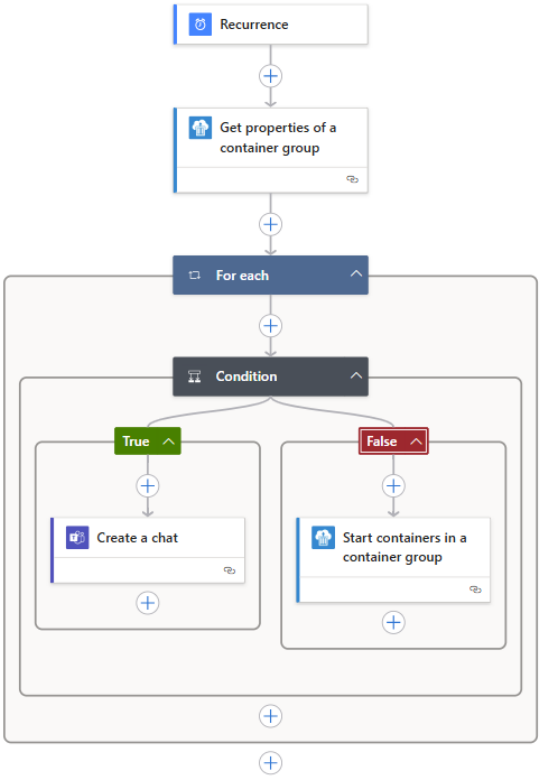
Deze configuratie zorgt ervoor dat scans efficiënt, gecontroleerd en veilig verlopen binnen de gecontaineriseerde omgeving, waarbij de resultaten centraal worden opgeslagen en later verwerkt voor rapportering of visualisatie.

4.7.5. Logic app

De Logic Apps vormen de automatiseringslaag binnen het systeem. Ze zijn zo geconfigureerd dat ze op vooraf ingestelde tijdstippen automatisch kwetsbaarheidsscans opstarten. Elke Logic App is gekoppeld aan een specifieke container instance waarin een scanner (Nikto, Nuclei of OWASP ZAP) wordt uitgevoerd. Dit gebeurt via een HTTP-aanroep of een Azure Function-trigger.

Tijdens de uitvoering controleert de Logic App eerst of de corresponderende container instance op dat moment al actief is. Indien dit het geval is, bijvoorbeeld omdat een vorige scan nog niet is afgerond, wordt er automatisch een waarschuwing verstuurd via Microsoft Teams. Zo worden problemen met de scanner snel gemeld en kunnen deze ook worden opgelost.

Deze meldingsfunctionaliteit werd gerealiseerd door gebruik te maken van de ingebouwde Teams-connector in Logic Apps, waarmee berichten in specifieke Teams-kanalen kunnen worden gepost. Dit verhoogt de zichtbaarheid van uitzonderingssituaties en laat toe om snel in te grijpen wanneer een scan afwijkt van de verwachte workflow.



Figuur 10: Logic app diagram

5. Besluit

Tijdens dit project werd onderzocht hoe een geautomatiseerd, open-source External Attack Surface Management-systeem kon worden opgezet ter ondersteuning van de beveiligingsstrategie van Baloise. Door middel van een gefaseerde aanpak, bestaande uit inventarisatie, detectie en rapportering, werd een schaalbare en flexibele oplossing gerealiseerd die publieke webapplicaties op een efficiënte manier monitort en scant.

De uiteindelijke oplossing combineert meerdere open-source scanners (Nikto, Nuclei en OWASP ZAP) met een monitoringsysteem (Uptime Kuma), aangestuurd via Azure Logic Apps en uitgerold via Terraform. De resultaten worden centraal opgeslagen en kunnen worden geraadpleegd voor verdere analyse of rapportering. Elk onderdeel werd afzonderlijk getest, gevalideerd en geautomatiseerd, met als resultaat een reproduceerbare infrastructuur die eenvoudig aanpasbaar is aan toekomstige noden.

Het project toont aan dat open-source tools, mits goed geïntegreerd, een waardevol alternatief kunnen vormen voor commerciële kwetsbaarheidsoplossingen. Dankzij de modulaire opbouw kan het systeem bovendien eenvoudig uitgebreid worden, bijvoorbeeld met extra scanners.

Toch zijn er ook nog groeimogelijkheden. Een belangrijke volgende stap is de volledige integratie met Power BI, zodat de scanresultaten automatisch en visueel worden weergegeven in dynamische dashboards. Dit zou het mogelijk maken om trends in kwetsbaarheden beter op te volgen en sneller in te grijpen bij kritieke bevindingen. Het gebruik van Azure Functions voor scriptuitvoering kunnen de schaalbaarheid en onderhoudsvriendelijkheid verder verbeteren.

Al met al werd een functioneel, toekomstgericht systeem opgeleverd dat niet alleen inspeelt op de huidige beveiligingsbehoeften van Baloise, maar ook ruimte laat voor verdere automatisering, visualisatie en uitbreiding.

6. Literatuurlijst

Alazmi, S., & De Leon, D. C. (2022). *A systematic literature review on the characteristics and effectiveness of web application vulnerability scanners.* IEEe Access.

Automation Guide. (2024). Opgehaald van zaproxy: https://www.zaproxy.org/docs/getting-further/automation/automation-options/

Automation of Nessus Scan via API. (2022, Maart 7). Opgehaald van Tenable: https://community.tenable.com/s/question/0D53a00008OeTHZCA3/automation-of-nessus-scan-via-api-without-tenableio?language=en_US

Awati, R. (2025, Februari). *What is the Nessus vulnerability scanning platform?* Opgehaald van Techtarget: https://www.techtarget.com/searchnetworking/definition/Nessus

Greenbone Community Edition – Documentation. (2024). Opgehaald van greenbone: https://greenbone.github.io/docs/latest/

Introduction to Nuclei Templates. (2025). Opgehaald van projectdiscovery: https://docs.projectdiscovery.io/templates/introduction

Network vulnerability scanners benchmark 2024. (2024, Mei 24). Opgehaald van pentest-tools: https://pentest-tools.com/benchmarks/network-vulnerability-scanners

Nikto: An Overview. (2023, December 28). Opgehaald van Range Force: https://www.rangeforce.com/blog/nikto-an-overview

Pricing calculator. (2025, Mei). Opgehaald van Microsoft: https://azure.microsoft.com/en-us/pricing/calculator/

Shash, M. (2024, November 18). *5 Best Uptime Kuma Alternatives: Exploring Uptime Monitoring Tools*. Opgehaald van Medium: https://medium.com/@max.shash/5-best-uptime-kuma-alternatives-exploring-uptime-monitoring-tools-f3d1a6d71b03

Sullo, C., & Lodge, D. (2024). *Nikto 2.5*. Opgehaald van cirt: https://www.cirt.net/Nikto2

Virtual machines in Azure. (2024, Augustus 22). Opgehaald van Microsoft: https://learn.microsoft.com/en-us/azure/virtual-machines/overview

What is Azure Container Instances? (2024, November 11). Opgehaald van Microsoft : https://learn.microsoft.com/en-us/azure/container-instances/container-instances-overview

What is Azure Kubernetes Service (AKS)? (2024, Mei 12). Opgehaald van Microsoft: https://learn.microsoft.com/en-us/azure/aks/what-is-aks