Optimalisatie van back-upstrategieën voor Azure PostgreSQL en MySQL databases bij Forvis Mazars met behulp van immutabele opslag en automatische back-ups: Een Proof-of-Concept.

Naoufal Bouazzaoui.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Martijn Saelens
Co-promotor: Rémy Tetaert
Academiejaar: 2024-2025
Eerste examenperiode

Departement IT en Digitale Innovatie .

HO
GENT

Woord vooraf

Met trots en voldoening presenteer ik deze bachelorproef, het sluitstuk van mijn opleiding Toegepaste Informatica aan de HoGent. In het begin toen ik begon met zoeken naar een onderwerp had ik geen idee waar ik moest beginnen en vond ik dit dus ook een grote uitdaging. Ik begon met het zoeken naar bepaalde interessante onderwerpen en toen kwam ik uiteindelijk op dit onderwerp. Ik heb sterke interesses in cybersecurity dus wou ik dit implementeren in mijn bachelorproef. Daarbij ontwikkelde ik ook interesse in back-ups tijdens het opleidingsonderdeel Cybersecurity Advanced. Het schrijven van deze bachelorproef was niet alleen een uitdagend leerproces, maar ook een unieke kans om mijn interesse in systeembeheer en cybersecurity verder te verdiepen. Dit project heeft me niet alleen geholpen mijn technische kennis te versterken, maar ook om praktijkervaring op te doen binnen een professionele context.

Ik wil graag mijn dankbaarheid uitdrukken aan mijn co-promotor, Rémy Tetaert, voor zijn waardevolle begeleiding, expertise en tijd tijdens dit proces. Zijn inzichten, ondersteuning en begeleiding waren heel belangrijk voor mij. Daarnaast wil ik mijn promotor, Martijn Saelens, bedanken voor zijn constructieve feedback en kritische blik, die me steeds hebben geholpen om mijn werk te verbeteren.

Tot slot wil ik ook mijn familie en vrienden bedanken voor hun geduld, steun en motivatie gedurende deze periode. Zonder hen zou dit niet mogelijk zijn geweest.

Samenvatting

Dit onderzoek heeft als doel de back-upstrategie van Forvis Mazars te analyseren en optimaliseren, met een focus op het verbeteren van de bescherming tegen cyberdreigingen zoals ransomware-aanvallen. Het bedrijf maakt gebruik van Azure voor het beheren van zijn data en heeft momenteel een back-upstrategie die bestaat uit automatische full back-ups van twee databases en een manueel uitgevoerd script voor het maken van back-ups van specifieke databases naar een Azure Storage-account.

De bescherming van bedrijfsdata is essentieel voor het waarborgen van de continuïteit van een organisatie, en back-ups spelen hierbij een cruciale rol. Het oorspronkelijke back-upplan van Forvis Mazars was echter kwetsbaar omdat het geen gebruik maakte van immutable storage, waardoor de back-ups gevoelig waren voor manipulatie door aanvallers, bijvoorbeeld in het geval van ransomware.

Om de back-upstrategie te verbeteren, werd een Proof-of-Concept (PoC) ontwikkeld in een lokale testomgeving met VirtualBox, waarin de effectiviteit van immutable storage werd getest. In deze simulatie werd aangetoond dat immutable storage het mogelijk maakt om back-ups te beschermen tegen manipulatie, zelfs wanneer een aanvaller volledige controle heeft over het systeem. Dit werd bevestigd door het encrypten van back-upbestanden en het verwijderen van de actieve database, waarbij de integriteit van de immutabele back-ups behouden bleef.

Verder werd de huidige back-upstrategie van Forvis Mazars geanalyseerd, waarbij verschillende verbeterpunten werden geïdentificeerd. Aanbevelingen zoals het implementeren van immutable storage in de Azure-omgeving en het automatiseren van manuele back-ups via Kubernetes cronjobs werden geformuleerd. Deze verbeteringen dragen bij aan het vergroten van de veiligheid en betrouwbaarheid van de back-ups en zorgen voor snellere en effectievere herstelmogelijkheden.

In dit onderzoek werd ook een literatuurstudie uitgevoerd waarin verschillende back-upmethoden, technieken en ransomware-resistente oplossingen werden onderzocht. De opgedane kennis heeft bijgedragen aan het ontwikkelen van een geoptimaliseerde back-upstrategie voor Forvis Mazars, die niet alleen de veiligheid verhoogt, maar ook de bedrijfscontinuïteit beschermt tegen cyberaanvallen. De inzichten en technieken die in dit onderzoek zijn gepresenteerd, kunnen ook waardevolle toepassingen hebben voor andere organisaties die hun back-upstrategieën willen versterken, met name binnen Azure-omgevingen.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Lijst van codefragmenten

1

Inleiding

De beveiliging van gegevens is van cruciaal belang voor organisaties, vooral gezien de toenemende dreigingen van cyberaanvallen zoals ransomware. Gezien de digitale transformatie die veel bedrijven doormaken, zijn betrouwbare en veilige back-up oplossingen essentieel om de continuïteit van de bedrijfsvoering te waarborgen. Dit geldt in het bijzonder voor bedrijven die werken met cloudplatformen zoals Microsoft Azure, waar databases zoals PostgreSQL en MySQL vaak cruciaal zijn voor het dagelijks functioneren. Bij Forvis Mazars worden momenteel back-ups van Azure-databases gemaakt via een combinatie van automatische volledige back-ups en handmatige back-ups via scripts. Deze aanpak kent echter enkele beperkingen, zoals de onregelmatige uitvoering van de handmatige back-ups en een gebrek aan geautomatiseerde processen, wat de veiligheid en efficiëntie van het systeem in gevaar kan brengen.

1.1. Probleemstelling

In deze bachelorproef wordt de huidige back-upstrategie van Forvis Mazars geanalyseerd en geoptimaliseerd. De focus ligt hierbij op het verbeteren van de back-upstrategie voor de Azure PostgreSQL en MySQL databases, met aandacht voor ransomware-resistentie en de integratie van immutable opslagtechnieken. Het doel is om de bestaande strategie te versterken en de kans op dataverlies door cyberaanvallen te minimaliseren. Daarnaast zal er een Proof-of-Concept worden uitgevoerd om de effectiviteit van immutable opslag te testen in een scenario waarbij een ransomware-aanval wordt nagebootst op mock-up bestanden. De probleemstelling van dit onderzoek is dat de huidige back-upstrategie bij Forvis Mazars niet voldoende robuust is om het risico op dataverlies door ransomware effectief te mitigeren. De doelgroep van dit onderzoek bestaat uit Forvis Mazars

1.2. Onderzoeksvraag

De onderzoeksvraag van deze bachelorproef luidt:

Hoe kan de back-upstrategie voor Azure PostgreSQL en MySQL databases bij Forvis Mazars worden geoptimaliseerd met behulp van immutable opslag en automatische back-ups?

1.3. Deelvragen

De onderzoeksvraag kan verder opgedeeld worden in de volgende deelvragen.:

- Hoe veilig en betrouwbaar zijn de huidige back-upoplossingen van Forvis Mazars voor Azure PostgreSQL en MySQL databases?
- Welke rol speelt immutable opslag in het beschermen van back-ups tegen ransomware en andere vormen van dataverlies?
- Wat zijn de belangrijkste uitdagingen bij het integreren van immutable opslag met Azure cloud back-upsystemen?

1.4. Onderzoeksdoelstelling

Het doel van dit onderzoek is om de back-upstrategie van Forvis Mazars te optimaliseren door de huidige back-upmethoden te analyseren en te verbeteren. Het onderzoek richt zich specifiek op het implementeren van immutable opslag om de bescherming tegen ransomware-aanvallen te versterken. Daarnaast wordt de automatisering van de manuele back-ups onderzocht en geïmplementeerd, aangezien deze momenteel niet frequent genoeg worden uitgevoerd. Een Proof-of-Concept (PoC) zal worden uitgevoerd door virtuele machines te gebruiken en een ransomware-aanval na te bootsen om de effectiviteit van de immutable opslag te testen. Ook zal het herstelproces getest worden om een beter beeld te krijgen van de snelheid waarmee een restore kan worden uitgevoerd. Dit proefproject zal verder bijdragen aan het verbeteren van de bestaande automatische back-upstructuur door het toevoegen van meer geautomatiseerde processen, wat de efficiëntie en de veiligheid van het back-upbeheer binnen het bedrijf zal vergroten.

1.5. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

Hoofdstuk 2 biedt een overzicht van de huidige kennis en technologieën rondom back-upstrategieën, ransomware-beveiliging en immutable opslag. De literatuur helpt de basis te leggen voor het verbeteren van de back-upbeveiliging bij Forvis Mazars.

In hoofdstuk 3 worden de stappen van het onderzoek beschreven. Een requirementsanalyse werd uitgevoerd om de huidige back-upstrategie van Forvis Ma-

zars te evalueren en verbeterpunten te identificeren. Vervolgens werd de opzet voor een Proof-of-Concept (PoC) uitgewerkt.

Hoofdstuk 4 onderzoekt de huidige back-upstrategie bij Forvis Mazars en stelt verbeteringen voor, zoals de automatisering van handmatige back-ups en het implementeren van immutable opslag voor verhoogde veiligheid.

In hoofdstuk 5 wordt de uitvoering van de Proof-of-Concept beschreven, waarin immutable opslag wordt getest door een ransomware-aanval na te bootsen op een lokale omgeving.

In hoofdstuk 6, tenslotte, wordt de conclusie gegeven van het onderzoek.

2

Stand van zaken

2.0.1. Back-ups in het kader van bedrijfscontinuïteit en disaster recovery

Bedrijfscontinuïteit verwijst naar de aanpak en procedures dat een bedrijf gebruikt om de voortgang van zijn werkzaamheden te bewaren, zelfs in het geval van incidenten. Deze incidenten kunnen variëren van relatief kleine problemen, zoals een gebroken netwerkverbinding, tot grote natuurrampen zoals aardbevingen. Omdat er zoveel soorten incidenten kunnen gebeuren is het moeilijk om een oplossing te vinden die ervoor zorgt dat bedrijven in alle gevallen beschermt zijn. In plaats daarvan gebruiken bedrijven een mix van strategieën en technologieën om de continuïteit van hun processen te beschermen.

De twee belangrijkste concepten voor de bedrijfscontinuïteit zijn hoge beschikbaarheid en disaster recovery. Volgens Zhu et al. (2015) duidt hoge beschikbaarheid op het feit dat een bedrijf zodanig is ingericht dat het kan blijven draaien, zelfs als bepaalde systemen of componenten uitvallen.

Een voorbeeld hiervan zijn twee routers die zijn geconfigureerd in een actieve-passieve opstelling. In deze configuratie is één router de primaire router die al het inkomende en uitgaande verkeer verwerkt, terwijl de andere router als reserve werkt. In het geval dat de primaire router faalt door een hardware storing of netwerkprobleem, neemt de tweede router automatisch de rol van de primaire router over zonder dat dit merkbare impact heeft op de netwerkverbindingen van de organisatie. Hierdoor blijft de beschikbaarheid van het netwerk gegarandeerd en blijft de downtime laag.

Disaster recovery (DR) is een onderdeel van bedrijfscontinuïteit dat zich specifiek richt op het herstellen van bedrijfsactiviteiten na een incident zoals een cyberaanval of een ernstige storing. Terwijl bedrijfscontinuïteit zich richt op bredere preventieve maatregelen om de continuïteit te waarborgen, focust disaster recovery zich juist op de praktische stappen en hulpmiddelen die nodig zijn om de or-

ganisatie na een verstoring weer snel operationeel te maken. Het doel van disaster recovery is om schade zoveel mogelijk te beperken en de normale gang van zaken zo snel mogelijk te herstellen. Back-ups spelen een belangrijke rol voor de continuïteit van een bedrijf en zijn vaak de eerste stap bij het opstellen van een disaster recovery plan (DRP).

Bij een optimale situatie is er na een incident geen data verloren en is alle data relatief snel terug beschikbaar. Indien een bedrijf geen back-ups heeft van belangrijke data zal de data in het geval van een incident verloren raken. Zonder back-ups zal het ook een grotere uitdaging zijn voor het bedrijf om de normale bedrijfsactiviteiten terug uit te voeren. Een belangrijke doelstelling van een bedrijf is winst maken. In het geval van een incident waarbij de bedrijfsactiviteiten niet normaal kunnen verlopen zal deze doelstelling verhinderd worden en zal er dus financieel verlies optreden.

Bij specifieke bedreigingen, zoals ransomware-aanvallen spelen ransomware-resistente back-ups een cruciale rol. Door back-ups te beveiligen tegen ransomware-aanvallen kunnen bedrijven hun data herstellen zonder losgeld te betalen. Dit benadrukt het belang van back-ups die niet alleen snel toegankelijk zijn, maar ook bestand zijn tegen digitale bedreigingen (Ghazi & H. O. Nasereddin, 2013).

2.0.2. Back-upmethoden en -technieken

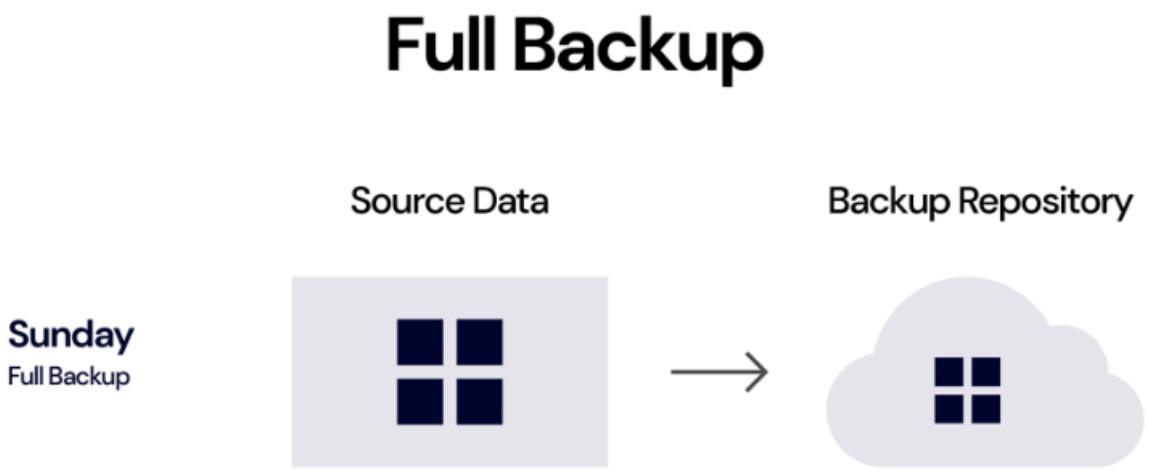
Back-ups zijn een belangrijk onderdeel voor het beheren en beveiligen van data binnen organisaties. Ze zorgen voor de continuïteit van bedrijfssystemen in het geval van een incident zoals een cyberaanval. Back-ups zijn snapshots van gegevens die op een bepaald tijdstip zijn gemaakt, opgeslagen in een wereldwijd gebruikelijk formaat en gedurende een bepaalde periode van bruikbaarheid worden bijgehouden, waarbij elke volgende kopie van de gegevens onafhankelijk van de eerste wordt bewaard (Nelson & Brown, 2011).

Door een aparte kopie van de gegevens te bewaren, kunnen bedrijven en individuen na een incident hun systemen of bestanden herstellen naar een eerdere, veilige staat. Hierbij kunnen back-ups zowel volledige datasets als selectieve bestandstypen omvatten, afhankelijk van de strategie en de specifieke behoeften van de organisatie.

Back-ups zijn een preventieve maatregel en het doel ervan is om dataverlies tegen te gaan. Dataverlies kan optreden door menselijke fouten, cyberaanvallen, en natuur- of bedrijfsrampen. Daarbij speelt beveiliging een belangrijke rol in een tijd waarin ransomware-aanvallen en datalekken frequenter voorkomen. Door back-ups versleuteld op te slaan en te beveiligen tegen ongeautoriseerde toegang, kunnen bedrijven zich beschermen tegen het verliezen van data.

Full back-ups

Een full back-up is een back-upmethode waarbij alle gegevens van een systeem op een specifiek moment volledig worden gekopieerd en opgeslagen.



Figuur 2.1: Representatie van een full back-up (Rivas, 2022)

Dit betekent dat elk bestand zonder uitzonderingen wordt gekopieerd, zodat er een exacte kopie van de volledige dataset ontstaat, zoals Beard (2018) beschrijft. Wanneer er zich een probleem voordoet, zoals het falen van een harde schijf, kan het hele bestandssysteem vanaf deze back-up volledig worden hersteld op een nieuwe schijf. Daarnaast kunnen ook individuele bestanden die verloren zijn gegaan, gemakkelijk worden teruggehaald uit de back-up.

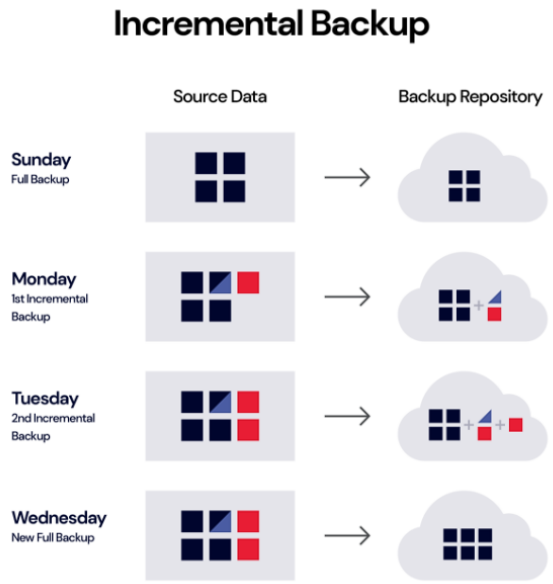
Dit soort back-up zorgt ervoor dat alle gegevens veilig zijn opgeslagen (Chervenak e.a., 1998). Full back-ups vormen vaak de basis van een back-upstrategie en worden regelmatig uitgevoerd om ervoor te zorgen dat alle gegevens volledig hersteld kunnen worden. Het concept en de implementatie van een full back-up is relatief eenvoudig omdat alle gegevens op één locatie zijn opgeslagen.

Aan de andere kant is er het probleem van opslagcapaciteit. Stel bijvoorbeeld dat een bedrijf elke nacht een full back-up maakt van zijn servers naar een opslagplaats in de cloud, waarbij per keer 500 GB aan data wordt opgeslagen. Na een week is er al 3,5 terabyte aan gegevens in de cloud opgeslagen. Aangezien cloudproviders vaak kosten in rekening brengen op basis van gebruikte opslagcapaciteit en dataverkeer, kan dit snel leiden tot aanzienlijke maandelijkse kosten. Bedrijven met een beperkt IT-budget kunnen hierdoor in de problemen komen of worden gedwongen om strenger te selecteren welke gegevens ze precies opslaan in de back-up. Daarbij kan het proces zelf ook veel tijd innemen, wat een uitdaging kan zijn in omgevingen waar snelle gegevensbeschikbaarheid nodig is. Vanwege deze nadelen is het vaak beter om full back-ups aan te vullen met andere methoden.

Stel bijvoorbeeld dat een groot bedrijf tijdens kantooruren een full back-up wil maken van alle gegevens. Omdat deze back-up meerdere uren in beslag kan nemen, worden de systemen gedurende die tijd zwaar belast. Dit kan ertoe leiden dat andere processen vertraging oplopen of dat de server tijdelijk minder goed be-

schikbaar is voor werknemers die ook van die systemen afhankelijk zijn voor hun dagelijkse taken (Nelson & Brown, 2011).

Incrementele back-up



Figuur 2.2: Representatie van een incremental back-up (Rivas, 2022)

Een incrementele back-upstrategie houdt in dat na een initiële full back-up slechts de gegevens worden opgeslagen die sinds de laatste back-up zijn gewijzigd (Zhao e.a., 2024). Dit betekent dat een incrementele back-up alleen de veranderingen in de bestanden opneemt, in plaats van telkens een volledige kopie te maken van alle gegevens.

Dit is vooral handig voor bedrijven die relatief vaak back-ups moeten maken, maar de opslag- en tijdskosten van een full back-up willen vermijden. Bijvoorbeeld, stel dat een bedrijf op maandag een full back-up uitvoert met al hun gegevens. Op dinsdag doet het bedrijf een incrementele back-up, waarbij enkel de wijzigingen sinds maandag worden opgeslagen. Dit gaat elke dag van de week zo verder, elke dag wordt enkel de nieuwe of gewijzigde data opgeslagen ten opzichte van de dag ervoor. De volgende week doet het bedrijf op maandag weer een full back-up en herhaalt het de stappen. Omdat bedrijven steeds meer data beheren, biedt deze methode een efficiënte manier om opslagkosten te beperken, vooral wanneer gebruik wordt gemaakt van een cloudservice.

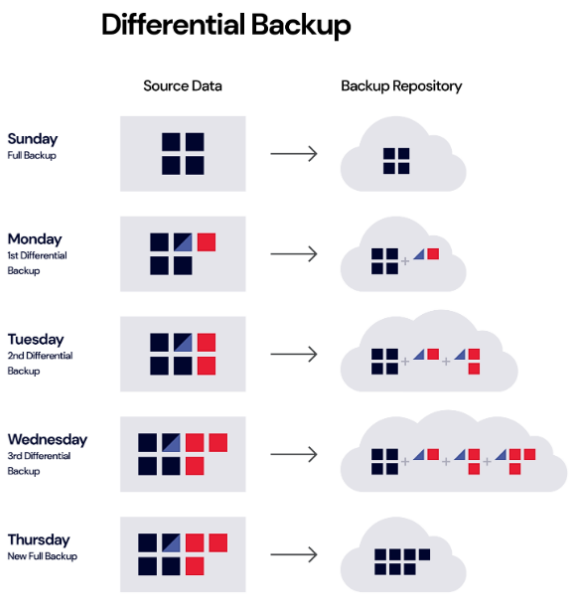
Stel dat een bedrijf dagelijks slechts 1% van zijn gegevens wijzigt; in plaats van elke dag een volledige kopie van bijvoorbeeld 1 TB te maken, slaat een incrementele back-up slechts de nieuwe 1% op, wat 990 GB aan opslagruimte per dag bespaart. Dit maakt incrementele back-ups heel aantrekkelijk voor bedrijven die grote hoeveelheden data verwerken en frequente back-ups willen uitvoeren.

Naast de besparing op opslagcapaciteit, zorgen incrementele back-ups voor

kortere back-uptijden omdat alleen de gewijzigde bestanden worden opgeslagen. Dit betekent dat bedrijven vaker back-ups kunnen uitvoeren zonder hun systemen te vertragen. Een mediabedrijf dat met grote bestanden werkt, kan hierdoor bijvoorbeeld elk uur een incrementele back-up maken, in plaats van dagelijks een volledige back-up. Dit minimaliseert het risico op dataverlies, omdat in het geval van een storing, slechts maximaal een uur aan data verloren gaat in plaats van een hele dag. Hoewel incrementele back-ups voordelen bieden op het gebied van opslag en back-uptijden, brengen ze ook nadelen met zich mee, zoals langere hersteltijden (Chervenak e.a., 1998). Om een systeem te herstellen, heb je de laatste volledige back-up en alle volgende incrementele back-ups nodig en dit kan veel tijd kosten.

Een ander nadeel is de complexiteit van het beheer. Elke incrementele back-up hangt af van de vorige, wat betekent dat een fout in één back-up de hele herstelketen kan verstoren. Een IT-bedrijf dat dagelijks incrementele back-ups maakt, kan bijvoorbeeld problemen ondervinden als de back-up van woensdag beschadigd blijkt te zijn. Alle latere back-ups zijn afhankelijk van die ene back-up, wat het herstelproces moeilijker maakt. Dit vraagt om extra monitoring en beheer, zodat eventuele beschadigingen of herstelproblemen tijdig kunnen worden opgemerkt en opgelost.

Differentiële back-ups



Figuur 2.3: Representatie van een differentiële back-up (Rivas, 2022)

Een differentiële back-up is een soort back-up waarbij enkel de data die sinds de laatste full back-up is veranderd of toegevoegd, wordt gekopieerd. In tegenstelling tot een incrementele back-up, die enkel de veranderingen sinds de laatste back-up opslaat, wordt er bij een differentiële back-up enkel de wijzigingen opge-

slagen sinds de laatste full back-up (Zhu e.a., 2015). Een differentiële back-up zal dus elke keer groter en groter worden naarmate er meer wijzigingen zijn omdat elke wijziging sinds de full back-up opgeslagen wordt.

Een eerste voordeel van deze soort back-up is dat er in het geval van een recovery slechts twee back-ups nodig zijn: de laatste full back-up en de meest recente differentiële back-up. Wanneer hersteltijden belangrijk zijn zullen differentiële back-ups dus handig zijn. Bijvoorbeeld, een organisatie die dagelijks een differentiële back-up uitvoert, heeft na een week slechts de volledige back-up van de eerste dag en de laatste differentiële back-up nodig om alles te herstellen. Dit zorgt voor een relatief eenvoudig en snel herstelproces.

Incrementele back-ups daarentegen slaan alleen de veranderingen op die sinds de laatste back-up zijn gemaakt van eender welke soort, of het nu een volledige of incrementele back-up is. Hierdoor zijn incrementele back-ups meestal kleiner en sneller uit te voeren dan differentiële back-ups, omdat ze alleen de allerlaatste wijzigingen bevatten.

Een eerder besproken nadeel is echter dat bij herstel alle opeenvolgende back-ups nodig zijn om de data volledig terug te zetten: de laatste volledige back-up en alle incrementele back-ups tot de meest recente back-up. Dit maakt incrementele back-ups soms trager en complexer bij recovery, omdat elk back-upbestand moet worden doorlopen.

Een voorbeeld om het verschil tussen incrementele back-ups en differentiële back-ups duidelijk te maken: stel dat een bedrijf aan het begin van de week een volledige back-up maakt. Bij het gebruik van een differentieel back-upschema zou elke back-up in de loop van de week groter worden, omdat elke back-up alle wijzigingen sinds die eerste dag bevat. Bij een incrementeel schema daarentegen blijft elke dagelijkse back-up klein, omdat elke nieuwe back-up alleen de nieuwste wijzigingen bevat. Als het systeem aan het einde van de week moet worden hersteld, zou met een differentieel schema enkel de full back-up en de laatste differentiële back-up nodig zijn. Bij het gebruik van incrementele back-ups zijn alle back-ups van de week vereist (Beard, 2018).

Cloud back-ups

Cloud back-ups zijn een populaire methode waarbij data op externe servers wordt opgeslagen, beheerd door een derde partij. In plaats van lokale fysieke opslagapparaten te gebruiken, worden de gegevens overgebracht naar een cloud-omgeving, zoals die van Amazon Web Services, Microsoft Azure of Google Cloud. Cloud back-ups bieden verschillende voordelen, zoals schaalbaarheid, eenvoud in beheer en de mogelijkheid om gegevens veilig op afstand op te slaan (Rahumed e.a., 2011). Bedrijven hoeven hierdoor geen geld te investeren in fysieke hardware.

Stel dat een bedrijf snel groeit of opeens veel meer data heeft, dan kan het makkelijk zijn om de cloud-opslag uit te breiden zonder de IT-infrastructuur aan te passen wat veel geld en moeite zou kosten. Een van de belangrijkste voorde-

len van cloud back-ups is toegankelijkheid. Aangezien de gegevens zich op een externe server bevinden, kan een bedrijf op elk moment en vanaf elke locatie toegang krijgen tot zijn data, zolang er een internetverbinding is. Dit is vooral handig voor bedrijven die meerdere fysieke locaties hebben.

Stel dat een middelgroot marketingbureau zijn klantgegevens in de cloud opslaat; de back-ups zijn dan beschermd tegen onvoorziene omstandigheden, zoals fysieke schade aan hun eigen kantoren. Echter, cloud back-ups hebben ook nadelen, waaronder de afhankelijkheid van een stabiele internetverbinding. Omdat cloud back-ups vereisen dat data over het internet wordt verzonden, kunnen problemen met de internetverbinding de back-uptijd vertragen of de overdracht volledig onderbreken. Voor een organisatie die bijvoorbeeld grote hoeveelheden videobestanden moet opslaan, kan dit tijdsverlies betekenen, vooral wanneer zij gevestigd zijn op een locatie met beperkte bandbreedte. Dit kan een probleem vormen wanneer er een strikte back-upfrequentie vereist is. Een ander nadeel is de kostprijs, vooral wanneer grote hoeveelheden gegevens vaak worden geüpdatet en opgeslagen (Obrutsky, 2016).

Cloud-providers baseren hun prijs meestal op de hoeveelheid opslagruimte die gebruikt wordt, het dataverkeer en extra functies zoals betere encryptie of de frequentie van de back-ups. Voor een bedrijf dat veel wijzigingen aanbrengt in grote databases kunnen de maandelijkse kosten aanzienlijk oplopen. Dit maakt het noodzakelijk om een goede keuze te maken over de frequentie van back-ups om de kosten beheersbaar te houden. Tot slot biedt de cloud niet altijd dezelfde mate van controle als on-premise oplossingen. Hoewel cloudproviders doorgaans een goede service garanderen, blijft het bedrijf afhankelijk van de beschikbaarheid en het onderhoudsbeleid van de provider.

Dit betekent dat, in het geval van een storing bij de cloudprovider, bedrijven geen directe toegang hebben tot hun eigen back-ups. Dit benadrukt het belang van goede service level agreements (SLA's) en mogelijk zelfs een hybride strategie die cloud-opslag combineert met een bepaalde vorm van lokale back-ups om het risico te spreiden.

On-premise back-ups

On-premise back-ups zijn lokale back-ups die op fysieke servers binnen het bedrijf zijn opgeslagen. Het bedrijf is dus zelf verantwoordelijk voor het beheer, de beveiliging en het onderhoud van de back-upomgeving, zoals Trovato et al. (2019) uitleggen. Dit biedt bedrijven vrijheid en flexibiliteit, maar vereist wel een hoger niveau van technische kennis en onderhoud.

On-premise back-ups bieden volledige controle over de gegevens, wat vooral belangrijk is in sectoren waar veiligheid en privacy cruciaal zijn, zoals bijvoorbeeld de gezondheidszorg en financiële sector. Een groot voordeel is dat er geen nood is aan het internet, waardoor de snelheid van het back-upproces afhangt van de hardware dat het bedrijf bezit. Dit is ideaal voor bedrijven die snel grote hoeveel-

heden data moeten opslaan. Toch hebben on-premise back-ups ook nadelen. Ze vereisen een hoge initiële investering in hardware en onderhoud, zoals servers en netwerkinfrastructuur.

Daarnaast zijn ze kwetsbaar voor fysieke risico's zoals brand, diefstal of natuurrampen, wat vraagt om extra beveiligingsmaatregelen, zoals off-site back-ups. Daarbij is er ook een IT-expert nodig om de back-upsystemen op te zetten.

2.0.3. Ransomware

Ransomware is een groeiende dreiging dat ervoor kan zorgen dat bedrijven hun gegevens voor een bepaalde tijd kwijt zijn of in het slechtste geval voor altijd kwijt zijn. Daarom moeten bedrijven zich sterk inzetten op het implementeren van een sterke back-upstrategie. Back-ups zijn het laatste redmiddel tegen ransomware-aanvallen, omdat ze een veilige kopie van data kunnen herstellen zonder in te gaan op de eisen van de aanvallers.

Ransomware is een type malware dat data vergrendelt of de toegang tot gegevens blokkeert door middel van privé-sleutel encryptie, totdat er losgeld wordt betaald, meestal in Bitcoin (Richardson & North, 2017). Malware is een software-programma dat opzettelijk voldoet aan de schadelijke bedoelingen van kwaadwillende aanvallers (Yanfang e.a., 2017). Deze aanvallen kunnen niet alleen bestanden versleutelen, maar soms ook volledige systemen blokkeren, waardoor de toegang tot cruciale data verloren gaat. De gevolgen zijn vaak ernstig, omdat slachtoffers pas weer controle krijgen als ze aan de eisen van de aanvallers voldoen.

Het betalen van de criminelen biedt echter geen garantie voor de toegang van de data en dit kan eindigen in een eindeloze cirkel waarbij de aanvaller elke keer opnieuw geld vraagt.

Evolutie

De evolutie van ransomware laat een constante groei zien sinds het einde van de jaren '80. In 1989 verscheen het eerste ransomwarevirus, de AIDS Trojan, die eenvoudige versleuteling gebruikte. In 2005 kwam de moderne ransomware met Trojan. Vanaf 2006 nam de populariteit van ransomware toe, met varianten zoals Trojan.Cryzip en Trojan.Archiveus. Rond 2011 begon ransomware wereldwijd uit te breiden dankzij anonieme betalingsdiensten. In 2013 werd CryptoLocker gelanceerd, een beruchte ransomware die complexe encryptie gebruikte en grote sommen losgeld eiste. Dit leidde tot een explosieve groei in ransomware-aanvallen en verfijnde technieken. Tegen 2016 bereikte ransomware een piek, waarbij het zich richtte op meerdere platformen, waaronder Linux en MacOS, en geavanceerdere strategieën gebruikte om detectie te vermijden en meer schade aan te richten.

Een belangrijk aspect van deze evolutie was de introductie van Bitcoin als betaalmethode voor losgeld. De anonimiteit van Bitcoin-transacties maakte het moeilijker voor autoriteiten om aanvallers te traceren. De inzet van cryptocurrencies zoals Bitcoin blijft een cruciaal onderdeel in de succesvolle verspreiding van

moderne ransomware (Richardson & North, 2017).

Impact van ransomware op organisaties

Ransomware-aanvallen hebben een aanzienlijke impact op organisaties. Ten eerste is er de financiële schade, die kan oplopen door gegevensverlies, dure herstelprocessen en de mogelijke betaling van losgeld. Naast directe kosten kunnen bedrijven ook te maken krijgen met verloren klantenvertrouwen en juridische gevolgen, wat de financiële impact verder vergroot.

Ten tweede zorgen ransomware-aanvallen voor operationele verstoringen: systemen worden vaak volledig vergrendeld, wat leidt tot stilstand van cruciale bedrijfsprocessen en verlies van productieve tijd. Deze verstoringen kunnen ernstige gevolgen hebben, vooral in sectoren waar tijdige toegang tot gegevens essentieel is.

2.0.4. Ransomware-resistente back-upoplossingen

Immutable storage

Immutable storage is een techniek waarbij opgeslagen gegevens na het opslaan niet kunnen worden gewijzigd of verwijderd gedurende een vooraf vastgelegde periode. Dit zorgt voor een sterke bescherming tegen ransomware-aanvallen omdat de opgeslagen data niet meer kan worden aangepast (Wahl, 2023). Het concept van immutable storage komt vooral van pas bij organisaties die te maken hebben met zeer gevoelige gegevens en die moeten kunnen garanderen dat hun data altijd veilig en betrouwbaar blijft.

Een belangrijk nadeel van immutable opslag heeft te maken met data throughput. Volgens Miao et al. (2016) verwijst data throughput naar de snelheid waarmee data kan worden overgedragen of verwerkt binnen een bepaald tijdsperiode.

Immutable opslag kan trager zijn bij het schrijven van gegevens omdat de immutability extra processen vereist om ervoor te zorgen dat data niet kan worden aangepast. Dit kan de snelheid van gegevensoverdracht vertragen, vooral bij systemen die software-gebaseerde immutability gebruiken, waar een extra laag van computationele controle nodig is.

Ten derde zorgt het implementeren van immutable storage vaak voor een verhoogde management overhead. Hoe meer opgeslagen data er is, hoe complexer het is om dit te beheren. Administrators moeten hierdoor meer tijd en middelen besteden aan het onderhouden van een efficiënt en veilig opslagbeheer.

Ten vierde kan beveiliging ook een aandachtspunt zijn in het geval dat de er met fysieke opslagapparaten gewerkt wordt. Bij immutability die via software is geïmplementeerd kan er sprake zijn van gevaar indien het hele besturingssysteem is aangevallen.

Ten slotte kunnen de kosten snel oplopen. De initiële investering in immutable opslag kan hoog zijn, vooral als er gekozen wordt voor dure opslagmedia of gespecialiseerde hardware. Naarmate de hoeveelheid gegevens toeneemt, nemen

ook de kosten voor opslag en onderhoud toe (Hasan e.a., 2005).

Air-gapped storage

Air-gapped back-ups bieden sterke bescherming tegen ransomware door back-ups fysiek of virtueel te isoleren van het netwerk. Dit betekent dat, zelfs als het netwerk wordt aangevallen, de back-ups veilig blijven omdat ze niet verbonden zijn met de geïnfecteerde systemen. Deze back-ups worden vaak opgeslagen op externe media zoals harde schijven (Bryant, 2015).

Air-gapping zorgt ervoor dat het onmogelijk is voor ransomware om de back-ups te infecteren, waardoor een bedrijf snel kan herstellen van een aanval en de bedrijfscontinuïteit kan behouden. Het maakt bedrijven minder afhankelijk van cloud-opslag en netwerkverbindingen, wat de risico's vermindert. Hoewel air-gapped back-ups een goede bescherming bieden tegen ransomware, kunnen ze minder snel toegankelijk zijn wanneer gegevensherstel nodig is.

Deze back-ups moeten namelijk fysiek worden opgehaald en aangesloten, wat veel tijd kan kosten. Desondanks bieden air-gapped back-ups een extra laag van beveiliging die belangrijk is voor organisaties die gevoelig zijn voor ransomware-aanvallen (Park e.a., 2023).

Offline back-ups

Edwards (2022) beschrijft offline back-ups als back-ups die worden opgeslagen op externe media, zoals harde schijven, die na het back-uppen van het netwerk worden losgekoppeld. Dit maakt ze immuun voor online bedreigingen zoals ransomware, in tegenstelling tot on-premise back-ups die meestal verbonden blijven met het netwerk.

Het grootste voordeel van offline back-ups is de extra beveiliging tegen cyberaanvallen, aangezien ze fysiek losgekoppeld zijn en daardoor buiten bereik van hackers blijven. Dit biedt bedrijven met gevoelige gegevens een betrouwbare manier om data te beschermen tegen digitale bedreigingen. Een ander voordeel is de fysieke controle over de opslaglocatie, waardoor bedrijven precies kunnen bepalen wie toegang heeft tot de gegevens.

Toch hebben offline back-ups ook nadelen: ze moeten handmatig worden bijgewerkt, wat tijdrovend is, en zijn kwetsbaar voor fysieke schade zoals brand of diefstal. Daarnaast kan het herstelproces langer duren, omdat de gegevens fysiek aangesloten en overgezet moeten worden, wat minder efficiënt is voor bedrijven die snel dataherstel nodig hebben (James, 2019).

Het verschil tussen air-gapped en offline back-ups ligt in de mate van isolatie van het netwerk. Air-gapped back-ups zijn volledig fysiek gescheiden van netwerken en kunnen niet op afstand worden benaderd, wat ze zeer veilig maakt tegen cyberaanvallen. Offline back-ups zijn ook niet constant verbonden, maar kunnen tijdelijk worden aangesloten voor het maken of herstellen van back-ups. Air-gapped back-ups bieden doorgaans een sterkere bescherming, omdat ze volledig geïso-

leerd zijn van potentiële aanvallen.

2.0.5. Technologische basis voor de Proof-of-Concept

Voor de Proof-of-Concept wordt gebruik gemaakt van verschillende technologische tools en platforms. Deze worden ingezet om de geplande back-upstrategie en de beveiligingsmaatregelen te testen en te optimaliseren. Hierbij wordt specifiek gewerkt met tools zoals Azure, VirtualBox en Vagrant. Deze technologieën worden gekozen vanwege hun flexibiliteit, schaalbaarheid en ondersteuning bij het simuleren van realistische scenario's.

Azure

Azure wordt gezien als het openbare cloudplatform van Microsoft en maakt gebruik van virtualisatietechnologie (Ekuan e.a., 2023). Door middel van virtualisatietechnologieën, ook wel hypervisors genoemd (Een hypervisor is software waarmee meerdere virtuele machines (VM's), elk met hun eigen besturingssysteem (OS), op één fysieke server kunnen draaien (Susnjara & Smalley, 2024).), is het mogelijk voor Azure om hardware na te bootsen in software. Dit gebeurt in datacenters die zijn opgebouwd uit serverrekken met onder andere netwerkswitches en voldoende stroomvoorzieningen.

Binnen een Azure-datacenter bevinden zich serverrekken, die elk uit meerdere serverblades bestaan. Deze serverrekken bevatten ook netwerkhardware, zoals netwerkswitches, en een PDU (Power Distribution Unit), die stroomvoorziening biedt. Voor extra schaalbaarheid en efficiëntie worden deze serverrekken vaak gegroepeerd in clusters.

Servers met speciale software, zoals infrastructuurcontrollers, zorgen ervoor dat services efficiënt worden toegewezen en storingen worden opgelost. Azure is meer dan alleen een verzameling servers. Het is een complex netwerk van toepassingen die samenwerken om gevirtualiseerde hardware en software te configureren en beheren. Dit maakt Azure een krachtig en flexibel platform voor gebruikers.

Azure Blob Storage is de objectopslagoplossing van Microsoft voor de cloud, geoptimaliseerd voor het opslaan van grote hoeveelheden ongestructureerde data (Dubey e.a., 2023). Azure Blob Storage biedt immutable storage in een WORM-status (Write Once, Read Many), waarmee data niet kan worden aangepast of verwijderd gedurende een ingestelde periode. Er zijn twee immutability policies beschikbaar:

- **Tijdgebonden retentiebeleid:** Data blijft gedurende een specifieke periode onveranderlijk. Na afloop kunnen bestanden worden verwijderd, maar niet overschreven. Dit beleid kan op account-, container- of versieniveau worden toegepast en kan van "unlocked" naar "gelocked" worden gezet. Eenmaal gelocked, kan de retentieperiode alleen worden verlengd.
- **Legal holds:** Houdt data onveranderlijk tot de hold expliciet wordt opgehe-

ven. Dit is nuttig bij onbepaalde bewaartermijnen, zoals juridische onderzoeken, en kan worden toegepast op container- of blobversieniveau (Estabrook e.a., 2024).

Azure ondersteunt immutability op twee niveaus: container-level, waarbij alle blobs in een container hetzelfde beleid volgen, en version-level, dat flexibiliteit biedt voor individuele blobs met verschillende retentievereisten. Een blob (Binary Large Object) is een type dataopslag dat gebruikt wordt om grote hoeveelheden ongestructureerde gegevens op te slaan, zoals tekst, afbeeldingen, video's, audio of binaire bestanden (Kemp, 2007). Samen zorgen deze opties voor veilige en conforme opslag.

Vagrant

Vagrant is een open-source tool ontwikkeld door HashiCorp die het proces van het beheren en configureren van virtuele machines automatiseert (Hashicorp, z.d.). HashiCorp is een bedrijf dat tools maakt voor infrastructuurbeheer.

Het zorgt ervoor dat gebruikers virtuele machines snel kunnen creëren en configureren door gebruik te maken van gestandaardiseerde configuratiebestanden, genaamd Vagrantfiles. In de Vagrantfiles kun je allerlei configuraties kiezen zoals netwerkconfiguraties, besturingssystemen en softwarepakketten.

Het biedt ondersteuning voor verschillende virtualisatieplatforms, zoals VirtualBox, VMware en Hyper-V en kan worden geïntegreerd met provisioning-tools zoals Ansible. Vagrant wordt vaak gebruikt voor het opzetten van test- en ontwikkelomgevingen.

Virtualbox

VirtualBox is een open-source virtualisatiesoftware die ervoor zorgt dat gebruikers meerdere besturingssystemen tegelijkertijd op één fysieke machine kunnen draaien (Oracle, 2024). Virtualbox biedt veel functionaliteiten aan, waaronder ondersteuning voor diverse gastbesturingssystemen zoals Windows en Linux, daarnaast zijn er ook geavanceerde netwerkmogelijkheden beschikbaar. VirtualBox maakt gebruik van virtuele netwerken, zoals NAT (Network Address Translation) en interne netwerken, waarmee gebruikers flexibele en gescheiden infrastructuren kunnen opzetten.

Dankzij de grafische interface en command-line tools is het een toegankelijk platform voor zowel beginners als gevorderde IT-professionals.

BorgBackup

BorgBackup is een krachtige back-uptool die efficiëntie en beveiliging combineert door gebruik te maken van compressie en geverifieerde encryptie (BorgBackup, 2024).

Het maakt gebruik van deduplicatie, waarbij alleen nieuwe of gewijzigde data worden opgeslagen, dit zorgt ervoor dat er veel opslagruimte bespaart kan worden.

Deze tool maakt gebruik van client-side encryptie met AES-256 en HMAC-SHA256.

Daarnaast heeft BorgBackup verschillende compressieopties zoals LZ4, Zstd en LZMA. Hierdoor kunnen gebruikers kiezen tussen snelheid en compressieniveau.

Als laatste is BorgBackup geschikt voor offsite back-ups via SSH, waardoor het ideaal is voor zowel lokale als externe back-upoplossingen.

MySQL

MySQL is een populair open-source relationeel databasesysteem gemaakt om data op te slaan en te beheren (Erickson, 2024). Het staat bekend om zijn snelheid, betrouwbaarheid en gebruiksvriendelijkheid.

MySQL wordt veel gebruikt voor het opslaan en beheren van gestructureerde data in zowel kleine toepassingen als grote, complexe systemen. MySQL ondersteunt SQL (Structured Query Language) voor het beheren van data en biedt functies zoals transacties, opslag op basis van verschillende database-engines en ondersteuning voor grote datasets. Een opslag-engine (of database-engine) is het onderdeel van een databasesysteem dat bepaalt hoe data fysiek wordt opgeslagen, beheerd, en opgehaald in een database. De database-engine is verantwoordelijk voor het uitvoeren van operaties zoals invoegen, bijwerken, verwijderen en opvragen van data (Cabral & Murphy, 2011).

Voor het maken van back-ups biedt MySQL tools zoals mysqldump, een utility in de CLI waarmee gebruikers eenvoudig een logische back-up (een logische back-up is een soort back-up die de tabelstructuur en gegevens reproduceert, zonder de daadwerkelijke gegevensbestanden te kopiëren (MySQL, z.d.)) kunnen maken van de structuur en data van een database naar een SQL-bestand. Dit bestand kan worden gebruikt om de database later te herstellen in geval van nood.

Bij het gebruik van een opslag-engine die transacties ondersteunt wordt het mogelijk om consistente back-ups te maken, zelfs als de database live in gebruik is. Met de `–single-transaction-optie` in mysqldump kan een momentopname (snapshot) van de database worden gemaakt op een specifiek tijdstip. Dit voorkomt problemen waarbij updates aan meerdere tabellen niet synchroon in de back-up worden opgenomen. Hierdoor bevat de back-up altijd de gegevens zoals die waren op het moment dat het proces begon, zelfs als er daarna wijzigingen plaatsvinden.

PostgreSQL

PostgreSQL is een krachtig open-source object-relationeel databasesysteem dat ontworpen is om data veilig op te slaan en complexe gegevensstructuren efficiënt te beheren (Drake & Worsley, 2002). Het staat bekend vanwege de betrouwbaarheid, schaalbaarheid en uitbreidbaarheid, en wordt vaak gebruikt in zowel kleine toepassingen als grootschalige systemen.

PostgreSQL biedt ondersteuning voor SQL (Structured Query Language) en

voegt hierop een breed assortiment aan extra functies toe, zoals geavanceerde datatypes, transacties en ondersteuning voor JSON.

Voor het maken van back-ups heeft PostgreSQL een tool genaamd pg_dump, dit is een command-line tool waarmee zowel de data als de structuur van een database kan worden opgeslagen in een dumpbestand (Drake & Worsley, 2002).

Bij het gebruik van de –no-lock optie bij het uitvoeren van het pg_dump commando kan een momentopname (snapshot) van de database worden gemaakt terwijl deze actief in gebruik is. Deze optie bestaat om een consistent beeld van de data op het moment van de back-up te bieden, zelfs als er tegelijkertijd wijzigingen plaatsvinden. Hierdoor blijft de back-up betrouwbaar en bevat deze altijd een consistente weergave van de database zoals deze was op het moment dat het proces begon.

Kubernetes

Kubernetes is een open-source container-orkestratiesysteem dat containergebaseerde applicaties beheert over meerdere hosts. Het wordt gebruikt voor het implementeren, monitoren en schalen van containers (Gupta & Hohn, 2019).

Een belangrijk concept binnen Kubernetes zijn pods, de kleinste uitvoerbare eenheden die één of meerdere containers bevatten. Containers binnen een pod delen dezelfde netwerkruimte en opslag, wat samenwerking en communicatie tussen de containers vergemakkelijkt. Pods vormen de basis voor het draaien van applicaties binnen Kubernetes.

Voor geautomatiseerde en geplande taken biedt Kubernetes CronJobs. Hiermee kunnen periodieke processen worden uitgevoerd, zoals het maken van backups, het opschonen van oude gegevens of het draaien van onderhoudstaken. CronJobs gebruiken een tijdsschema, vergelijkbaar met Linux-cron, om taken automatisch uit te voeren op specifieke momenten.

Om gevoelige gegevens, zoals wachtwoorden of API-sleutels, veilig te beheren, gebruikt Kubernetes Secrets. Secrets stellen gebruikers in staat om vertrouwelijke informatie op een veilige manier beschikbaar te maken voor pods zonder deze expliciet in configuratiebestanden op te nemen.

3

Methodologie

Het onderzoek begint met een uitgebreide literatuurstudie over back-upstrategieën, ransomware-resistente opslag, en immutable storage. Hierbij wordt een overzicht gegeven van de state of the art, waarbij de nieuwste technieken en strategieën voor databeveiliging in kaart worden gebracht. Deze literatuurstudie biedt de fundamentele kennis die nodig is om het bestaande back-upplan te analyseren en geeft een goed beeld van hoe organisaties effectief hun back-upsystemen kunnen beveiligen. In de tweede fase zal de huidige back-upstrategie van Forvis Mazars worden geanalyseerd en verbeterd. Momenteel wordt er elke dag één volledige back-up door Azure automatisch uitgevoerd, wat zorgt voor een basisbeveiliging. Door de bestaande methode te optimaliseren, wordt zowel de veiligheid als de efficiëntie van het back-upproces verhoogd.

3.0.1. Requirements-analyse

1. Must Have (Essentiële vereisten)

Deze vereisten zijn cruciaal voor de verbetering van de back-upstrategie en moeten absoluut worden geïmplementeerd om een werkbare en veilige oplossing te garanderen:

- **Regelmatige en betrouwbare automatische back-ups:** Er moet gezorgd worden voor de implementatie van een automatische back-upstrategie die dagelijks volledige back-ups van de databases uitvoert. Dit moet kunnen worden geïmplementeerd in de bestaande Azure-omgeving van Forvis Mazars met minimale aanpassingen, zodat er altijd een up-to-date herstelpunt beschikbaar is bij systeemfouten of cyberaanvallen.
- **Beveiliging tegen ransomware:** De vernieuwde back-upstrategie moet bestand zijn tegen ransomware, wat betekent dat back-ups moeten worden beschermd tegen externe aanvallen die de back-ups zelf kunnen infecteren. Dit

vereist de implementatie van technieken zoals immutable storage, zodat back-ups niet gewijzigd of verwijderd kunnen worden tijdens een ransomware-aanval.

- **Versiebeheer van back-ups:** Het invoeren van versiebeheer voor back-ups maakt het mogelijk om verschillende versies van data op te slaan. Dit kan nuttig zijn voor het herstellen van data naar een specifieke eerdere versie (bijvoorbeeld na een fout die niet meteen werd opgemerkt).
- **Herstelcapaciteit (Restore from backup):** Het herstelproces moet efficiënt en snel kunnen worden uitgevoerd vanuit de back-ups. De back-upstrategie moet testen hoe snel en betrouwbaar de systemen kunnen worden hersteld in geval van dataverlies.

2. Should Have (Aanbevolen vereisten)

Deze vereisten dragen bij aan de effectiviteit van de back-upstrategie, maar zijn niet strikt noodzakelijk voor de eerste versie van de oplossing:

- **Differentiële en incrementele back-ups:** Hoewel volledige back-ups cruciaal zijn, moeten incrementele en/of differentiële back-ups overwogen worden om de belasting op de opslagcapaciteit en netwerkinfrastructuur te verminderen. Dit kan bijdragen aan de optimalisatie van de back-upstrategie door slechts gewijzigde gegevens te back-uppen in plaats van de volledige dataset.
- **Documentatie:** Er moet gedetailleerde documentatie beschikbaar zijn over het back-upproces, de gebruikte technieken, en de herstelprocedures.

3. Could Have (Wenselijke vereisten)

Deze vereisten kunnen de back-upstrategie verder verbeteren, maar kunnen in eerste instantie worden uitgesteld als er beperkingen zijn in tijd of middelen:

- **Geautomatiseerd herstelproces:** Een geautomatiseerd herstelproces kan worden ontwikkeld, zodat de systemen automatisch kunnen worden hersteld in geval van dataverlies, wat de downtime minimaliseert.

4. Won’t Have (Niet noodzakelijke vereisten)

Deze vereisten worden niet opgenomen in de huidige verbeteringsronde van de back-upstrategie vanwege beperkingen in tijd, middelen, of prioriteit:

- **Multi-cloud back-upoplossingen:** Hoewel multi-cloud back-upstrategieën voordelen kunnen bieden, is het implementeren van een complexe multi-cloud-oplossing voor Forvis Mazars op dit moment niet noodzakelijk, aangezien Azure al gebruikt wordt voor back-ups en de primaire focus ligt op het verbeteren van de huidige strategie binnen de Azure-omgeving.

- **Fysieke back-ups op externe schijven:** Aangezien Forvis Mazars voornamelijk gebruik maakt van een cloud-gebaseerde infrastructuur voor back-ups, sluit het gebruik van fysieke externe schijven of on-premise hardware-oplossingen momenteel niet aan bij hun threatmodel en operationele behoeften. Hoewel dergelijke oplossingen voordelen kunnen bieden in specifieke scenario's, wegen deze voor Forvis Mazars niet op tegen de extra complexiteit en kosten die ermee gepaard gaan.

Conclusie

Deze requirementsanalyse geeft de noodzakelijke vereisten voor de verbetering van de back-upstrategie van Forvis Mazars weer, met een focus op beveiliging tegen ransomware, automatisering van de back-ups, en het herstelproces. Door de integratie van automatisering, immutable storage en verbeterde back-uptechnieken zal Forvis Mazars in staat zijn om zowel de veiligheid als de efficiëntie van hun gegevensbeheer te verhogen. Verdere verbeteringen, zoals incrementele back-ups en cloud-integratie, kunnen op een later moment worden geïmplementeerd, afhankelijk van de beschikbare middelen en de prioriteiten van het bedrijf.

3.0.2. Proof-Of-Concept

In de Proof-of-Concept (PoC) is een virtuele omgeving opgezet met behulp van VirtualBox en Vagrant om een testomgeving te creëren. Binnen deze omgeving werden drie virtuele machines geconfigureerd: een primaire server die als databaseserver functioneert, een back-upserver voor het opslaan van de back-ups, en een attacker VM voor het simuleren van een ransomware-aanval. Het doel van deze opzet was om aan te tonen hoe het implementeren van immutable storage kan helpen bij het beschermen van belangrijke data tegen ransomware-aanvallen.

Voor de back-upstrategie werd gebruik gemaakt van een combinatie van MySQL en BorgBackup. Op de primaire server werd een testdatabase aangemaakt en deze werd geëxporteerd naar een back-upbestand. Dit back-upbestand werd vervolgens veilig opgeslagen in een Borg-repository op de back-upserver. De back-updirectory werd beschermd met het Linux-commando `chattr`, waarmee de immutability werd geïmplementeerd. Dit attribuut voorkomt dat bestanden in de back-updirectory kunnen worden gewijzigd of verwijderd en simuleert een vorm van immutable storage.

Om een ransomware-aanval na te bootsen, werd een Bash-script gebruikt dat de bestandsnamen van de back-ups aanpaste en de bestanden versleutelde. Dit simuleerde een malware-aanval. Daarnaast werd de actieve database onbruikbaar gemaakt om te demonstreren dat deze niet meer bruikbaar is na een aanval, terwijl de immutabele back-ups intact bleven en als herstelpunt konden dienen.

Nadien werd ook het herstelproces van de back-ups getest. Een back-up werd vanuit de Borg-repository teruggehaald en gebruikt om de oorspronkelijke database succesvol te herstellen. Dit proces demonstreert de integriteit van de back-

ups.

Deze methode biedt een praktische Proof-of-Concept waarmee kan worden aangetoond hoe immutable storage kan bijdragen aan de beveiliging en integriteit van back-ups in een productieomgeving.

4

Analyse van de huidige back-upstrategie bij Forvis Mazars

4.1. Huidige back-upstrategie

De huidige back-upstrategie van Forvis Mazars maakt gebruik van een Azure-omgeving voor het beheer van hun databases. Binnen deze omgeving draaien twee verschillende databaseservers: Azure Database for PostgreSQL flexible server en Azure Database for MySQL flexible server. Deze servers bevatten meerdere databases die door de organisatie worden beheerd. De back-upstrategie van het bedrijf omvat twee verschillende methoden voor het maken van back-ups, namelijk automatische full back-ups door Azure zelf en handmatige back-ups per database, deze manuele back-ups worden niet frequent genoeg uitgevoerd.

De automatische full back-ups worden dagelijks uitgevoerd door de ingebouwde Azure-functie voor databasebeheer. Deze back-ups zijn servergebaseerd, wat betekent dat een volledige back-up van de server wordt genomen. Dit heeft als nadeel dat bij een fout in een specifieke database, zoals Database A, alle andere databases op dezelfde server, zoals Database B en Database C, eveneens moeten worden hersteld. Dit kan onpraktisch zijn, omdat het terugzetten van een database die geen problemen had leidt tot onnodig verlies van gegevens of verstoorde bedrijfsprocessen.

Voor de handmatige back-ups is een Python-script ontwikkeld. Dit script wordt uitgevoerd op een pod binnen het Kubernetes-cluster van de organisatie. Het doel van dit script is om op verzoek een back-up te maken van een specifieke database, hetzij een PostgreSQL- of MySQL-database, en deze op te slaan in een Azure Storage Account. Het voordeel van deze aanpak is dat het script per database werkt

en daarmee gerichter is dan de servergebaseerde automatische back-up. Echter, de handmatige uitvoering van dit script werd niet geautomatiseerd in de huidige omgeving. Dit leidde tot een onbetrouwbare uitvoering, omdat de back-ups afhankelijk waren van de handmatige interventie van medewerkers. Dit zorgt voor incomplete back-ups.

Een ander belangrijk punt is dat de specifieke duur van een herstel uit een back-up onbekend is, aangezien dit proces nog niet is getest. Hoewel er regelmatig back-ups worden gemaakt, is er geen duidelijk inzicht in hoeveel tijd het kost om een database volledig te herstellen na een incident. Dit gebrek aan testervaring met het herstelproces kan problematisch zijn, aangezien de snelheid van herstel cruciaal is voor het minimaliseren van downtime en het waarborgen van de bedrijfscontinuïteit.

De opslag van de gemaakte back-ups vindt plaats in een Azure Storage Account, waarvoor toegang strikt gereguleerd is. Alleen gebruikers met de juiste machtigingen, via een specifiek Service Principal binnen het interne netwerk van het bedrijf, kunnen de back-ups benaderen. Deze benadering biedt een zekere mate van beveiliging, maar de opslag wordt niet beschermd door mechanismen zoals immutable storage, wat een aanzienlijke zwakte is in de context van moderne cyberdreigingen zoals ransomware-aanvallen. Zonder immutable storage kunnen back-ups worden gemanipuleerd of verwijderd.

Daarnaast beschikt Forvis Mazars over een Kubernetes-cluster dat verschillende pods draait, die verschillende taken uitvoeren, waaronder het back-uppen van databases. Door de flexibele aard van Kubernetes kunnen processen zoals het back-uppen eenvoudig geautomatiseerd worden, bijvoorbeeld door middel van CronJobs. Deze mogelijkheid wordt echter niet benut binnen de bestaande back-upstrategie. Het Python-script dat wordt gebruikt voor de handmatige back-ups kan worden aangepast en geautomatiseerd via Kubernetes CronJobs, wat zou zorgen voor een veel efficiëntere en betrouwbaardere back-upstrategie.

De huidige setup van Forvis Mazars biedt een basisinfrastructuur voor de back-ups van de databases, maar heeft belangrijke tekortkomingen op het gebied van automatisering, beveiliging en herstel. De afwezigheid van immutable storage maakt de back-ups kwetsbaar voor moderne cyberdreigingen, terwijl het gebrek aan geautomatiseerde handmatige back-ups de betrouwbaarheid van het systeem vermindert. De bestaande strategie biedt ruimte voor aanzienlijke verbetering door te focussen op meer geavanceerde back-uptechnieken, zoals het implementeren van immutable storage, het verbeteren van de opslagbeveiliging, en het automatiseren van handmatige back-ups via Kubernetes en geoptimaliseerde CronJobs.

4.2. Mogelijke bijkomende verbeteringen voor de back-upstrategie

De volgende tools en verbeteringen zouden nog extra kunnen worden geïmplementeerd:

- **Automatisering**: Het manuele back-upscript kan worden geautomatiseerd met tools zoals Azure Automation of Logic Apps.
- **Incrementele en Differentiële Back-ups**: Het implementeren van incrementele of differentiële back-ups kan opslagkosten verlagen en het herstel versnellen.
- **Back-upretentie**: Een uitgebreider retentiebeleid kan zorgen voor langere bewaartermijnen voor wekelijkse en maandelijkse back-ups.
- **Beveiliging**: Immutable storage kan worden toegepast om back-ups te beschermen tegen ransomware-aanvallen.
- **Hersteltesten**: Periodieke hersteltests kunnen de effectiviteit van de back-ups garanderen.
- **Monitoring en Rapportage**: Azure Monitor of Log Analytics kan helpen bij het monitoren van back-ups en het genereren van rapporten.

5

Proof-of-Concept

Voor het eerste praktische deel van deze bachelorproef werden er drie virtuele machines opgezet binnen VirtualBox met behulp van Vagrant om een gecontroleerde testomgeving te creëren. Deze virtuele machines (VM's) simuleren een scenario waarin een ransomware-aanval gericht wordt op databases die door het bedrijf worden beheerd. Het primaire doel van deze simulatie is aan te tonen dat het gebruik van immutable storage een effectieve maatregel kan zijn om belangrijke data te beschermen tegen ransomware-aanvallen.

5.0.1. Relevantie van de PoC voor de Azure-omgeving van Forvis Mazars

De Proof-of-Concept (PoC) in VirtualBox simuleert een ransomware-aanval in een lokale omgeving, gericht op het evalueren van beveiligingsmaatregelen zoals immutable storage. Deze aanpak sluit nauw aan bij de Azure-omgeving van Forvis Mazars, waar databases en back-ups worden beheerd.

De technieken uit de PoC, zoals immutable storage, kunnen direct worden toegepast in Azure via functies zoals immutable blobs in Azure Storage. Dit maakt het mogelijk om gegevens beter te beschermen tegen wijzigingen of verwijdering. Daarnaast biedt de PoC een veilig platform om de impact van een ransomware-aanval te begrijpen en te testen hoe snel en effectief back-ups kunnen worden hersteld, wat een cruciaal aspect is voor de bedrijfscontinuïteit.

Forvis Mazars kan de PoC gebruiken om risico's te analyseren en beveiligingsoplossingen eerst kleinschalig te testen, alvorens deze op grotere schaal binnen hun cloudinfrastructuur toe te passen. Hiermee helpt de PoC bij het verfijnen en optimaliseren van hun bestaande Azure-back-upstrategie.

5.0.2. Technische uitwerking

Voor het opzetten van de virtuele machines in de Proof-of-Concept (PoC) werd gebruik gemaakt van een Vagrantfile, zie bijlage B, die de specificaties en configuraties van de VM's bepaalt.

In de onderstaande tabel worden de specificaties van de drie virtuele machines weergegeven die in de Proof-of-Concept zijn gebruikt. Elke VM heeft een specifieke functie binnen het netwerk. De tabel bevat details over de hoeveelheid toegewezen RAM, het aantal CPU-cores, het gebruikte besturingssysteem, de toegewezen IP-adressen en de configuratie van de netwerkadapter. Deze configuratie zorgt ervoor dat de VM's binnen hetzelfde interne netwerk met elkaar kunnen communiceren, wat essentieel is voor het testen van de ransomware-aanval en de back-upstrategieën.

Functie	RAM	CPU Cores	IP	Besturingssysteem	Netwerkadapter
Primary server	2 GB	1	192.168.0.10	Ubuntu 22.04.5 LTS	NAT + Internal
Back-up server	1 GB	1	192.168.0.20	Ubuntu 22.04.5 LTS	NAT + Internal
Attacker VM	2 GB	1	192.168.0.30	Ubuntu 22.04.5 LTS	NAT + Internal

Tabel 5.1: Beschrijving van de virtuele machines in de Proof-of-Concept

De Primary VM stelt een actieve databankserver voor binnen een bedrijfsomgeving. Deze server bevat de operationele data en vertegenwoordigt de belangrijkste bron die beschermd moet worden tegen dataverlies of aanvallen.

De Back-up VM fungeert als een back-upserver waarop regelmatig de back-ups worden opgeslagen. Deze back-upserver is cruciaal voor bedrijfscontinuïteit en disaster recovery, omdat ze in geval van een aanval of fout de herstelmogelijkheden biedt.

De Attacker VM vertegenwoordigt een hacker met slechte intenties binnen de testomgeving. Deze machine wordt gebruikt om een ransomware-aanval te simuleren, waarbij de functionaliteit van zowel de Primary VM als de Back-up VM wordt bedreigd.

Het doel van deze opstelling is om te demonstreren hoe immutable storage een bedrijf kan beschermen tegen de gevolgen van een ransomware-aanval.

Aanmaken van de database

Op de primary VM werd een eenvoudige SQL-database geïnstalleerd en de volgende tabel aangemaakt om als testdata te dienen:

Listing 5.1: MySQL-code voor het aanmaken van de testdatabank.

```
1 CREATE TABLE employees (
2     id INT AUTO_INCREMENT PRIMARY KEY,
3     name VARCHAR(50),
4     role VARCHAR(50)
```

```
);
INSERT INTO employees (name, role) VALUES
    ('Alice', 'Engineer'),
    ('Bob', 'Manager'),
    ('Charlie', 'Analyst');
```

Back-up van de database

Nadien werd de database geëxporteerd naar een `.sql`-bestand met het volgende `mysqldump`-commando:

Listing 5.2: Mysqldump commando om een databank te exporteren.

```
mysqldump -u testuser -p testdb > /home/vagrant/backup.sql
```

Het resulterende bestand, backup.sql, werd vervolgens met BorgBackup opgeslagen in een back-uprepository op de back-up VM. De repository werd vooraf geïnitialiseerd met het volgende commando:

Listing 5.3: Borg commando om een map te initialiseren als Borg repository.

```
borg init --encryption=repokey /home/vagrant/backups
```

Vervolgens werd de back-up gemaakt:

Listing 5.4: Borg commando om een back-up te nemen.

```
borg create --progress
ssh://vagrant@192.168.0.20/home/vagrant/backups::backup-$(date +%Y-%m-%d)
/home/vagrant/backup.sql
```

Beveiliging van de back-up directory

Om de back-up directory ransomware-resistent te maken, werd het Linux-commando chattr gebruikt om het immutable-attribuut toe te passen op de back-up directory. Dit attribuut zorgt ervoor dat er geen wijzigingen aan de bestanden in de directory kunnen gebeuren, zelfs door gebruikers met root-rechten. Het commando:

Listing 5.5: Linux commando om de map immutable te maken.

```
sudo chattr +i /home/vagrant/backups/
```

Simulatie van de ransomware-aanval

Op de aanvaller-VM werd een script gebruikt om de ransomware-aanval te simuleren. Het script probeert alle bestanden in de back-up directory te versleutelen met behulp van een versleutelingstechniek (AES-256-CBC) door de inhoud van de bestanden te encrypteren en op te slaan met de extensie .enc. Na het encrypteren worden de oude bestanden verwijderd zodat deze niet meer bereikbaar zijn. Dit simuleert een ransomware-aanval waarbij de back-upbestanden worden versleuteld, waardoor ze niet meer bruikbaar zijn zonder het juiste decryptiesleutel. Zie bijlage C voor de code van het script dat de aanval uitvoert.

Voor het gemak heeft de Attacker VM volledige controle gekregen over de back-up VM en de primary VM. Dit is gedaan omdat de scope van deze bachelorproef niet is om toegang te verkrijgen tot een server, maar eerder om een gecontroleerde omgeving te creëren waarin een Attacker VM een ransomware-aanval nabootst. Het doel is om te demonstreren hoe een ransomware-aanval een back-up directory kan beïnvloeden, en niet om de daadwerkelijke methoden voor het verkrijgen van toegang tot een server in detail uit te werken.

Oorspronkelijk werd het script zo opgezet dat het alleen de bestandsnamen wijzigde door de extensie .malware toe te voegen, wat werd gepresenteerd als een simulatie van een ransomware-aanval. Echter, deze benadering bood geen realistische weergave van een daadwerkelijke aanval, aangezien de inhoud van de bestanden onaangetast bleef en nog steeds toegankelijk was. Vervolgens werd het script aangepast zodat niet alleen de bestandsnamen werden gewijzigd, maar de daadwerkelijke inhoud van de bestanden werd versleuteld met AES-256-CBC encryptie. Hierdoor werden de bestanden daadwerkelijk onleesbaar zonder de juiste decryptiesleutel, wat een authentiekere simulatie van een ransomware-aanval mogelijk maakte.

Naast een poging om de back-up map op de back-up VM aan te vallen is de actieve database op de primary VM ook aangevallen. Dit werd gedaan om een realistische simulatie van een malware-aanval na te bootsen. Eerst werd er een hash berekend van de tabel in de MySQL-shell zodat deze later kan worden vergeleken om de integriteit te testen. Dit werd gedaan met het volgende commando:

Listing 5.6: MySQL commando om de hash van de tabel te berekenen voor de restore.

```
mysql> SELECT SUM(CRC32(CONCAT_WS('#', id, name, role))) AS checksum FROM
    employees;
+------------+
| checksum   |
+------------+
| 6854392278 |
+------------+
1 row in set (0.00 sec)
```

Nadien werd de actieve database verwijderd in een MySQL-shell met het volgende commando:

Listing 5.7: Commando om de actieve MySQL database te droppen.

```
mysql> DROP DATABASE testdb;
```

Voor de uitvoering van de malware-aanval werd een hash van de volledige back-upfolder gegenereerd om de integriteit ervan te controleren. Dit zal ook na het uitvoeren van de aanval gedaan worden om te bewijzen dat de inhoud niet is aangetast. De hash werd berekend door een script dat over elk bestand in de back-up map gaat en de hashes van de bestanden combineert.

Listing 5.8: Bash script om de hash te berekenen van de back-up map.

```
#!/bin/bash

FOLDER_PATH="/home/vagrant/backups"

# Controleer of de folder bestaat
if [ ! -d "$FOLDER_PATH" ]; then
  echo "Error: Folder $FOLDER_PATH does not exist."
  exit 1
fi

# Genereer een hash van alle bestanden in de folder
find "$FOLDER_PATH" -type f -exec sha256sum {} \; | sort | sha256sum
```

Bij het uitvoeren van dit script krijgen we volgende output:

Listing 5.9: Output van het script om de hash te berekenen.

```
vagrant@ubuntu-jammy:~$ ./check.sh
65ac54d466e08669a8a6ec1651f6563684ac386d7a55d07846088bd933af1f7e  -
```

Bij het uitvoeren van het ransomware bash-script zien we dat dit niet mogelijk is en krijgen we volgende output:

Listing 5.10: Output van het bash-script op de immutable map.

```
vagrant@ubuntu-jammy:~$ ./malware.sh
Can't open "/home/vagrant/backups/README.enc" for writing, Operation not permitted
40A7F5F4427F0000:error:80000001:system library:BIO_new_file:Operation not
    permitted:../crypto/bio/bss_file.c:67:calling fopen(/home/vagrant/backups/
    README.enc, wb)
40A7F5F4427F0000:error:10080002:BIO routines:BIO_new_file:system lib:../crypto/bio
    /bss_file.c:77:
Error: Could not encrypt /home/vagrant/backups/README
Can't open "/home/vagrant/backups/config.enc" for writing, Operation not permitted
4057A1DC8D7F0000:error:80000001:system library:BIO_new_file:Operation not
    permitted:../crypto/bio/bss_file.c:67:calling fopen(/home/vagrant/backups/
    config.enc, wb)
4057A1DC8D7F0000:error:10080002:BIO routines:BIO_new_file:system lib:../crypto/bio
    /bss_file.c:77:
Error: Could not encrypt /home/vagrant/backups/config
Can't open "/home/vagrant/backups/hints.5.enc" for writing, Operation not
    permitted
402707FC577F0000:error:80000001:system library:BIO_new_file:Operation not
    permitted:../crypto/bio/bss_file.c:67:calling fopen(/home/vagrant/backups/hints
    .5.enc, wb)
402707FC577F0000:error:10080002:BIO routines:BIO_new_file:system lib:../crypto/bio
    /bss_file.c:77:
Error: Could not encrypt /home/vagrant/backups/hints.5
Can't open "/home/vagrant/backups/index.5.enc" for writing, Operation not
    permitted
40578D98227F0000:error:80000001:system library:BIO_new_file:Operation not
    permitted:../crypto/bio/bss_file.c:67:calling fopen(/home/vagrant/backups/index
```

```
    .5.enc, wb)
40578D98227F0000:error:10080002:BIO routines:BIO_new_file:system lib:../crypto/bio
    /bss_file.c:77:
Error: Could not encrypt /home/vagrant/backups/index.5
Can't open "/home/vagrant/backups/integrity.5.enc" for writing, Operation not
    permitted
40D7008EF47F0000:error:80000001:system library:BIO_new_file:Operation not
    permitted:../crypto/bio/bss_file.c:67:calling fopen(/home/vagrant/backups/
    integrity.5.enc, wb)
40D7008EF47F0000:error:10080002:BIO routines:BIO_new_file:system lib:../crypto/bio
    /bss_file.c:77:
Error: Could not encrypt /home/vagrant/backups/integrity.5
Can't open "/home/vagrant/backups/nonce.enc" for writing, Operation not permitted
40570E33907F0000:error:80000001:system library:BIO_new_file:Operation not
    permitted:../crypto/bio/bss_file.c:67:calling fopen(/home/vagrant/backups/nonce
    .enc, wb)
40570E33907F0000:error:10080002:BIO routines:BIO_new_file:system lib:../crypto/bio
    /bss_file.c:77:
Error: Could not encrypt /home/vagrant/backups/nonce
```

Als we de inhoud van de map bekijken zien we dat er niets is aangepast en de back-ups nog aanwezig zijn. Daarbij is de hash nog steeds hetzelfde na het uitvoeren van de aanval:

Listing 5.11: Output voor het tonen van de inhoud en het berekenen van de hash.

```
vagrant@ubuntu-jammy:~$ ls /home/vagrant/backups/
README  config  data  hints.5  index.5  integrity.5  nonce
vagrant@ubuntu-jammy:~$ ./check.sh
65ac54d466e08669a8a6ec1651f6563684ac386d7a55d07846088bd933af1f7e  -
```

Dit toont aan dat de immutability heeft gewerkt en de back-ups niet zijn aangetast. Om dit verder te testen zal het bash-script ook eens uitgevoerd worden op een map waarbij de immutability niet is geïmplementeerd. Bij het berekenen van de hash van de tweede map voor het uitvoeren van de malware krijgen we volgende output:

Listing 5.12: Output na het berkenen van de hash van de tweede map voor de aanval.

```
vagrant@ubuntu-jammy:~$ ./check.sh
40dfc3b0b216f34e5992a16f3f9b21a0d5eae4c7fcdc5951c28d9d6b4b51590f  -
```

Bij het uitvoeren van de malware op de tweede map zien we dat de bestanden zijn aangetast en krijgen we volgende output:

Listing 5.13: Output van het bash-script op de tweede map zonder immutability

```
vagrant@ubuntu-jammy:~$ ./malware.sh
Encrypted /home/vagrant/testdir/README and saved as /home/vagrant/testdir/README.
    enc
Original file /home/vagrant/testdir/README is deleted.
Encrypted /home/vagrant/testdir/config and saved as /home/vagrant/testdir/config.
    enc
```

```
Original file /home/vagrant/testdir/config is deleted.
Encrypted /home/vagrant/testdir/hints.5 and saved as /home/vagrant/testdir/hints
    .5.enc
Original file /home/vagrant/testdir/hints.5 is deleted.
Encrypted /home/vagrant/testdir/index.5 and saved as /home/vagrant/testdir/index
    .5.enc
Original file /home/vagrant/testdir/index.5 is deleted.
Encrypted /home/vagrant/testdir/integrity.5 and saved as /home/vagrant/testdir/
    integrity.5.enc
Original file /home/vagrant/testdir/integrity.5 is deleted.
Encrypted /home/vagrant/testdir/nonce and saved as /home/vagrant/testdir/nonce.enc
Original file /home/vagrant/testdir/nonce is deleted.
vagrant@ubuntu-jammy:~$ ls /home/vagrant/testdir/
README.enc  config.enc  data  hints.5.enc  index.5.enc  integrity.5.enc  nonce.enc
```

De hash is ook niet meer hetzelfde en bij het berekenen van de hash krijgen we een andere waarde:

Listing 5.14: Output na het berekenen van de hash van de tweede map

```
vagrant@ubuntu-jammy:~$ ./check.sh
491ceb5435661759feac9c1196430f3940f85df06019e086fa37eaaa52b6f5f6  -
```

Toen het script werd uitgevoerd op de back-up map van de back-up VM, werd duidelijk dat het encrypteren van de bestanden niet lukte vanwege het immutable-attribuut. Dit toont aan dat de ransomware-aanval niet slaagde en de bestanden in de back-up directory beschermd bleven.

Herstellen van de back-ups

Om te bewijzen dat de back-ups nog steeds bruikbaar waren, werd een herstelproces uitgevoerd op de primary VM vanuit de Borg-repository:

Listing 5.15: Borg commando om een back-up te herstellen

```
borg extract
ssh://vagrant@192.168.0.20/home/vagrant/backups::backup-2024-12-05
```

De databank werd opnieuw opgezet vanuit het bestand dat uit de Borg-repository werd gehaald. Als eerst werd een nieuwe lege databank aangemaakt in een MySQL-shell door `CREATE DATABASE testdb;` uit te voeren. Nadien werd de database opnieuw opgezet met het volgende commando:

Listing 5.16: MySQL commando om een databank te herstellen vanuit een .sql-bestand

```
mysql -u root -p restored_db < /home/vagrant/backup.sql
```

Om aan te tonen dat het herstelproces succesvol is verlopen is de hash opnieuw berekend in de MySQL-shell en deze kwam overeen met de hash die voor het uitvoeren van de restore werd berekend. Dit werd gedaan met het volgende commando:

Listing 5.17: MySQL commando om de hash van de tabel te berekenen na de restore.

```
mysql> SELECT SUM(CRC32(CONCAT_WS('#', id, name, role))) AS checksum FROM
    employees;
+-------------+
| checksum    |
+-------------+
| 6854392278 |
+-------------+
1 row in set (0.00 sec)
```

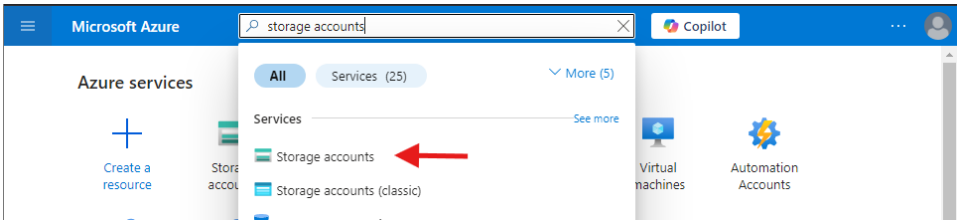
In eerste instantie werd geprobeerd de hashes van de .sql-bestanden vóór en na de hersteloperatie te vergelijken. Echter, de verkregen hashes kwamen niet overeen, dit kwam doordat de tijdstempels, die tijdens het exporteren van de gegevens werden toegevoegd, niet identiek waren. Om deze inconsistentie te omzeilen, werd besloten om de integriteit van de gegevens direct binnen de MySQL-database te controleren.

Het herstelproces verliep succesvol, wat bewijst dat de immutable storage de integriteit van de back-ups had behouden en dat de bestanden veilig waren gebleven ondanks de ransomware-aanval.

5.0.3. Implementatie van immutable storage in de Azure-omgeving

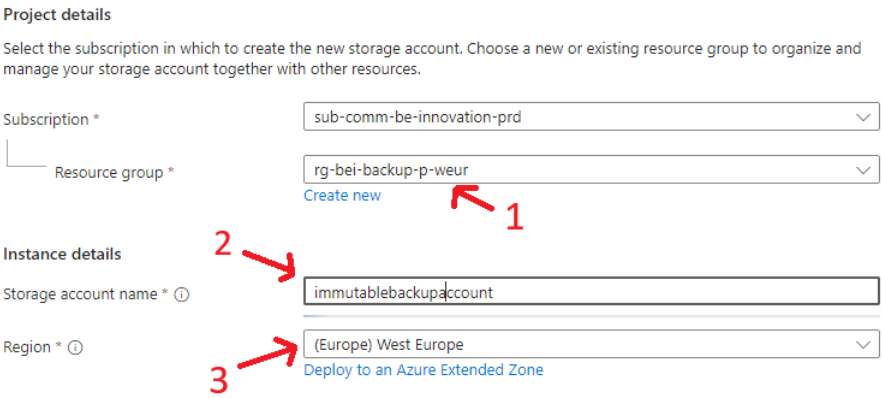
Aanmaken van een storage account

De eerste stap in het implementeren van Immutable Storage in Azure is het aanmaken van een `Storage Account` in de Azure Portal. Bij het aanmaken van het storage

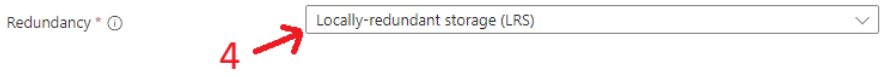


Figuur 5.1: Storage account zoekopdracht binnen de Azure Portal

account kiezen we de correcte resource group, naam die het account moet krijgen, regio en bij redundancy kiezen we voor `Locally-redundant storage (LRS)`. Bij de optie `Account kind` kiezen we voor `General-purpose v2`, omdat deze versie alle benodigde functionaliteit biedt, zoals het ondersteunen van de blob storage en het configureren van immutability policies.



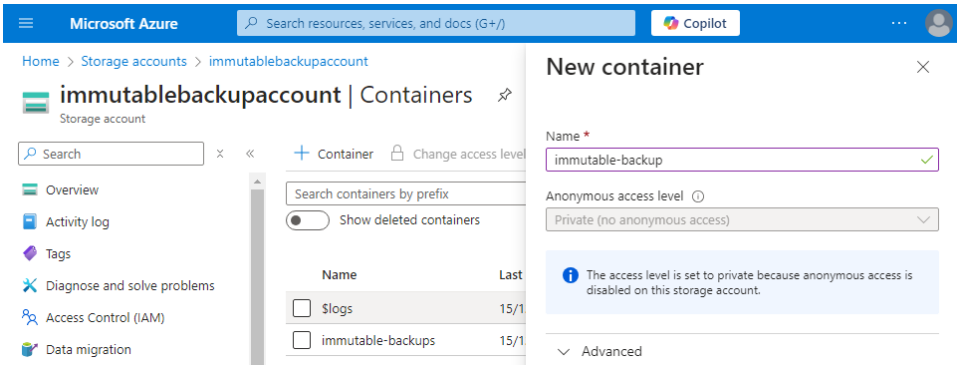
Figuur 5.2: Configuratie voor het nieuwe storage account



Figuur 5.3: Tweede deel van de configuratie voor het nieuwe storage account

Aanmaken van een container binnen het storage account

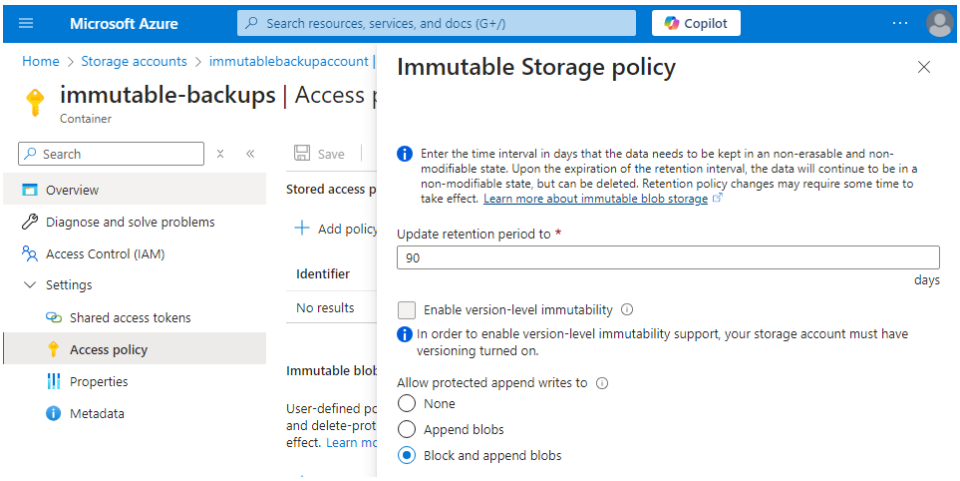
Na het aanmaken van het storage account moet er een container geconfigureerd worden binnen het nieuwe storage account om de gegevens op te slaan. Bij het aanmaken van de container moet de `public access level` op private staan voor de veiligheid. Containers in Azure werken als mappen waarin je blobs kunt opslaan zoals back-ups.



Figuur 5.4: Configuratie voor de container in het storage account

Opzetten van een time-based retention policy

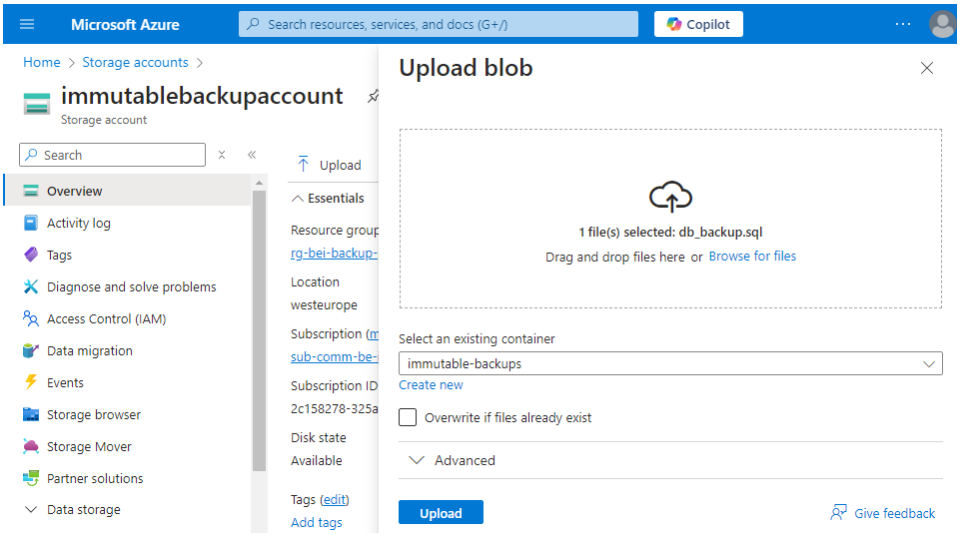
Nadien moet er een immutability policy opgezet worden. Hierbij werd gekozen voor een time-based retention policy van 90 dagen, de back-up is met andere woorden bescherm tegen verwijdering of wijziging voor deze periode. Dit is een veilige en praktische keuze voor back-ups. Daarnaast is er bij de optie `Allow protected append writes to` gekozen voor `Block and append blobs`, dit maakt het mogelijk om gegevens te blijven toevoegen aan de blob zonder de bestaande gegevens te wijzigen of te verwijderen, wat ideaal is voor scenario's zoals logbestanden of incrementele back-ups.



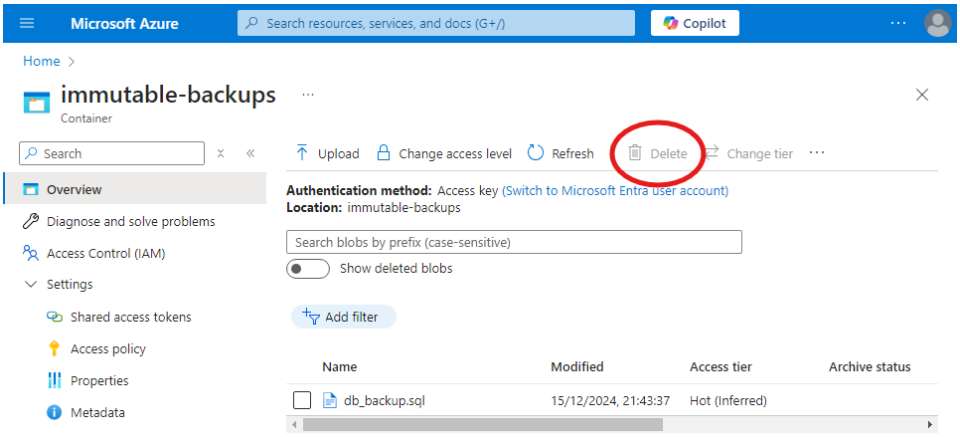
Figuur 5.5: Configuratie voor time-based retention policy

Testen van de immmutable storage

Om de immutable storage te testen is er gekozen om een back-up van een MySQL-database de uploaden naar de container. Na het uploaden van het bestand was er geen optie om dit bestand te verwijderen of te wijzigen. Met andere woorden is deze back-up dus beschermd tegen een ransomware-aanval.



Figuur 5.6: Uploaden van het back-up bestand van de MySQL-databank



Figuur 5.7: Screenshot van de poging om het bestand te verwijderen, waarbij de actie wordt geblokkeerd

Conclusie

De implementatie van immutable storage op het azure storage account is succesvol afgerond, waardoor de opgeslagen back-ups nu beschermd zijn tegen onverwachte wijzigingen of verwijderingen. Daarnaast worden er dagelijks automatische back-ups van de databanken genomen, welke een retentieperiode van 7 dagen hebben. Dit zorgt ervoor dat er altijd een versie van de back-up beschikbaar is, zelfs als de meest recente back-up beschadigd of onbruikbaar blijkt. Deze com-

binatie van immutable storage en versiebeheer versterkt de bescherming tegen dataverlies en maakt het mogelijk om eerdere, werkende back-ups snel te herstellen.

5.0.4. Implementatie van de Kubernetes-gebaseerde automatische back-upoplossing

Inleiding

In dit deel wordt een Kubernetes-omgeving ingezet om een schaalbare en geautomatiseerde back-upoplossing te ontwikkelen voor PostgreSQL- en MySQL-databases. Voor testdoeleinden worden de databases lokaal gehost op een Vagrant Virtual Machine (VM), wat een geïsoleerde omgeving biedt voor het opzetten van de back-upworkflow. De productieomgeving van Forvis Mazars maakt echter gebruik van Azure, waardoor de lokale testomgeving het proces simuleert onder vergelijkbare principes.

De back-ups worden uitgevoerd door containers die draaien binnen Kubernetes pods. Voor deze containers wordt gebruik gemaakt van een op maat gemaakte Docker-image. Deze image bevat een Python-script voor zowel het uitvoeren van de back-ups als het opschonen van verouderde bestanden. Kubernetes automatiseert de back-upworkflow verder met behulp van CronJobs, terwijl opslag wordt beheerd via Persistent Volumes (PV) en Persistent Volume Claims (PVC). Gevoelige gegevens worden beveiligd met Kubernetes Secrets.

5.0.5. Relevantie van de Kubernetes-setup voor Forvis Mazars

De Kubernetes-setup die in deze Proof-of-Concept (PoC) is ontwikkeld, blijkt direct relevant voor de werkomgeving van Forvis Mazars, aangezien ook daar Kubernetes wordt ingezet voor het beheer van de IT-infrastructuur. In deze sectie worden de overeenkomsten besproken, wordt toegelicht hoe de oplossing aansluit bij de behoeften van Forvis Mazars, en worden mogelijke aanpassingen beschreven om de setup volledig te integreren in hun bestaande workflows. Het belangrijkste verschil is dat bij Forvis Mazars de back-ups niet worden opgeslagen in een Persistent Volume, maar rechtstreeks worden geüpload naar een Azure Storage Container.

Overeenkomsten met de Forvis Mazars Setup

Bij Forvis Mazars wordt gebruikgemaakt van Kubernetes pods om databases te beheren en back-ups te genereren. De belangrijkste onderdelen van de ontwikkelde oplossing, zoals het gebruik van CronJobs en containerisatie, komen sterk overeen met de gebruikte werkwijze. Dit zorgt ervoor dat Forvis Mazars deze oplossing met minimale aanpassingen kan overnemen. Dit zijn de voornaamste gelijkenissen tussen de uitgewerkte oplossing en de productieomgeving van Forvis Mazars:

- **Automatisering via CronJobs**
  Bij zowel de PoC als de setup van Forvis Mazars worden Kubernetes CronJobs

toegepast voor het plannen van taken. In de PoC worden deze CronJobs ingezet om dagelijkse back-ups uit te voeren en oudere back-ups automatisch te verwijderen. Deze aanpak wordt gekenmerkt door schaalbaarheid en sluit goed aan bij de architectuur van Kubernetes die door Forvis Mazars wordt gebruikt.

- **Beveiliging van wachtwoorden**
  In de PoC worden Kubernetes Secrets gebruikt om gevoelige informatie, zoals databasewachtwoorden, door te geven aan het Python-script dat de back-ups uitvoert. Dit zorgt ervoor dat wachtwoorden niet hardcoded hoeven te worden in configuratiebestanden of scripts. Hoewel niet bekend is of Forvis Mazars eveneens Secrets gebruikt, biedt deze aanpak een veilige en flexibele oplossing die eenvoudig kan worden geïntegreerd in een vergelijkbare omgeving.

- **Schaalbaarheid en containerisatie**
  Het gebruik van een gestandaardiseerde Docker-image voor de uitvoering van de back-ups weerspiegelt de wijze waarop Kubernetes pods bij Forvis Mazars worden beheerd. Containers kunnen eenvoudig worden aangepast of opnieuw worden ingezet, zonder dat dit aanpassingen aan de volledige infrastructuur vereist.

Verschil in Back-upopslag

Het grootste verschil tussen de PoC en de setup van Forvis Mazars ligt in de wijze van back-upopslag. Bij de PoC worden back-ups lokaal opgeslagen in een Persistent Volume (PV), terwijl deze bij Forvis Mazars direct worden geüpload naar een Azure Storage Container. Dit verschil heeft enkele belangrijke implicaties:

- **Opslaglocatie**
  Persistent Volumes bieden een eenvoudige manier om gegevens lokaal binnen een Kubernetes-cluster op te slaan. Dit maakt ze geschikt voor testomgevingen en lokale implementaties, zoals in de PoC. Forvis Mazars maakt echter gebruik van een cloudopslagoplossing in de vorm van Azure Storage Containers. Deze oplossing biedt voordelen zoals schaalbaarheid, hoge beschikbaarheid en eenvoud in beheer binnen een productieomgeving.

- **Integratie met Azure**
  Bij Forvis Mazars wordt de Kubernetes-setup geïntegreerd met Azure door gebruik te maken van tools zoals Azure CLI of SDK's voor het uploaden van back-ups naar Azure Storage. Binnen de PoC zou een vergelijkbare integratie kunnen worden gerealiseerd door het Python-script aan te passen, zodat back-ups direct naar Azure worden geüpload in plaats van lokaal te worden opgeslagen. Dit kan worden bereikt door de Azure Storage SDK voor Python te implementeren of CLI-opdrachten toe te voegen aan de Docker-image.

Door deze overeenkomsten en verschillen te analyseren, kan worden geconcludeerd dat de ontwikkelde Kubernetes-setup een solide basis biedt die volledig kan worden afgestemd op de operationele vereisten van Forvis Mazars.

Python-script voor databaseback-ups

Het Python-script, zie bijlage E, is ontworpen om automatisch back-ups te maken van MySQL- en PostgreSQL-databases en biedt tevens functionaliteit om oude back-ups te verwijderen op basis van een ingestelde retentieperiode. Daarbij zorgt het script voor versiebeheer van de back-ups door in de bestandsnaam de datum en het type database op te nemen, wat het mogelijk maakt om verschillende versies van de back-ups te onderscheiden op basis van de tijd waarop ze zijn gemaakt. Het maakt gebruik van een object-georiënteerde aanpak, waarbij de `Backup`-klasse de kern van het script vormt. Bij de initialisatie van de klasse worden verschillende parameters ingesteld, waaronder de naam van de database, de gebruikersnaam en het type database (MySQL of PostgreSQL), evenals de locatie van de back-ups. De bestandsnaam voor de back-up wordt dynamisch gegenereerd op basis van de datum en het type database, en krijgt de volgende vorm:

`backup-{type}-{YYYY-MM-DD}-{database_name}.dump`

Daarnaast worden omgevingsvariabelen gebruikt voor de IP-adressen van de servers waarop de databanken draaien en poorten van zowel MySQL als PostgreSQL, wat zorgt voor een flexibele configuratie van het script.

Listing 5.18: Backup-klasse van het python-script.

```
1  class Backup:
2  def __init__(self, database_name, database_user, type, backup_dir):
3  timestr = datetime.datetime.now().strftime('%Y-%m-%d')
4  self.filename = f'backup-{type}-{timestr}-{database_name}.dump'
5  self.database_name = database_name
6  self.database_user = database_user
7  self.type = type
8  self.password = None
9  self.backup_dir = backup_dir
10 self.hostname_mysql = os.environ.get("HOSTNAME_MYSQL", "192.168.1.62")
11 self.hostname_psql = os.environ.get("HOSTNAME_PSQL", "192.168.1.62")
12 self.port_mysql = os.environ.get("PORT_MYSQL", "3306")
13 self.port_psql = os.environ.get("PORT_PSQL", "5432")
```

Een essentieel onderdeel van het script is de `set_password`-methode, die het databasewachtwoord uit een omgevingsvariabele haalt. Indien het wachtwoord niet is ingesteld, wordt een waarschuwingsbericht weergegeven en stopt het script om onveilige of onvolledige back-ups te voorkomen. Dit maakt de back-upprocedure veiliger en garandeert dat alleen bevoegde gebruikers toegang hebben tot de databasegegevens.

De `create_backup`-methode is verantwoordelijk voor het daadwerkelijk uitvoeren van de back-up. Afhankelijk van het type database wordt ofwel het `mysqldump`-

commando voor MySQL of het `pg_dump`-commando voor PostgreSQL uitgevoerd. Bij het maken van een MySQL-back-up wordt het `mysqldump`-commando gebruikt met de juiste databasegegevens, zoals gebruikersnaam, wachtwoord, hostnaam en poort. Voor PostgreSQL wordt het `pg_dump`-commando gebruikt. Beide commando's worden uitgevoerd via de `subprocess.run`-functie, die de uitvoer naar een bestand schrijft. Mocht een van de commando's mislukken, dan zorgt de foutafhandeling ervoor dat het script stopt en een foutmelding wordt weergegeven.

Listing 5.19: create_backup-methode van het python-script.

```
    def create_backup(self, type):
    # Create a backup of the database.

    # Ensure the backup directory exists
    if not os.path.exists(self.backup_dir):
    os.makedirs(self.backup_dir)
    # Prepend the backup directory to the filename
    full_path = os.path.join(self.backup_dir, self.filename)

    if type == "MYSQL":
    try:
    cmd = [
    'mysqldump',
    '--single-transaction',
    '-u', self.database_user,
    f'-p{self.password}',
    '-h', self.hostname_mysql,
    '-P', str(self.port_mysql),
    '--no-tablespaces',
    '-B', self.database_name,
    ]
    with open(full_path, 'w') as backup_file:
    result = subprocess.run(cmd, stdout=backup_file, check=True)  # nosec
    if result.returncode != 0:
    logging.warning(f'Command failed. Return code: {result.returncode}')
    exit(1)
    return full_path
    except Exception as e:
    logging.warning(e)
    exit(1)

    elif type == "PSQL":
    try:
    cmd = [
    'pg_dump',
    '-U', self.database_user,
    '-h', self.hostname_psql,
    '-p', str(self.port_psql),
    '-F', 'c',
    '-f', full_path,
    self.database_name
```

```
    ]
    result = subprocess.run(cmd, check=True)  # nosec
    if result.returncode != 0:
    logging.warning(f'Command failed. Return code: {result.returncode}')
    exit(1)
    return full_path
    except Exception as e:
    logging.warning(e)
    exit(1)
```

Daarnaast bevat het script de `delete_old_backups`-methode, die ervoor zorgt dat oude back-ups automatisch worden verwijderd om opslagruimte te beheren. Het script vergelijkt de wijzigingsdatum van elk bestand in de opgegeven back-up directory met de huidige datum en verwijdert bestanden die ouder zijn dan de ingestelde retentieperiode. Dit maakt het mogelijk om ongewenste en verouderde back-ups op te schonen, wat belangrijk is voor het beheer van systeembronnen.

Listing 5.20: delete_old_backups-methode van het python-script.

```
    @staticmethod
    def delete_old_backups(directory, retention_days):
    """
    Deletes backup files older than the specified retention period.
    """
    now = datetime.datetime.now()
    logging.info(f"Starting cleanup of backups in {directory} with retention
        period: {retention_days} days")
    for file in os.listdir(directory):
    if file.startswith("backup-") and file.endswith(".dump"):
    file_path = os.path.join(directory, file)
    file_mtime = datetime.datetime.fromtimestamp(os.path.getmtime(file_path))
    age = (now - file_mtime).days
    logging.info(f"Checking file {file_path}: age = {age} days")
    if age >= retention_days:
    try:
    os.remove(file_path)
    logging.info(f"Deleted old backup: {file_path}")
    except Exception as e:
    logging.warning(f"Failed to delete {file_path}: {e}")
    else:
    logging.info(f"File {file_path} is not old enough to delete (age = {age} days)
        .")
```

Het script maakt gebruik van uitgebreide logging om de voortgang van de back-up- en opruimprocessen te documenteren. Logging biedt inzicht in de status van de uitgevoerde handelingen, zoals succesvolle back-ups, waarschuwingen voor ontbrekende omgevingsvariabelen, foutmeldingen bij mislukte bewerkingen en informatie over de bestanden die worden gecontroleerd of verwijderd.

Ten slotte wordt het script uitgevoerd vanuit de `main`-functie, die de ingevoerde argumenten verwerkt en de bijbehorende methoden aanroept. Als een

back-up wordt aangevraagd, wordt een instantie van de `Backup`-klasse gemaakt en worden de methoden `set_password` en `create_backup` uitgevoerd. Als de gebruiker de optie voor het verwijderen van oude back-ups heeft opgegeven, wordt de statische methode `delete_old_backups` aangeroepen om de oude bestanden te verwijderen. Op deze manier biedt het script een oplossing voor het beheren van back-ups en het efficiënt verwijderen van verouderde back-ups.

Dockerfile

Het Dockerfile, zoals weergegeven in bijlage D, is ontworpen om een container te creëren die de benodigde software bevat voor het uitvoeren van de geautomatiseerde back-ups via het eerder besproken Python-script binnen Kubernetes. Het begint met het ophalen van de laatste versie van Ubuntu als basisimage. Vervolgens worden de MySQL- en PostgreSQL-clients geïnstalleerd, samen met Python3 voor het uitvoeren van het script. Het Python-script wordt vanuit de lokale src-map naar de container gekopieerd. Deze Docker-image is uiteindelijk gepushed naar een Docker-account, zodat deze gebruikt kan worden in de Kubernetes-omgeving. Dit Dockerfile biedt de nodige omgeving voor het uitvoeren van de back-ups in de Kubernetes pods.

Kubernetes cronjobs en werking

Elke pod bevat een of meer containers die een specifieke taak uitvoeren. In dit project wordt een pod geconfigureerd om:

- De back-up van een database uit te voeren.
- Data op te slaan in een Persistent Volume.
- Gevoelige informatie, zoals wachtwoorden, te beheren via Secrets.

Voor elke database (PostgreSQL en MySQL) is een aparte CronJob geconfigureerd. Deze CronJobs maken gebruik van dezelfde Docker-image, maar passen specifieke parameters toe om de juiste database te back-uppen.

Het proces verloopt als volgt:

1. De CronJob wordt volgens een gepland tijdschema uitgevoerd.
2. De CronJob spint een pod op waarin de container draait die de back-up uitvoert.
3. De container voert het Python-script uit dat de database-back-up naar een Persistent Volume schrijft.
4. Na voltooiing wordt de pod beëindigd, terwijl de back-upbestanden bewaard blijven.

Het eerste CronJob YAML-bestand, dat wordt gepresenteerd in bijlage F.1, configureert een geautomatiseerde taak die elke nacht om 2 uur een back-up maakt

van een MySQL-database. Deze taak wordt uitgevoerd in een Kubernetes-omgeving en maakt gebruik van een container die het eerder genoemde Python-script uitvoert om de back-up te maken. De cronjob is opgezet om periodiek de database te back-uppen zonder menselijke tussenkomst.

De `apiVersion: batch/v1` en `kind: CronJob` geven aan dat het bestand een Kubernetes CronJob definieert, een soort geplande taak die op regelmatige tijdstippen wordt uitgevoerd. De `schedule` is ingesteld op "0 2 * * *", wat betekent dat de back-up elke dag om 2:00 uur 's nachts wordt uitgevoerd. Dit schema volgt de standaard cron-vergelijkingsnotatie, waarbij de tijd en frequentie worden gedefinieerd.

Binnen de `jobTemplate` wordt de daadwerkelijke taak gespecificeerd die door de cronjob moet worden uitgevoerd. De container die wordt gebruikt voor deze taak is de `bouzewazi/ubu4` Docker-image, die het Python-script bevat dat verantwoordelijk is voor het maken van de back-up. Het script wordt uitgevoerd met een aantal argumenten, waaronder de database-naam (`testdb`), gebruikersnaam (`root`), het type database (`MYSQL`), en de locatie van de back-up (`/data`). Dit zorgt ervoor dat de juiste database op het juiste moment wordt geback-upt en de back-up wordt opgeslagen in een gedeelde opslaglocatie.

De container maakt gebruik van een persistent volume claim, gedefinieerd onder `volumes`, om back-upbestanden op te slaan. Het volume wordt gekoppeld aan een specifieke directory binnen de container via `volumeMounts`. Dit zorgt ervoor dat de opgeslagen gegevens behouden blijven, zelfs als de container opnieuw wordt opgestart.

Verder wordt het databasewachtwoord veilig doorgegeven via een Kubernetes Secret. Het wachtwoord wordt opgehaald uit een Kubernetes Secret als een environment variable genaamd `DB_PASSWORD`. Dit zorgt ervoor dat gevoelige gegevens op een veilige manier aan de container worden verstrekt zonder dat ze hardcoded in de YAML worden opgenomen.

Daarbij is er ook een andere CronJob die dezelfde functionaliteit biedt, maar dan voor het maken van back-ups van een PostgreSQL-database. Deze CronJob is vergelijkbaar met de MySQL CronJob, met aanpassingen die specifiek zijn voor de PostgreSQL-databaseconfiguratie, zoals te zien in bijlage F.2.

Tenslotte is er een CronJob die oude back-ups verwijdert die ouder zijn dan 14 dagen, wat de retentieperiode is voor de dagelijkse back-ups. Dit zorgt ervoor dat alleen back-ups binnen de laatste 14 dagen bewaard blijven, waardoor opslag efficiënt wordt beheerd, zoals beschreven in bijlage F.3.

Overstap van Docker naar Kubernetes voor cronjob automatisering

Aanvankelijk werd gekozen om de automatisering van de back-ups te beheren via een Docker-container, waarin CronJobs werden ingesteld om dagelijks back-ups te maken en oude back-ups te verwijderen. Binnen de container werd een setup.sh-script uitgevoerd om de cron-service te starten en de benodigde cronjobs te con-

figureren. Dit proces gebruikte een Dockerfile, zie bijlage D.2, die de benodigde software installeerde, zoals MySQL-client, PostgreSQL-client, cron en Python3. Ook werd het Python-script en setup-script, zie bijlage H, in de container gekopieerd. De CronJobs werden geconfigureerd om dagelijks back-ups te maken om 2:00 uur 's nachts en oude back-ups, ouder dan 14 dagen, te verwijderen.

Hoewel dit functioneel was, bood deze oplossing beperkte persistentie voor de back-ups, aangezien de opslag binnen de container plaatsvond. Uiteindelijk werd besloten om de CronJobs op Kubernetes te draaien, wat beter aansluit bij de infrastructuur van Forvis Mazars. Kubernetes biedt schaalbaarheid, persistente opslag en een efficiënte integratie binnen de bestaande setup van het bedrijf, waardoor het de logische keuze was voor het beheren van de geautomatiseerde back-uptaken.

5.0.6. Herstelcapaciteit (Restore from Backup)

Een belangrijk onderdeel van het praktijkgedeelte van dit onderzoek is het testen van de herstelcapaciteit (restore from back-up). Dit was een essentiële vereiste in het onderzoek, waarbij de focus lag op het efficiënt en snel herstellen van systemen vanuit de back-ups. Het doel was om een idee te krijgen over hoe snel en betrouwbaar een databank up-and-running kan zijn na het ondergaan van een incident.

Voor deze must-have werd het herstelproces gesimuleerd in een testomgeving waarbij de back-ups werden opgehaald uit de immutable Azure storage container en deze dan herstelt werden in een MySQL- en PostgreSQL-databank. De database die hier gebruikt werd had een tabel van 6 kolommen met ongeveer 2000 rijen. Deze testomgeving komt grotendeels overeen met de actieve omgeving die Forvis Mazars gebruikt en de database komt overeen met een database die Forvis Mazars ook zou gebruiken. Het proces omvatte de volgende stappen:

Back-ups ophalen uit de immutable storage in Azure

De back-ups werden veilig opgeslagen in een Azure Storage Account met een immutable opslagbeleid, wat een cruciale maatregel is tegen ransomware-aanvallen. Door het gebruik van het `az storage blob download`-commando werden de back-ups gedownload naar de lokale virtuele machine.

Herstellen van de PostgreSQL- en MySQL-databanken

Na het downloaden werden de PostgreSQL- en MySQL-databases opnieuw opgezet met herstelcommando's. Voor PostgreSQL werd het bestand hersteld met `pg_restore`, terwijl voor MySQL gebruik werd gemaakt van het `mysql`-commando. Beide hersteltaken verliepen succesvol en zonder dataverlies. Om aan te tonen dat de integriteit van de back-up is behouden werd er een hash berekend van de tabel in de MySQL-shell van de database voor de restore en deze werd vergeleken met de hash na het herstellen van de database vanuit de back-up. Dit is de hash van de database voor de restore:

Listing 5.21: MySQL commando om de hash van de tabel te berekenen voor de restore.

```
mysql> SELECT SUM(CRC32(CONCAT_WS('#', firstName, lastName, gender, address, phone
    , email))) AS checksum FROM testdb;
+---------------+
| checksum      |
+---------------+
| 4336568446292 |
+---------------+
1 row in set (0.00 sec)
```

De hash van de database na het uitvoeren van de restore is identiek en geeft dezelfde output:

Listing 5.22: MySQL commando om de hash van de tabel te berekenen voor de restore.

```
mysql> SELECT SUM(CRC32(CONCAT_WS('#', firstName, lastName, gender, address, phone
    , email))) AS checksum FROM testdb;
+---------------+
| checksum      |
+---------------+
| 4336568446292 |
+---------------+
1 row in set (0.00 sec)
```

Dit hele proces is gemeten en duurde 48 seconden, wat wijst op een snelle toegang tot de back-ups in noodsituaties. Het back-upbestand voor de MySQL-databank had een bestandsgrootte 383 KiB en het bestand voor de PostgreSQL-databank had een grootte van 45,16 KiB. Deze bestanden werden gedownload met een internetsnelheid van 75.33 Mbit/s. Dit is getest met de tool speedtest-cli.

Link met Herstelcapaciteit

De testprocedure toont aan dat de opgestelde back-upstrategie niet alleen effectief is in het beschermen van gegevens, maar ook voldoet aan de eisen van herstelcapaciteit:

- **Efficiëntie:** Het herstelproces, vanaf het downloaden van de back-ups tot het opnieuw opzetten van de databases, verliep in een relatief korte tijd. Dit bevestigt dat de back-ups eenvoudig en snel toegankelijk zijn, wat essentieel is in een bedrijfsomgeving waar down-time zo laag mogelijk moet zijn.
- **Betrouwbaarheid:** Door gebruik te maken van immutable opslag in Azure worden de back-ups beschermd tegen ongewenste acties en aanvallen, wat de kans op succesvol herstel sterk verhoogt.

Conclusie

De uitgevoerde tests bieden waardevolle inzichten in de herstelcapaciteit van de back-upstrategie. Het toont aan hoe een goed ontworpen back-upplan bedrijven in staat stelt om snel te reageren op noodsituaties en hun systemen betrouwbaar

te herstellen. Hiermee is aangetoond dat het systeem niet alleen beschermt tegen gegevensverlies, maar ook operationeel herstel mogelijk maakt binnen een acceptabele tijd. Dit maakt het een reële oplossing voor bedrijven die ransomware-resistente back-upstrategieën willen implementeren.

6

Conclusie

In het kader van deze bachelorproef werd een robuuste back-upstrategie ontwikkeld en getest om de veiligheid van databases bij Forvis Mazars te waarborgen, met een sterke focus op het beschermen tegen ransomware-aanvallen. De proef werd opgedeeld in vier hoofdonderdelen, die elk bijdroegen aan de evaluatie van de effectiviteit en efficiëntie van de voorgestelde oplossingen.

In het eerste deel van het onderzoek werd een gecontroleerde testomgeving gecreëerd door drie virtuele machines (VM's) op te zetten met behulp van Vagrant en VirtualBox. Deze VMs simuleerden een ransomware-aanval waarbij de back-upbestanden werden versleuteld en de actieve database op de actieve VM werd verwijderd. Het gebruik van immutable storage bleek effectief in het beschermen van de back-up, aangezien de integriteit van de gegevens behouden bleef. Door het vergelijken van de hashes van de originele en herstelde back-ups werd aangetoond dat de gegevens niet waren aangetast, wat de effectiviteit van immutability in een ransomware-aanval bevestigde.

In het tweede deel werd een Azure-oplossing geïmplementeerd in de actieve productieomgeving van Forvis Mazars, waarbij een immutable Azure Storage-container werd opgezet voor het veilig opslaan van back-ups. Dit deel van het onderzoek bevestigde dat immutable opslag effectief kan bijdragen aan het beschermen van back-ups tegen wijzigingen en dataverlies door externe bedreigingen.

Het derde onderdeel betrof de implementatie van een schaalbare en geautomatiseerde back-upoplossing in een Kubernetes-omgeving. Hiervoor werden PostgreSQL- en MySQL-databases lokaal gehost op een Vagrant VM voor testdoeleinden, terwijl de productieomgeving van Forvis Mazars gebruik maakt van Azure. Door containers binnen Kubernetes Pods te draaien, konden back-ups automatisch worden uitgevoerd met behulp van CronJobs. Daarnaast werden de back-ups opgeslagen in Persistent Volumes (PV) en beveiligd met Kubernetes Secrets. Dit zorgde voor een geautomatiseerde en veilige oplossing die zich goed leent voor

schaalbaarheid en integratie met de bestaande infrastructuur van Forvis Mazars.

Het laatste onderdeel van het onderzoek richtte zich op het testen van het herstelproces van back-ups. Zowel de MySQL- als PostgreSQL-databases werden hersteld uit de back-ups, waarbij de integriteit van de gegevens werd gecontroleerd door het vergelijken van de hashes van de data voor en na het herstel. Het herstelproces werd uitgevoerd over een internetsnelheid van 75,33 Mbit/s, waarbij de MySQL-database (383 KiB) binnen 48 seconden werd hersteld, evenals de PostgreSQL-database (45,16 KiB). Het herstel toonde aan dat zowel de snelheid als de integriteit van de gegevens behouden bleef, wat de betrouwbaarheid van het back-up- en herstelproces benadrukte.

Samenvattend heeft deze bachelorproef aangetoond dat het gebruik van immutabele opslag, geautomatiseerde back-ups via Kubernetes en betrouwbare herstelprocessen de veiligheid en effectiviteit van de back-upstrategie bij Forvis Mazars aanzienlijk verbeteren. De voorgestelde oplossing biedt niet alleen bescherming tegen ransomware-aanvallen, maar zorgt ook voor een efficiënte en schaalbare back-upworkflow die snel herstel mogelijk maakt, met behoud van de integriteit van de gegevens.

A

Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.0.1. abstract

In deze bachelorproef wordt een optimalisatie van de back-upstrategie voor de Azure PostgreSQL en MySQL databases bij Forvis Mazars onderzocht, de focus ligt voornamelijk op immutabele opslag en automatische back-ups. Het doel is om de back-upstrategie van Forvis Mazars te optimaliseren en het resistent te maken tegen ransomware-aanvallen. Daarnaast wordt er ook een Proof-of-Concept (PoC) uitgevoerd, waarin immutabele opslag wordt geïmplementeerd om ervoor te zorgen dat back-ups onveranderlijk zijn na opslag. Verder worden geautomatiseerde back-ups geïmplementeerd om de back-ups efficiënter te maken en de consistentie van de back-ups te verbeteren. In de state-of-the-art ligt de focus op bestaande back-upstrategieën, zoals cloud back-ups, on-premise back-ups, en offline back-ups. De methodologie omvat een literatuurstudie, een analyse van de huidige back-upstrategie bij Forvis Mazars, en de ontwikkeling van een PoC. De verwachte resultaten zullen de verbeterde beveiliging en efficiëntie van de back-upstrategie aantonen, met als doel het minimaliseren van de risico's op dataverlies en het waarborgen van bedrijfscontinuïteit.

Inhoudsopgave

A.0.2. Inleiding

Ransomware-aanvallen zijn één van de meest voorkomende aanvallen dat een organisatie kan treffen de dag van vandaag. Om gegevensverlies tegen te gaan in geval van een aanval moeten bedrijven altijd een back-upplan klaar hebben in geval van een incident. Het niet hebben van een back-upplan of het hebben van een suboptimaal plan kan leiden tot een groot verlies op financieel vlak. Daarnaast kan dit ook zorgen voor het verliezen van cruciale informatie en als laatste kan dit de reputatie van een organisatie sterk doen dalen, aangezien niemand in zee wilt gaan met een bedrijf dat niet goed beveiligd is of niet goed voorbereid is op uitzonderlijke incidenten. Om deze redenen is het van groot belang voor een bedrijf om een doordacht, robuust en veilig back-upplan te hebben.

Het doel van deze bachelorproef is het optimaliseren van het back-upplan van Forvis Mazars. Dit bedrijf maakt gebruik van 2 soorten databases in Azure, eenderzijds een PostgreSQL databank en anderzijds een MySQL databank. Van deze databanken worden er automatische alsook manuele back-ups gemaakt. De automatische back-ups gebeuren door Azure zelf en de manuele back-ups worden per database uitgevoerd. Stel dat applicatie A een nieuwe versie heeft, dan zal er eerst een back-up genomen worden van de database vooraleer de nieuwe versie uitgerold wordt. Echter zijn er nog bepaalde verbeteringen mogelijk, zoals het veiliger opslaan van deze back-ups met behulp van technieken als immutable storage en het instellen van geautomatiseerde dagelijkse, wekelijkse en maandelijkse back-ups. Daarbij is een belangrijk aandachtspunt dat de databanken beter beveiligd moeten worden tegen cyberaanvallen aangezien gegevens in zo'n situatie versleuteld of vernietigd kunnen worden.

De doelgroep van dit onderzoek bestaat uit de IT-professionals en vooral de systeembeheerders van Forvis Mazars, die verantwoordelijk zijn voor het beheer van de back-ups en de beveiliging van gegevens binnen de organisatie alsook het herstellen van alle gegevens na een incident.

De onderzoeksvraag die onderzocht zal worden is: "Hoe kan de back-upstrategie van de Azure PostgreSQL en MySQL databases bij Forvis Mazars worden geoptimaliseerd door het implementeren van automatische back-ups en het veilig opslaan van deze back-ups om gegevensverlies te minimaliseren?" In dit onderzoek wordt onderzocht hoe de bestaande back-upoplossingen kunnen worden verbeterd, zodat Forvis Mazars in geval van een incident goed voorbereid is en geen informatie

verliest. De onderzoeksvraag kan onderverdeeld worden in volgende kleinere deelvragen:

- Hoe veilig en betrouwbaar zijn de huidige back-upoplossingen van Forvis Mazars voor Azure PostgreSQL en MySQL databases?
- Welke rol speelt immutabele opslag in het beschermen van back-ups tegen ransomware en andere vormen van dataverlies?
- Wat zijn de belangrijkste uitdagingen bij het integreren van immutabele opslag met Azure cloud back-upsystemen?
- Hoe kan er voor de Azure PostgreSQL en MySQL databases een automatische back-upstrategie worden geïmplementeerd?

Het uiteindelijke doel van dit onderzoek is om ervoor te zorgen dat de Azure databanken van Forvis Mazars een geoptimaliseerd back-upplan hebben dat veilig en efficient is. Het plan moet immuun zijn tegen ransomware-aanvallen en daarnaast moet het ook geautomatiseerd zijn. Daarbij zal de tijd bij een herstel vanuit een back-up ook onderzocht worden. Om het back-upplan te testen zal er een Proof-of-Concept (PoC) opgesteld worden om alles grondig te testen in een testomgeving.

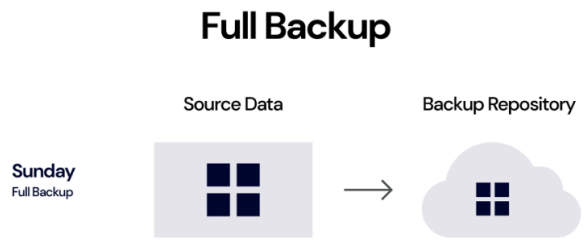
A.0.3. Literatuurstudie

Bedrijven moeten hun data goed beschermen om succesvol te zijn. Echter vormen back-ups van de databases vaak een zwakke schakel in de beveiligingsketen van gegevensbescherming. Hoewel veel organisaties zich richten op het beveiligen van hun actieve databases, worden de back-ups vaak over het hoofd gezien, wat een groot risico met zich meebrengt. Back-ups worden meestal offsite opgeslagen, bijvoorbeeld op tape, en zijn daardoor vatbaar voor verlies of diefstal (Cherry, 2015). Dit maakt het van essentieel belang om back-ups goed te beveiligen, bijvoorbeeld door encryptie. Echter, bij het kiezen van een encryptieoplossing is het belangrijk om een evenwicht te vinden tussen de sterkte van de encryptie en de impact op de prestaties van de server omdat sterke encryptie meer resources nodig heeft en het kan de beschikbaarheid van de back-ups veranderen. In de afgelopen jaren is er een scherpe toename van ransomware-aanvallen gericht op bedrijven, waarbij het aantal getroffen organisaties is gestegen van ruim 2.700 naar bijna 4.900 in slechts twaalf maanden. Deze toename laat zien hoe vastberaden en steeds slimmer ransomwaregroepen worden in hun aanvallen. Wat bijzonder zorgwekkend is voor de bedrijfswereld, is de trend van herhaalde aanvallen op bedrijven, waarbij sommige organisaties binnen korte tijd door meerdere ransomwaregroepen worden getroffen (Dikbiyik e.a., 2024). Dit wijst erop dat cybercriminelen actief profiteren van momenten van kwetsbaarheid om bedrijven in hun zwakste momenten

opnieuw aan te vallen, wat de noodzaak voor robuuste preventieve maatregelen benadrukt.

back-upstrategieën

Full back-up



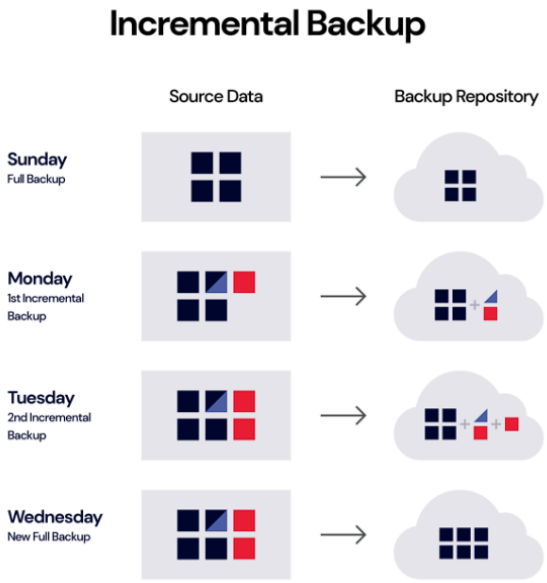
Figuur A.1: Representatie van een full back-up (Rivas, 2022)

Een full back-up is een back-upmethode waarbij alle gegevens van een systeem op een specifiek moment volledig worden gekopieerd en opgeslagen. Dit betekent dat elk bestand zonder uitzonderingen wordt gekopieerd, zodat er een exacte kopie van de volledige dataset ontstaat (Beard, 2018). Wanneer er zich een probleem voordoet, zoals het falen van een harde schijf, kan het hele bestandssysteem vanaf deze back-up volledig worden hersteld op een nieuwe schijf. Daarnaast kunnen ook individuele bestanden die verloren zijn gegaan, gemakkelijk worden teruggehaald uit de back-up. Dit soort back-up zorgt ervoor dat alle gegevens veilig zijn opgeslagen. Ten eerste is het proces van het lezen en schrijven van het volledige bestandssysteem tijdsintensief, vooral bij grote hoeveelheden data. Ten tweede gebruikt het opslaan van een volledige kopie van het bestandssysteem veel opslagruimte, wat inefficiënt kan zijn wanneer de back-ups regelmatig worden gemaakt (Chervenak e.a., 1998).

Incrementele back-ups

Incremental back-ups zijn een efficiënte methode om alleen gewijzigde data sinds de laatste back-up op te slaan, wat tijd en opslag bespaart. In tegenstelling tot een volledige back-up, die alle data kopieert, richten incrementele back-ups zich enkel op nieuwe of aangepaste bestanden. Dit maakt ze sneller, maar hersteltijden kunnen langer zijn omdat meerdere incrementele back-ups nodig zijn naast de laatste volledige back-up. Recente onderzoeken hebben zich gericht op het optimaliseren van back-upstrategieën, met name voor databasesystemen. Zo zijn er modellen ontwikkeld die bepalen hoe vaak volledige en incrementele back-ups moeten worden uitgevoerd op basis van factoren zoals systeem-betrouwbaarheid, de hoeveelheid dataveranderingen en back-up-kosten (Zhao e.a., 2024). Er zijn ook varianten zoals differentiële back-ups, die alle veranderingen sinds de laatste volledige back-up bevatten, waardoor de hersteltijd korter kan zijn dan bij traditionele incrementele back-ups. Daarnaast zorgen moderne geautomatiseerde oplos-

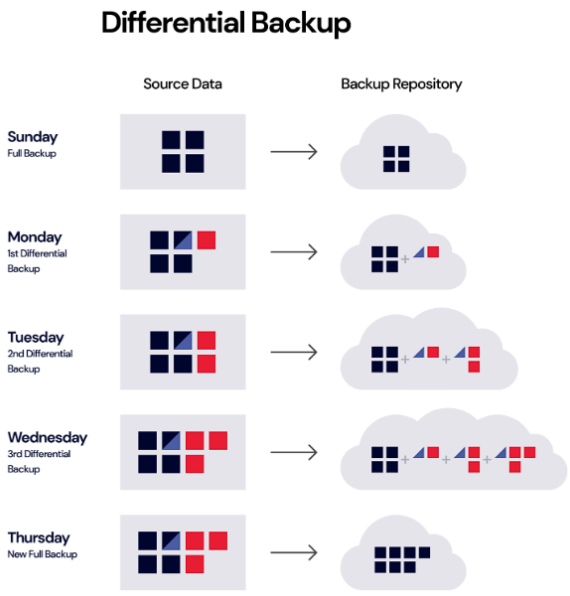
singen voor continue incrementele back-ups, wat realtime herstelmogelijkheden biedt zonder noemenswaardige belasting van de productieomgeving (Qian e.a., 2010).



Figuur A.2: Representatie van een incrementele back-up (Rivas, 2022)

Differentiële back-up

Een differentiële back-up is een soort back-up waarbij alleen de data die sinds de laatste volledige back-up is veranderd of toegevoegd, wordt gekopieerd. In tegenstelling tot een incrementele back-up, die enkel de veranderingen sinds de laatste back-up opslaat, wordt er bij een differentiële back-up enkel de wijzigingen opgeslagen sinds de laatste full back-up. Dit komt doordat elke differentiële back-up alle wijzigingen sinds de meest recente volledige back-up bevat, waardoor de grootte van de back-up groter wordt naarmate er meer wijzigingen plaatsvinden. Een belangrijk voordeel van differentiële back-ups is de relatief snelle hersteltijd (Beard, 2018). Om data te herstellen, is alleen de laatste volledige back-up en de meest recente differentiële back-up nodig, wat het herstelproces eenvoudiger en sneller maakt dan bij incrementele back-ups. Differentiële back-ups zijn bijzonder nuttig in omgevingen waar een snel herstelproces cruciaal is, zoals bij bedrijven die minimale downtime vereisen. Het nadeel van differentiële back-ups is dat de back-ups groter worden naargelang de tijd tussen de volledige back-ups. Elke nieuwe differentiële back-up bevat namelijk alle wijzigingen sinds de laatste volledige back-up, wat betekent dat deze geleidelijk groter wordt totdat er een nieuwe volledige back-up wordt gemaakt. Daarom is het belangrijk om een goede balans te vinden tussen de frequentie van volledige back-ups en differentiële back-ups.



Figuur A.3: Representatie van een differentiële back-up (Rivas, 2022)

On-premise back-ups

Bedrijven staan vaak voor de uitdaging om te beslissen of ze hun data on-premise opslaan of de voorkeur geven aan een cloud-service (Ali e.a., 2024). On-premise back-ups slaan gegevens lokaal op, meestal op fysieke schijven binnen het bedrijf zelf. Deze methode biedt bedrijven volledige controle over hun back-up- en gegevensbeheer. Een belangrijk voordeel van on-premise back-ups is dat data altijd beschikbaar is, zelfs zonder toegang tot het internet, wat nuttig is bij netwerkproblemen (Trovato e.a., 2019). Daarnaast hebben bedrijven volledige eigendom over de beveiliging van hun gegevens, aangezien de opslag lokaal blijft binnen het bedrijf. Hoewel deze methode geen terugkerende kosten aan externe providers met zich meebrengt, brengt het wel risico's met zich mee, zoals schade door brand of overstromingen, en vraagt het om regelmatige onderhoud van de hardware. Het herstelproces is doorgaans sneller dan bij een cloudservice, wat van cruciaal belang kan zijn na een ransomware-aanval of een ander incident.

Cloud-gebaseerde back-ups

Cloud-gebaseerde back-ups zijn een populaire oplossing waarbij data extern wordt opgeslagen bij een bedrijf dat cloudservices aanbiedt. Dit biedt individuen en bedrijven de mogelijkheid om hun gegevens veilig op afstand te bewaren, zonder dat ze hoeven te investeren in fysieke opslagapparaten. Hoewel dit handig is om gegevensverlies te voorkomen bij hardware- of softwarestoringen, of onverwachte rampen, brengt het gebruik van cloud-opslag vaak aanzienlijke kosten met zich mee, vooral op de lange termijn (Obrutsky, 2016). Naarmate je meer opslag nodig hebt is er altijd een mogelijkheid om te opschalen, dit is een groot voordeel van het gebruiken van een cloudservice. Echter, het waarborgen van de veiligheid van

deze data is een cruciaal aandachtspunt, vooral omdat cloudproviders vaak niet open zijn over hoeveel kopieën van de data er zijn en waar deze precies worden opgeslagen. Om problemen zoals datalekken en foutieve verwijdering te voorkomen, zijn er nieuwe methoden zoals “assured deletion” ontwikkeld, waarmee klanten zeker weten dat hun gegevens permanent worden verwijderd op verzoek. Hierdoor kunnen bedrijven hun data met zekerheid beheren in de cloud terwijl gevoelige informatie veilig blijft (Rahumed e.a., 2011).

Offline back-ups

Offline back-ups zijn een traditionele methode waarbij data wordt opgeslagen op fysieke media, meestal externe harde schijven zonder tussenkomst van het internet. Het voornaamste voordeel is dat de data dan beveiligd is tegen online bedreigingen en er geen internettoegang nodig is om aan de data te geraken (Edwards, 2022). Een belangrijk voordeel is dat offline back-ups niet beïnvloed worden door stroomstoringen of internetuitval, waardoor ze een robuuste back-upoptie vormen voor gevoelige data. Echter, in tegenstelling tot on-premise back-ups, die vaak op dezelfde fysieke locatie als de IT-infrastructuur van een bedrijf worden opgeslagen, kunnen offline back-ups eenvoudig meegenomen en elders bewaard worden, waardoor ze extra bescherming bieden tegen fysieke rampen. Toch delen beide methoden het nadeel dat ze kwetsbaar zijn voor schade door ongelukken, diefstal of verlies, en moeten de fysieke apparaten op een veilige locatie opgeslagen worden (James, 2019).

Immutable storage

Immutable storage is een type opslag waarbij data niet meer kan worden gewijzigd of aangepast vanaf het geback-upt is. Dit concept is cruciaal voor het waarborgen van de integriteit van belangrijke gegevens. Het idee achter immutability is dat bepaalde bestanden, nadat ze zijn gemaakt, niet meer mogen worden gewijzigd zonder de juiste autorisatie. Dit biedt een sterke bescherming tegen ongewenste wijzigingen en hierdoor kunnen hackers de gegevens niet aanpassen. Immutable storage speelt dus een belangrijke rol in het beschermen van systemen tegen cyberaanvallen. Bij aanvallen, waarbij hackers volledige toegang verkrijgen, kunnen onbeveiligde systemen worden gemanipuleerd of misbruikt. Immutable storage voorkomt dit, omdat de opgeslagen gegevens niet kunnen worden gewijzigd, zelfs niet door iemand met volledige toegang. Hierdoor wordt de integriteit van de data behouden en is het risico op schade door hackers aanzienlijk kleiner (Hasan e.a., 2005).

A.0.4. Methodologie

In de eerste fase van het onderzoek zal er een grondige literatuurstudie worden uitgevoerd rond back-upstrategieën, ransomware-resistentie, en immutable storage met een overzicht van de state of the art van back-upstrategieën en immutabele

opslag als deliverable. Wetenschappelijke papers, bedrijfscasussen en technische artikels zullen gebruikt worden om een theoretische basis aan te leggen en om de best-practices te achterhalen. Dit zal ook helpen om de onderzoeksvragen te beantwoorden. Daarnaast zal er een Proof-Of-Concept ontworpen worden waarbij onderzocht zal worden hoe er immutable back-ups gemaakt kunnen worden voor de back-ups van Forvis Mazars. Daarnaast zal er een testomgeving opgezet worden om een ransomware-aanval na te bootsen en het systeem opnieuw op gang te krijgen. De deliverable voor de PoC is een werkende immutable back-upoplossing in Azure die tegen een ransomware-aanval bestendig is. Verder zal er een optimale back-upstrategie opgesteld worden met de state-of-the-art technieken. De literatuurstudie zal ongeveer 4 weken duren, de Proof-Of-Concept zal 4 weken duren en als laatst zal het rapport met de optimale verbeteringen 2 weken duren.

A.0.5. Verwacht resultaat, conclusie

Het verwachte resultaat is dat door de implementatie van immutable storage en automatische back-ups, de back-upstrategie van Forvis Mazars zal worden verbeterd. Vooral de bescherming tegen ransomware en andere bedreigingen zal beter zijn door het gebruik van immutable storage, waarbij back-ups onveranderlijk worden opgeslagen en niet kunnen worden gemanipuleerd. Daarnaast zorgt de automatisering van de back-ups voor een efficiënter beheer, waarbij de manuele taken van het IT-team verminderd worden. Dit kan in de praktijk leiden tot meer consistente back-ups en een verbeterde betrouwbaarheid van het systeem. De resultaten zullen waarschijnlijk aantonen dat de combinatie van deze twee oplossingen zorgen voor een sterkere, efficiëntere en beter back-upstrategie.

B

Vagrantfile

```
Vagrant.configure("2") do |config|

  # Define the first VM - Primary
  config.vm.define "primary" do |primary|
    primary.vm.box = "ubuntu/jammy64"

    # Network configuration
    primary.vm.network "private_network", ip: "192.168.0.10", virtualbox__intnet:
        "internal_network"

    # Hardware resources
    primary.vm.provider "virtualbox" do |vb|
      vb.memory = "2048" # 2GB RAM
      vb.cpus = 1        # 1 CPU
    end
  end

  # Define the second VM - Backup
  config.vm.define "backup" do |backup|
    backup.vm.box = "ubuntu/jammy64"

    # Network configuration
    backup.vm.network "private_network", ip: "192.168.0.20", virtualbox__intnet: "
        internal_network"

    # Hardware resources
    backup.vm.provider "virtualbox" do |vb|
      vb.memory = "2048" # 2GB RAM
      vb.cpus = 1        # 1 CPU
    end
  end

  # Define the third VM - Attacker VM
```

```
  config.vm.define "attacker" do |attacker|
    attacker.vm.box = "ubuntu/jammy64"

    # Network configuration
    attacker.vm.network "private_network", ip: "192.168.0.30", virtualbox__intnet:
        "internal_network"

    # Hardware resources
    attacker.vm.provider "virtualbox" do |vb|
      vb.memory = "1024" # 1GB RAM
      vb.cpus = 1        # 1 CPU
    end
  end

end
```

C

Script voor het simuleren van een ransomware-aanval

Listing C.1: Bash script om een ransomware-aanval na te bootsen

```
#!/bin/bash

BACKUP_DIR="/home/vagrant/backups"

# Ensure the encryption key is set as an environment variable
if [ -z "$ENCRYPTION_KEY" ]; then
  echo "Error: ENCRYPTION_KEY environment variable is not set."
  exit 1
fi

for file in "$BACKUP_DIR"/*; do
  if [ -f "$file" ]; then
    # Encrypt the file using OpenSSL
    if openssl enc -aes-256-cbc -salt -in "$file" -out "${file}.enc" -pass pass:"
        $ENCRYPTION_KEY" -pb>      echo "Encrypted $file and saved as ${file}.enc"
      rm "$file"
      echo "Original file $file is deleted."
    else
      echo "Error: Could not encrypt $file"
    fi
  fi
done
```

D

Dockerfile

Listing D.1: Dockerfile gebruikt voor de images van de Kubernetes pods in de POC.

```
FROM ubuntu:latest

RUN apt update -y \
&& apt -y install mysql-client postgresql-client nano \
&& apt install -y python3

COPY src/* /

CMD ["tail", "-f", "/dev/null"]
```

Listing D.2: Dockerfile gebruikt voor het veranderen naar Kubernetes.

```
FROM ubuntu:latest

WORKDIR /src

USER root
RUN apt update -y \
    && apt -y install mysql-client postgresql-client cron nano \
    && apt install -y python3

COPY src/* /src/
COPY crontab.txt /etc/cron.d/my-cron-job
COPY src/setup.sh /src/setup.sh

RUN chmod 0644 /etc/cron.d/my-cron-job && \
    chmod +x /src/backup_script.py
RUN chmod +x /src/setup.sh

ENTRYPOINT ["tail"]
CMD ["-f", "/dev/null"]
```

E

Python-script

Listing E.1: Python-script voor back-ups en retentiebeleid.

```
import datetime
import os
import subprocess  # nosec
import argparse
import logging


class Backup:

    def __init__(self, database_name, database_user, type, backup_dir):
        timestr = datetime.datetime.now().strftime('%Y-%m-%d')
        self.filename = f'backup-{type}-{timestr}-{database_name}.dump'
        self.database_name = database_name
        self.database_user = database_user
        self.type = type
        self.password = None
        self.backup_dir = backup_dir
        self.hostname_mysql = os.environ.get("HOSTNAME_MYSQL", "192.168.1.53")
        self.hostname_psql = os.environ.get("HOSTNAME_PSQL", "192.168.1.53")
        self.port_mysql = os.environ.get("PORT_MYSQL", "3306")
        self.port_psql = os.environ.get("PORT_PSQL", "5432")

    def set_password(self):
        """
        Retrieve the database password from an environment variable.
        """
        self.password = os.environ.get("DB_PASSWORD")
        if not self.password:
            logging.warning("Database password not set. Please provide the
                DB_PASSWORD environment variable.")
            exit(1)
```

```
31
32      def create_backup(self, type):
33          """
34          Create a backup of the database.
35          """
36          # Ensure the backup directory exists
37          if not os.path.exists(self.backup_dir):
38              os.makedirs(self.backup_dir)
39
40          # Prepend the backup directory to the filename
41          full_path = os.path.join(self.backup_dir, self.filename)
42
43          if type == "MYSQL":
44              try:
45                  cmd = [
46                      'mysqldump',
47                      '--single-transaction',
48                      '-u', self.database_user,
49                      f'-p{self.password}',
50                      '-h', self.hostname_mysql,
51                      '-P', str(self.port_mysql),
52                      '--no-tablespaces',
53                      '-B', self.database_name,
54                  ]
55
56                  with open(full_path, 'w') as backup_file:
57                      result = subprocess.run(cmd, stdout=backup_file, check=True)
                            # nosec
58
59                  if result.returncode != 0:
60                      logging.warning(f'Command failed. Return code: {result.
                            returncode}')
61                      exit(1)
62
63                  return full_path
64
65              except Exception as e:
66                  logging.warning(e)
67                  exit(1)
68
69          elif type == "PSQL":
70              try:
71                  cmd = [
72                      'pg_dump',
73                      '-U', self.database_user,
74                      '-h', self.hostname_psql,
75                      '-p', str(self.port_psql),
76                      '-F', 'c',
77                      '-f', full_path,
78                      self.database_name
79                  ]
```

```
80
81                  result = subprocess.run(cmd, check=True)  # nosec
82
83                  if result.returncode != 0:
84                      logging.warning(f'Command failed. Return code: {result.
                            returncode}')
85                      exit(1)
86
87                  return full_path
88
89              except Exception as e:
90                  logging.warning(e)
91                  exit(1)
92
93      @staticmethod
94      def delete_old_backups(directory, retention_days):
95          """
96          Deletes backup files older than the specified retention period.
97          """
98          now = datetime.datetime.now()
99          logging.info(f"Starting cleanup of backups in {directory} with retention
                period: {retention_days} days")
100         for file in os.listdir(directory):
101             if file.startswith("backup-") and file.endswith(".dump"):
102                 file_path = os.path.join(directory, file)
103                 file_mtime = datetime.datetime.fromtimestamp(os.path.getmtime(
                        file_path))
104                 age = (now - file_mtime).days
105                 logging.info(f"Checking file {file_path}: age = {age} days")
106                 if age >= retention_days:
107                     try:
108                         os.remove(file_path)
109                         logging.info(f"Deleted old backup: {file_path}")
110                     except Exception as e:
111                         logging.warning(f"Failed to delete {file_path}: {e}")
112                 else:
113                     logging.info(f"File {file_path} is not old enough to delete (
                            age = {age} days).")
114
115
116 def main():
117     # Configure logging
118     logging.basicConfig(
119         level=logging.INFO,
120         format="%(asctime)s - %(levelname)s - %(message)s"
121     )
122
123     parser = argparse.ArgumentParser(description="Execute a local backup of a
            database.")
124     parser.add_argument("-dn", "--database_name", required=True, help="Enter the
            name of the database for the backup.")
```

```
125     parser.add_argument("-du", "--database_user", required=True, help="Enter the
            username of the database for the backup.")
126     parser.add_argument("-b", "--backup", action="store_true", help="Backup the
            database.")
127     parser.add_argument("-t", "--type", choices=["MYSQL", "PSQL"], required=True,
            help="MYSQL or PSQL")
128     parser.add_argument("-bd", "--backup_dir", default=".", help="Directory where
            backups are stored.")
129     parser.add_argument("-r", "--retention", type=int, help="Retention period in
            days for keeping backups.")
130     args = parser.parse_args()
131
132     db_name = args.database_name
133     db_user = args.database_user
134     backup = args.backup
135     type = args.type
136     backup_dir = args.backup_dir
137     retention = args.retention
138
139     if backup:
140         b = Backup(db_name, db_user, type, backup_dir)
141         try:
142             b.set_password()
143             backup_file = b.create_backup(type=type)
144         except Exception as e:
145             logging.warning(e)
146         else:
147             logging.info(f"Backup successful: {backup_file}")
148
149     if retention is not None:
150         try:
151             Backup.delete_old_backups(directory=backup_dir, retention_days=
                    retention)
152             logging.info(f"Old backups older than {retention} days were
                    successfully deleted from {backup_dir}.")
153         except Exception as e:
154             logging.warning(f"Failed to clean up old backups: {e}")
155
156
157 if __name__ == '__main__':
158     main()
```

F

Kubernetes Cronjobs

Listing F.1: YAML-bestand voor de configuratie van de Kubernetes CronJob die een back-up neemt van de MySQL databank.

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: backup-mysql-cronjob
spec:
  schedule: "0 2 * * *"
  jobTemplate:
    spec:
      ttlSecondsAfterFinished: 3600
      template:
        spec:
          containers:
          - name: backup-mysql-container
            image: bouzewazi/ubu4
            command:
              - "python3"
              - "/backup_script.py"
              - "-dn"
              - "testdb"
              - "-du"
              - "root"
              - "-t"
              - "MYSQL"
              - "-b"
              - "-bd"
              - "/data"
            volumeMounts:
              - name: backup-storage
                mountPath: /data
            env:
```

```
              - name: DB_PASSWORD
                valueFrom:
                  secretKeyRef:
                    name: db-secret
                    key: DB_PASSWORD

            resources:
              requests:
                memory: "1Gi"
                cpu: "250m"
              limits:
                memory: "2Gi"
                cpu: "500m"
          restartPolicy: OnFailure
          volumes:
          - name: backup-storage
            persistentVolumeClaim:
              claimName: backup-pvc
```

Listing F.2: YAML-bestand voor de configuratie van de Kubernetes CronJob die een back-up neemt van de PostgreSQL databank.

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: backup-psql-cronjob
spec:
  schedule: "0 2 * * *"
  jobTemplate:
    spec:
      ttlSecondsAfterFinished: 3600
      template:
        spec:
          containers:
          - name: backup-mysql-container
            image: bouzewazi/ubu4
            command:
              - "python3"
              - "/backup_script.py"
              - "-dn"
              - "testdb"
              - "-du"
              - "root"
              - "-t"
              - "PSQL"
              - "-b"
              - "-bd"
              - "/data"
            volumeMounts:
              - name: backup-storage
                mountPath: /data
            env:
```

```
              - name: DB_PASSWORD
                valueFrom:
                  secretKeyRef:
                    name: db-secret
                    key: DB_PASSWORD
              - name: PGPASSWORD
                valueFrom:
                  secretKeyRef:
                    name: db-secret2
                    key: PGPASSWORD

            resources:
              requests:
                memory: "1Gi"
                cpu: "250m"
              limits:
                memory: "2Gi"
                cpu: "500m"
          restartPolicy: OnFailure
          volumes:
          - name: backup-storage
            persistentVolumeClaim:
              claimName: backup-pvc
```

Listing F.3: YAML-bestand voor de configuratie van de Kubernetes CronJob die oude back-ups verwijdert volgens het retentiebeleid.

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: delete-cronjob
spec:
  schedule: "0 2 * * *"
  jobTemplate:
    spec:
      ttlSecondsAfterFinished: 3600
      template:
        spec:
          containers:
          - name: backup-mysql-container
            image: bouzewazi/ubu4
            command:
              - "python3"
              - "/backup_script.py"
              - "-dn"
              - "testdb"
              - "-du"
              - "root"
              - "-t"
              - "MYSQL"
              - "-r"
              - "14"
```

```
            - "-bd"
            - "/data"
          volumeMounts:
            - name: backup-storage
              mountPath: /data
          env:
            - name: DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: db-secret
                  key: DB_PASSWORD

          resources:
            requests:
              memory: "1Gi"
              cpu: "250m"
            limits:
              memory: "2Gi"
              cpu: "500m"
        restartPolicy: OnFailure
        volumes:
        - name: backup-storage
          persistentVolumeClaim:
            claimName: backup-pvc
```

G

Persistent Volume (PV) voor back-upopslag en bijbehorende Persistent Volume Claim (PVC) in de POC

Listing G.1: YAML-bestand voor de configuratie van het Persistent Volume (PV) dat opslag biedt voor de databaseback-ups in Kubernetes.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: backup-pv
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: ""
  hostPath:
    path: "/data"
```

Listing G.2: YAML-bestand voor de configuratie van de Persistent Volume Claim (PVC).

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: backup-pvc
spec:
  accessModes:
```

```
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: ""
```

H

Setup script voor Docker-container

Listing H.1: Bash-script om de cronjobs op te zetten.

```
#!/bin/bash
service cron start

mkdir -p /src/backups

# Voeg cronjobs toe
crontab - <<EOF
* * * * * DB_PASSWORD=root PGPASSWORD=root python3 /src/backup_script.py -dn
    testdb -du root -t MYSQL -b -bd /src/backups >> /var/log/cron.log 2>&1
* * * * * DB_PASSWORD=root PGPASSWORD=root python3 /src/backup_script.py -dn
    testdb -du root -t PSQL -b -bd /src/backups >> /var/log/cron.log 2>&1
0 2 * * * DB_PASSWORD=root PGPASSWORD=root python3 /src/backup_script.py -bd /src/
    backups -r 14 -dn testdb -du root -t MYSQL >> /var/log/cron.log 2>&1
EOF

exec "$@"
```

Bibliografie

Ali, A., Laghari, A. A., Kandhro, I. A., Kumar, K., & Younus, S. (2024). Systematic analysis of on-premise and cloud services. *International Journal of Cloud Computing*, *13*(3), 214–242. https://doi.org/https://doi.org/10.1504/IJCC.2024.139604

Beard, B. (2018). *Full Backups*. Apress. https://doi.org/https://doi.org/10.1007/978-1-4842-3456-3_1

BorgBackup. (2024, november 18). *Borg Documentation*. Verkregen december 14, 2024, van https://borgbackup.readthedocs.io/en/latest/

Bryant, W. D. (2015, juli 30). *International Conflict and Cyberspace Superiority*. https://books.google.be/books?id=LJ9GCgAAQBAJ&q=%22air+gapped%22&pg=PA107&redir_esc=y#v=onepage&q&f=false

Cabral, S. K., & Murphy, K. (2011, maart 4). *MySQL Administrator's Bible*. https://www.google.be/books/edition/MySQL_Administrator_s_Bible/PqZ6QytCemcC?hl=nl&gbpv=0

Cherry, D. (2015). Database Backup Security, 293–311. https://doi.org/https://doi.org/10.1016/B978-0-12-801275-8.00010-5

Chervenak, A., Vellanki, V., & Kurmas, Z. (1998). Protecting file systems: A survey of backup techniques. *Joint NASA and IEEE Mass Storage Conference*, *99*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4b6cfd832c2eb61c60ae0ab956e

Dikbiyik, F., Gul, F., Tapkan, G., Han, Y., Akdora, O., Budakoglu, G., Ciftci, B., Celik, E. S., Cengiz, S. E., Toprak, G., & Dogan, Y. (2024). State of Ransomware 2024:A Year of Surgesand Shuffling. *Black kite*. https://blackkite.com/wp-content/uploads/2024/05/BlackKite_Report_Ransomware-2024.05.14.pdf

Drake, J. D., & Worsley, J. C. (2002, januari 7). *Practical PostgreSQL*. https://www.google.be/books/edition/Practical_PostgreSQL/fI1lAgAAQBAJ?hl=nl&gbpv=0

Dubey, A., Jewell, P., Estabrook, N., Haas, S., Myers, T., Martin, J., Callison, D., Fernández, J. Á., Coulter, D., Lee, D., Rabeler, C., Anderson, B., Fowler, C., Kohli, A., Hopkins, M., Sharkey, K., Kumar, R., Irwin, J., Shahan, R., & Pratt, T. (2023). Introduction to Azure Blob Storage. *Microsoft*. https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction

Edwards, B. (2022). Why You Need an Offline Backup. *How-To Geek*. https://www.howtogeek.com/818193/why-you-need-an-offline-backup/

Ekuan, M., Buck, A., Zimmergren, T., Moore, G., Parker, D., & Coulter, D. (2023). Hoe werkt Azure? *Microsoft*. https://learn.microsoft.com/nl-nl/azure/cloud-adoption-framework/get-started/what-is-azure

Erickson, J. (2024). MySQL: Understanding What It Is and How It's Used. *Oracle*. https://www.oracle.com/be/mysql/what-is-mysql/

Estabrook, N., Nottingham, C., Pavan, P., Singh, A., Martin, J., Yoshioka, H., & Myers, T. (2024). Store business-critical blob data with immutable storage in a write once, read many (WORM) state. *Microsoft*. https://learn.microsoft.com/en-us/azure/storage/blobs/immutable-storage-overview

Ghazi, K., & H. O. Nasereddin, H. (2013). Business Continuity Based on Backup. *American Academic Scholarly Research Journal*, *5*, 253–258. https://www.aasrc.org/aasrj/index.php/aasrj/article/viewFile/1385/547

Gupta, A., & Hohn, A. (2019). Getting Started with Kubernetes. *Dzone*, *24*, 2019. https://dzone.com/storage/attachments/14131598-dzone-kubernetes-bundle.pdf

Hasan, R., Stanton, P., Yurcik, W., Brumbaugh, L., Rosendale, J., & Boonstra, R. (2005). The Techniques and Challenges of Immutable Storage with applications in Multimedia. *National Center for Supercomputing Applications*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=578ff4957d4fa2e550ec2a819b6500820d72

Hashicorp. (z.d.). *What is Vagrant?* Verkregen december 10, 2024, van https://developer.hashicorp.com/vagrant

James. (2019). Offline backups in an online world. *National Cyber Security Centre*. https://www.ncsc.gov.uk/blog-post/offline-backups-in-an-online-world

Kemp, K. (2007, december 26). *Encyclopedia of Geographic Information Science*. https://www.google.be/books/edition/Encyclopedia_of_Geographic_Information_S/FrUQHIzXK6EC?hl=nl&gbpv=0

Miao, G., Zander, J., Sung, K. W., & Slimane, S. B. (2016, maart 3). *Fundamentals of Mobile Data Networks*. https://books.google.be/books?id=ImeSCwAAQBAJ&printsec=frontcover&source=gbs_atb&redir_esc=y#v=onepage&q&f=false

MySQL. (z.d.). *MySQL 8.4 Reference Manual: MySQL Glossary*. Verkregen december 15, 2024, van https://dev.mysql.com/doc/refman/8.4/en/glossary.html

Nelson, S., & Brown, R. (2011, februari 23). *Pro Data Backup and Recovery*. https://www.google.be/books/edition/Pro_Data_Backup_and_Recovery/0lfehRoBOPkC?hl=nl&gbpv=0

Obrutsky, S. (2016). Cloud storage: Advantages, disadvantages and enterprise solutions for business. *Conference: EIT New Zealand*. https://www.researchgate.net/profile/Santiago-Obrutsky/publication/305508410_Cloud_Storage_Advantages_Disadvantages_and_Enterprise_Solutions_for_Business/links/5792976508ae33e89f7cc136/Cloud-Storage-Advantages-Disadvantages-and-Enterprise-Solutions-for-Business.pdf

Oracle. (2024, november 1). *User Guide for Release 7.1*. Verkregen december 11, 2024, van https://docs.oracle.com/en/virtualization/virtualbox/7.1/user/preface.html

Park, J., Yoo, J., Yu, J., Lee, J., & Song, J. (2023). A Survey on Air-Gap Attacks: Fundamentals, Transport Means, Attack Scenarios and Challenges. *Sensors*, *23*(6), 3215. https://doi.org/https://doi.org/10.3390/s23063215

Qian, C., Huang, Y., Zhao, X., & Nakagawa, T. (2010). Optimal Backup Interval for a Database System with Full and Periodic Incremental Backup. *Journal of Computers*, *5*(4), 557–564. https://doi.org/10.4304/jcp.5.4.557-564

Rahumed, A., Chen, H. C. H., Tang, Y., Lee, P. P. C., & Lui, J. C. S. (2011). A secure cloud backup system with assured deletion and version control. *40th International Conference on Parallel Processing Workshops*, 160–167. https://www.researchgate.net/publication/221617563_A_Secure_Cloud_Backup_System_with_Assured_Deletion_and_Version_Control

Richardson, R., & North, M. (2017). Ransomware: Evolution, mitigation and prevention. *International Management Review*, *13*(1), 10. https://digitalcommons.kennesaw.edu/cgi/viewcontent.cgi?article=5312&context=facpubs

Rivas, K. (2022). What's the Diff: Full, Incremental, Differential, and Synthetic Full Backups. *Backblaze*. https://www.backblaze.com/blog/whats-the-diff-full-incremental-differential-and-synthetic-full-backups/

Susnjara, S., & Smalley, I. (2024). What are hypervisors? *IBM*. https://www.ibm.com/topics/hypervisors

Trovato, F., Sharp, A., & Siman, T. (2019). Cloud, co-location, on-premises and hybrid disaster recovery solutions: Pros, cons, and a cost comparison. *Journal of Business Continuity & Emergency Planning*, *13*(2), 120–135. https://www.ingentaconnect.com/content/hsp/jbcep/2019/00000013/00000002/art00004

Wahl, C. (2023). Recovering Fast from Ransomware Attacks: The Magic of an ImmutableBackup Architecture. *Rubrik*. https://www.rubrik.com/content/dam/rubrik/en/resources/white-paper/rwp-recovering-fast-from-ransomware-attacks.pdf?ref=thestack.technology

Yanfang, Y., Tao, L., Donald, A., & Sitharama, I. (2017). A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)*, *50*(3), 1–40. https://dl.acm.org/doi/pdf/10.1145/3073559

Zhao, X., Bu, Y., Pang, W., & Cai, J. (2024). Periodic and random incremental backup policies in reliability theory. *Software Quality Journal*, *32*(3), 1325–1340. https://doi.org/https://doi.org/10.1007/s11219-024-09685-1

Zhu, W.-D., Allenbach, G., Battaglia, R., Boudreaux, J., Harnick-Shapiro, D., Kim, H., Kreuch, B., Morgan, T., Patel, S., & Willingham, M. (2015, april 13). *Disaster Recovery and Backup Solutions for IBM FileNet P8 Version 4.5.1 Systems*. IBM Redbooks. https://books.google.be/books?id=O1TAAgAAQBAJ