Automatische email rapportages met HTML en Qlik Automate

De achtergrond

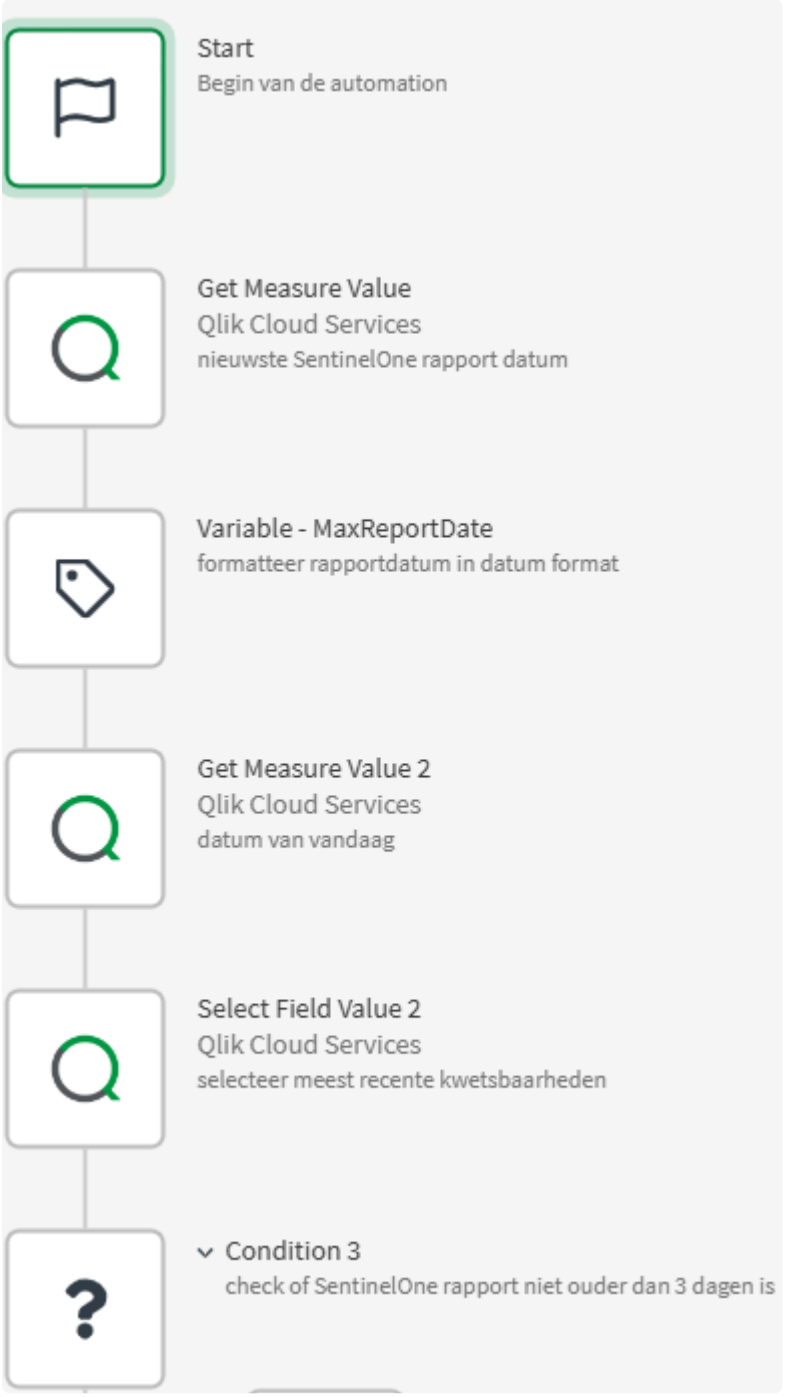
Bij HippoLine wordt SentinelOne gebruikt om bekende kwetsbaarheden in software te detecteren, zodat medewerkers weten welke programma's geupdatet moeten worden om cyberaanvallen en datalekken te voorkomen. Deze oplossing biedt echter geen manier om medewerkers te alerteren; een medewerker moet wekelijks handmatig een uitdraaisel maken, uitsplitsen per collega, en mailen of ze kunnen updaten. Dit tijdrovende proces kan efficiënter!

De ingebouwde Qlik Reporting connector binnen Qlik Automations lijkt hiervoor op het eerste gezicht een logische keuze. Maar de mogelijkheden zijn beperkt: het is niet mogelijk om de opmaak of indeling te veranderen, en tabellen worden alof het screenshots zijn in het rapport geplakt. Als de tabel meer rijen bevat dan er op 1 pagina passen, worden deze zonder pardon afgesneden.
Door mijn affiniteit met websites bouwen ontstond het idee om een HTML rapport te maken. Qlik biedt HTML template functionaliteit, maar ook hier hangen beperkingen aan; er is geen mogelijkheid om conditionele logica in te bouwen, of om op een vast schema rapporten te versturen. Qlik Automation is een krachtige tool om data uit Qlik apps te verwerken, maar de visualisatieopties binnen autmations voor rapporten schieten tekort. We moesten dus met onze eigen oplossing komen.

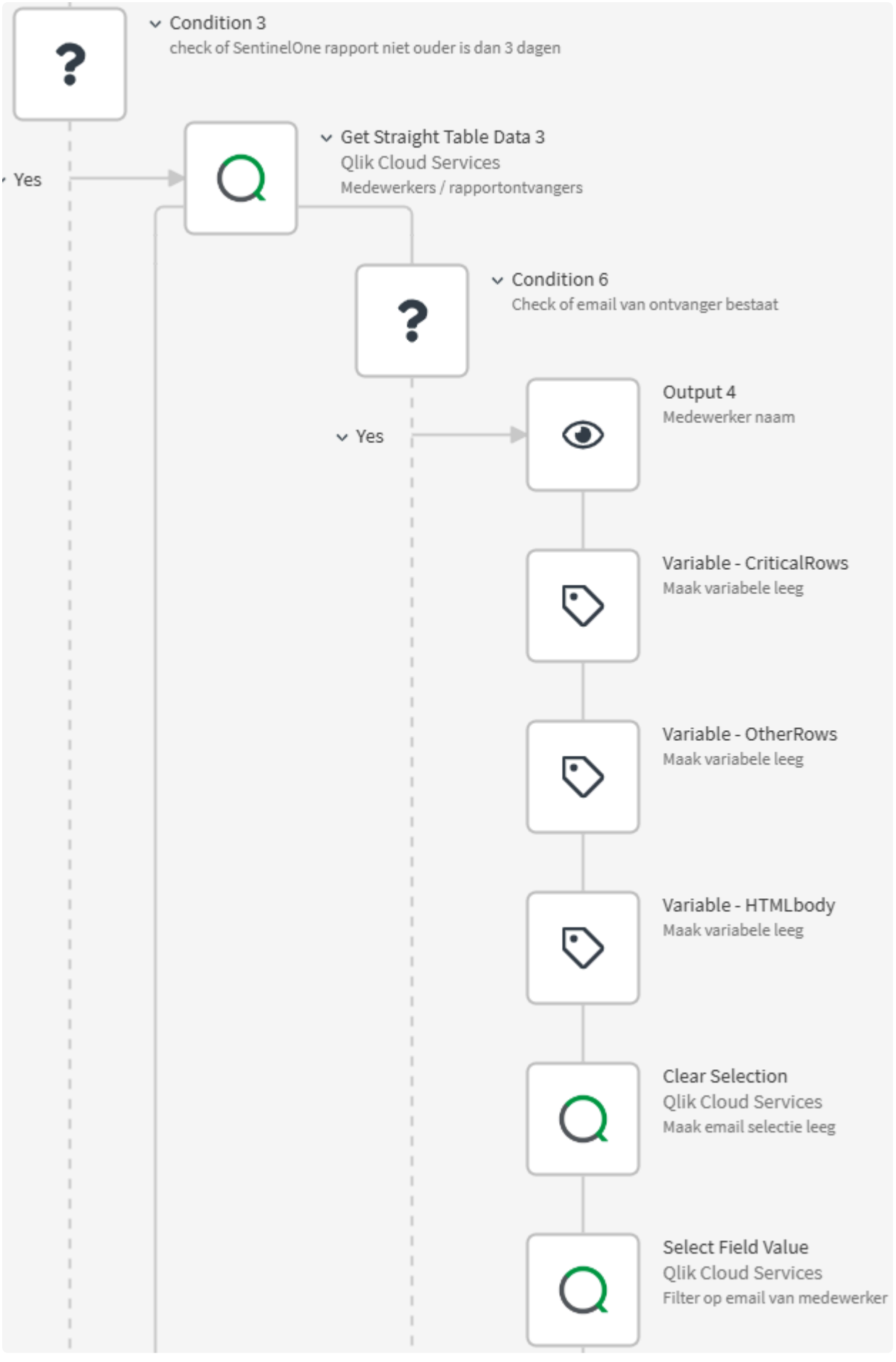
Wil je meer te weten komen over Qlik Automations, en hoe je zelf je eigen HTML rapporten kan samenstellen, precies zoals jij het wil? Lees dan vooral verder!

De implementatie

Automations werken als een soort flowchart, waarbij data opgehaald kan worden door blokjes, en de output hiervan kan weer gebruikt worden door blokjes later in de automation. We beginnen met wat voorbereidend werk; De datum van vandaag & het meest recente SentinelOne rapport worden opgehaald en vergeleken, om te zorgen dat er niet meerdere mailtjes gestuurd worden met dezelfde data.



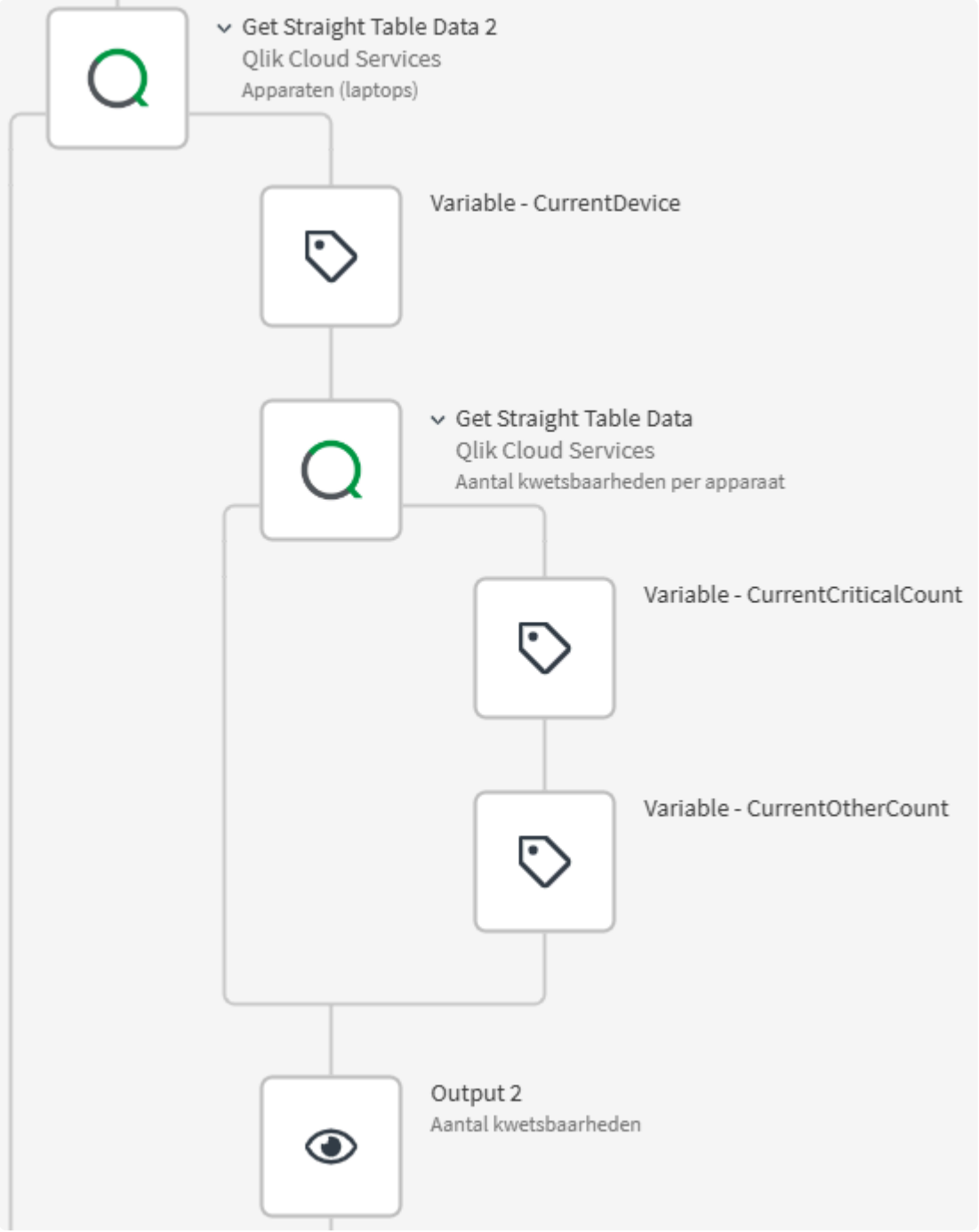
Het fijne aan automations is dat je op elk moment selecties kan maken op je data, alsof je in het Qlik dashboard zelf zit. Je hoeft je datamodel dus niet op 2 losse plekken samen te stellen; automation en dashboard werken samen in harmonie.



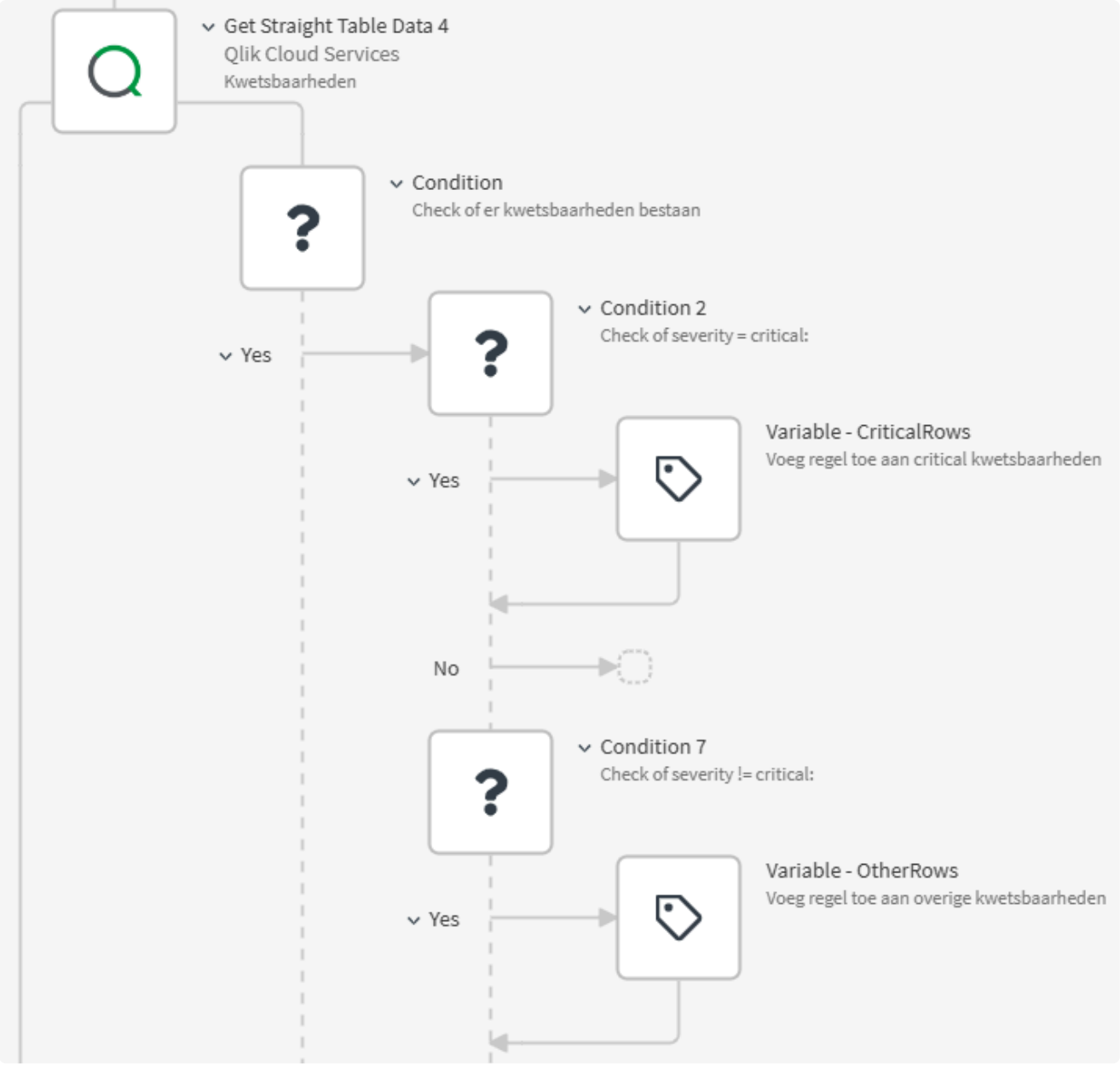
Hier begint de loop, waardoor een apart rapport naar elke medewerker gestuurd kan worden. Aan het begin van elke iteratie, worden alle gebruikte variabelen leeggemaakt. Wie bekend is met programmeertalen, zal niet verbaasd zijn waarom dit nodig is. Omdat rijen

met kwetsbaarheden aan de variabele worden 'geplakt', zou anders elke medewerker zijn eigen kwetsbaarheden ontvangen, plus alle kwetsbaarheden van de voorafgaande medewerkers.

De output blokjes printen bepaalde informatie naar het output scherm van de automation. Deze zijn niet strikt gezien nodig, maar helpen tijdens ontwikkeling enorm om te controleren of je automation doet wat je verwacht.



Sommige medewerkers hebben meerdere laptops, omdat ze bijvoorbeeld recentelijk zijn overgestapt. Dus de kwetsbaarheden worden uitgesplitst per apparaat. De counts worden gebruikt om bovenaan het rapport in 1 oogopslag te tonen hoeveel applicaties kwetsbaar zijn.



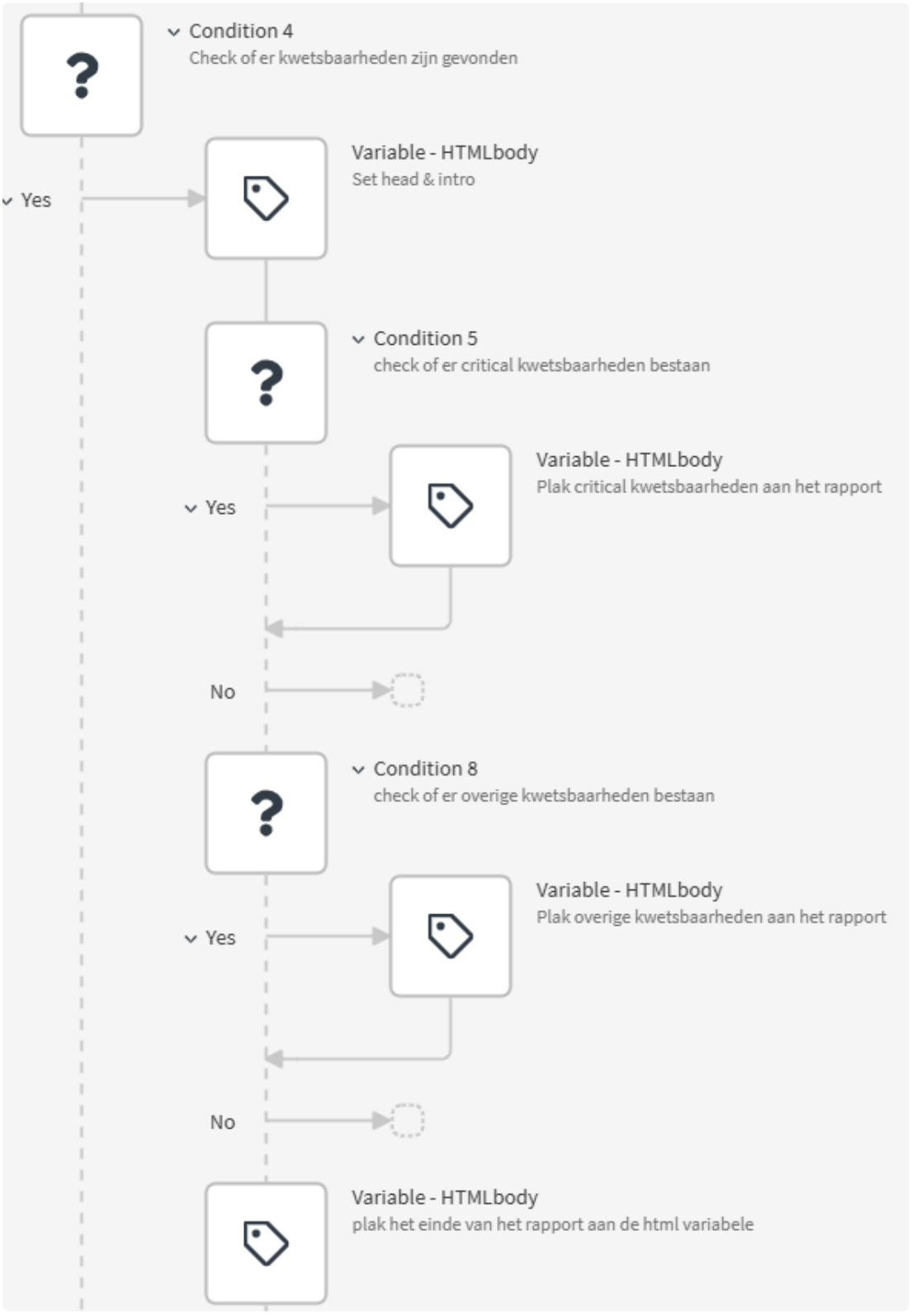
Na deze loop komt de daadwerkelijke loop over alle kwetsbaarheden. SentinelOne geeft severity scores aan alle kwetsbaarheden, en critical severity dient zo snel mogelijk aangepakt te worden. Deze kwetsbaarheden worden in een aparte variabele bijgehouden. Een rij is niks meer dan een HTML `<tr>` element, om later in een tabel te kunnen zetten. Deze rij ziet er als volgt uit:

```
{if: { Kwetsbaarheden.InPreviousReport } = 1,
    <tr class="tr-critical recurring">, <tr class="tr-critical">
}
    <td class="software">
        {if: { Kwetsbaarheden.InPreviousReport } = 1,
            <span class="recurring-badge">!</span>}
            {Kwetsbaarheden.Software}
    </td>
    <td class="versie">{Kwetsbaarheden.Versie}</td>
    <td class="severity">{Kwetsbaarheden.Severity}</td>
```

```
    <td class="beschrijving">{Kwetsbaarheden.Beschrijving}</td>
</tr>
```

Aan blokken in een automation kan een comment toegevoegd worden, maar de naam aanpassen is niet mogelijk. Voor de leesbaarheid heb ik daarom hierboven `$.GetStraightTableData4.item` vervangen met `Kwetsbaarheden`.

`InPreviousReport` is een veld in het dashboard waar de data vandaan komt, die bijhoudt of een bepaalde kwetsbaarheid bij het vorige rapport ook al bestond. Deze hardnekkige meldingen vereisen extra aandacht, en krijgen dus een andere achtergrondkleur, samen met een uitroepteken.



We zijn er bijna! Hier wordt de variabele opgebouwd die uiteindelijk de inhoud van de email gaat vormen. Aan het einde van dit proces wordt staat er dus een volwaardig html bestand in die variabele. De kop hiervan ziet er als volgt uit:

```
<html>
  <head>
    <style>
      { Hier komt alle styling die nodig is voor het rapport }
    </style>
  </head>
```

```
  <body>
  <div class="email-wrapper">
    <div class="email-header">
      <h2>Vulnerability rapport SentinelOne</h2>
    </div>

    <div class="email-content">
      <div class="summary-cards">
        <div class="summary-card">
          <h3 class="summary-title">{$.CurrentDevice}</h3>
          <p class="summary-values">
            Voor het laatst gecontroleerd op: <br /> { $.CheckedAtDate }
          </p>
        </div>

        <div class="summary-card">
          <h3 class="summary-title">Aantal gedetecteerde kwetsbaarheden</h3>
          <p class="summary-values">
            Critical: {$.CurrentCriticalCount} | Overig:
{$.CurrentOtherCount}
          </p>
        </div>

        <div class="summary-card">
          <h3 class="summary-title">Terugkomende kwetsbaarheden</h3>
          <p class="summary-values">
            Alle kwetsbaarheden met een "!" ervoor kwamen ook voor in het
vorige rapport.
            <br />
            Zorg ervoor dat deze er volgende keer niet meer tussen staan!
          </p>
        </div>
      </div>
```

De rijen lijken qua opmaak erg op elkaar, dus hier is de logica voor de kritieke kwetsbaarheden:

```
{if: { textlength: { $.CurrentCriticalCount } } > 0,
    <div class="section-callout callout-critical">
        <h3 class="heading heading-critical"> Kritieke kwetsbaarheden </h3>
        <p class="subheading">
            Loop alle software uit deze lijst na en update ze naar de meest
```

```
recente (stabiele) versie.
        </p>
    </div>
    <table>
        <thead>
            <tr class="header-critical">
                <th>Software</th>
                <th>Versie</th>
                <th>Score</th>
                <th>Omschrijving</th>
            </tr>
        </thead>
        <tbody> {$.CriticalRows} </tbody>
    </table> }
```

HTML elementen moeten altijd ook weer afgesloten worden, dus ongeacht of er kwetsbaarheden van beide categoriën zijn of niet, komt dit er aan het einde van de rit bij:

```
      <div class="email-footer">
        Dit rapport is automatisch gegenereerd. Reageren op deze e-mail is
niet nodig.
      </div>
    </div>
  </body>
</html>
```

Het resultaat

En dat is het! Hierna versturen we de email en begint de loop weer vanaf het begin bij de volgende ontvanger. Dit is de mail die verstuurd wordt:

Vulnerability rapport SentinelOne

APPLEBOOK PRO
Voor het laatst gecontroleerd op: 05-03-2025

TOTAAL GEDETECTEERDE KWETSBAARHEDEN
Critical: 4 | Overig: 3

TERUGKOMENDE KWETSBAARHEDEN
Alle kwetsbaarheden met een "!" ervoor kwamen ook voor in het vorige rapport. Zorg ervoor dat deze er volgende keer niet meer tussen staan!

Kritieke kwetsbaarheden

Loop alle software uit deze lijst na en update ze naar de meest recente (stabiele) versie.

SOFTWARE	VERSIE	SCORE	OMSCHRIJVING
! **Microsoft Windows Desktop Runtime**	`8.0.11.34221`	**9,9**	Inconsistent interpretation of http requests ('http request/response smuggling') in ASP.NET Core allows an authorized attacker to bypass a security feature over a network.
Google Chrome	`143.0.7499.194`	**9,8**	Incorrect security UI in Google Chrome on Android prior to 144.0.7559.59 allowed a remote attacker to spoof the contents of the Omnibox (URL bar) via a crafted HTML page
Google Chrome	`143.0.7499.194`	**9,8**	Incorrect security UI in Google Chrome on Android prior to 144.0.7559.59 allowed a remote attacker to spoof the contents of the Omnibox (URL bar) via a crafted HTML page
File Pilot	`0.6.8`	**9,3**	Integer underflow in the file_printf function in the "file" program before 4.20 allows user-assisted attackers to execute arbitrary code via a file that triggers a heap-based buffer overflow.

Overige kwetsbaarheden (Hoog, Middel, Laag, Onbekend)

Deze kwetsbaarheden dienen ook nagelopen te worden, maar zijn vaak van minder belang dan de kritieke kwetsbaarheden.

ERNST	SOFTWARE	VERSIE	SCORE	OMSCHRIJVING
! **High**	**Python**	`3.14`	**7,8**	A vulnerability has been found in the CPython `venv` module and CLI where path names provided when creating a virtual environment were not quoted properly, allowing the creator to inject commands into virtual environment "activation" scripts (ie "source venv/bin/activate")
High	Microsoft Windows Desktop Runtime	`10.0.1.50000`	**7,5**	Improper handling of missing special element in .NET allows an unauthorized attacker to perform spoofing over a network.

Dit rapport is automatisch gegenereerd. Reageren op deze e-mail is niet nodig.