

UNIVERSIDADE DE LISBOA  
FACULDADE DE CIÊNCIAS  
DEPARTAMENTO DE INFORMÁTICA



## **Autonomous Incident Response**

Juan Christian da Silva Siqueira

**Mestrado em Segurança Informática**

Trabalho de projeto orientado por:  
Professor Doutor Alan Oliveira de Sá

2022



## **Acknowledgments**

This work would not be a reality without the people mentioned here, so I would like to sincerely thank:

Professor Alan Oliveira de Sá for his guidance and availability. His suggestions were essential for the development of this work.

The OutSystems' Security Office team, in special the Chief Information Security Officer José Casinha and the managers Igor Antunes and Luis Paulino for their trust and support during the conception and implementation of this work.

Finally, and most importantly, my parents and sister for their encouragement, support and patience over these two years of intensive dedication.



*To my parents, my sister, and my grandfather.*



## Resumo

A segurança da informação é essencial para qualquer organização que almeja permanecer relevante e crescer. Ela desempenha um papel importante como facilitador de negócios, seja do ponto de vista regulatório ou uma perspectiva de reputação. Ter pessoas, processos e tecnologia para resolver o número cada vez maior de incidentes de segurança o mais rápido possível e com o mínimo de impacto é um desafio para pequenas e grandes organizações.

Uma pesquisa recente realizada pelo Ponemon Institute e patrocinada pela IBM estudou 550 organizações impactadas por violações de dados entre março de 2021 e março de 2022 e concluiu que 83% delas tiveram uma ou mais violações e que 60% dessas violações levaram a aumentos nos preços dos serviços/produtos vendidos por tais organizações [39]. O *2022 Data Breach Investigations Report* da Verizon também confirma o estado alarmante da segurança da informação das organizações. Eles analisaram 23.896 incidentes, 5.212 dos quais culminaram em violações de dados em 2021 [70].

A pesquisa do Ponemon Institute revelou também que apenas 63% das organizações analisadas tinham um *Security Operations Center (SOC)* e processos de *Incident Response (IR)* bem definidos e testados regularmente, tais organizações tiveram um custo médio de violação de dados de \$3,26 milhões, enquanto que as organizações sem um SOC tiveram um custo médio de \$5,92 milhões. É impossível prevenir todos os incidentes de segurança, portanto quando eles acontecem as organizações querem resolvê-los rapidamente e conter os impactos. Esta é uma das principais razões pelas quais as organizações investem em um SOC.

Para enfrentar esse desafio da segurança informática, as empresas começaram a investir em *Security Orchestration, Automation, and Response (SOAR)* [39, 68, 70]. A orquestração de segurança é o planejamento, integração, cooperação e coordenação das atividades de ferramentas e especialistas de segurança para produzir e automatizar as ações necessárias em resposta a qualquer incidente de segurança em vários paradigmas tecnológicos [40]. Em outras palavras, o uso do SOAR é uma forma de traduzir os procedimentos manuais seguidos pelos analistas de segurança em ações automatizadas, tornando o processo mais rápido e escalável e economizando no orçamento de recursos humanos.

Tal desafio é encontrado na OutSystems, uma empresa multinacional de larga escala que atua no mercado *low-code*. Resumidamente, este mercado é composto por empresas que vendem plataformas que permitem que os desenvolvedores criem aplicações *web* e *mobile* por meio de interfaces gráficas, em vez de métodos de desenvolvimento tradicionais utilizando linguagens de programação como Java, Python, PHP, etc.

A OutSystems tem milhares de clientes em 87 países e seu principal produto é a Plataforma OutSystems, uma solução *low-code* que gera código *back-end* e *front-end* a partir de modelos visuais em uma experiência *drag-and-drop*.

A Plataforma OutSystems é um *Platform-as-a-Service (PaaS)*, a empresa cuida da hospedagem, rede, armazenamento, *runtime*, etc. para que o cliente possa se concentrar no desenvolvimento de aplicações e sites usando *low-code*. Os recursos (computação, armazenamento, rede, etc.) são hospedados na Amazon Web Services (AWS) seguindo as melhores práticas de segurança e soberania de dados.

Plataformas e empresas *low-code* estão enfrentando um maior escrutínio à medida que os clientes se preocupam cada vez mais com a segurança de seus dados e os dados de seus clientes subsequentes. O relatório da Verizon [70] mostrou que o comprometimento da cadeia de suprimentos de tecnologia (ex.: *software* e *hardware*) foi responsável por 62% dos incidentes de intrusão do sistema em 2021.

À medida que a OutSystems cresce, contrata colaboradores, adquire clientes, faz parcerias e, conseqüentemente, expande sua pegada digital e superfície de ataque, o SOC deve acompanhar o ritmo e garantir a segurança dos ativos atuais e novos. Infelizmente, a equipa do SOC precisa executar uma quantidade considerável de tarefas de segurança mundanas e manuais, o que não é o melhor uso do tempo do analista e também é um risco para a eficácia e escalabilidade do SOC.

O SOC da OutSystems é composto por uma equipa multicultural que trabalha em vários fusos horários utilizando uma abordagem *Follow the Sun (FTS)*, garantindo total disponibilidade para resolver incidentes de segurança. O SOC é responsável por fazer a resposta a incidentes para os ativos internos da empresa e também para todos os ativos de PaaS dos clientes a nível global.

É economicamente e operacionalmente inviável resolver o problema da escalabilidade simplesmente contratando mais analistas. Conforme evidenciado por diversos materiais acadêmicos [36, 40, 51, 57], é preciso investir em orquestração e automação como forma de aumentar a capacidade de segurança da organização de forma sustentável.

Neste sentido, a OutSystems passou a investir na implementação de SOAR. O projeto iniciou-se com a compreensão do contexto atual da empresa, seguindo-se a definição do problema que teve em consideração o referido contexto, os desafios enfrentados pelo SOC e os requisitos do projeto.

Para definir os requisitos funcionais e não funcionais do sistema, as necessidades e expectativas típicas da equipa do SOC foram identificadas levando em consideração o



conhecimento da literatura disponível, pesquisas de mercado e a experiência profissional dos membros do SOC da OutSystems.

Várias abordagens foram analisadas, nomeadamente: comprar um SOAR *Commercial off-the-shelf* (COTS), adaptar o Apache Airflow como um SOAR, adaptar o Luigi como um SOAR, criar um SOAR usando a Plataforma OutSystems, criar um SOAR usando Kubernetes, usar serviços em nuvem para criar um SOAR. No fim, a abordagem mais alinhada aos requisitos do SOC foi usar serviços em nuvem para criar um SOAR.

Este projeto propõe um SOAR *serverless*, seguro, rápido, escalável, e económico. A arquitetura do sistema inclui muitas partes, como o repositório de código no GitHub, o *Security Information and Event Management* (SIEM) Splunk, o software de comunicação utilizado pela OutSystems e vários serviços de nuvem da AWS como o *AWS Simple Notification Service* (SNS), *Lambda*, *Secrets Manager*, *Step Functions*, *CodePipeline*, e *CloudFormation*.

O SOC decidiu os 11 casos de uso a serem automatizados com base em múltiplos fatores: viabilidade de automatização, número de ocorrências/mês, complexidade do *playbook*, complexidade do incidente e número de ações manuais.

O sistema foi avaliado em um ambiente de produção com incidentes reais. Durante a avaliação do sistema, o SOC teve um total de 364 incidentes de segurança correspondentes aos casos de uso. O caso de uso *Indicators of Compromise* (IoC) está relacionado a um processo de obtenção de inteligência de ameaças que executou mais de 9 mil vezes.

Antes do SOAR, o analista levava 75,84 horas para executar as tarefas manuais relacionadas aos 11 casos de uso definidos. Além disso, estima-se que 450 horas do seu tempo seriam usadas para executar o caso de uso *Update threat intelligence database*.

Sobre o caso de uso IoC, não é possível executá-lo manualmente na cadência que um sistema automatizado faz, então o analista simplesmente atualizaria a base de dados uma vez por dia através de importação em massa.

Depois do SOAR, as mesmas tarefas foram executadas automaticamente em 31,2 minutos e as 9.000 execuções do *Update threat intelligence database* em 5,3 horas.

É importante observar que não foi possível automatizar todas as ações em todos os casos de uso, pois algumas delas exigem pensamento crítico, são muito complexas para automatizar ou o sistema não fornece uma *Application Programming Interface* (API). Isso é esperado e reforça o que foi observado em alguns materiais académicos [28, 51], o SOAR não é uma bala de prata e não se deve tentar automatizar todos os processos de forma descontrolada.

Neste projeto propusemos um sistema SOAR nativo em nuvem de baixo custo, apresentando os detalhes subjacentes de seu *design*. O desempenho da solução proposta foi avaliado em um ambiente real e de produção realizada em uma grande empresa multinacional, a OutSystems. Os resultados mostram que a solução foi capaz de diminuir a duração das tarefas em uma média de 98,81% (99,14% se considerarmos apenas os casos

de uso relacionados a Incident Response (IR) e desconsiderar as métricas do caso de uso IoC) enquanto possui uma despesa operacional inferior a \$65/mês.

**Palavras-chave:** SOAR, Computação em nuvem, Segurança, Orquestração, Automação



# Abstract

Information security is a must-have for any organization willing to stay relevant and grow, it plays an important role as a business enabler, be it from a regulatory perspective or a reputation perspective. Having people, process, and technology to solve the ever growing number of security incidents as fast as possible and with the least amount of impact is a challenge for small and big companies.

To address this challenge, companies started investing in Security Orchestration, Automation, and Response (SOAR) [39, 68, 70]. Security orchestration is the planning, integration, cooperation, and coordination of the activities of security tools and experts to produce and automate required actions in response to any security incident across multiple technology paradigms [40]. In other words, the use of SOAR is a way to translate the manual procedures followed by the security analysts into automated actions, making the process faster and scalable while saving on human resources budget.

This project proposes a low-cost cloud native SOAR platform that is based on serverless computing, presenting the underlying details of its design. The performance of the proposed solution was evaluated through 364 real-world incidents related to 11 use cases in a large multinational enterprise. The results show that the solution is able to decrease the duration of the tasks by an average of 98.81% while having an operating expense of less than \$65/month.

Prior to the SOAR, it took the analyst 75.84 hours to perform manual tasks related to the 11 use cases. Additionally, an estimated 450 hours of the analyst's time would be used to run the Update threat intelligence database use case. After the SOAR, the same tasks were automatically ran in 31.2 minutes and the Update threat intelligence database use case ran 9.000 times in 5.3 hours.

**Keywords:** SOAR, Cloud Computing, Security, Orchestration, Automation





# Contents

<b>List of Figures</b>	<b>xvii</b>
<b>List of Tables</b>	<b>xix</b>
<b>List of Abbreviations</b>	<b>xxiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	3
1.2 Objectives . . . . .	4
1.3 Document structure . . . . .	5
<b>2 Related works</b>	<b>7</b>
<b>3 Background</b>	<b>15</b>
3.1 OutSystems and the Low-Code Market . . . . .	15
3.2 Security in the Low-Code Market . . . . .	17
3.3 OutSystems' SOC . . . . .	17
<b>4 SOAR System Requirements</b>	<b>21</b>
4.1 Functional Requirements . . . . .	22
4.2 Non-functional Requirements . . . . .	23
4.3 Alternative approaches to solution . . . . .	23
4.3.1 COTS SOAR . . . . .	24
4.3.2 Apache Airflow . . . . .	29
4.3.3 Luigi . . . . .	31
4.3.4 OutSystems Platform . . . . .	32
4.3.5 Kubernetes . . . . .	33
4.3.6 Cloud Services . . . . .	34
4.4 Selected approach . . . . .	36
<b>5 Architecture and Implementation</b>	<b>39</b>
5.1 Design . . . . .	39
5.2 Technology . . . . .	42

5.2.1	Splunk Cloud . . . . .	43
5.2.2	AWS Simple Notification Service . . . . .	45
5.2.3	AWS Lambda . . . . .	46
5.2.4	AWS Secrets Manager . . . . .	51
5.2.5	AWS Step Functions . . . . .	54
5.2.6	AWS CodePipeline . . . . .	56
5.2.7	AWS CloudFormation . . . . .	58
5.2.8	GitHub . . . . .	60
<b>6</b>	<b>Results</b>	<b>63</b>
6.1	Use Cases . . . . .	63
6.1.1	Impossible travel . . . . .	63
6.1.2	AWS EC2 changes . . . . .	64
6.1.3	AWS IAM changes . . . . .	64
6.1.4	Access to credentials without justification . . . . .	64
6.1.5	New digital certificate issued . . . . .	64
6.1.6	Possible SQL injection . . . . .	65
6.1.7	Login brute force against the OutSystems Platform . . . . .	65
6.1.8	Login brute force against an internal system . . . . .	65
6.1.9	Login brute force against an external system . . . . .	65
6.1.10	Partner registered as employee . . . . .	65
6.1.11	Update threat intelligence database . . . . .	65
6.2	Performance Assessment . . . . .	66
<b>7</b>	<b>Conclusion</b>	<b>73</b>
	<b>Bibliography</b>	<b>81</b>
<b>A</b>	<b>AWS Pricing Calculator quote</b>	<b>83</b>
<b>B</b>	<b>SOAR Modules and Actions</b>	<b>85</b>







# List of Figures

1.1	Key functionalities provided by Security Orchestration [40] . . . . .	3
2.1	Quality attributes of security orchestration platform [40] . . . . .	7
2.2	Challenges that promote security orchestration [40] . . . . .	8
2.3	Conceptual map of security orchestration and automation [41] . . . . .	9
3.1	OutSystems Development Environment [58] . . . . .	15
3.2	Gartner Magic Quadrant for Enterprise LCAP, 2021 [34] . . . . .	16
3.3	Sentry architecture on AWS (adapted from Sentry Datasheet) [61] . . . . .	17
3.4	OutSystems' SOC Incident Response Life Cycle . . . . .	18
4.1	Apache Airflow web interface [1] . . . . .	29
4.2	Apache Airflow workflow graph [1] . . . . .	30
4.3	Luigi web interface [47] . . . . .	32
4.4	Luigi workflow graph [47] . . . . .	33
5.1	SOAR system architecture diagram . . . . .	40
5.2	Configuration of the AWS SNS Alert in Splunk Cloud . . . . .	44
5.3	AWS SNS overview [4] . . . . .	45
5.4	AWS Lambda Destination configuration . . . . .	48
5.5	AWS Secrets Manager overview [13] . . . . .	52
5.6	AWS Step Functions Workflow Studio [16] . . . . .	55
5.7	SOAR system production pipeline . . . . .	57
5.8	AWS CloudFormation overview [5] . . . . .	60
5.9	SOAR system GitHub repository . . . . .	61



# List of Tables

2.1	Comparison of this SOAR implementation and related works . . . . .	13
4.1	Typical needs and expectations of the SOC personnel . . . . .	21
4.2	SOAR functional requirements (adapted) . . . . .	22
4.3	SOAR non-functional requirements . . . . .	23
4.4	SOAR vendors and products, according to Gartner [23] . . . . .	25
4.5	SOAR vendors grouped by market presence, according to Forrester [49] .	27
4.6	SOAR vendors grouped by market presence, according to G2 [32] . . . .	28
4.7	SOAR vendors grouped by momentum, according to G2 [33] . . . . .	28
4.8	Comparison of potential cloud services for building a SOAR . . . . .	35
4.9	Comparison of the alternative approaches . . . . .	37
5.1	Total cost of the solution per month . . . . .	43
5.2	AWS SNS limits . . . . .	46
5.3	AWS SNS costs . . . . .	46
5.4	AWS Lambda limits . . . . .	49
5.5	AWS Lambda costs . . . . .	51
5.6	AWS Secrets Manager vs Parameter Store (adapted) . . . . .	52
5.7	AWS Secrets Manager limits . . . . .	53
5.8	AWS Secrets Manager costs . . . . .	54
5.9	AWS Step Functions: Standard vs Express [17] . . . . .	55
5.10	AWS Standard Step Functions limits . . . . .	56
5.11	AWS Step Functions costs . . . . .	56
5.12	AWS CodePipeline limits . . . . .	59
5.13	AWS CodePipeline costs . . . . .	59
6.1	Comparison of manual and automated tasks duration . . . . .	67
6.2	Review of the system requirements completeness . . . . .	68
A.1	AWS Pricing Calculator quote . . . . .	83
B.1	SOAR Modules and Actions . . . . .	85



# List of Abbreviations

<b>A2A</b>	Application-to-Application
<b>A2P</b>	Application-to-Person
<b>AI</b>	Artificial Intelligence
<b>AKS</b>	Azure Kubernetes Service
<b>API</b>	Application Programming Interface
<b>ARK</b>	Automated Responder Knowledge
<b>ARN</b>	Amazon Resource Name
<b>AWS</b>	Amazon Web Services
<b>CAA</b>	Certificate Authority Authorization
<b>CapEx</b>	Capital Expenditures
<b>CA</b>	Certificate Authority
<b>CI/CD</b>	Continuous Integration and Continuous Delivery
<b>CISO</b>	Chief Information Security Officer
<b>COTS</b>	Commercial off-the-shelf
<b>CTI</b>	Cyber Threat Intelligence
<b>EBS</b>	Elastic Block Store
<b>EC2</b>	Elastic Compute Cloud
<b>EKS</b>	Elastic Kubernetes Service
<b>ETSI</b>	European Telecommunication Standard Institute
<b>FTE</b>	Full-Time Equivalent

<b>FTS</b>	Follow the Sun
<b>GKE</b>	Google Kubernetes Engine
<b>HTTPS</b>	Hypertext Transfer Protocol Secure
<b>IaC</b>	Infrastructure as Code
<b>IAM</b>	Identity and Access Management
<b>IoC</b>	Indicators of Compromise
<b>IR</b>	Incident Response
<b>ISMS</b>	Information Security Management System
<b>ISPEC</b>	International Conference on Information Security Practice and Experience
<b>PAN</b>	Palo Alto Networks
<b>ITS</b>	Incident Tracking System
<b>IT</b>	Information Technology
<b>JSON</b>	JavaScript Object Notation
<b>LCAP</b>	Low-Code Application Platforms
<b>ML</b>	Machine Learning
<b>MQ</b>	Magic Quadrant
<b>MSSP</b>	Managed security service providers
<b>MTTR</b>	Mean time to resolve
<b>MWAA</b>	Managed Workflows for Apache Airflow
<b>OpEx</b>	Operating Expenses
<b>PaaS</b>	Platform-as-a-Service
<b>PoC</b>	Proof of Concept
<b>PR</b>	Pull Request
<b>RBAC</b>	Role-based access control
<b>RFP</b>	Request for proposal



<b>RPA</b>	Robotic Process Automation
<b>S3</b>	Simple Storage Service
<b>SaaS</b>	Software-as-a-Service
<b>SAI</b>	Supervised Active Intelligence
<b>SAM</b>	Serverless Application Model
<b>SDOF</b>	Security Device Orchestration Framework
<b>SDS</b>	Software-Defined Security
<b>SLA</b>	Service-level agreement
<b>SNS</b>	Simple Notification Service
<b>SOAR</b>	Security Orchestration, Automation, and Response
<b>SOC</b>	Security Operations Center
<b>SSO</b>	Single sign-on
<b>TAM</b>	Technical Account Manager
<b>TLS</b>	Transport Layer Security
<b>TM</b>	Trimmed Mean
<b>VM</b>	Virtual Machines
<b>VPC</b>	Virtual Private Cloud
<b>XDR</b>	Extended Detection and Response
<b>SIEM</b>	Security Information and Event Management



# Chapter 1

## Introduction

There are countless headlines [24, 27] about security incidents that caused great damage to companies and their customers. It has become commonplace to see news of yet another cyber attack against private and public organizations. Ashley A. Hall et al. [38], for instance, reviewed major security breaches between 2014 and 2018, covering 182 organizations, including public, private, governmental, and educational institutions. The study showed that 80.8% of the reported breaches directly affected the organization’s customers and that since 2014 there has been an increase in breaches related to “hack of online system including malware”.

Other works confirm this challenging security landscape. A recent research conducted by the Ponemon Institute and sponsored by IBM studied 550 organizations impacted by data breaches between March 2021 and March 2022 and concluded that 83% of them have had one or more breaches and that 60% of those breaches led to increases in the prices of the services/products sold by them [39]. Verizon’s 2022 Data Breach Investigations Report also confirms the alarming state of the information security of the enterprises. They analyzed 23,896 incidents, 5,212 of which culminated in data breaches in 2021 [70].

It is impossible to prevent all security incidents from happening, but when they happen organizations want to solve them fast and keep the impacts low. This is one of the main reasons why they maintain a Security Operations Center (SOC). Ponemon’s research showed that 63% of the analyzed organizations had a SOC team with regularly tested IR process, this decreased breach costs to an average of 3.26 million dollars, which represents a 45% cost reduction when compared to the average cost of 5.92 million for organizations that did not invest in a SOC and IR process.

A SOC is composed of various security professionals that are responsible for the detection, analysis, containment, eradication, and recovery of security incidents. They must react fast to the security incidents, but that is not always possible due to many factors (*e.g.* lack of personnel, expertise, automation, etc.). To overcome that, companies started enhancing their SOC with the capabilities provided by SOAR systems [23, 68].

The term SOAR was coined by Gartner in 2017 in their research “Innovation Insight for Security Orchestration, Automation, and Response” [25]. It has since become a relevant topic in the information security field, but without clear and common definition as many security practitioners have their own definition of SOAR, as evidenced by C. Islam et al. [40]. The authors, therefore, came up with their own definition, as follows: “Security Orchestration is the planning, integration, cooperation, and coordination of the activities of security tools and experts to produce and automate required actions in response to any security incident across multiple technology paradigms.”

Security professionals from more than 170 companies from different industries, locations, and size were interviewed in the 2022 edition of the SANS Institute survey on cloud security. The survey showed that SOAR solutions have been used by 53.8% of the analyzed companies (it was 48.4% in 2021) and that the use of automation and orchestration tools has increased across the board, SANS expect this trend to continue as organizations improve the speed and efficiency of cloud deployments [68].

A typical SOAR system make use of built-in and custom integrations to orchestrate and automate many tasks that would be manually executed by a security analyst during an incident response. According to C. Islam [40] and detailed in Figure 1.1, three major functionalities are provided by security orchestration systems:

- **Act as a Middleware/Hub:** vendors have mentioned security orchestration as a platform that acts as a hub for unification, coordination, data sharing and analysis for disparate cybersecurity and Information Technology (IT) solutions. The security analyst can easily integrate multivendor security tools, share threat intelligence, and collaborate with the external organizations to get an insight of an organization’s security state through an orchestration platform.
- **Orchestrate Security Activities:** after receiving alerts, security experts need to perform multiple steps to find the attacks, vulnerabilities, affected endpoints, and mitigation solutions. The workflow is designed to mimic human activities of threat investigation to reduce the cumbersome manual process, human errors, and improve staff capabilities to incident response. Orchestrating and integrating security tools’ activities allow experts to simplify complex workflow, coordinate the flow of data and tasks, and enable the powerful machine to machine automation.
- **Enable Automated Response:** security orchestration automate incident response activities. HEXADITE, for instance, has automated 800,000 man-hours of work in 2 years that is equivalent to \$38.5 million in customer savings. Several papers have reported that security orchestration automates the entire threat defense lifecycle and provides intelligence automation services. The European Telecommunication Standard Institute (ETSI) has considered automating the control of deployment and

configuration of the security functions as a substantial prerequisite of orchestration. Vendors use a security orchestration platform to automate repeatable tasks and remove duplicate incidents to optimize security staff's capability and reduce the overall cost.

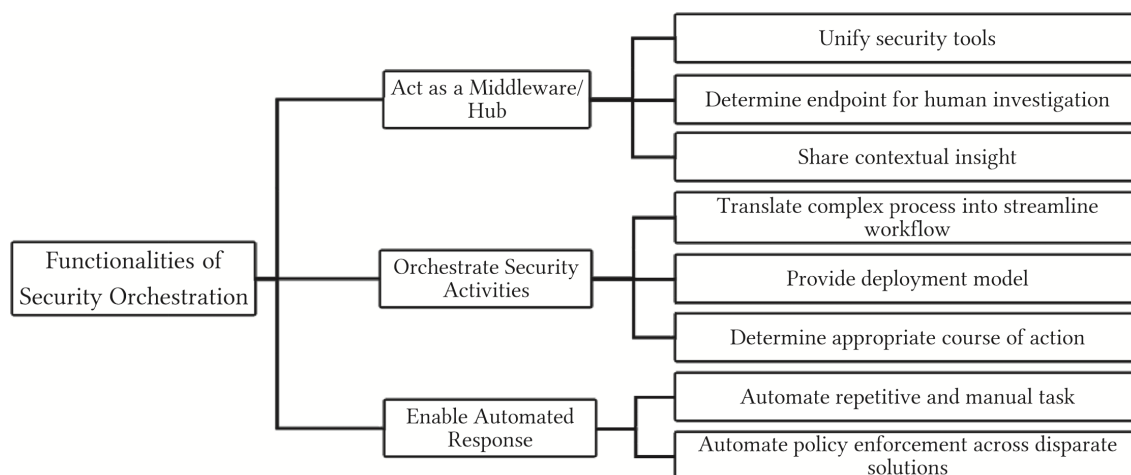


Figure 1.1: Key functionalities provided by Security Orchestration [40]

For example, a SOAR can automate many mundane and repeated tasks required by an impossible travel alert. This alert typically occur when the same user logs into a system using two different IPs from distinct countries in a short time span. During the resolution of the alert the analyst has to check the reputation of the IPs, check if the IPs are in the company's internal threat intelligence database (if not, add the IPs), perform a WHOIS lookup on the IPs, get user details, send a message to the user, post internal comments in the ticket, update the ticket status, etc.

The sequence of tasks described above for the impossible travel case can be automated by a SOAR system, and this automation concept can also be expanded to other use cases. In general terms, the SOAR system executes the tasks according to a preconfigured workflow that is strictly followed, so the outcome is predictable. Response times are also shortened as there is no human fatigue involved and the system is able to do repeated tasks indefinitely with the same accuracy and with a speed that is impossible for a human to achieve.

## 1.1 Motivation

As previously discussed, companies maintain a SOC to detect and resolve security incidents while containing its impacts, but it's a challenge to keep up with the growing quantity of the incidents and keep the SOC staff motivated. The use of security orchestration and automation is a way to increase SOC capacity and effectiveness, while optimizing

human resources costs. A well-trained team with a SOAR is capable of achieving great results as evidenced by this project's results in Chapter 6.

A SOAR system usually has a higher upfront cost as they are expensive and demand lots of person-hour for planning, implementation, initial configuration, and training of the team. But, this initial cost pays off when the system is up and running and doing the job of multiple analysts at lower costs and without worrying about sickness, fatigue, etc.

The business motivation behind this project is to deploy a SOAR system in order to increase the capacity of the SOC, enabling them to handle more incidents in a faster and more scalable way, while still providing the team a good work-life balance. Additionally, this will save the company financial investments in new hires as the SOAR can do the work of multiple analysts for a fraction of the cost.

Finally, to the best of our knowledge, the majority of the academic materials about security orchestration and automation focuses on theoretical concepts, frameworks development, and experiments in controlled environments. Moreover, there is a need to promote the development of low-cost SOAR solutions as the commercial ones are very expensive, ranging from \$10K to \$40K per month, and in the rare cases where they provide a free version, they are very limited and inviable for production usage. In this context, the main contributions of this project are:

- it proposes a low-cost and cloud native SOAR solution that decreased the duration of the security incident tasks by an average of 98.81% while having an operating expense of less than \$65/month.
- it fills the gap between the theory and practice by evaluating the performance of the proposed solution in the production environment of OutSystems, a large multinational company with a diverse technology stack, thousands of IT assets, and thousands of customers and employees around the world.

The aforementioned contributions are registered in a paper accepted for publication [26] in the 17th International Conference on Information Security Practice and Experience (ISPEC) held in Taipei, Taiwan in 2022.

## 1.2 Objectives

This document describes the implementation of a SOAR solution at OutSystems, a private multinational company in the technology sector. The project started with the understanding of the current context of OutSystems, followed by the definition of the problem statement which took into consideration said context, the challenges faced by the SOC, and the project requirements. The general objective of this project is: automate repetitive security tasks and increase the SOC capacity and effectiveness while maintaining the same headcount. In a more granular way, the objectives of this project are:

- Acquire or build a SOAR solution that is cost-effective
- Deploy a SOAR solution to the production environment
- Integrate the solution with at least 10 systems in the first year of operations
- Create at least 5 automated playbooks in the first year of operations

Many solutions were considered and the one that most aligned with the aforementioned objectives and the team's expectations and constraints was selected to be implemented.

## 1.3 Document structure

The remainder of this document is organized as follows:

- Chapter 2 presents the related works on security orchestration and automation.
- Chapter 3 provides context about the company, the market it is inserted into, and the project stakeholders.
- Chapter 4 presents the problem statement, the project requirements, and the proposed solutions.
- Chapter 5 provides details about the selected solution, including its architecture and the technologies used.
- Chapter 6 describes the use cases considered to validate and assess the solution and the practical results obtained with its implementation.
- Finally, Chapter 7 brings the conclusions and discusses possible improvements and future works.





# Chapter 2

## Related works

There is a good number of articles related to security orchestration and automation, some focuses on the review of the state of the art as well as the conceptual and social implications of SOAR [40, 28, 51, 43], others on the creation of frameworks and solutions [41, 48, 71], and finally, some discuss the implementation of security orchestration and automation in experimental and real-world scenarios [57, 45, 36, 53].

One of the most comprehensive academic works related to security orchestration and automation was done in 2019 by C. Islam et al. [40] when they identified and analyzed critical aspects of security orchestration solutions found in 95 papers. Their review addressed three research questions: what is security orchestration; what challenges security orchestration intend to solve; and what types of solutions have been proposed. The authors identified the key functionalities provided by the solutions as previously shown in Figure 1.1 and also listed the quality attributes of a typical security orchestration solution as shown in Figure 2.1.

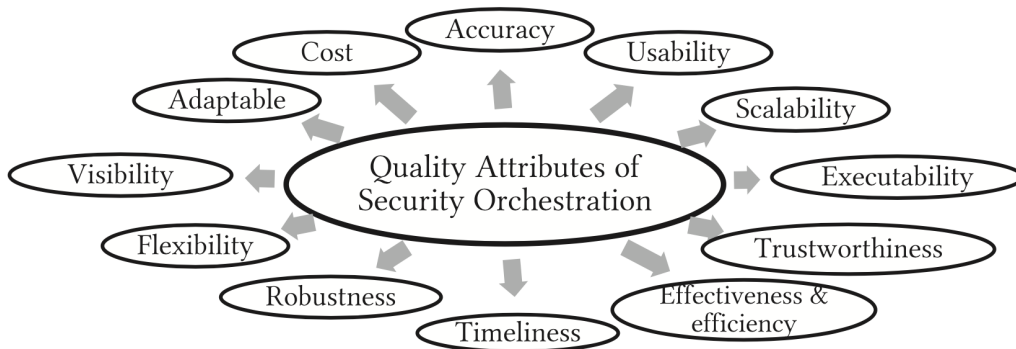


Figure 2.1: Quality attributes of security orchestration platform [40]

For C. Islam et al., security orchestration is “the planning, integration, cooperation, and coordination of the activities of security tools and experts to produce and automate required actions in response to any security incident across multiple technology paradigms”. The authors assert that having a common working definition of security orchestration will

help practitioners and researchers to define a discipline of research and practice for promoting practices, processes, and tools.

The authors listed the challenges that security orchestration intend to solve as shown in Figure 2.2. The challenges are divided into two main categories: technical and socio-technical. According the the authors, the technical challenges are related to limitations of the tools to accurately detect and respond to threats, limited interaction between diverse tools, and conflicts among tools while running simultaneously. On the other hand, the socio-technical challenges are related to the organizational processes, policies and rules with respect to cybersecurity and interaction of people with the technical system.

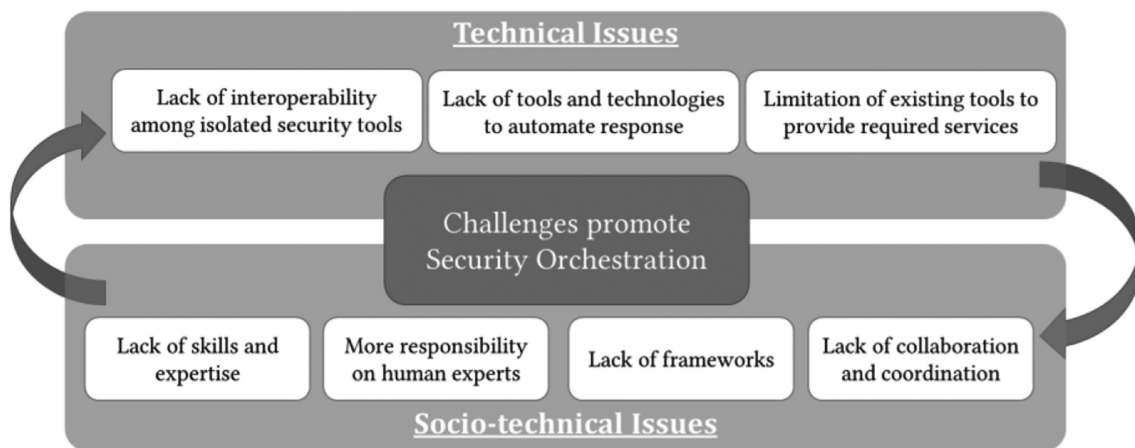


Figure 2.2: Challenges that promote security orchestration [40]

Finally, the authors have highlighted the key techniques, tools, and strategies used by practitioners and researchers in the realization of security orchestration. Most of the reviewed studies have proposed platform-based architecture as a strategy for incorporating security solutions to support their integration, orchestration, and automation [40].

A year later, in 2020, C. Islam et al. [41] presented their work on architecture-centric support for designing a SOAR platform, which has the objective of reducing the design complexity of a SOAR by modularizing the functional and non-functional requirements. This work has some overlaps with their previous publication [40], but there are some new and relevant content such as the conceptual map of SOAR (see Figure 2.3) and the evaluation of a Proof of Concept (PoC) designed and implemented based on the architecture.

For the PoC, the authors selected seven tools: Snort (Intrusion Detection and Prevention System - IDPS); Splunk (Security Information and Event Management - SIEM); LimaCharlie (Endpoint Detection and Response - EDR); MISP (Open-source intelligence - OSINT); Windows Defender Firewall (firewall); Wireshark and WinPCap (packet monitoring and logging).

The system has executed 45 IR processes out of 48, because for three of the processes, the orchestrator could not find any security tool with the required capabilities to execute

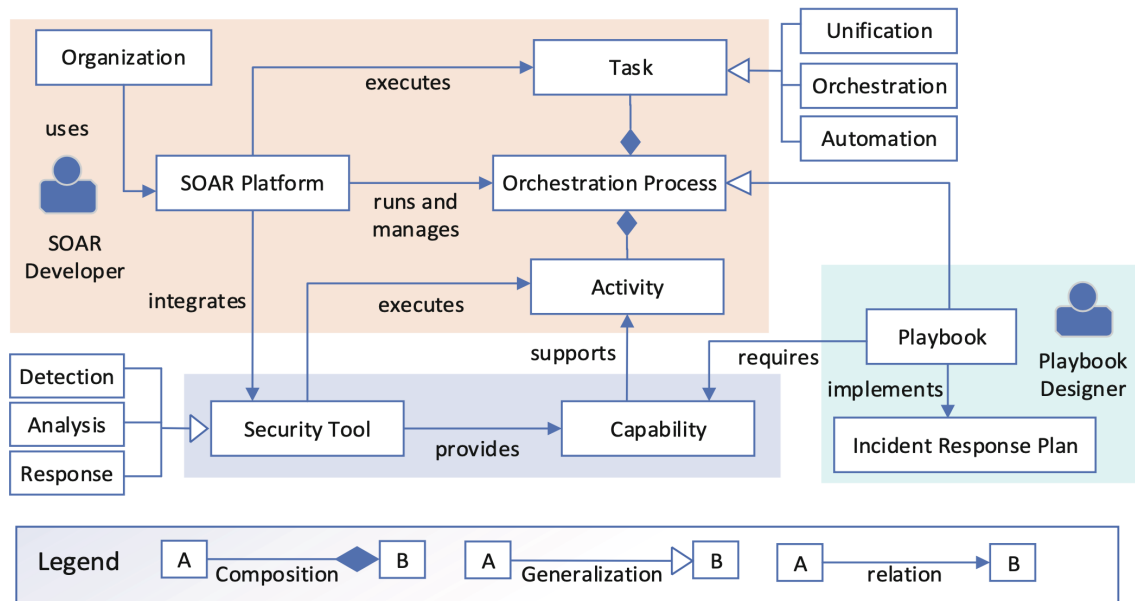


Figure 2.3: Conceptual map of security orchestration and automation [41]

the activities, hence those were executed partially. Overall, the authors affirm that the PoC proved the feasibility of the proposed architecture.

Johnson Kinyua et al. [43] reviewed works published in academic journals, conferences, sites, blogs, white papers, etc. that were related to Artificial Intelligence (AI) and Machine Learning (ML) implementation in SOAR solutions. The authors analyzed the maturity of AI/ML capabilities in major commercial SOAR solutions (according to Gartner and other sources) and their main observations were:

- FireEye: has a ML pipeline that evolves with adversaries through re-labeling and re-training.
- IBM Resilient: has ML models to predict the severity of new incidents, estimate time to resolve, and find similar incidents that were closed previously.
- Splunk: uses ML to do anomaly detection, user behavior analytics, events classification and clustering, and forecasting and prediction.
- Siemplify: uses ML to prioritize and investigate alerts and assign the best analyst to a case.
- D3 Security: the authors did not mention how AI/ML is used.
- DFLabs: uses ML in the form of Supervised Active Intelligence (SAI) and Automated Responder Knowledge (ARK).
- Rapid7: combines ML and ongoing human input to detect attacks and provide context about both the user and adversary in order to accelerate incident response.

- ThreatConnect: uses AI/ML for behavioral modeling, advanced analytics, and automated incident response.
- Palo Alto XSOAR (formerly Demisto): uses AI to support incident triage and to offer the analysts suggestions for next steps.
- ATAR Labs: uses ML to detect unusual behaviors that signal an attempt at data exfiltration.
- ServiceNow: uses AI/ML to automatically model normal behavior for performance metrics and detect anomalies for new metrics that fall outside predicted thresholds.

Dimitrios Lalos [45] analyzed three major SOAR solutions available in the market: Siemplify, Splunk SOAR, and Palo Alto Networks (PAN) XSOAR. He deployed the solutions in an experimental environment and presented the solutions' features. His work focused on showcasing the interface and main functionalities of the SOAR solutions and explaining their utility in a SOC environment.

W. Keith Edwards et al. [28] did a critical analysis on the use of security automation and how it is not a one-size-fits-all solution to solve all the scalability issues of the information security field. The authors also suggested a definition of security automation for end-users along a spectrum of rigidity, which provides a basis for discussing three sources of limits in automation: situational and social dependencies, accommodation of end-user values, and user interface costs deriving from automation failures.

First, Edwards explains that along the spectrum of security automation, the more rigid the security policy is, the more it assumes a "one-size-fits-all" strategy. The fixed policy approach assumes that the security decisions embedded in the design are a fit for the widest range of users; in contrast, the dynamic policy approach assumes a fit for individual users. A security policy that takes into consideration the situational and social dependencies of the users is essential to avoid creating a "social-technical gap". For example, a policy that blocks access to social media sites does not make sense for the Marketing team, as this is part of their job, but makes sense for other teams that do not require it to perform their daily job.

Second, the accommodation of end-user values refers to the contrast between the security team values and the values of the rest of the areas within the company (*e.g.* Sales, Marketing, Legal, etc.). When security decisions are made, the values behind these decisions are those of "empowered" sources (usually the IT or Security teams) rather than the users who will be affected by these policies. For example, whenever a malicious artefact is found in the laptop of an employee, a system isolation (*i.e.* the endpoint protection solution blocks all network connections and USB ports) is enforced until an automatic full-scan reports no additional findings, this may severely impact the end-user productivity in the name of proactive and "overzealous" security.

Third and last, the effects of automation on the user experience may also limit the implementation and usage of automation. The accuracy of the system must be high so the user does not need to be constantly dealing with incomplete/failed tasks. Additionally, in the rare cases where the automation fails the user must be able to easily detect and overcome the failure. For example, an aggressive anti-malware configuration that automatically deletes files deemed suspicious without any notification will generate backlash from the unsuspecting users that have no idea what happened to their files.

Song Luo et al. [48] presented a service-oriented approach to security orchestration for software-defined infrastructure that abstracts security controls as security services. Their approach interprets assets' security policies as service requirements. The approach aims to simplify security management by separating it from the actual security controls and infrastructure details, making enterprise security more agile. The authors evaluated their approach against two experimental use cases: Portable Asset Security and Autonomous Attack Containment. Unfortunately, they did not provide the results or a comparison of the performance before and after the implementation, so it was not possible to assess with certainty the effectiveness of the system.

Raydel Montesino et al. [51] analyzed three widely used information security standards and best practice guidelines, showing that about 30% of the security controls included in the ISO/IEC 27001 [31] and the NIST Special Publication 800-53 [55] can be automated by existing tools. The analysis has shown that no single tool exploits the full security control automation potential. Instead a combination of different tools is required to achieve the maximum automation degree.

Weijia Wang et al. [71] presented a data driven Security Device Orchestration Framework (SDOF) for Software-Defined Security (SDS). In SDOF, they put forward uniform interfaces for security devices so that they could be orchestrated by software and their data could be collected and processed centrally. According to the authors, one of the key challenges of SDS is how to schedule and orchestrate security appliances according to huge and heterogeneous threat information. The SDOF was implemented in a test environment and their analysis showed that it introduced a delay of 114ms on average, which is an acceptable cost considering that SDOF does real-time dynamic orchestration that can't be done in normal SDS environments.

Motoyuki Ohmori [57] created a solution to orchestrate several information systems and automate the initial incident response. The Incident Tracking System (ITS) automatically locates and isolates a suspicious host, and sends an e-mail notification to the person in charge of handling the incident. The system was evaluated by comparing the IR metrics of the critical incidents faced by the Tottori University before and after the implementation of the system. The system reduced the time required for the initial incident response to automatically isolate a suspicious host to less than 40 seconds while a manual operation required more than 30 minutes, several hours or even several days in some cases.

Gibadullin et al. [36] presented a system for automated incident management based on Apache Airflow, an open-source workflow management platform. The solution consisted of a high-availability environment with two HAProxy load balancers, two application servers running Apache Airflow, and two PostgreSQL database servers. To evaluate the system, three undisclosed security incidents were taken as statistical data. The automated incident management system decreased the average time spent on security incidents from 34.5 to 12 minutes, meaning the incidents were resolved 2.9 times faster.

Sara Nascimento [53] explored concepts of Robotic Process Automation (RPA) and SOAR to implement automation in three security use cases of MEO's SOC, one of the biggest telecommunications companies in Portugal, by using the RPA solution Blue Prism. The use cases were: Daily Reports from ArcSight to Tableau; Notify Customers of DoS Attacks with Reports; and Notify Customers of DoS Attacks with Screenshots. The RPA implementation provided a higher availability of execution and made the execution of some tasks 2 to 6 times faster.

It is important to note that SOAR is not the same as RPA, even though they share many similarities and the same end goal: automate tasks. The main difference between them is the fact that SOAR solutions are heavily dependent on the availability of API. On the other hand, RPA solutions can be used in situations where there are no APIs as they are able to mimic human actions (*e.g.* move the mouse, click on a button, write a text, etc.). If available, using the API is highly suggested as they are easier to interact with and provide an “official” way to programmatically interact with the system. The use of RPA is valid for cases where the system does not provide an API, so the only option left to programmatically interact with it is by using RPA solutions.

As demonstrated in this Chapter, there is academic and business interest in SOAR solutions as a way to improve the efficiency of security teams. The presented works confirm the need of orchestration and automation of security processes, and that having manual processes clearly defined is a necessary condition to a successful SOAR implementation. The security field is moving towards the use of orchestration and automation, there is an emerging market for such solutions.

This project takes into consideration the insights provided by the related works to create a low-cost and cloud native SOAR solution that is implemented and evaluated in a real-world scenario. The related works focused on the conceptual and social aspects of SOAR, creating frameworks, developing solutions that are not cloud native, and implementing a limited number of use cases using custom-built or commercial SOAR solutions. Although the custom-built solutions provided successful results, they were not created with the same cloud native requirements that are present in this project. Finally, commercial SOAR solutions are relatively costly when implemented in a large-scale environment.

To the best of the author's knowledge, this project differs from the currently available

academic materials in the sense that it focuses on creating a low-cost and cloud native SOAR solution that is implemented and evaluated using a vast number of use cases in the production environment of a large multinational company with real-world incidents.

Table 2.1 shows a comparison of the SOAR implementation done in the scope of this project and the implementation done by four of the related works with greater proximity to the present project, followed by a brief explanation.

	<b>This work</b>	<b>Dimitrios Lalos [45]</b>	<b>Ohmori [57]</b>	<b>Gibadullin et al. [36]</b>	<b>Sara Nascimento [53]</b>
Environ.	Large company	Personal	University	Small company	Large company
Solution	Custom SOAR	Siemplify, Splunk, PAN	Custom SOAR	Apache Airflow	Blue Prism (RPA)
Use cases	11	3	1	1	3
Executions	9,364	13	4	3	2,685
Monthly Cost	\$63.26	<i>Unknown</i>	<i>Unknown</i>	<i>Unknown</i>	<i>Unknown</i>

Table 2.1: Comparison of this SOAR implementation and related works

Regarding the environment, this work and Sara Nascimento’s work implemented their respective solutions in the production environment of a large company. Lalos implemented his solution in a personal test/lab environment. Ohmori implemented his solution in the production environment of a university. Finally, Gibadullin et al. implemented their solution in a small company.

With regard to the proposed solution, this work proposes a custom SOAR that is low-cost and cloud native. Lalos proposed multiple solutions based on commercial SOAR products such as Siemplify, Splunk SOAR, and Palo Alto Networks XSOAR. Ohmori proposed a custom SOAR, the ITS. Gibadullin et al. proposed a SOAR solution based on Apache Airflow. Finally, Sara Nascimento proposed a solution based on a commercial RPA product, Blue Prism.

Concerning the use cases, this work implemented 11 use cases while the other works implemented between 1 and 3. All the use cases are related to security incident response or repetitive security tasks such as threat intelligence collection and security reporting. The use cases implemented in this work are described in Section 6.1.

Regarding the number of executions using the implemented solution, this work and Sara Nascimento’s work were the ones with most executions, 9,364 and 2,685 respectively, which provides a more accurate idea of the system performance. The other works ranged from 3 to 13 executions.

Regarding costs, to the best of the author’s knowledge, this was the only work that explicitly explained the costs involved in a SOAR solution implementation, the other works related to SOAR implementation provided vague statements about costs or did not

address the subject at all.



# Chapter 3

## Background

This Chapter provides context about OutSystems and the market in which it is inserted, with a focus on the security perspective. Additionally, the OutSystems' SOC and their IR process is presented and briefly explained.

### 3.1 OutSystems and the Low-Code Market

The project took place at OutSystems, a large multinational company in the low-code market. The company has thousands of customers in 87 countries and across 22 industries. Their main product is the OutSystems Platform, a low-code solution that generates optimized back-end and front-end code from visual application models in a drag-and-drop experience, as presented in Figure 3.1.

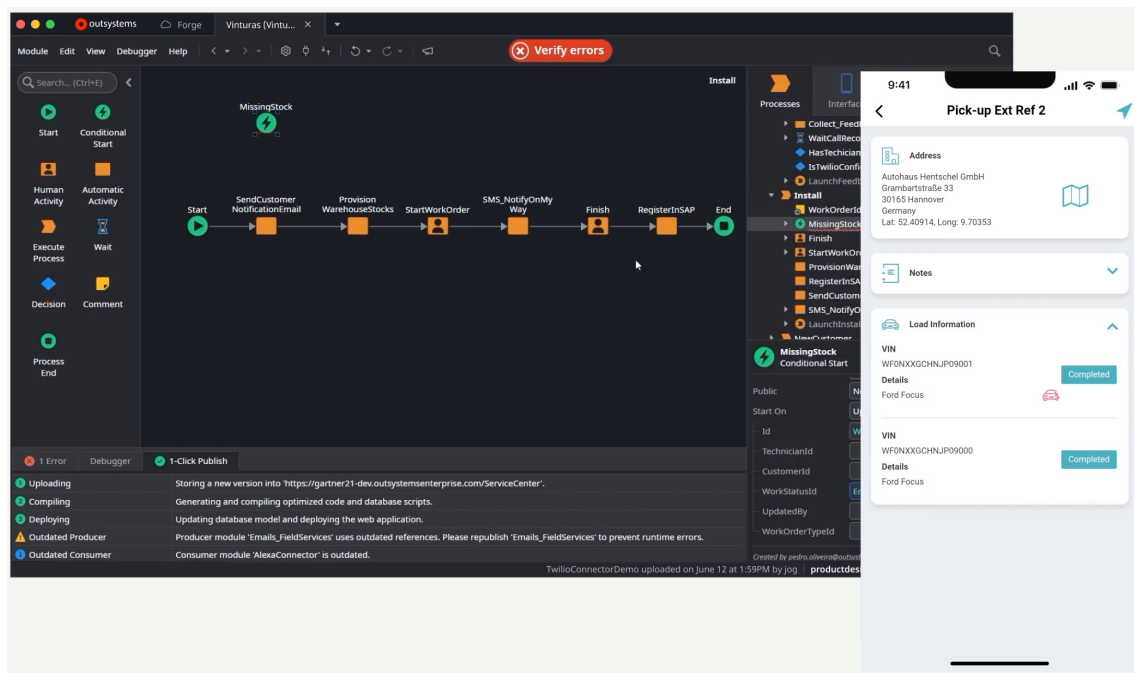


Figure 3.1: OutSystems Development Environment [58]

The OutSystems Platform is a PaaS, the company takes care of the hosting, networking, storage, runtime, etc. so the customer can focus on developing low-code applications. The resources (computing, storage, network, etc.) are hosted on AWS following security and data sovereignty best practices.

The company subscribes to the AWS Enterprise Support plan which provides benefits such as enhanced technical support and a designated Technical Account Manager (TAM). Parallel to that, there is also a multi-year strategic collaboration between OutSystems and AWS for investing in people, processes, and technologies to facilitate the value perception of the customers when they use the OutSystems Platform [59].

The company is inserted into the Enterprise Low-Code Application Platforms (LCAP) market, a highly competitive market with lots of vendors. According to Gartner [35], a LACP provides rapid application development and deployment using low-code and no-code techniques such as declarative, model-driven application design and development together with the simplified one-button deployment of applications. OutSystems is considered a market leader, as shown in Figure 3.2.



Figure 3.2: Gartner Magic Quadrant for Enterprise LCAP, 2021 [34]

## 3.2 Security in the Low-Code Market

Low-code platforms and companies are facing bigger scrutiny as customers increasingly worry about the security of their data and the data of their subsequent customers. Verizon’s Report [70] showed that supply chain compromise was responsible for 62% of system intrusion incidents in 2021.

As cybersecurity supply chain risk management becomes more mainstream with publications such as the NIST SP 800-161 [56] and the ISO/IEC 27036 [30], the security of the vendor and its platform has become a sales differential and a decision factor for the customers. Multiple vendors have dedicated pages explaining their security approach and compliance achievements, including OutSystems [60].

On this aspect, the OutSystems Platform has a straightforward pricing model: Free, Standard, and Enterprise. Within the Enterprise plan, there is also an optional offering called Sentry, which “provides additional security features designed for organizations working with sensitive data and/or working in highly regulated markets that requires compliance with ISO 27001, SOC2, HIPAA, etc.” [61]. Figure 3.3 shows a high level architecture of a Sentry environment. Customers that subscribe to the Sentry offering have a Service-level agreement (SLA) that defines distinguished access to the OutSystems’ SOC.

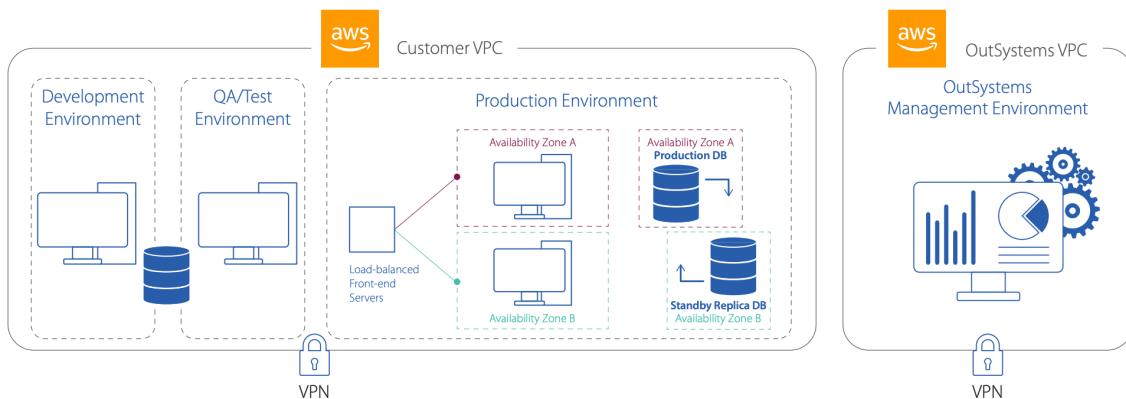


Figure 3.3: Sentry architecture on AWS (adapted from Sentry Datasheet) [61]

## 3.3 OutSystems’ SOC

The OutSystems’ SOC is composed of a multicultural team working in multiple time zones using a FTS approach, ensuring full availability to resolve security incidents. The SOC has a flat organizational chart, with analysts and engineers reporting to a manager, which then reports to the Chief Information Security Officer (CISO). Besides the SOC, there are other teams within the OutSystems’ Security Office, namely Risk and Compliance, Security Automation, Identity and Access Management (IAM), Security Architecture, and Application Security.

The SOC is responsible for doing incident response for internal company assets and also for all the customers' PaaS assets, with distinguished SLAs for the Sentry customers. The SOC has an IR process and hundreds of playbooks to deal with a wide range of attack vectors and threats, each playbook is written in English and contains the phases shown in Figure 3.4.

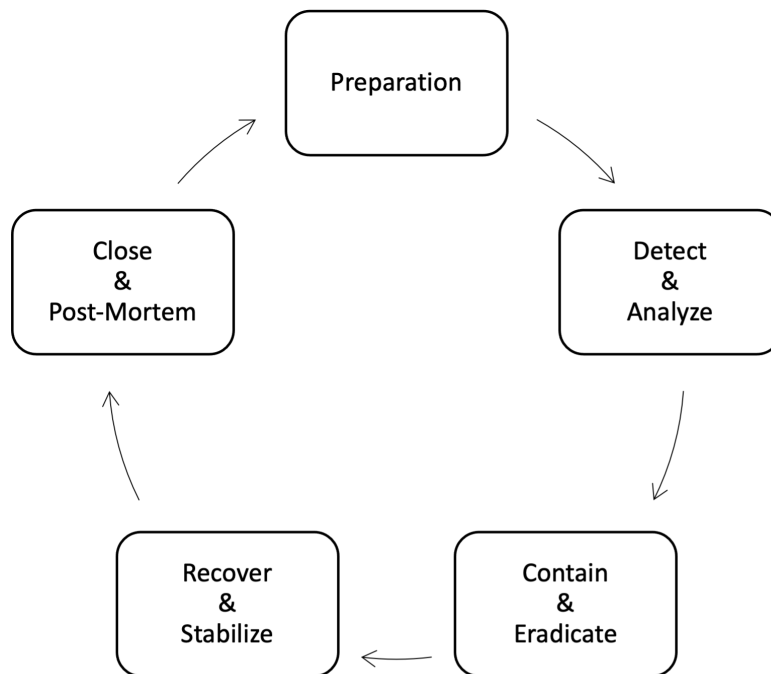


Figure 3.4: OutSystems' SOC Incident Response Life Cycle

The OutSystems' SOC Incident Response Life Cycle is based on the NIST's Incident Response Life Cycle [54] and its phases are described as follows:

- **Preparation:** this is an essential phase of the incident life cycle, but that often goes unnoticed by less mature companies security-wise. This phase involves the identification and classification of the assets, definition and continuous maintenance of the Information Security Management System (ISMS) including policies, procedures, and standards, continuous employee awareness and training, and finally, but not less important, continuous technical training and IR exercises for the SOC team.
- **Detect & Analyze:** this phase involves the collection of data from many sources (*e.g.* cloud services logs, network flows, audit logs, threat intelligence feeds, etc.), the correlation of this data, and the triggering of alerts when behaviours deemed suspicious/malicious are detected according to the rules defined by the SOC. This phase also involves the use of many tools to investigate and document the incident and relies on the critical-thinking and analytical capabilities of the analyst.

- **Contain & Eradicate:** the goal of this phase is to contain the security incident to avoid a bigger impact and to also make the eradication and recovery faster and easier. The containment may include tasks such as blocking IPs in the firewall, quarantine a network node or even an entire subnet, quarantine malicious artefacts, block user accounts, etc. After the incident has been contained, the SOC starts removing all the malicious artefacts from the environment.
- **Recover & Stabilize:** this phase involves the recovery of systems and data that were affected by the incident, for most cases the recovery is straightforward as there is no relevant data loss and the removal of the malicious artefacts is enough, but there can be more severe incidents where the company must restore the systems and data using the backups, which must be promptly available in the first place, considering the company did a good job at creating and maintaining an ISMS. This phase also involves the stabilization of the systems and networks, so they go back to their previous state, as it is expected to have anomalous resource usage and even short downtimes during and shortly after the recovery.
- **Close & Post-Mortem:** finally, this phase involves the final revision and documentation of everything done during the incident, including the aspects that could be improved so they can be addressed. Post-Mortem sessions are held for all the major incidents, but can also be held for smaller incidents that are of interest. Periodic “lessons learned” meetings are also held to understand the current context and state of the SOC and use it as an input for the Preparation phase so that the appropriate measures are taken (*e.g.* fine-tune an alert, update the IR process, etc.).

The playbooks are created and maintained considering the MITRE’s ATT&CK framework, which is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. This knowledge base is used as a foundation for the development of threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community [50].

The SOC deals with a wide range of security solutions on a daily basis (*e.g.* SIEM, IDPS, CASB, etc.). In basic terms, the SOC analysts are responsible for operating said solutions and doing preventive and evolutive maintenance that requires low to medium effort, as their focus is security monitoring and response. On the other hand, the SOC engineers are focused on designing and implementing security solutions, like the SOAR solution described in this project, and maintenance tasks that requires more experience and time.



# Chapter 4

## SOAR System Requirements

This Chapter presents the functional and non-functional requirements for the SOAR system and the alternative approaches to solution that were considered by the OutSystems' SOC. The selected solution is presented in Section 4.4.

Role	Typical needs and expectations
Analyst	Spend less time doing repetitive tasks that could be automated, so it is possible to allocate more time to tasks that require critical-thinking.
	Spend less time working on security incidents, so it is possible to participate in more high-visibility and strategic security projects.
	Decrease the time it takes to solve security incidents, so it is possible to decrease the potential impacts to the company and customers.
Engineer	Have a solution that is vendor-neutral and capable of interacting with systems from multiple vendors, to avoid vendor lock-in.
	Have a solution that is extensible using high-level programming languages, to allow the creation of custom integrations.
	Have a solution with low maintenance needs and that promotes a shift from Capital Expenditures (CapEx) to Operating Expenses (OpEx).
Manager	Remove low-value and boring tasks from the analyst's daily routine, to improve the team's morale and retention.
	Have a solution that has reporting and metrics capabilities, so it is easy to evaluate its performance and share it with business stakeholders.
	Orchestrate and automate as many tasks as possible, to increase the SOC capacity while maintaining the same headcount.

Table 4.1: Typical needs and expectations of the SOC personnel

As the company grows, it hires employees, acquires customers, does partnerships and, consequently, expands its digital footprint and attack surface, the SOC must keep up with the pace and ensure the security of the current and new assets. Unfortunately, the SOC team has to execute a considerable amount of mundane and manual security tasks, which is not the best use of the analyst's time and is also a risk to the effectiveness and scalability of the SOC.

It is economically and operationally inviable to solve this problem by simply hiring more analysts. As evidenced by the academic materials [36, 40, 51, 57], one has to invest

in orchestration and automation as a way to increase the organization's security capacity in a sustainable way.

To define the functional and non-functional requirements of the SOAR System, the typical needs and expectations of SOC personnel were first identified by taking into consideration the knowledge from the available literature, market surveys, and the professional experience of the SOC team members. Table 4.1 shows the typical needs and expectations of the SOC personnel from different perspectives.

## 4.1 Functional Requirements

The OutSystems' SOC defined a set of functional requirements for the SOAR solution, which were properly noted and had weights associated to them (ranging from optional to mandatory). Table 4.2 contains an adapted version of the original list, as some of the requirements were omitted or slightly changed to avoid the risk of exposing internal company information.

REQ	Description
F.R.01	Accessible via HTTPS, no agent is required to operate the SOAR
F.R.02	Have playbook automation with visual editing capabilities
F.R.03	Be able to run orchestration actions in parallel and sequentially
F.R.04	Audit logs can be exported to external systems in popular formats
F.R.05	Can be deployed in the cloud
F.R.06	Provide a REST API to interact with the SOAR
F.R.07	Have auditing capabilities
F.R.08	Have Single sign-on (SSO) capabilities
F.R.09	Supports Role-based access control (RBAC)
F.R.10	Integrate with Directory systems ( <i>e.g.</i> Azure AD, Active Directory)
F.R.11	Integrate with SIEM systems ( <i>e.g.</i> QRadar, Splunk, Sentinel)
F.R.12	Integrate with email systems ( <i>e.g.</i> Gmail, Exchange)
F.R.13	Integrate with communication systems ( <i>e.g.</i> Slack, Teams, Zoom)
F.R.14	Integrate with ticketing systems ( <i>e.g.</i> BMC, ServiceNow, Zendesk)
F.R.15	Integrate with on-call management systems ( <i>e.g.</i> Opsgenie, PagerDuty)
F.R.16	Integrate with major cloud provider services ( <i>e.g.</i> AWS, GCP, Azure)
F.R.17	Integrate with operating systems ( <i>e.g.</i> Windows, Linux, macOS)
F.R.18	Integrate with device management systems ( <i>e.g.</i> Jamf, Intune, Google)
F.R.19	Integrate with endpoint protection systems ( <i>e.g.</i> Crowdstrike, Defender)
F.R.20	Integrate with networking devices ( <i>e.g.</i> Fortinet, Meraki, Cisco)
F.R.21	Integrate with the OutSystems Platform
F.R.22	Automated playbooks have version control capabilities
F.R.23	Possible to create custom actions using high-level languages ( <i>e.g.</i> Python)
F.R.24	Technical support 24x7 with tiered SLA (only applicable to COTS SOAR)
F.R.25	Can be subscribed "as a service" (only applicable to COTS SOAR)

Table 4.2: SOAR functional requirements (adapted)



## 4.2 Non-functional Requirements

A list of non-functional requirements was also defined and is presented in Table 4.3.

REQ	Description
NF.R.01	Maintenance: the SOC wants to avoid the operational overhead of deploying and maintaining servers. The solution should be based on one of the following: a COTS product that is sold as a Software-as-a-Service (SaaS); a system built on top of managed Kubernetes services; or a system built using serverless services such as AWS Lambda, Azure Functions, or Google Cloud Functions.
NF.R.02	Security: the solution must guarantee data security at rest and in transit, auditing, access control, data sovereignty, disaster recovery, and high availability. Additionally, if it uses cloud services, it must follow the security best practices provided by the selected cloud provider.
NF.R.03	Velocity: the automated actions performed by the solution must be, at least, ten times faster than the same actions performed by the security analyst.
NF.R.04	Elasticity: the solution must be able to handle, at least, 10,000 security incidents/month. This number was intentionally extrapolated to avoid disclosing internal company metrics and to guarantee that the solution is able to handle a very high volume of incidents. It's important to note that being elastic means the solution is able to automatically use more resources to handle a higher volume of incidents and release said resources when the volume is lower.
NF.R.05	Cost efficiency: for COTS, the maximum monthly cost must be 2.5 FTE (\$7,047.50). For custom-built solutions, the maximum monthly cost must be 0.25 FTE (\$704.75) as there are other indirect costs related to the system's development and maintenance. Both situations consider the average monthly brute salary of a security analyst in Lisbon, Portugal, which at the time of this publication was \$2,819 [37].

Table 4.3: SOAR non-functional requirements

## 4.3 Alternative approaches to solution

This Section describes the solutions that were proposed to solve the problem. Each solution was analyzed considering the requirements defined.

The market for SOAR products is rather new when compared to more mainstream and mature markets like SIEM, Endpoint Protection, Network Security, etc. There are multiple vendors with COTS products like Palo Alto, IBM, Splunk, etc. These products have a high cost as they demand lots of person-hour for planning, implementation, initial configuration, and training of the team. But, this initial cost pays off when the solution is properly set up and doing the job of multiple Full-Time Equivalent (FTE) employees at lower costs and without worrying about sickness, fatigue, etc.

There are also open-source workflow management tools like Apache Airflow and Spotify's Luigi that are not considered SOAR products because their focus is not security and they lack many, if not all, built-in integrations with various security systems. Even so, they can be used as a SOAR system if one decides to build the necessary integrations and make use of their orchestration capabilities, as evidenced by Gibadullin et al. [36].

Finally, there are other ways to achieve security orchestration and automation by leveraging the capabilities of managed cloud services. AWS has shown [69] that services like Step Functions and Lambda can orchestrate and automate various aspects of the incident response process. The same could be achieved by using equivalent services from other cloud providers such as Microsoft Azure and Google Cloud.

Along with the SOC team, this project considered six solutions to solve the problem by implementing a security orchestration and automation solution. The proposed solutions are listed below and are explained in-depth in Sub-Sections 4.3.1 until 4.3.6.

- Buy a COTS SOAR product
- Adapt Apache Airflow as a SOAR system
- Adapt Luigi as a SOAR system
- Build a SOAR system using the OutSystems Platform
- Build a SOAR system using Kubernetes
- Build a SOAR system using cloud services

### 4.3.1 COTS SOAR

The first proposed solution was buying a SOAR product from the market as this meant having a system that is ready for use and with dedicated support. Following the company's procurement procedures, a market research was performed to identify vendors, products, and trends in the SOAR market. Publicly available market research from Gartner, Forrester, IDC, and G2 were considered and are discussed below.

Gartner is well known for its Magic Quadrant (MQ) research methodology which groups the vendors into four groups: Leaders, Visionaries, Niche Players and Challengers, but there is no MQ for SOAR as of the date of this publication. For new and emerging markets (the case of SOAR), Gartner provides a different research piece called Market Guide which shares an understanding of the current status of the market. The Gartner Market Guide for SOAR Solutions [23] provides valuable insights and a list of the prominent vendors and products in the market, which are listed in Table 4.4.

Gartner defines SOAR as solutions that combine incident response, orchestration, automation, and Cyber Threat Intelligence (CTI) management capabilities in a single platform. SOAR tools are also used to document and implement processes (aka playbooks,

<b>Vendor</b>	<b>Product</b>
Anomali	ThreatStream
Cyware	Virtual Cyber Fusion Center
D3 Security	D3 SOAR
DFLabs	IncMan SOAR
EclecticIQ	EclecticIQ Platform
Fireeye	Helix
Fortinet (CyberSponse)	FortiSOAR
Honeycomb	SOCAutomation
IBM Security	Resilient
LogicHub	SOAR+
Micro Focus (ATAR Labs)	ArcSight SOAR
Palo Alto Networks	Cortex XSOAR
Rapid7	InsightConnect
ServiceNow	Security Operations
Google (Siemplify)	Siemplify SOAR Platform
Splunk	Splunk SOAR
Swimlane	Swimlane SOAR Platform
ThreatConnect	ThreatConnect SOAR Platform
ThreatQuotient	ThreatQ
Tines	Tines

Table 4.4: SOAR vendors and products, according to Gartner [23]

workflows and processes); support security incident management; and apply machine-based assistance to human security analysts and operators. Workflows can be orchestrated via integration with other technologies, and automated to achieve desired outcomes.

The main customers of SOAR solutions are medium to large SOC teams and Managed security service providers (MSSP) that already have clearly defined incident response processes and want to enhance their capabilities and decrease the Mean time to resolve (MTTR) with the use of automation and orchestration. A successful SOAR solution implementation depends on the current maturity level of the SOC, as one can only automate clearly defined tasks.

The increasing use of SOAR solutions is also putting pressure on the various security vendors to add API capabilities to their products so it can exchange data using tried and tested interfaces like REST, SOAP, GraphQL, etc. The availability of an API plays a major role in defining if the SOAR solution can interact with a given product or not.

In the past, the vast majority of security vendors focused on integrations limited to their ecosystem of products and sold this as an advantage for adopting their range of products. Now, products who can not interact with other vendors are considered limited and a bottleneck for the SOC activities. This forced a narrative change, now many vendors consider the openness and extensibility of their product as a selling point.

Gartner mentions that “a lack of mature processes and procedures in the security operations team, combined with budget and staff constraints, presents obstacles to the adoption of SOAR solutions by a wider audience. This, combined with the varying degrees of maturity in vendors’ APIs as integration options, remains a key reason for SOAR not achieving higher rates of adoption.” [23]

Finally, Gartner strongly recommends a list of requirements to consider when selecting a SOAR solution, which are listed below in a concise way:

- Have the ability to be deployed either on-premises or as a cloud solution (like SaaS).
- Support a wide range of security products across multiple existing point solution markets (*e.g.* endpoint, firewalls, IDPS, SIEM, email gateways, etc.).
- Support low-code “playbooks” as a way to do event correlation and aggregation to improve security operations processes and alerting with better event enrichment.
- Support the ingestion of a variety of sources and formats of threat intelligence from third-party sources, including open-source, government, and commercial providers.
- Bidirectional integrations with IT solutions like ticketing systems for case management and messaging applications for better real-time communications.

Another popular research company is Forrester, it is well known for its Forrester Wave, which consists of a buyers guide that groups vendors into four groups: Leaders, Strong Performers, Contenders, and Challengers, but there is no Forrester Wave for SOAR as of the date of this publication. Fortunately, they also provide another type of research called Forrester Now Tech which gives insights into a given segment of the market and the vendors included on it, and there is one for SOAR [49]. All in all, it provided similar insights as Gartner did, their main takeaways were:

- Process automation for inconsistent workflows is challenging and cannot address every manual task security analysts perform. It is necessary to identify and automate processes that are consistent and repeatable.
- Define — prior to purchase — detection and response processes that can be automated. SOAR playbooks are only as useful as the underlying processes they are defined by.
- Allot resources for continuous upkeep. SOAR is not a set-it-and-forget-it tool.

Forrester Now Tech for SOAR also provided a list of vendors grouped by their market presence as visible in Table 4.5.

Market Presence	Vendors
Large (> \$30M/year revenue)	Cisco, Exabeam, IBM, Micro Focus, Microsoft, Palo Alto Networks, ServiceNow, Splunk
Medium (\$10M-\$30M/year revenue)	Cyware, D3 Security, LogRhythm, Logpoint, Rapid7, Securonix, Siemplify, Sumo Logic, Swimlane, ThreatQuotient
Low (< \$10M/year revenue)	DTonomy, Fortinet, Gurukul, LogicHub, Logsign, ManageEngine, QI-ANXIN, Securaa, Shuffle, SIRP, ThreatConnect, Tines, Torq

Table 4.5: SOAR vendors grouped by market presence, according to Forrester [49]

IDC is yet another popular research company that we used to gain insight on the SOAR market. They provide a wide range of market research materials, but this project focuses on the ones that are publicly available, which unfortunately are very limited. In the conference “Automation, incident response and orchestration: a modern security framework” [42] hosted by IDC and Palo Alto Networks on September 2021, they presented a very brief analysis of the SOAR market and the results of a security automation survey:

- IDC anticipates that orchestration and automation technologies will experience double-digit growth during next 5 years (+11.4% CAGR).
- Remote work is the biggest impact of COVID-19 which also resulted in additional demand for third-party support in security and incident response.
- Lack of automation, skills and budget are currently the biggest challenges for the respondents when it comes to security orchestration and automation technologies.
- The majority of the survey respondents are ready to invest money into SOAR over the next 12 months including automation and forensics technologies.

G2 is a lesser-known research company, but was included in the analysis for completeness. It provides three types of reports: Grid Reports (provides a high-level overview of a category); Index Reports (provides a deep dive into a range of products); and Momentum Reports (provides a list of products that are innovating, the ones that are stagnant, and the ones that are falling behind).

For the SOAR market, G2 created a Grid Report [32] and a Momentum Grid Report [33]. All the reports are based on the review of real users along with data aggregated from online sources and social networks, hence the findings can be imprecise if not enough reviews and public data are available.

The SOAR Grid Report provided a list of vendors grouped by their market presence and customer satisfaction as visible in Table 4.6. The SOAR Momentum Grid Report provided a list of vendors grouped by their momentum as visible in Table 4.7.

<b>Grid Placement</b>	<b>Vendors</b>
Leaders	PhishER, Tines, LogPoint
High Performers	CrowdSec, LogicHub, Blumira, SIRP
Contenders	Microsoft Sentinel, Demisto
Niche	Swimlane, D3 Security, IBM Resilient, Siemplify

Table 4.6: SOAR vendors grouped by market presence, according to G2 [32]

<b>Grid Placement</b>	<b>Vendors</b>
Momentum Leaders	Blumira, SIRP
Others	Siemplify, Swimlane, Demisto, D3 Security, IBM Resilient

Table 4.7: SOAR vendors grouped by momentum, according to G2 [33]

G2 findings were lacking and not aligned with the findings from the other research companies and the overall market sentiment. Well-known SOAR vendors like Splunk, Micro Focus, Rapid7 and many others were not even listed in G2's research. They also placed PhishER as a SOAR Leader, but this platform focuses only on phishing orchestration and automation, hence the scope of action and capabilities are very limited for this platform to be considered a competitive SOAR vendor, let alone a category leader.

Taking into consideration the system requirements, market researches, OutSystems' current systems and partners, and previous experience with some of the vendors, the SOC came up with a shortlist of vendors that were invited to participate in a Request for proposal (RFP) process. The RFP had the following phases:

- Create and validate the RFP with the stakeholders
- Send the RFP to the selected vendors
- Host a 1-hour session with each vendor so they can show their product
- Test the product capabilities in a demo environment
- Receive documents and pricing from the vendors
- Review and document all the findings and conclusions
- Finish the RFP, and name a winner (optional)

The RFP process took roughly three months to finish. All the vendors were evaluated following the same principles and placed in a rank according to their capabilities to meet the requirements defined. The top three candidates received additional scrutiny from the Risk and Compliance team and entered the negotiations phase with the Procurement team.

The pricing of a COTS SOAR varies a lot depending on the number of users, SLA agreed, services included, deployment model, discounts, etc. To avoid disclosing internal

information, the author was not allowed to provide the exact prices that were achieved during the RFP. For the sake of completeness of the analysis, we will consider a safe cost range of \$10,000 to \$30,000 per month for a SaaS SOAR that includes 20 concurrent users, unlimited playbooks and executions, 24x7 technical support, ramp-up courses, and the initial setup done by the vendor.

The RFP conclude that three vendors were able to meet most of the functional requirements, but no one was able to met the non-functional requirement NFR.05 so the company decided to not proceed with a COTS SOAR at the moment, which ultimately made this approach unavailable. Fortunately, the meetings with the vendors and the hands-on demos with the products provided the OutSystems' SOC with insights about the SOAR market and capabilities.

### 4.3.2 Apache Airflow

Apache Airflow is a platform developed and maintained by the Apache Software Foundation to programmatically create, schedule, and monitor workflows. Being one of the most mature open-source tools for workflow management [3, 67], it has been included in the service catalog of major cloud providers such as AWS and Google Cloud, so customers can deploy it using an on-demand pricing model and abstract the infrastructure maintenance tasks. The platform's web interface is shown in Figure 4.1.

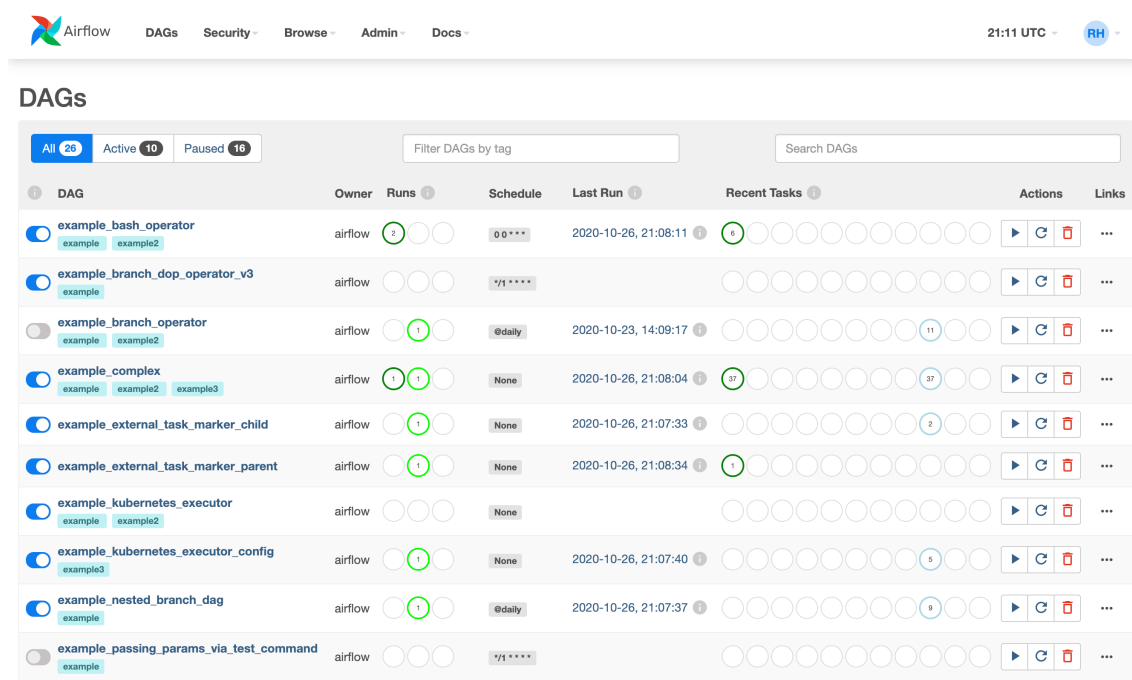


Figure 4.1: Apache Airflow web interface [1]

Adapting Apache Airflow as a SOAR solution is possible and has already been done by Gibadullin et al. [36]. They developed a high-availability solution consisting of two

HAProxy load balancers, two application servers running Apache Airflow, and two PostgreSQL database servers.

The OutSystems' SOC did not want the burden of maintaining the underlying Apache Airflow infrastructure as well as load balancers and database servers, so this solution considered managed Apache Airflow services only, instead of deploying and maintaining the Apache Airflow infrastructure ourselves like Gibadullin et al. [36] did. AWS offers Amazon Managed Workflows for Apache Airflow (MWAA) while Google Cloud offers Cloud Composer. Azure does not provide a service like MWAA or Cloud Composer, instead they recommend [66] the deployment of your own Apache Airflow using App Service, Kubernetes, and Azure Storage.

Even by using a managed Apache Airflow service, there is one big drawback with this solution: the lack of visual editing capabilities (F.R.02). The platform displays the finished workflow in a visual way as seen on Figure 4.2, but this is not editable. To create and update the workflows, the analyst must edit the underlying Python code of the workflow.

There are some workarounds for visually authoring workflows such as Elyra [29] and Rabix Composer [64]. The first has an active community but focus on artificial intelligence pipelines, while the second can be used for general-purpose pipelines but has not been maintained since Q2 2021. In the end, they are just workarounds and not something official and built into Apache Airflow. The SOC can not rely on third-party workarounds that can stop working or lose support any time in the future.

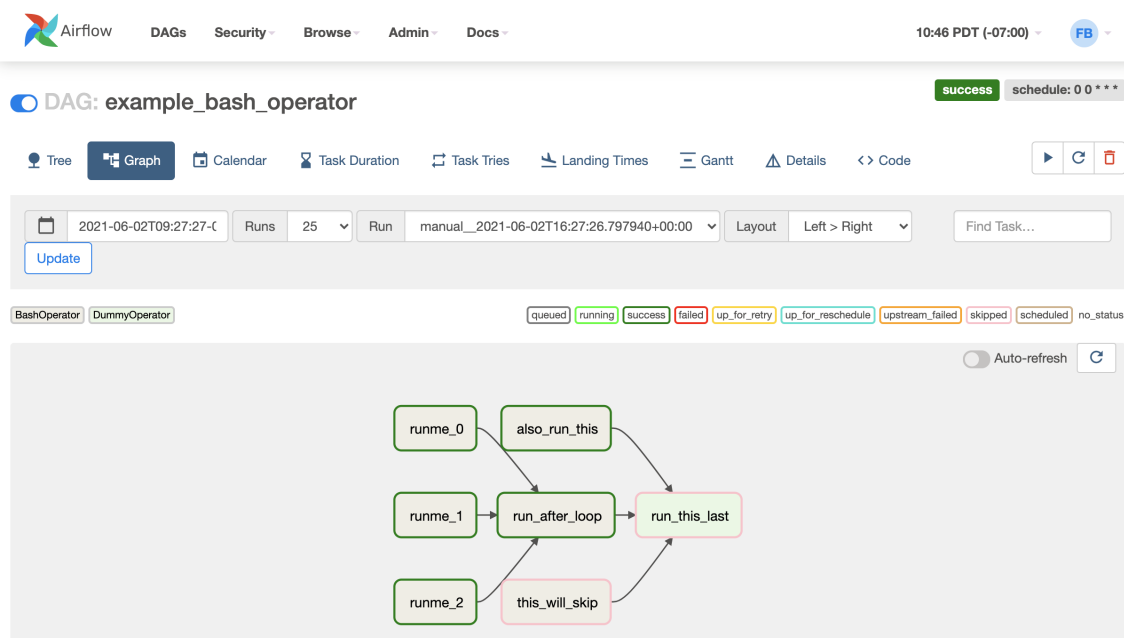


Figure 4.2: Apache Airflow workflow graph [1]

Azure did not provide a managed service and, therefore, cannot be chosen as a solu-



tion. Google Cloud Composer was also not chosen because the SOC had low experience working with Google Cloud. AWS MWAA was the only option left and the cost was \$1,931.38/month for the configuration below. However, this solution was also disregarded due to the lack of visual editing capabilities, which is a mandatory requirement.

- Production environment
  - Large MWAA environment
  - Workers: 2 (min) to 4 (max)
  - Hours/day at maximum workers: 6
  - Schedulers: 4
  - 1x 30 GB EBS gp2 volume
- Development environment
  - Small MWAA environment
  - Workers: 2 (min) to 4 (max)
  - Hours/day at maximum workers: 3
  - Schedulers: 2
  - 1x 30 GB EBS gp2 volume

### 4.3.3 Luigi

Luigi is a Python 3 tool that provides capabilities to build pipelines of batch jobs and workflow management. The tool was built by Spotify to be internally used for tasks automation, but later on was released as an open-source tool. Even though it has a large community, it does not have a mature open-source organization behind it, like Apache Airflow has. The tool could work as a SOAR if tweaked to address the security orchestration and automation needs. Luigi's web interface is shown in Figure 4.3.

There are no managed services based on Luigi, so the SOC would be responsible for the deployment, updates, and maintenance of the underlying infrastructure. Also, because there is no managed service option and this is an open-source tool, there is no dedicated support nor SLAs. The cost of hosting Luigi on AWS was \$364.01/month for the configuration listed below.

- Production environment
  - 1x `t3a.2xlarge` Elastic Compute Cloud (EC2) instance
  - 1x 30 GB EBS gp2 volume

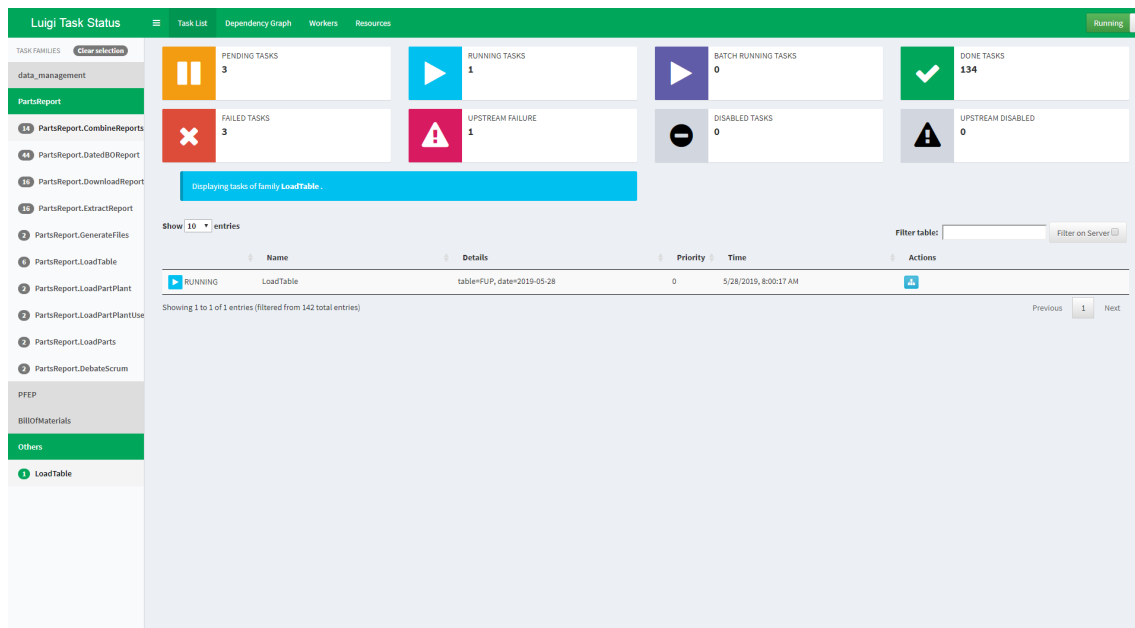


Figure 4.3: Luigi web interface [47]

- Development environment
  - 1x `t3a.xlarge` EC2 instance
  - 1x 30 GB EBS `gp2` volume

It is important to note that the configuration listed above did not include a load balancer or EC2 auto scaling. This is due to the fact that Luigi does not support distribution of execution [46].

Luigi has the same drawback as Apache Airflow: the lack of visual editing capabilities. The tool displays the finished workflow in a visual way as seen on Figure 4.4, but this is not editable and has an interface that is less usable than the one provided by Apache Airflow. Therefore, this solution was disregarded due to the lack of visual editing capabilities and scalability, which are mandatory requirements.

#### 4.3.4 OutSystems Platform

As highlighted by Gartner [23], some organizations are using orchestration and automation capabilities for both security and non-security use cases, as there are some crossover with business processes automation typically delivered by low-code platforms, such as the OutSystems Platform. The ability to create workflows in a visual and low-code way is a key enabler of adoption.

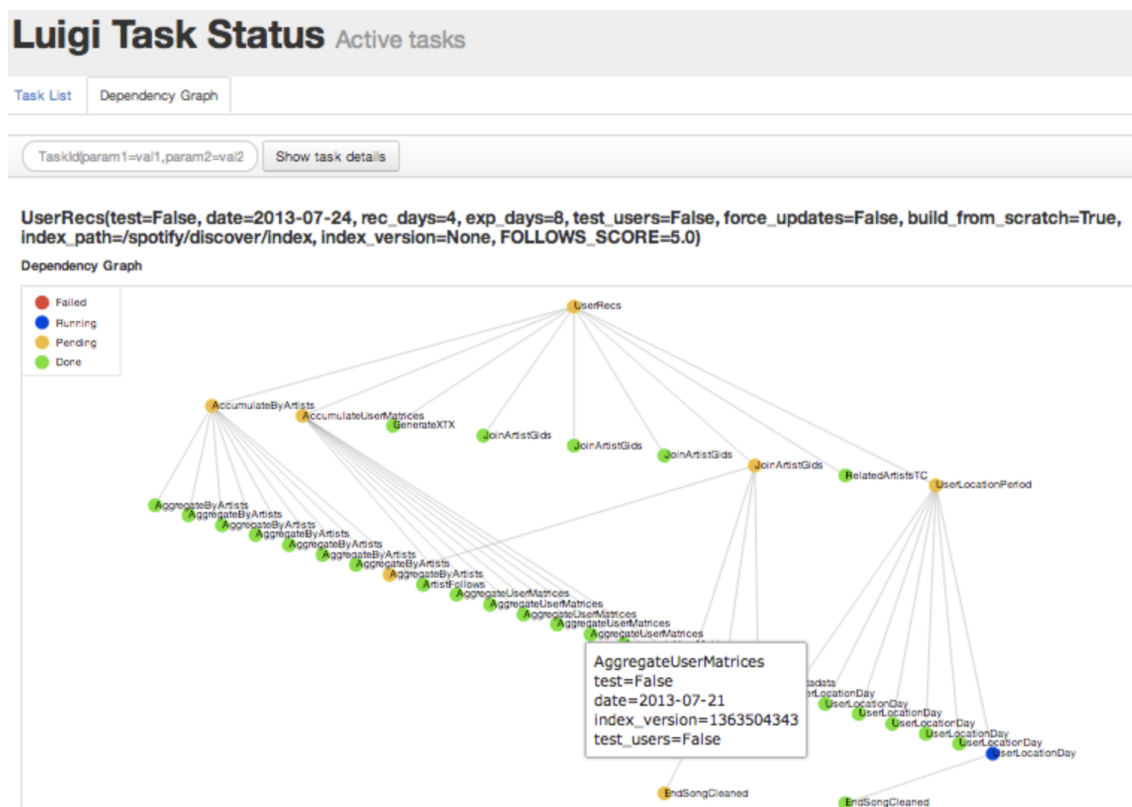


Figure 4.4: Luigi workflow graph [47]

Using the OutSystems Platform to build a SOAR is also a possible solution, but ultimately was disregarded because the whole SOC team do not have vast experience in building entire systems using the Platform, the team's know-how is about the Platform's underlying infrastructure security and not developing in OutSystems. For the sake of completeness of the analysis, the author considered the cost of this solution as \$1,512.50/month [62], which is the starting price of the OutSystems Platform Standard version that includes one production and one development environment.

### 4.3.5 Kubernetes

Kubernetes is a popular open-source container orchestration solution used by many companies to build fast, scalable and manageable systems. The SOC did not want the burden of maintaining the underlying Kubernetes infrastructure, so this solution only considered managed services such as Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS), and Google Kubernetes Engine (GKE).

GKE and AKS were crossed out because OutSystems did not have any managed Kubernetes clusters on those cloud providers by the time of this publication and for the sake of alignment with internal company policies and architectures the SOC wanted to

be aligned with the *modus operandi* of the rest of the company, which relied on EKS for managed Kubernetes.

Amazon EKS runs a single tenant Kubernetes control plane for each cluster. The control plane infrastructure isn't shared across clusters or AWS accounts. The control plane consists of at least two API server instances and three etcd instances that run across three Availability Zones within an AWS Region. A cluster consists of a control plane and the Amazon EC2 or AWS Fargate compute where the pods run [21].

There are two pricing models for EKS: on-demand and savings plan. For on-demand, you pay for what you use by the second, with no long-term commitments or upfront payments. On the other hand, by using a savings plan you make a commitment to a consistent usage for a term length of 1 or 3 years, which grants you a significant discount in the hourly price (up to 60% in some cases).

Even though the savings plan provided the best deal, the decision was to only consider the on-demand pricing model to avoid any long term commitment. The EKS cost was \$586.28/month for the configuration listed below.

- Production environment
  - 1x EKS cluster
  - 4x `c5.large` EC2 instances (managed nodes)
  - 4x 30 GB EBS `gp2` volumes
- Development environment
  - 1x EKS cluster
  - 2x `c5.large` EC2 instances (managed nodes)
  - 2x 30 GB EBS `gp2` volumes

This solution was disregarded because the front-end for providing visual playbook editing would need to be created and maintained by the SOC, which is an undesired overhead for the OutSystems' SOC team.

### 4.3.6 Cloud Services

There are many companies that provides cloud services (e.g. computing, storage, network, etc.) such as AWS, Microsoft, Google, and others. One of the most sought-after service is cloud computing. There are many definitions for cloud computing, below are the ones given by the three major cloud providers. Their definitions share the same key concepts: resources/services delivered via the Internet, as a service business model, on-demand, pay-per-use, and the shift from CapEx to OpEx.

- AWS: “Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider.”
- Microsoft Azure: “Simply put, cloud computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the Internet (“the cloud”) to offer faster innovation, flexible resources, and economies of scale. You typically pay only for cloud services you use, helping you lower your operating costs, run your infrastructure more efficiently, and scale as your business needs change.”
- Google Cloud: “Cloud computing is the on-demand availability of computing resources as services over the internet. It eliminates the need for enterprises to procure, configure, or manage resources themselves, and they only pay for what they use.”

As previously stated, AWS has shown [69] that it is possible to orchestrate the security incident response by using a combination of cloud services. The same concept applies to other major cloud providers such as Microsoft Azure and Google Cloud. The use of cloud services inherently accomplishes many of the functional and non-functional requirements and facilitate the others. Table 4.8 shows the cloud services that could be used for the purpose of building a cloud native SOAR solution followed by a brief analysis. A more detailed analysis of the services and the motivation behind their use is in Chapter 5.

<b>Svc. / Provider</b>	<b>AWS</b>	<b>Azure</b>	<b>Google Cloud</b>
Messaging	SNS	Service Bus	Pub/Sub
Compute	Lambda	Functions	Functions
Secrets Mgmt.	Secrets Manager	Key Vault	Secret Manager
Workflows	Step Functions	Logic Apps	Workflows
CI/CD	CodePipeline	DevOps	Build
Deployment	CloudFormation	Deployment Manager	Deployment Manager

Table 4.8: Comparison of potential cloud services for building a SOAR

In regards to the functional requirements, the SOAR system is cloud native (F.R.05) and built using Python (F.R.23). The cloud services can be managed via a web browser to access the HTTPS management console (F.R.01) and also via dedicated APIs for each service (F.R.06). The playbooks’ visual editing capabilities (F.R.02) are provided by AWS Step Functions including the ability to create parallel and sequential actions (F.R.03) and

playbooks' version control (F.R.22) is provided by making use of Git via GitHub and CI/CD capabilities from AWS CodePipeline.

Additionally, the SOAR system provides audit logs in JSON format and the cloud services provides audit logs in popular/parsable formats (F.R.07), both of which can be exported to external systems (F.R.04). SSO capabilities (F.R.08) and RBAC (F.R.09) are provided by AWS Identity Center (formerly AWS SSO) and AWS IAM, respectively.

Finally, the various functional requirements regarding integration with systems (F.R.10 to F.R.21) are possible to implement in this approach, but they are highly dependent on the availability of the systems' APIs and plenty of time for building the integrations from scratch as there is nothing available from day one as it happens with COTS SOAR.

In regards to the non-functional requirements, the SOAR system: uses AWS Lambda functions (NF.R.01); implements security best practices such as input validation, error handling, data encryption, secrets management via AWS Secrets Manager, network isolation, high availability, access control, etc. (NF.R.02); executes tasks 112 times faster than a human and with greater accuracy (NF.R.03); uses AWS CloudFormation to deploy assets and is also able to scale up and down according to the demand (NF.R.04); and is cost efficient, costing less than 0.25 FTE per month (NF.R.05). Such accomplishments will be detailed in Chapter 6.

It is important to note that the use of cloud services does come with intrinsic risks that must be addressed to avoid breaches or to at least have lesser impacts if a breach occurs. SANS cloud survey [68] listed the biggest concerns of cloud customers and below are the top five:

1. Unauthorized (rogue) application components or compute instances (54.4%)
2. Poorly configured or insecure interfaces or APIs (52.4%)
3. Inability to respond to incidents traversing our cloud apps and data (51.2%)
4. Unauthorized access by outsiders (50.8%)
5. Lack of skills within the organization for specific public cloud services (46.8%)

## 4.4 Selected approach

Table 4.9 shows a summary of the alternative approaches to solution, their cost, and a brief conclusion about each. The selected approach, using AWS cloud services for building a low-cost and cloud native SOAR system, was the most aligned with the requirements defined in Sections 4.1 and 4.2. The architecture details of the selected approach are presented in Chapter 5.

Approach	Cost/month	Conclusion
COTS SOAR	\$10K - \$30K	Three vendors met most of the requirements, but this approach was eliminated because no vendor met the non-functional requirement NF.R.05 “cost efficiency”.
Apache Airflow	\$1,931.38	Mature workflow platform with an active community and available as a cloud managed service, but was eliminated due to the lack of built-in visual editing capabilities (F.R.02) and for not being cost-efficient (NF.R.05).
Luigi	\$364.01	Eliminated due to the lack of visual editing capabilities (F.R.02) and elasticity (NF.R.04). Not as mature as Apache Airflow, between these two open-source options, Luigi would be the least preferred anyway.
OutSystems	\$1,512.50	Eliminated due to the team’s lack of know-how in “programming” with OutSystems. The SOC mainly uses Python and learning a new programming paradigm is out of the scope of this project.
Kubernetes	\$586.28	Eliminated due to the need of building and maintaining the web front-end for the visual editor ourselves. The SOC team is not a web development team and such tasks are out of the scope.
AWS Services	\$63.26	Selected approach. The best fit for the SOC needs, was able to meet all the non-functional requirements and most of the functional requirements while having an expressively lower cost.

Table 4.9: Comparison of the alternative approaches





# Chapter 5

## Architecture and Implementation

This Chapter is divided into two sections: design and technologies. The first focuses on presenting the system architecture and a brief overview of it. The second dives into the technologies used, their limits, costs, and the motivation behind their use.

### 5.1 Design

This project proposes a SOAR system that is serverless, secure, fast, scalable, and cost-efficient. As shown in Figure 5.1, the system architecture includes many parties such as the code repository system, the SIEM system, the chat application, and multiple AWS services. These parties are numbered and briefly described below and a more in-depth description is presented in Section 5.2.

1. **Code repository:** the source code and supporting files are hosted in a private repository in GitHub Cloud. There are 2 branches: dev (development) and main (production). Branch protection is activated for the main branch, so changes are only possible via Pull Request (PR) that must be analyzed and approved by the repository maintainers. When the PR is applied to the main branch, it automatically triggers the production pipeline in AWS CodePipeline.
2. **AWS CodePipeline:** the pipeline has 3 stages: source, build, and deploy. The source stage is triggered via the GitHub Cloud integration that was mentioned above. The build stage receives the code and supporting files from the source stage to perform the compilation and sends it to the next stage. In the final stage, deploy, AWS CloudFormation takes care of the deployment.
3. **AWS CloudFormation:** creates, updates, or deletes the resources according to the changes that were pushed. All the SOAR system resources such as the Lambda functions, Lambda Layers, Step Functions, and many others are centrally managed by CloudFormation. The entire system can be deployed and decommissioned with

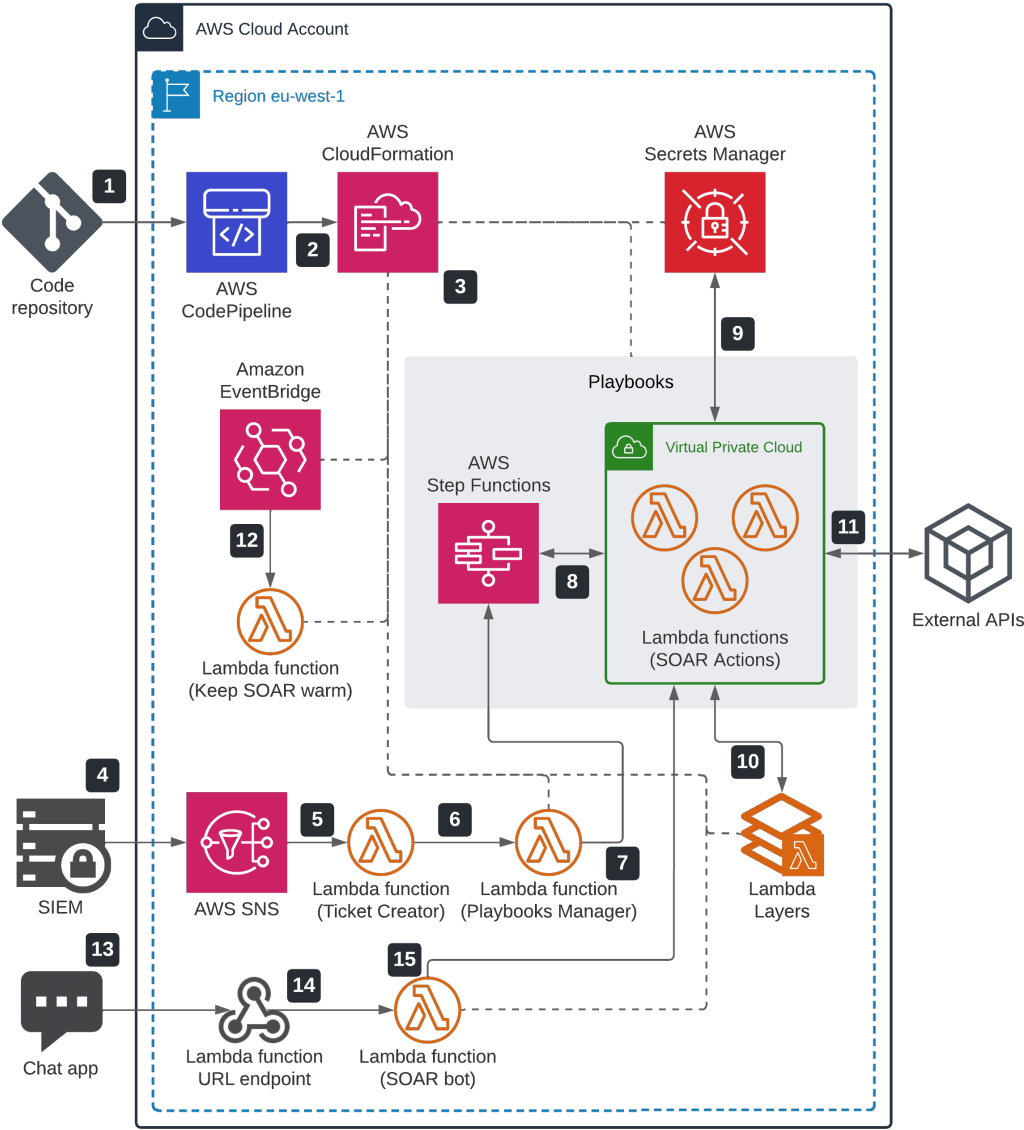


Figure 5.1: SOAR system architecture diagram

one-click. The dotted lines in the diagram connects the CloudFormation to all the resources it manages.

4. **SIEM:** the SOAR system is SIEM-agnostic. The only requirement is that the system is able to send messages to an AWS SNS topic (all major SIEM vendors are capable of doing that natively or via add-ons). For every new security incident alert, the SIEM sends a message with the relevant data (*e.g.* username, URL, IP, hash, etc.) to the pre-configured SNS topic.
5. **SNS:** the SNS topic receives the alert message and automatically triggers the Ticket Creator Lambda function and passes the message to it. This is a publish–subscribe messaging pattern where the SIEM is the publisher and the Ticket Creator is the subscriber.
6. **Ticket Creator:** this Lambda function creates the security incident ticket in the ticketing system used by the organization. If the execution is successful, this function will trigger another one, the Playbooks Manager, and pass the alert message to it.
7. **Playbooks Manager:** this Lambda function triggers the execution of an AWS Step Function’s state machine and pass the alert message to it, if there is one created for that given type of alert. The state machines are the automated playbooks. In case there is no automated playbook, the function logs the execution details (as it always does) and exits gracefully.
8. **AWS Step Functions:** the automated playbooks are basically a collection of sequential and concurrent actions placed in a visual workflow that has logical capabilities (*e.g.* if-else, loops, etc.). The playbook’s original input is the alert message, but within the workflow the output of one action can be used as input for another one. An automated playbook may use one or more actions (*e.g.* check IP reputation, get user details, add a hash to the threat intel database, etc.).
9. **AWS Secrets Manager:** the actions are grouped into modules (*e.g.* the module AbuseIPDB has the actions check IP, check subnet, report IP, etc.). Each module has one dedicated Secrets Manager object to securely store sensitive data such as username, password, API key, etc. The actions have read-only access to their module’s secret object to retrieve the necessary data during runtime.
10. **AWS Lambda Layers:** the modules are Lambda layers that are imported into the Action Lambda functions. There are also other supporting Lambda layers such as the SOAR utils (utility code that is used by all the actions) and the AWS Lambda Powertools [11].

11. **SOAR Actions:** these Lambda functions use the dedicated Virtual Private Cloud (VPC) with 2 static public IPs to interact with the Internet. By default, a Lambda interacts with the Internet using a random IP allocated by AWS during runtime, but this is not viable for the SOAR system because some APIs require adding the IP to an allowlist and ultimately because using a VPC provides more control and isolation capabilities.
12. **Amazon EventBridge / Keep SOAR warm:** when talking about Lambda usage at scale, which is the case of this work, cold start avoidance is a recurring topic [22]. To avoid cold start and to make the SOAR Actions promptly available, there is a scheduled EventBridge rule that triggers the Keep SOAR warm Lambda function which then sends a heartbeat to all the SOAR Actions to keep them cached within the AWS infrastructure.
13. **Chat app:** this and the remaining items are not mandatory for the full operation of the SOAR system, they are part of an optional feature that provides the capability of executing the SOAR Actions in an ad hoc manner via chatbot. The analyst interacts with the SOAR bot using a chat application which triggers a webhook.
14. **Lambda function URL endpoint:** the webhook is a Lambda function URL endpoint that is exposed to the Internet via HTTPS. The endpoint receives the message payload from the Chat app and triggers the SOAR bot Lambda function and pass the payload to it.
15. **SOAR bot:** this Lambda function ensures authentication and authorization of the request before calling the intended action. Once the action execution is complete, which usually takes milliseconds due to the caching, the response is sent back to the chat where the user invoked the bot.

It's important to note that the deployment, configuration, and usage of the SIEM (item 4), the AWS SNS topic (item 5), and the Ticket Creator (item 6) are out of the scope of this work. These resources were already set up and in production prior to this project. They were included in the diagram for better understanding of the entire architecture, but they were not deployed nor configured by the author. Except for these three resources, all the other resources in the diagram were fully deployed and configured by the author.

## 5.2 Technology

The technologies used are mainly AWS services, but there is also the SIEM system (*e.g.* Azure Sentinel, IBM QRadar, Splunk, etc.) and the code repository (*e.g.* Bitbucket, GitHub, GitLab, etc.) which are important pieces of the solution.

A single company can own multiple AWS accounts at the same time and only pay for the deployed resources, this is usually the case for large companies with specific business and technical demands. This capability is achieved by using AWS Organizations. The author considered a dedicated AWS account for the SOAR system to avoid taking into consideration unpredictable usage patterns from other solutions that might be deployed into a multipurpose account. The service limits presented herein took into consideration the AWS service endpoints and quotas documentation [20].

To calculate the costs of the solution the author considered a volume of 10K security incidents/month, intentionally extrapolating the number of security incidents to prove that the system can scale while keeping the costs low. To avoid long-term commitment, the solution considered the on-demand pricing available in the AWS Pricing Calculator [19] and not the Savings Plans pricing as they require a 1- or 3-year commitment. The discounts provided by the AWS Free Tier were also intentionally avoided.

Finally, the cost and availability of some services vary according to the region, so the author used the region `eu-west-1` (Ireland, Europe) as reference. The total cost of the solution is presented in Table 5.1 and a more detailed pricing structure is described in the following sub-sections. The full AWS Pricing Calculator quote is in Appendix A.

Description	Cost/month
SIEM	-
AWS SNS	\$0.27
AWS Lambda	\$3.72
AWS Secrets Manager	\$40.82
AWS Step Functions	\$12.50
AWS CodePipeline	\$6.05
AWS CloudFormation	\$0.00
Code repository	-
<b>TOTAL</b>	<b>\$63.26</b>

Table 5.1: Total cost of the solution per month

### 5.2.1 Splunk Cloud

Splunk Cloud is a SaaS data platform where you can ingest machine data to implement monitoring and observability use cases. OutSystems uses it as a SIEM to detect and alert on security incidents. Each alert can be configured to do one or more actions as a result of it being triggered, there are many built-in actions provided by Splunk itself (*e.g.* send an email, call a webhook, etc.) and additional actions can be installed via add-ons or created by the customer.

For the security incident alerts, the SOC uses a custom action provided by one of the installed add-ons, which sends the alert payload to an AWS SNS topic. As mentioned


before, this process has been in place before this project started and there was no need to change it to integrate with the SOAR system, this part was included as-is in the architecture. Figure 5.2 shows the action configuration in Splunk Cloud.

#### Trigger Actions

+ Add Actions ▾

When triggered

▼

 AWS SNS Alert Remove

Account \*

[AccountName]

Region \*

[AwsRegion]

Topic Name \*

[TopicName]

Correlation ID

\$result.correlation\_id\$

Entity

\$result.entity\$

Source

\$result.source\$

Timestamp

\$result.\_time\$

Event

\$result.\_raw\$

Message \*

[Alert specific data goes here]

Figure 5.2: Configuration of the AWS SNS Alert in Splunk Cloud

The alert payload contains data that is included by default such as time, title, description, and severity, but also alert specific data such as username, email, source IP, destination IP, hash, path, etc., that can be present or not depending on the type of the alert. Regardless of the type, the payload size sits between 1 and 8 KB as it's just a plain text with delimiters to separate the data for the subsequent parsing.

The costs related to the SIEM are not considered in the SOAR system's total cost as having one is not even mandatory, other services and systems like Amazon CloudWatch alarms and Extended Detection and Response (XDR) systems can also generate incident alerts. The only input the system needs to start the whole orchestration and automation process is the alert payload that is delivered to the Playbooks Manager Lambda function.

### 5.2.2 AWS Simple Notification Service

Amazon SNS is a fully managed messaging service for both Application-to-Application (A2A) and Application-to-Person (A2P) communication. The A2A pub/sub functionality provides topics for high-throughput, push-based, many-to-many messaging between distributed systems, microservices, and event-driven serverless applications [4]. An overview of the main features of this service is presented in Figure 5.3. In this project, Splunk Cloud is the publisher and the Ticket Creator Lambda function is the subscriber.



Figure 5.3: AWS SNS overview [4]

Even though AWS SNS was used in this project due to the already deployed solution for creating tickets, there are other managed messaging services like Google Cloud Pub/Sub and Azure Service Bus that have similar features and pricing.

As previously mentioned, the SIEM Splunk Cloud sends the alert payload to a specific SNS topic. Every new message added to the topic triggers the Lambda subscriber which receives the alert payload and interacts with the company's ticketing system to create the ticket. The system uses this setup to receive the alert payload that triggers the execution of the associated automated playbook.

There are some relevant service limits that the author took into consideration when designing the architecture, the main one being the 256 KB maximum payload size allowed by SNS. The relevant limits are presented in Table 5.2.

AWS SNS pricing is competitive, they don't charge to deliver messages to Lambda subscribers, there are only data transfer costs as shown in Table 5.3. For cost reference, the author considered the maximum alert payload size of 256 KB, which is 32 times bigger than the biggest alert payload currently in production (8 KB), so the cost estimate errs on the side of caution. Even with the extrapolated payload size and the large numbers of messages, the SNS costs were just \$0.27/month.

Item	Limit	Comment
Standard topics/account	100K	The system only needs 1 standard topic (the one that receives the alerts payload).
Subscriptions/std. topic	100M	The system only needs 1 Lambda subscriber configured in the std. topic mentioned above.
Messages/second	9K	Each message is equivalent to one security incident and we considered a high volume of 10K security incidents/month, which is still way below the limit. To reach this limit the company would have to have 9K security incidents/second, which is extremely unusual.
Payload size/message	256 KB	The majority of the alert payload sizes are between 1 and 3 KB, with some outliers going as high as 8 KB, which is still safely below the limit. As mentioned, the alert payloads are plain text with delimiters, so it is expected to have a small size.

Table 5.2: AWS SNS limits

Description	Cost/month
10K messages to a Lambda subscriber (1 message is equivalent to 1 alert)	\$0.00
3 GB outbound data transfer (256 KB * 10K messages = 2.56 GB, rounded upward to 3 GB)	\$0.27
<b>TOTAL</b>	<b>\$0.27</b>

Table 5.3: AWS SNS costs

### 5.2.3 AWS Lambda

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and SaaS applications, and only pay for what you use. It natively supports many programming languages like Python, Java, Node.js, C#, Go, etc. [10]

There are other managed compute services, some are serverless like Google Cloud Functions and Azure Functions and others are based on virtual machines such as AWS EC2, Azure Virtual Machines (VM), and Google Compute Engine. Additionally, AWS also has a serverless compute service focused on containers called Fargate.

In short, the decision to go with Lambda was straightforward: services based on VMs were dismissed due to the serverless requirement, AWS was the cloud provider of choice for this project thus eliminating Google Cloud and Azure, and AWS Fargate focused on containers, which is a technology the SOC did not want to implement in this project as explained in the subsection 4.3.5.



The author created the system using Python, but this is not a requirement as the logic behind the code can be easily implemented in any high-level programming language. An important feature that was also used is the AWS Lambda Layers [18], as they can store custom libraries and dependencies which are then imported into the main Lambda function, making the update and maintenance process easier and faster. The Lambda functions and Lambda Layers that are presented in Figure 5.1 are described below:

- **Ticket Creator:** this Lambda is triggered by the SNS topic to create the ticket in the SOC queue within the ticketing system used by OutSystems. Nothing was changed in the actual code, it works independently from the SOAR. On the other hand, we did a simple configuration change in the Lambda settings to enable a Lambda Destination, so the Playbooks Manager Lambda can be triggered after this one is executed successfully.
- **Playbooks Manager:** this Lambda is responsible for receiving the alert payload from the Ticket Creator Lambda and running the automated playbook that is associated with that alert. There can be cases where this is triggered but there is no automated playbook for that specific alert, in such cases the Lambda will just log the attempt and finish silently as this behaviour is expected.
- **SOAR Action:** there is one SOAR Action Lambda function for each action (*e.g.* check an IP on AbuseIPDB, report an IP to the AbuseIPDB, etc.). These actions work like building blocks to construct the automated playbooks. These functions are responsible for input validation and for calling the intended module to interact with the API.
- **SOAR Module:** there is one SOAR Module Lambda Layer for each system the SOAR integrates with (*e.g.* AbuseIPDB, VirusTotal, PhishTank, etc.). The actual code that implements the logic for interacting with the APIs is in the modules, the actions are just callers.
- **SOAR Utils:** this Lambda Layer contains libraries and dependencies that are transversal to all Lambda functions.
- **Keep SOAR warm:** this Lambda function runs on a schedule (by default, it is every 3 minutes) and sends an asynchronous heartbeat to all the SOAR Action Lambda functions to keep them cached within the AWS infrastructure to avoid cold-start delays. The Amazon EventBridge rule that triggers this function has no cost, as it is using the default bus.
- **AWS Lambda Powertools:** this Lambda Layer was created and is maintained by AWS. It is a suite of utilities to ease adopting best practices such as tracing, struc-

tured logging, custom metrics, and more [11]. The system uses it for logging, input validation, and parameters.

- **SOAR bot:** this Lambda function is responsible for receiving the message payload from the SOAR bot webhook and ensuring the authentication and authorization of the request before passing it to the intended SOAR action.

It's important to note that the communication between the SOAR Actions and the External APIs shown in Figure 5.1 is encrypted using Transport Layer Security (TLS) 1.2 or higher to protect data in transit. To guarantee isolation and traceability, the Lambda functions are inside a dedicated VPC and all the communications are initiated by the SOAR system using its two dedicated public IPs. Also, like any service, the AWS Lambda has limits that were taken into consideration when designing the architecture. Table 5.4 shows the limits that are relevant to the system.

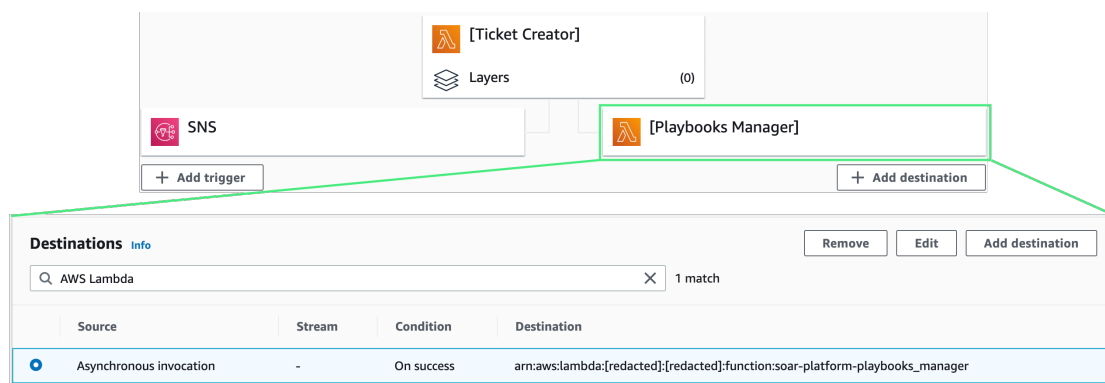


Figure 5.4: AWS Lambda Destination configuration

Finally, estimating the Lambda costs was more complex than the other services as there are many variables involved. For cost reference, the author considered the 10K security incidents/month, an average of 30 actions/incident, and 100 modules with an average of 10 actions each. As mentioned in Table 5.4, each Lambda function has 512 MB of RAM and 512 MB of ephemeral storage. The average execution durations were taken from the SOAR system metrics for July 2022. The Lambda usage is described below:

- **Playbooks Manager**
  - Executions (each security incident is 1 execution): 10K executions/month
  - Average execution duration: 11 milliseconds
- **Keep SOAR warm**
  - Executions (scheduled to run every 3 minutes): 14.6K executions/month

Item	Limit	Comment
Execution timeout	15 min.	The configured value is 1 minute. The fastest actions take less than a second to execute and the slowest ones take less than a minute. Actions that are intrinsically more time consuming ( <i>e.g.</i> SSL/TLS site scan) are built using a job queue structure to avoid idling Lambda resources and hitting the timeout.
Concurrent executions	1K	The configured value is 1K. The actions are executed in a very short time span, so having 1K actions running at the same time is unlikely. This is a soft limit that can be increased to tens of thousands.
RAM	10 GB	The configured value is 512 MB. The actions are not resource intensive. Analysis were made using AWS's Lambda Power Tuning utility, which indicated that 512 MB is the sweet spot for balancing cost and performance.
Ephemeral storage	10 GB	The configured value is 512 MB. The actions do not save temporary files to the ephemeral storage, so this value was left as the minimum possible.
Environment variable size	4 KB	The actions use environment variables to store the execution log level ( <i>e.g.</i> debug, info, warning, etc.) and the Amazon Resource Name (ARN) of the Secrets Manager's object that is associated with that action. Considering the env vars set by AWS and the ones created by the SOAR, the final size is less than 1 KB.
Invocation payload size	256 KB	As mentioned, the biggest alert payload is just 8 KB and once this payload arrives at the automated playbook, each action receives only the data that is relevant to it and not the complete payload ( <i>e.g.</i> AbuseIPDB's check IP only receives the IP).
Deployment package size	250 MB	Both the Lambda and the Lambda Layers deployment package sizes are way below this limit, the biggest one is around 3 MB (unzipped).
Attached Lambda Layers	5	This is the only concerning limit as the actions need 3 layers attached to the Lambda at all times: the AWS Lambda Powertools library, the SOAR utils library, and the SOAR module library. This leaves only 2 free slots. Unfortunately, this is a hard limit that can not be increased via request.

Table 5.4: AWS Lambda limits

- Average execution duration: 3 seconds
- SOAR Action (Keep SOAR warm calls)
  - Executions (14.6K heartbeat calls \* 1K actions): 14.6M executions/month
  - Average execution duration: 1 millisecond
- SOAR Action (Playbook calls)
  - Executions (10K incidents/month \* 30 actions each): 300K executions/month
  - Average execution duration: 105 milliseconds
- SOAR Action (SOAR bot calls)
  - The executions triggered by the SOAR bot were not considered as this is an optional feature. The bot is used in cases where there is no automated playbook for a given alert, which is out of the scope of this project. For transparency and to show how well the system scales, even if we consider a SOC team with 30 analysts working 8 hours/day and each of them using the bot 5 times every minute of every day (including weekends), we would have a rate of 72K executions/day which represents a cost of \$0.24/month.
  - It is important to note that the average SOAR Action execution duration remains as 105 milliseconds for the SOAR bot calls, as it is the same SOAR Action Lambda function that is executed, it is just triggered in a different way.
- SOAR bot
  - For the same reason explained above, these executions were not considered in the costs estimate. Taking into consideration the 72K executions/day and that the average execution duration of this Lambda function is 563 milliseconds, we would have a cost of \$0.44/month.

The lambda costs are \$3.72/month as described in Table 5.5. It is worth noticing that the “Keep SOAR warm” and the “SOAR Action (Keep SOAR warm calls)” combined represent 91.3% (\$3.40) of the monthly Lambda costs, but these costs are stable and predictable as they only depend on the number of actions and the heartbeat cadence (every 3 minutes by default).

The use of Lambda caching is paramount as it improves the user experience when interacting with the SOAR bot. With caching, the user consistently receives the replies in less than 2 seconds, instead of waiting 5+ seconds for the answer as there would be cold start delay for the SOAR bot Lambda and for the SOAR Action Lambda.

The caching also decreases the playbooks execution time. For example, a playbook with 30 actions without caching would have 30 to 90 seconds added to its total execution time due to the cold start delay of each action. The cold start would also affect the Playbooks Manager Lambda, adding 1 to 3 seconds to each playbook execution as well.

To prove the caching costs are manageable and reasonable, we can multiply the number of modules by ten, culminating in 1,000 modules with 10 actions each, which is an extremely high rate of integrations that is not even provided by major COTS SOAR products such as Palo Alto XSOAR (800+), Splunk SOAR (370+), and Siemplify (320+) [23] and end up with a cost of only \$34.07/month for maintaining this caching setup.

When considering the 1,000 modules, the caching costs would not even be the major concern, instead it would be the unreasonable maintenance efforts to keep all those modules up-to-date and working as expected as the External APIs change over time.

Description	Cost/month
Playbooks Manager	\$0.00
Keep SOAR warm	\$0.36
SOAR Action (Keep SOAR warm calls)	\$3.04
SOAR Action (Playbook calls)	\$0.32
TOTAL	\$3.72

Table 5.5: AWS Lambda costs

## 5.2.4 AWS Secrets Manager

AWS Secrets Manager securely stores applications, services, and IT resources secrets. It enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets with a call to Secrets Manager API, eliminating the need to hardcode sensitive information in plain text [13]. The main aspects of the service are presented in Figure 5.5.

The system needs to secure sensitive data such as usernames, API keys, and access tokens both in transit and at rest. Data security in transit was already discussed in the sub-section 5.2.3. Now, for data security at rest, the chosen solution was AWS Secrets Manager due to its usability and scalability. Each secret is a JavaScript Object Notation (JSON) with key-values for storing the data.

This service was released to the general public in 2018, so it is a relatively new service offered by AWS. Before this, developers had only one option for storing secrets: the Systems Manager Parameter Store. Both services are very similar, but there are key differences that make the use of Secrets Manager more reasonable for storing and managing secrets, as presented in Table 5.6.

There are many Secrets Manager limits and the ones related to API calls are quite restrictive when compared to other services provided by AWS. They recommend avoid

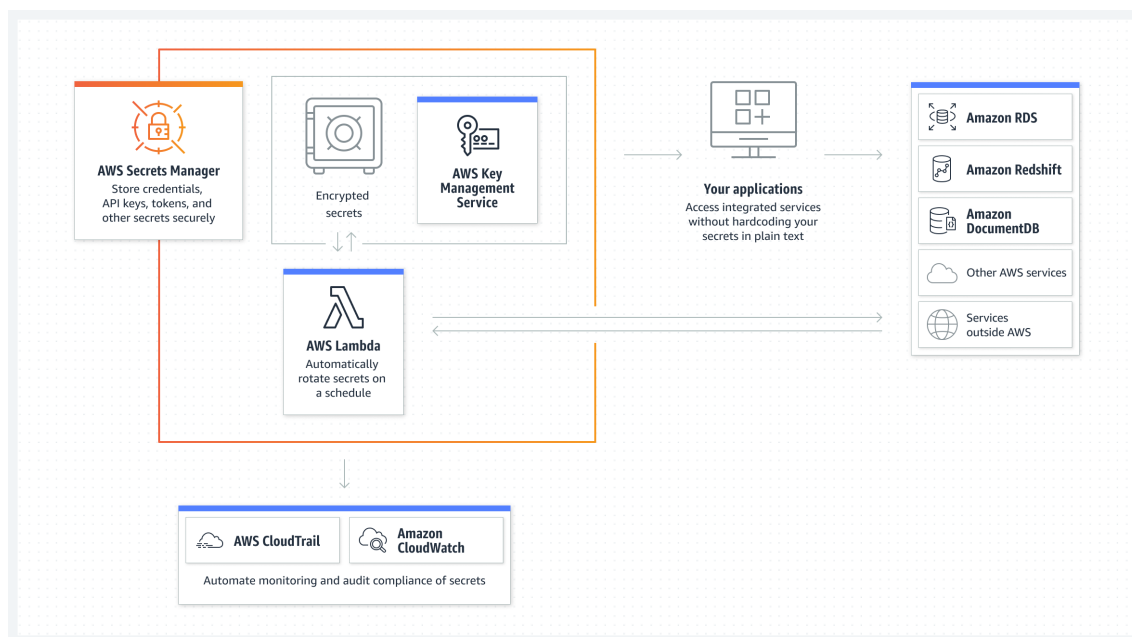


Figure 5.5: AWS Secrets Manager overview [13]

	Secrets Manager	Parameter Store (Standard)
Encryption of data at rest	Supported	Supported
Automatic secret rotation	Supported	Not supported
Generate random secrets	Supported	Not supported
Secrets/region	500K	10K
Maximum secret size	64 KB	4 KB
Pricing	Paid	Free

Table 5.6: AWS Secrets Manager vs Parameter Store (adapted)

calling `PutSecretValue` or `UpdateSecret` at a sustained rate of more than once every 10 minutes to avoid facing issues related to secret versions limit [14]. In the case of the SOAR system, the restrictive limits are not a problem as the architecture design took them into consideration and even at high load the system is safely below the limits. The relevant limits are described in Table 5.7.

Even though Secrets Manager is paid while the Parameter Store is free, the benefits it offers outweighs the cost. The pricing model is simple: pay for the number of secrets stored/hour and the number of API calls. For reference, the author considered the same 100 modules as previously used, which translates to 100 stored secrets as each module has 1 dedicated secret object. Add to that the 2 secrets used by the Playbooks Manager and the SOAR Utils library, culminating in a total of 102 secrets that are stored 24x7.

There are other ways to securely store data at rest in AWS, with services like Elastic Block Store (EBS), DynamoDB, and Simple Storage Service (S3) to name a few. The justification for not opting for them is described below.

Item	Limit	Comment
Secrets/region	500K	Apart from the global secret that is shared by all the modules ( <i>e.g.</i> platform name, version, etc.), each module contains only 1 secret object.
Secret size	64 KB	The secrets are JSONs with key-values such as URL, username, password, user-agent, etc. The biggest secret object in use is not even 1 KB in size.
GetSecretValue calls/sec	5K	The Lambda functions have a cache of the secrets that is updated just when necessary. Because of this setup, hitting this limit is unlikely.
CreateSecret calls/sec	50	The secrets are created in two moments: when the system is first deployed and when a new module is deployed. In both cases, AWS CloudFormation handles the rate limiting transparently.
UpdateSecret calls/sec	50	The secrets are updated according to the necessity ( <i>e.g.</i> API key rotation, username change, etc.). Even considering an unlikely high-rate of daily secret updates of all the secrets, this limit would not be reached.
DeleteSecret calls/sec	50	The secrets are deleted in two moments: when a module is deleted and when the system is decommissioned. In both cases, AWS CloudFormation handles the rate limiting transparently.

Table 5.7: AWS Secrets Manager limits

- EBS: this service is focused on block-level storage, so the secrets would be stored in files (*e.g.* JSON, INI) inside the encrypted EBS disk. For the Lambdas to be able to read the files, we would have to use yet another AWS service (EFS: Elastic File System) to bridge this communication.

There is also a need to create a custom solution to segregate the access to the files by Lambda or create 1 EBS disk per integration, in order to mimic the access control capability provided out of the box by Secrets Manager. All of this would add unnecessary complexity to the architecture.

- S3: this service is focused on object storage, so we would have issues similar to the ones mentioned in the EBS case.
- DynamoDB: this is a NoSQL key-value database service, making it a very good option for storing key-value data. Specifically for this project we decided to not use it because: the pricing model is way more complex than Secrets Manager (number of read and writes, peak read/write rate, storage usage, etc.); we would need to design a database model; and we would need to create a CRUD interface to abstract the interaction between the Lambdas and the database. In the end, that's a lot of

overhead for achieving the same results as Secrets Manager.

As previously mentioned, the SOAR Action Lambda functions are cached via the Keep SOAR warm heartbeats and that cache includes the secret object to avoid a high volume of unnecessary calls to the Secrets Manager. On the other hand, this approach may cause sync issues if a secret is updated in the Secrets Manager (*e.g.* replace an API key) as the heartbeat calls do not include a secret refresh. To mitigate that, the system has a cache invalidation mechanism to purge the cache of all the actions or just the ones related to a given module whenever necessary.

Considering the daily secret updates of all the secrets, we would have 102 API calls/day which added to the 102 secrets stored 24x7 ends up with a cost of \$40.82/month as shown in Table 5.8.

Description	Cost/month
102 secrets	\$40.80
3.1K API calls	\$0.02
TOTAL	\$40.82

Table 5.8: AWS Secrets Manager costs

The Secrets Manager's cost is by far the highest one when compared to all the other AWS services used by the system, but data security at rest and secrets isolation are a must-have and using any other method for storing data such as Lambda environment variables or Parameter Store were not acceptable due to security, maintainability, and/or scalability concerns.

### 5.2.5 AWS Step Functions

AWS Step Functions is a low-code, visual workflow service that developers use to build distributed applications, automate IT and business processes, and build data and machine learning pipelines using AWS services. Workflows manage failures, retries, parallelization, service integrations, and observability so developers can focus on higher-value business logic. [16]. The Step Functions Workflow Studio is show in Figure 5.6.

There are two types of Step Functions workflows, Standard and Express, the main differences between them are presented in Table 5.9. Most importantly, the maximum duration of 5 minutes of the Express workflows made them inviable for the system as there can be automated playbooks that needs to run for more than that, for example, a playbook that is waiting for the analyst's feedback to proceed with a given action.

The system uses the Step Functions' state machines as automated playbooks in a 1:1 correspondence. These playbooks contains one or more SOAR Action Lambda functions to execute the actions the analyst would do during an incident response. The playbooks contains logical flows such as if-else, loops, sleep, etc. and are able to make simple



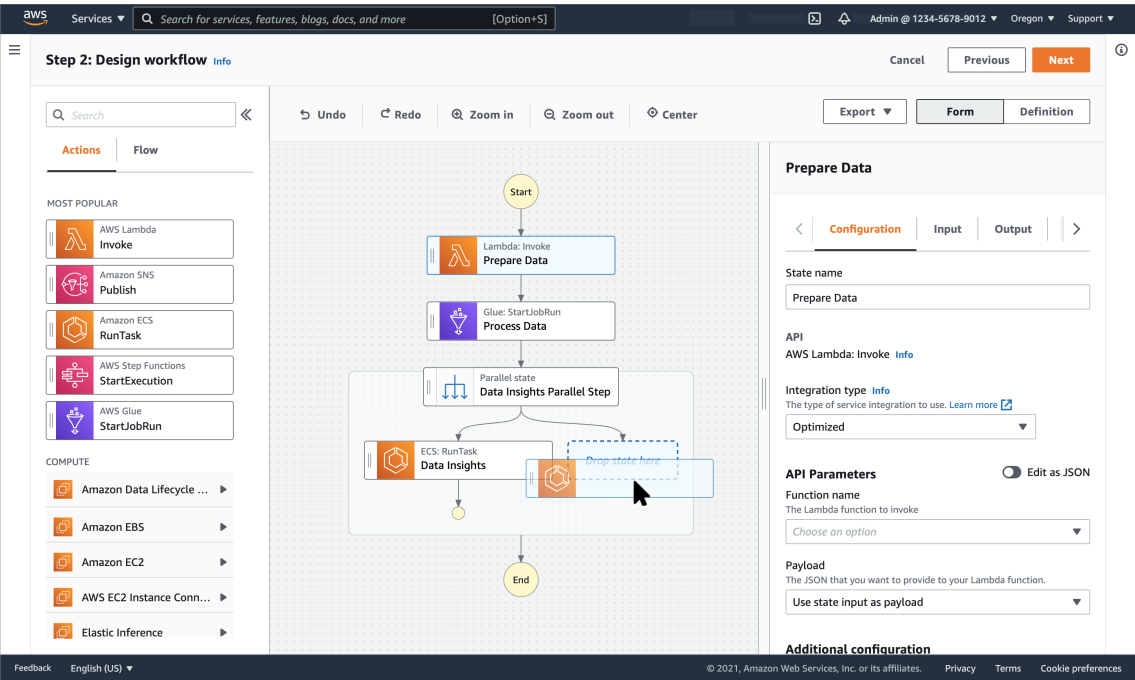


Figure 5.6: AWS Step Functions Workflow Studio [16]

	Standard	Express
Max duration	1 year	5 minutes
Execution start rate	2K/second	100K/second
State transition rate	4K/second	“nearly unlimited”
Pricing	Priced per transition. It is counted each time a step is completed	Priced by the number of executions, duration, and memory consumption

Table 5.9: AWS Step Functions: Standard vs Express [17]

decisions. On the other hand, tasks that requires critical-thinking or are too complex to be automated are not included.

There are many Step Functions limits: state throttling, API action throttling, state machine executions, and task executions to name a few. First, all the limits related to the creation, update and deletion of the state machines are handled by AWS CloudFormation in a transparent way. Second, the limits that are relevant to the system are presented in Table 5.10.

The system uses Standard Workflows so the costs estimate is straightforward. It’s important to note that there is no cost for having a state machine, you are only charged when they are executed. Considering the 10K security incidents/month (that is 10K state machine executions/month) and an average of 50 state transitions/state machine, we have a cost of \$12.50/month as described in Table 5.11.

Item	Limit	Comment
State machines/account	10K	Hitting this limit is unlikely as one security playbook is equivalent to one state machine, and the SOC team have less than 1K playbooks. This is a soft limit that can be increased to 23K.
States/state machine	10K	Roughly, each state represents a manual action that would be done by the analyst and there is no playbook that requires more than 50 actions. This is a soft limit that can be increased to tens of thousands.
Request size (header included)	1 MB	The alert payload is capped at 256 KB due to the SNS limit, leaving 768 KB for the header, which is plenty already. This is a hard limit.
Concurrent executions/region	1M	The 9K messages/second SNS limit would be a bottleneck long before the system reaches this limit. This is a soft limit that can be increased to millions.
Execution time-out	1 year	Most automated playbooks do not require human intervention so they finish in minutes, at most. For the ones that require, it is expected that an analyst will be available in minutes or hours. This is a hard limit.
Execution start rate	2K/sec.	To reach this limit, the system would have to ingest 2K security incidents/second, which is unlikely. This is a soft limit that can be increased (the maximum value was not disclosed by AWS).
State transition rate	4K/sec.	This limit is relevant when the state machine has large scale concurrent loops. This is not the case of the automated playbooks, where even the biggest concurrent loops are in the hundreds transitions/second. This is a soft limit that can be increased (the maximum value was not disclosed by AWS).

Table 5.10: AWS Standard Step Functions limits

Description	Cost/month
10K Standard Workflow requests	-
50 state transitions/workflow	-
TOTAL	\$12.40

Table 5.11: AWS Step Functions costs

### 5.2.6 AWS CodePipeline

AWS CodePipeline is a fully managed continuous delivery service that automates release pipelines for fast and reliable application and infrastructure updates. It automates the build, test, and deploy phases of the release process every time there is a code change [8].

Figure 5.7 shows the Platform's production pipeline.

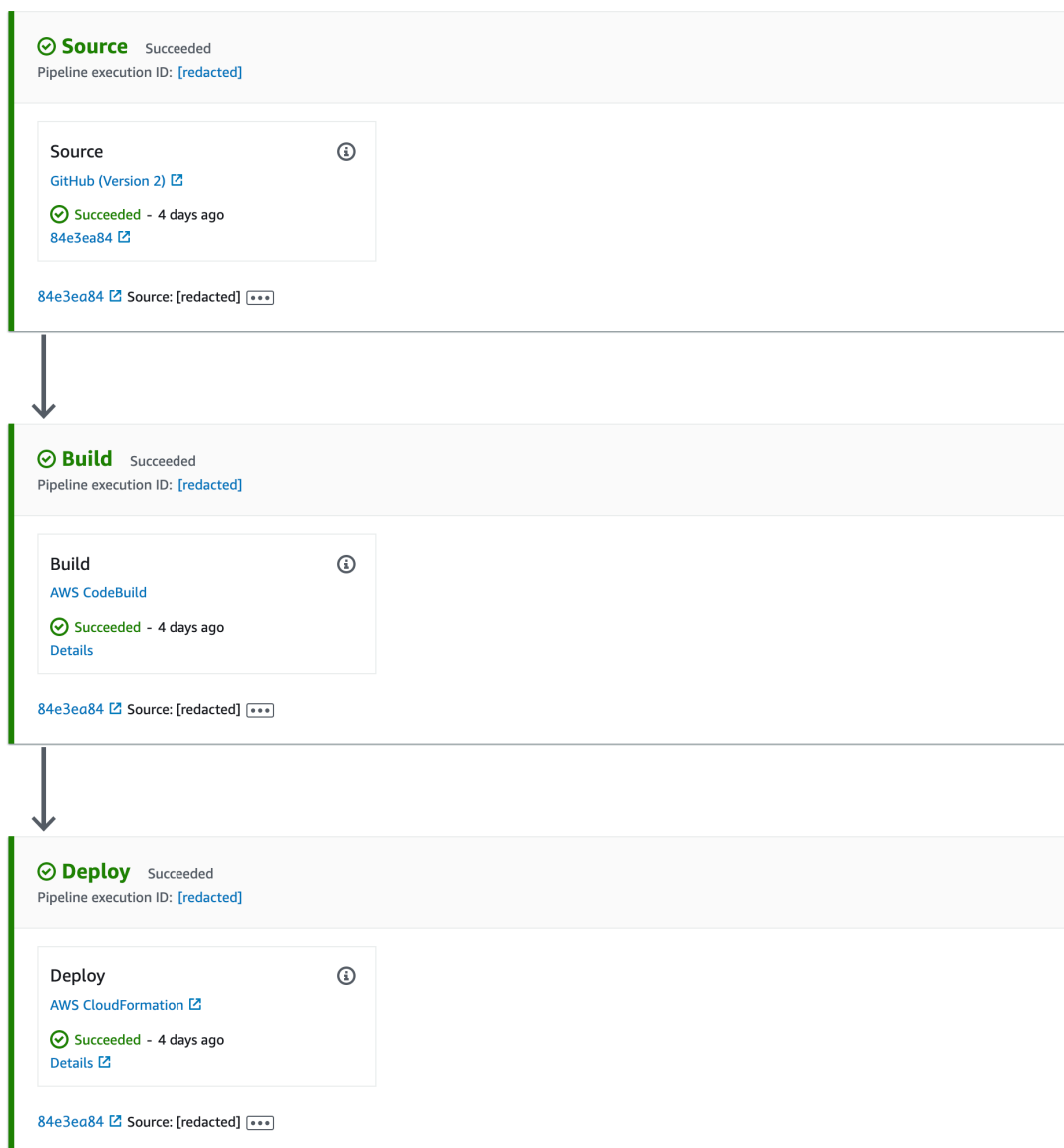


Figure 5.7: SOAR system production pipeline

There are other Continuous Integration and Continuous Delivery (CI/CD) solutions like Jenkins, TeamCity, CircleCI, etc. The decision to use AWS CodePipeline was pretty straightforward as the OutSystems' SOC already had experience with it. It made sense to use a tried and tested solution instead of investing resources (budget and time) into evaluating, buying and learning a new CI/CD solution just for this project.

The system has 2 pipelines: production and development. Both pipelines have the same stages (Source, Build, and Deploy) and very similar configurations. In the Source stage, GitHub was configured as the action provider, it points to the SOAR system code repository and automatically starts the pipeline when a source code change is detected. Each pipeline points to a different branch in the repository (development points to the dev

branch and production points to the main branch).

In the Build stage, for each pipeline AWS CodeBuild compiles the source code and the CloudFormation template to create the package that is sent to the Deploy stage. A compute instance with 3 GB of RAM, 2 vCPU, and the operating system Amazon Linux 2 (the same one that is used by the Lambda functions) was configured for the build process, it was the cheapest one available and had plenty of compute power for the job. The build instance is managed by AWS, we do not have direct access to it nor need to worry about the operating system configuration and maintenance.

The build times are billed by the minute and were in most cases around 1 minute, with some outliers going as high as 3 minutes. For cost reference, the author decided to err on the side of caution and considered the worst-case scenario and a high deployment rate of multiple deployments per day, even in the production environment, which is unlikely as new features are generally introduced in a weekly or monthly basis. The following usage pattern was considered:

- Development
  - Compute instance `general1.small` (3 GB RAM, 2 vCPU)
  - 180 builds/month
  - 3 minutes/build
- Production
  - Compute instance `general1.small` (3 GB RAM, 2 vCPU)
  - 90 builds/month
  - 3 minutes/build

Finally, in the Deploy stage, the system uses AWS CloudFormation to automatically create, update, and delete the resources, as described in detail in the sub-section 5.2.7.

AWS CodePipeline has limits for the number of pipelines, stages, actions, and web-hooks by region, but they can be increased via request. The limits that are relevant to the system are listed in Table 5.12.

The costs related to the AWS CodePipeline are divided into 3 parts: the pipeline itself, the build process, and the deployment process. The deployment costs are not considered in this sub-section as they are discussed in the sub-section 5.2.7. Table 5.13 presents the costs, which are \$6.05/month.

### 5.2.7 AWS CloudFormation

AWS CloudFormation is an Infrastructure as Code (IaC) service that allows the provisioning and management of resources in an orderly and predictable fashion. Developers can

Item	Limit	Comment
Pipelines/region	1K	The solution was designed considering the use of 2 pipelines (dev and prod), but even with a more granular deployment model with dev, test, qa, stage, and prod, the system would still be under the limit.
Webhooks/region (Source stage)	300	The GitHub action provider configured in the pipelines uses webhook for detecting source code changes. The above comment applies here as well, the system is well below the limit.
CodeBuild timeout (Build stage)	8 hours	It depends on the quantity of changes that were included in the pull request. The build times ranged from a few seconds for small hofixes, 2 minutes for adding a new module with multiple actions, and 5 minutes for a full system build.
CloudFormation timeout (Deploy stage)	3 days	The CloudFormation deployment times ranged from 1 minute for small hofixes, 2 minutes for adding a new module with multiple actions, and 9 minutes for a full system deployment.

Table 5.12: AWS CodePipeline limits

Description	Cost/month
Development pipeline	\$1.00
Production pipeline	\$1.00
Development build time (180 builds, 3 minutes each)	\$2.70
Production build time (90 builds, 3 minutes each)	\$1.35
<b>TOTAL</b>	<b>\$6.05</b>

Table 5.13: AWS CodePipeline costs

deploy and update compute, database, and many other resources in a simple, declarative style that abstracts away the complexity of specific resource APIs [5].

All the resources that compose the system are created, updated, and deleted via AWS CloudFormation, there are no manual tasks involved in the deployment and decommissioning of the system.

Regarding service costs, there is no additional charge for using AWS CloudFormation with resource providers in the following namespaces: `AWS::*`, `Alexa::*`, and `Custom::*`. In these cases, you only pay for the created resources the same as if you had created them manually [6]. For the SOAR system, only resources in the namespace `AWS::*` were created, so the cost of using CloudFormation was zero.

Both the development and the production resources were deployed from the same AWS CloudFormation YAML template. A prefix is dynamically added to the resource names according to the environment and some resources are only created if a pre-defined

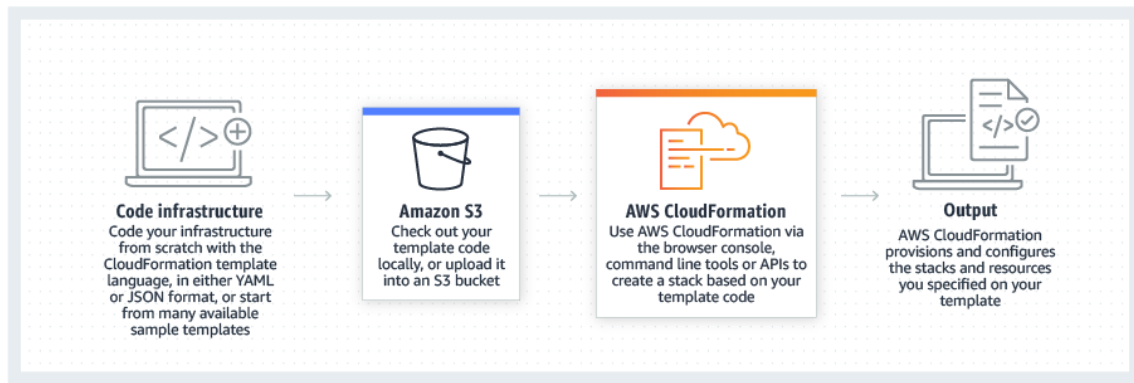


Figure 5.8: AWS CloudFormation overview [5]

condition is met as they are environment-specific. The template is divided into the following sections:

- **Transform**: this is where AWS CloudFormation macros are defined. The template uses AWS Serverless Application Model (SAM) [15].
- **Parameters**: this is where input values are configured. They are used during the stack creation and update process. The template has many parameters such as environment prefix, Python version, Lambda timeout, etc.
- **Conditions**: they define the circumstances under which resources are created or configured. For example, the template has the condition `DevelopmentEnv` that defines that a resource must only be created if the parameter `EnvPrefix` is set to `DEV`.
- **Resources**: this is where the resources used by the SOAR system are declared.

### 5.2.8 GitHub

This is a very popular a code hosting platform used by many individuals and companies for version control and collaboration. It offers free and paid products for storing and collaborating on code. Some products apply only to personal accounts, while other plans apply only to organization and enterprise accounts.

There are other code hosting platforms like AWS CodeCommit, BitBucket, SourceForge, etc. The decision to use GitHub is motivated by the fact that this is the platform used at OutSystems for code hosting and version control, the company subscribes to the Enterprise plan to have access to paid features and premium support covered by a SLA.

As in the SIEM system case, the costs related to GitHub subscriptions are not considered in the SOAR's total cost. Even though OutSystems pays for an Enterprise plan, everything done on GitHub related to this project is available in the Free plan as well, there is no dependency on the paid features as they were not used.

The SOAR system’s code and supporting files are hosted on GitHub Cloud, under the account of the company where this project took place. The repository is private so even within the company only the SOC team is able to see it. The repository contains 2 branches (main and dev) and has branch protection enabled to block direct changes to the main branch, instead they are done via pull requests that are analyzed and approved.

In order to make the SOAR system scalable and modular, each integration with a given system (*e.g.* AbuseIPDB, VirusTotal, PhishTank, etc.) is a self-contained module that has all the code necessary for it to work properly. Each module contains one or more actions to interact with the target system (*e.g.* the AbuseIPDB module has the `check_ip` action). The main branch of the GitHub repository created for the SOAR system is shown in Figure 5.9.

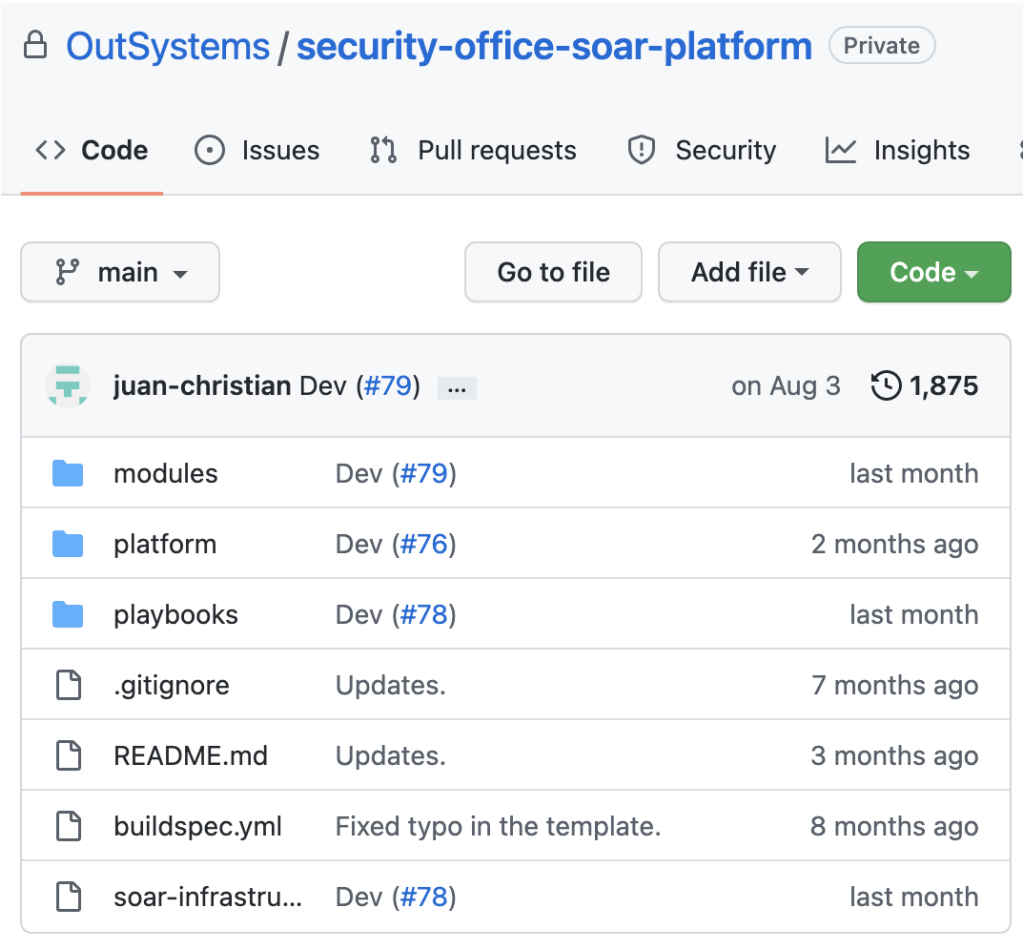


Figure 5.9: SOAR system GitHub repository





# Chapter 6

## Results

This Chapter is divided into two sections: use cases and performance assessment. The first presents the use cases considered to validate and assess the solution, while the second presents the practical results obtained with the proposed SOAR system, focusing on quantitative metrics.

The SOAR system integrates with many systems to be able to orchestrate and automate the tasks that would otherwise be done manually by the analysts. The functional requirements shown in Table 4.2, the defined use cases, and the complexity of the integrations were taken into consideration when deciding which modules should be developed. In the end, 18 modules and a total of 43 actions were developed, which are listed and described in Appendix B.

The actions can be used as building blocks to create automated playbooks using the AWS Step Functions, but can also be executed ad-hoc by the analyst via the SOAR bot. The bot responds the analyst in the same chat and can also send the response to the ticking system in order to expedite the incident response process for tickets that do not have an automated playbook yet.

### 6.1 Use Cases

The OutSystems' SOC decided on the 11 manual playbooks that should be the first ones to have an automated version, based on multiples factors: feasibility of automation, number of occurrences/month, complexity of the playbook, complexity of the incident, and number of manual actions. The selected use cases are described below and it is important to note that they are not detailed so as to not disclose sensitive information about the OutSystems' IR process.

#### 6.1.1 Impossible travel

This use case has the objective of detecting authentications (successful or not) being done against the OutSystems Platform from very far locations in a short time span, something

impossible to be accomplished by taking into consideration the current transportation infrastructure. It is important to note that even though there is a continuous improvement process to fine-tune this use case, there is always a small number of false-positives due to the unique contexts and demands of some Sentry customers.

### **6.1.2 AWS EC2 changes**

The company has many AWS accounts under its control, each one of them with one or more AWS EC2 instances that run important workloads. Changes to said instances must be justified and applied according to the company's maintenance window. This use case has the objective of detecting suspicious changes made to the EC2 instances (*e.g.* changes on network security groups, instance status, attached storage, etc.).

### **6.1.3 AWS IAM changes**

This use case is similar to the one presented in the Sub-Section 6.1.2, but focuses on identity security of the many AWS accounts owned by the company. This use case has the objective of detecting suspicious changes made on the AWS IAM service (*e.g.* changes on users, groups, policies, etc.).

### **6.1.4 Access to credentials without justification**

The company has a process for when an eligible employee needs to access customers' infrastructure. Every access is made using an ephemeral credential with time-limited session and the employee must provide a reason for the access, for example, a support ticket number. In the rare occasions where the reason is not valid (*e.g.* typo in the ticket number), the OutSystems' SOC is alerted to investigate the situation.

### **6.1.5 New digital certificate issued**

The company has many domains under its responsibility and must ensure their security and legitimacy. There is monitoring in place to track the issuance of new digital certificates to all the owned domains. In the rare occasions where a Certificate Authority (CA) issues an illegitimate certificate, the OutSystems' SOC team is alerted to investigate the situation.

It is important to note that even though there are measures such as Certificate Authority Authorization (CAA) that allows domain owners to define which CAs are authorized to issue certificates for the company, this does not exempt the company of monitoring the issued certificates as even the trusted/allowed CAs could be eventually breached and issue bad certificates.

### **6.1.6 Possible SQL injection**

The company's main commercial product, the OutSystems Platform, is intrinsically exposed to the Internet on port TCP 443 - Hypertext Transfer Protocol Secure (HTTPS), so it is subject to attacks. This use case has the objective of detecting possible SQL injections against the product.

### **6.1.7 Login brute force against the OutSystems Platform**

This use case has the objective of detecting login brute force attempts against the OutSystems Platform. As previously mentioned, the Platform is intrinsically exposed to the Internet on port TCP 443 (HTTPS) so it is subjected to such attacks.

### **6.1.8 Login brute force against an internal system**

This use case has the objective of detecting login brute force attempts against one of the systems the company uses internally. The system is not exposed to the Internet so a lower rate of occurrence is expected.

### **6.1.9 Login brute force against an external system**

This use case has the objective of detecting login brute force attempts against one of the systems the company uses. The system is intrinsically exposed to the Internet so a higher rate of occurrence is expected.

### **6.1.10 Partner registered as employee**

OutSystems' workforce is composed of interns, employees, contractors, partners, etc. During the on-board, the entity is assigned the necessary permissions according to their position in the company and the job requirements. The on-board process involves many teams and has some manual operations, hence it is prone to human-error. This use case has the objective of detecting inconsistencies in the employees directory.

### **6.1.11 Update threat intelligence database**

This use case is not related to a security incident alert, but rather a security process. A common practice of SOCs is sharing threat intelligence with partners and security networks. Most of the security tools provide built-in capabilities to automatically collect and act on IoC, for example, when the endpoint protection solution automatically blocks connections to a continuously updated list of malicious IPs.

The issue is that there are many threat intelligence feeds that are not included in said tools and adding/updating the IoCs manually is a monotonous task, even with the bulk

import functionality offered by some of the tools. Moreover, as evidenced by C. Islam et al. [40], companies usually fail to adequately update their threat intelligence databases due to the lack of automation.

## 6.2 Performance Assessment

This project took place in OutSystems, a large multinational company with a 24x7 SOC team composed of analysts located in different continents and timezones. The SOAR system was properly tested in the development environment before being shipped to production. The assessment was conducted in the production environment with real-world security incidents. The system was deployed to production on July 1, 2022.

Table 6.1 shows the use cases in a random and anonymous way to avoid disclosing sensitive information about OutSystems' security incidents, the "Update threat intelligence database", shortened as IoC, is the only use case disclosed as it is easy to identify it by the large number of occurrences and because it is not related to security incidents.

The table compares how many seconds, on average, a selected range of actions per use case took to complete before and after implementing the SOAR system. The presented values are not the MTTR, as there are other actions executed during the incident response that are out of the scope of the SOAR, namely tasks that requires critical-thinking from the analysts.

The manual tasks duration considered the average time an analyst would take to complete a group of actions needed in a given use case, for example, do an IP reputation check, WHOIS lookup, get user details, etc. To calculate the duration, we did a practical experiment with one SOC team member, this person did each task 5 times, we then excluded the 2 worst times and calculated the average of the remaining 3 and this value is what is listed in the column "Manual tasks duration (s)" in Table 6.1.

Additionally, for the manual tasks, we did not even took into consideration the delays that might occur during their execution, such as lunch, coffee break, bathroom, meetings, loss of concentration, etc. The values are the best-case scenario.

The automated tasks duration considered the execution time of the automated playbooks for doing the same actions without human intervention. These values are accurate as they were obtained from the state machine's execution times and by doing a Trimmed Mean (TM) that ignores the 10% top and bottom values to exclude the outliers.

The performance assessment considered the SOAR system usage between 2022-07-01 00:00:00 UTC and 2022-08-13 23:59:59 UTC, inclusive. Even though the SOAR bot was used by the OutSystem's SOC during that time range, its performance and usage metrics, although very positive, were disregarded as the assessment focused on the use cases described in Section 6.1 and the bot was used in other use cases.

During the assessment, the OutSystems' SOC had a total of 364 security incidents

Use Case	Manual tasks duration (s)	Automated tasks duration (s)	Decrease in duration	Occurrences (via SOAR)
A	900	8	99.1%	19
B	780	11.5	98.5%	9
C	240	1.6	99.3%	5
D	1,560	6	99.6%	1
E	120	1.7	98.6%	39
F	1,020	6.2	99.4%	207
G	120	1.6	98.7%	46
H	120	1.7	98.6%	17
IoC	180	2.12	98.8%	9000
J	1,080	7.2	99.3%	18
K	1,110	6.9	99.4%	3

Table 6.1: Comparison of manual and automated tasks duration

matching the use cases, with the exception of the IoC use case as this one is not related to incidents, but rather a threat intelligence process that ran more than 9K times (one execution every few minutes). Considering the time range of the assessment, the number of executions, and the time saved by the SOAR system, we have the following:

- Before: it used to take 75.84 hours of the analyst's time to execute the manual tasks related to 11 use cases defined.

Also, an hypothetical 450 hours of the analyst's time to execute the use case "Update threat intelligence database", but realistically it is not possible to execute this manually in the cadence that an automated system does, so the SOC would simply update the IoC database once a day or so.

- After: the same tasks were automatically executed by the SOAR in 31.2 minutes and the 9K executions (one execution every few minutes) of the "Update threat intelligence database" was completed in 5.3 hours.

The SOAR system met all the non-functional requirements defined in Section 4.2 and most of the functional requirements defined in Section 4.1, as demonstrated in Table 6.2.

It is expected that the unfulfilled functional requirements will be addressed in future works as they were not accomplished due to the limited time of the project and the complexity of the integration needed to be developed and maintained.

Table 6.2: Review of the system requirements completeness

REQ	Done	Description
F.R.01	YES	The management interface of all the used cloud services are accessible via the AWS Console site which uses HTTPS and is accessible via any modern web browser, the analyst does not need to install any software in his machine.
F.R.02	YES	The automated playbooks are created using AWS Step Functions, a visual “playbook editor”.
F.R.03	YES	AWS Step Functions supports parallel and sequential actions. Automated playbooks were created with both parallel and sequential actions in the same workflow.
F.R.04	YES	The audit logs are in JSON format and are sent to AWS Cloud-Watch, which can be exported to many different systems.
F.R.05	YES	The SOAR is cloud native and deployed in the cloud.
F.R.06	YES	The SOAR itself does not expose a REST API, but it is possible to interact (following access control best practices) with all the aspects of the system via the AWS API ( <i>e.g.</i> call the Secrets Manager API to change a secret).
F.R.07	YES	The SOAR has auditing capabilities with logs ranging from debug to critical level. The cloud services themselves also have audit logs.
F.R.08	YES	The SOAR supports SSO inherently due to IAM Identity Center (formerly AWS SSO).
F.R.09	YES	The SOAR supports RBAC inherently due to AWS IAM.
F.R.10	YES	The SOAR integrates with the Directory system used at OutSystems.
F.R.11	YES	The SOAR integrates with the SIEM system used at OutSystems.
F.R.12	NO	The SOAR does not integrate with the email system used at OutSystems. There is an API, but due to the limited project duration and lower priority (the selected use cases have no dependency on this system) the integration was not developed.
F.R.13	YES	The SOAR integrates with the communication system used at OutSystems.
F.R.14	YES	The SOAR integrates with the ticketing system used at OutSystems.
F.R.15	NO	The SOAR does not integrate with the on-call management system used at OutSystems. There is an API, but due to the limited project duration the integration was not developed.
F.R.16	NO	The SOAR does not integrate with the cloud provider services used at OutSystems. There are APIs, but due to the limited project duration the integration was not developed.

F.R.17	NO	The SOAR does not integrate with the operating systems used at OutSystems. There are no official APIs and this integration has lower priority (the selected use cases have no dependency on these systems), so the integration was not developed.
F.R.18	NO	The SOAR does not integrate with the device management systems used at OutSystems. There are APIs, but due to the limited project duration and lower priority (the selected use cases have no dependency on these systems) the integration was not developed.
F.R.19	YES	The SOAR integrates with the endpoint protection system used at OutSystems.
F.R.20	NO	The SOAR does not integrate with the networking devices used at OutSystems. There are APIs, but due to the limited project duration and lower priority (the selected use cases have no dependency on these systems) the integration was not developed.
F.R.21	NO	The SOAR does not integrate with the OutSystems Platform. There is an API, but due to the limited project duration and lower priority (the selected use cases have no dependency on this system) the integration was not developed.
F.R.22	YES	Version control is achieved inherently due to having the automated playbooks YAML files in the SOAR Git repository.
F.R.23	YES	Because the SOAR is custom-built, all modules and actions were developed using Python.
F.R.24	N/A	Only applicable to COTS SOAR
F.R.25	N/A	Only applicable to COTS SOAR
NF.R.01	YES	The system uses Lambda functions, which is a form of serverless computing that has no back-end maintenance requirements.
NF.R.02	YES	The system uses a dedicated AWS VPC with 2 dedicated public IPs, the network communications are encrypted with TLS 1.2 and above, the secrets are encrypted at rest and segregated by module, there is authentication and authorization for interacting with the SOAR bot, and there is input validation for all inputs received from the user.
NF.R.03	YES	As described in the column “Decrease in duration” in Table 6.1, the automated actions are, on average, 112 times faster than the manual actions.
NF.R.04	YES	All the cloud services used are able to scale up and down according to the demand.
NF.R.05	YES	The SOAR system costs around \$65/month while the average monthly salary of a security analyst in Lisbon, Portugal is \$2,819 [37], meaning the system is equivalent to the cost of 0.02 FTE.

Finally, it is important to remember the AWS services limits that were discussed in Section 5.2 and how far from the limits the SOAR system is, even considering the high

volume of 10K security incidents/month and the 100 secrets objects that were used as reference for calculating the costs. The cost of \$65/month already considers this high volume and scale as reference, and not only the 364 security incidents and 18 modules that were created in the scope of the project.







# Chapter 7

## Conclusion

In this project, we proposed a low-cost cloud native SOAR system, presenting the underlying details of its design. The performance of the proposed solution was evaluated through real-world assessment performed in a large multinational company. The results show that the solution was able to decrease the duration of the tasks in a weighted average of 98.81% (99.14% if we only consider the use cases related to IR and disregard the IoC use case metrics) while having an operating expense of less than \$65/month.

It is important to note that it was not possible to automate all the actions in all the use cases, as some of them require critical-thinking, are too complex to automate, or the system does not provide an API. This is expected and reinforce what was observed in the academic materials [28, 51], SOAR is not a silver bullet and one must not try to automate everything blindly.

Finally, it is worth mentioning that to avoid cloud vendor lock-in, the ideal choice would be to use a vendor-neutral IaC platform such as Terraform or Ansible, but that was not possible in the limited time-frame of this project. As future work, the author plans to move from AWS CloudFormation to a vendor-neutral IaC solution to avoid cloud vendor lock-in and to make the SOAR solution cloud-agnostic.

Along the use of a vendor-neutral IaC platform, the author also intends to finish the development of the pending integrations and expand the list of modules and actions to cover even more use cases.



# Bibliography

- [1] Apache. Apache airflow github repository. <https://github.com/apache/airflow>. [Online; accessed Aug 14, 2022].
- [2] Craig Arcuri. An inside look at aws secrets manager vs aws systems manager parameter store. <https://acloudguru.com/blog/engineering/an-inside-look-at-aws-secrets-manager-vs-parameter-store>. [Online; accessed Aug 14, 2022].
- [3] Astronomer. What is apache airflow? <https://www.astronomer.io/airflow/>. [Online; accessed Feb 25, 2023].
- [4] AWS. Amazon simple notification service. <https://aws.amazon.com/sns/>. [Online; accessed Aug 14, 2022].
- [5] AWS. Aws cloudformation. <https://aws.amazon.com/cloudformation/>. [Online; accessed Aug 14, 2022].
- [6] AWS. Aws cloudformation pricing. <https://aws.amazon.com/cloudformation/pricing/>. [Online; accessed Aug 14, 2022].
- [7] AWS. Aws codebuild. <https://aws.amazon.com/codebuild/>. [Online; accessed Aug 14, 2022].
- [8] AWS. Aws codepipeline. <https://aws.amazon.com/codepipeline/>. [Online; accessed Aug 14, 2022].
- [9] AWS. Aws codepipeline pricing. <https://aws.amazon.com/codepipeline/pricing/>. [Online; accessed Aug 14, 2022].
- [10] AWS. Aws lambda. <https://aws.amazon.com/lambda/>. [Online; accessed Aug 14, 2022].
- [11] AWS. Aws lambda powertools for python. <https://awslabs.github.io/aws-lambda-powertools-python/latest/>. [Online; accessed Aug 14, 2022].

- [12] AWS. Aws organizations. <https://aws.amazon.com/organizations/>. [Online; accessed Aug 14, 2022].
- [13] AWS. Aws secrets manager. <https://aws.amazon.com/secrets-manager/>. [Online; accessed Aug 14, 2022].
- [14] AWS. Aws secrets manager quotas. [https://docs.aws.amazon.com/secretsmanager/latest/userguide/reference\\_limits.html](https://docs.aws.amazon.com/secretsmanager/latest/userguide/reference_limits.html). [Online; accessed Aug 14, 2022].
- [15] AWS. Aws serverless application model. <https://aws.amazon.com/cloudformation/pricing/>. [Online; accessed Aug 14, 2022].
- [16] AWS. Aws step functions. <https://aws.amazon.com/step-functions/>. [Online; accessed Aug 14, 2022].
- [17] AWS. Aws step functions standard vs. express workflows. <https://docs.aws.amazon.com/step-functions/latest/dg/concepts-standard-vs-express.html>. [Online; accessed Aug 14, 2022].
- [18] AWS. Creating and sharing lambda layers. <https://docs.aws.amazon.com/lambda/latest/dg/configuration-layers.html>. [Online; accessed Aug 14, 2022].
- [19] AWS. Pricing calculator. <https://calculator.aws/>. [Online; accessed Aug 14, 2022].
- [20] AWS. Service endpoints and quotas. <https://docs.aws.amazon.com/general/latest/gr/aws-service-information.html>. [Online; accessed Aug 14, 2022].
- [21] AWS. What is amazon eks? <https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html>. [Online; accessed Aug 14, 2022].
- [22] James Beswick. Operating lambda: Performance optimization – part 1. <https://aws.amazon.com/blogs/compute/operating-lambda-performance-optimization-part-1/>. [Online; accessed Aug 14, 2022].
- [23] Claudio Neiva; Craig Lawson; Toby Bussa. Market guide for security orchestration, automation and response solutions. <https://www.gartner.com/en/documents/3990720>. [Online; accessed Aug 14, 2022].
- [24] Long Cheng, Fang Liu, and Danfeng Daphne Yao. Enterprise data breach: Causes, challenges, prevention, and future directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(5), 2017.

- [25] et al. Claudio Neiva. Innovation insight for security orchestration, automation and response. <https://www.gartner.com/en/documents/3834578/innovation-insight-for-security-orchestration-automation>. [Online; accessed Aug 14, 2022].
- [26] Juan Christian da Silva Siqueira; Luis Paulino; Alan Oliveira de Sá. A low-cost and cloud native solution for security orchestration, automation, and response (accepted for publication). *17th International Conference on Information Security Practice and Experience (ISPEC)*, -, 2022.
- [27] et al. David McCandless. World's biggest data breaches & hacks. <https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/>. [Online; accessed Aug 14, 2022].
- [28] W. Keith Edwards, Erika Shehan Poole, and Jennifer Stoll. Security automation considered harmful? In *Proceedings of the 2007 Workshop on New Security Paradigms*, NSPW '07, page 33–42, New York, NY, USA, 2008. Association for Computing Machinery.
- [29] Elyra. Elyra github repository. <https://github.com/elyra-ai/elyra>. [Online; accessed Aug 14, 2022].
- [30] International Organization for Standardization. Information technology — Security techniques — Information security for supplier relationships. Standard ISO/IEC 27036, International Organization for Standardization, November 2013.
- [31] International Organization for Standardization. Information technology — Security techniques — Information security management systems — Requirements. Standard ISO/IEC 27001, International Organization for Standardization, October 2013.
- [32] G2. Grid report for security orchestration, automation, and response (soar). <https://www.g2.com/reports/7ff18f7b-3496-4b24-a28a-6bb725055da5/preview>. [Online; accessed Aug 14, 2022].
- [33] G2. Momentum grid report for security orchestration, automation, and response (soar). <https://www.g2.com/reports/35873864-e60f-4fdd-b2db-61135f327c36/preview>. [Online; accessed Aug 14, 2022].
- [34] Gartner. Gartner magic quadrant for enterprise low-code application platforms, 2021. <https://www.outsystems.com/1/low-code-application-platforms-gartner/>. [Online; accessed Aug 14, 2022].
- [35] Gartner. What is an enterprise low-code application platforms (lcap)? <https://www.gartner.com/reviews/market/enterprise-low-code-application-platform>. [Online; accessed Aug 14, 2022].

- [36] R. F. Gibadullin and V. V. Nikonorov. Development of the system for automated incident management based on open-source software. In *2021 International Russian Automation Conference (RusAutoCon)*, pages 521–525, 2021.
- [37] Glassdoor. Average security analyst salary in lisbon. [https://www.glassdoor.com/Salaries/lisbon-security-analyst-salary-SRCH\\_IL\\_0,6\\_IC3192045\\_KO7,23.htm](https://www.glassdoor.com/Salaries/lisbon-security-analyst-salary-SRCH_IL_0,6_IC3192045_KO7,23.htm). [Online; accessed Aug 14, 2022].
- [38] Ashley A Hall and Carol S Wright. Data security: A review of major security breaches between 2014 and 2018. *Federation of Business Disciplines Journal*, 6:50–63, 2018.
- [39] Ponemon Institute. Cost of a Data Breach Report 2022. Technical report, IBM, 2022.
- [40] Chadni Islam, Muhammad Ali Babar, and Surya Nepal. A multi-vocal review of security orchestration. *ACM Comput. Surv.*, 52(2), apr 2019.
- [41] Chadni Islam, Muhammad Ali Babar, and Surya Nepal. Architecture-centric support for integrating security tools in a security orchestration platform. In Anton Jansen, Ivano Malavolta, Henry Muccini, Ipek Ozkaya, and Olaf Zimmermann, editors, *Software Architecture*, pages 165–181, Cham, 2020. Springer International Publishing.
- [42] Jakub Jiříček Jaroslav Babický. Automation, incident response and orchestration: a modern security framework. <https://www.idc.com/eu/events/69153-automation-incident-response-and-orchestration-a-modern-security-framework>. [Online; accessed Aug 14, 2022].
- [43] Lawrence Awuah Johnson Kinyua. Ai/ml in security orchestration, automation and response: Future research directions. *Intelligent Automation & Soft Computing*, 28(2):527–545, 2021.
- [44] et al. Keith Turpin. Owasp secure coding practices-quick reference guide. [https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/migrated\\_content](https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/migrated_content). [Online; accessed Aug 14, 2022].
- [45] Dimitrios Lalos. Analysis on Security Orchestration Automation and Response (SOAR) platforms for Security Operation Centers. Master’s thesis, University of Piraeus, Greece, 2022.
- [46] Luigi. Design and limitations. [https://luigi.readthedocs.io/en/stable/design\\_and\\_limitations.html](https://luigi.readthedocs.io/en/stable/design_and_limitations.html). [Online; accessed Aug 14, 2022].



- [47] Luigi. Luigi github repository. <https://github.com/spotify/luigi>. [Online; accessed Aug 14, 2022].
- [48] Song Luo and Malek Ben Salem. Orchestration of software-defined security services. *IEEE International Conference on Communications Workshops (ICC)*, -:436–441, 2016.
- [49] Allie Mellen. Now tech: Security orchestration, automation, and response (soar), q2 2022. <https://www.forrester.com/report/now-tech-security-orchestration-automation-and-response-soar-q2-2022/RES177298>. [Online; accessed Aug 14, 2022].
- [50] MITRE. Att&ck. <https://attack.mitre.org/>. [Online; accessed Aug 14, 2022].
- [51] Raydel Montesino and Stefan Fenz. Information security automation: How far can we go? In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 280–285, 2011.
- [52] Donna Dodson Murugiah Souppaya (NIST), Karen Scarfone (Scarfone Cybersecurity). Sp 800-218, secure software development framework (ssdf). <https://csrc.nist.gov/publications/detail/sp/800-218/final>. [Online; accessed Aug 14, 2022].
- [53] Sara Vanessa Sanches Lima do Nascimento. Robotização de tarefas repetitivas e críticas nos centros de operação de ciber segurança. *FC-DI - Master Thesis (projects)*, -, 2019.
- [54] National Institute of Standards and Technology. Computer security incident handling guide. Technical Report SP 800-61 Rev. 2, U.S. Department of Commerce, Washington, D.C., 2012.
- [55] National Institute of Standards and Technology. Security and privacy controls for information systems and organizations. Technical Report SP 800-53 Rev. 5, U.S. Department of Commerce, Washington, D.C., 2020.
- [56] National Institute of Standards and Technology. Cybersecurity supply chain risk management practices for systems and organizations. Technical Report SP 800-161 Rev. 1, U.S. Department of Commerce, Washington, D.C., 2022.
- [57] Motoyuki Ohmori. On automation and orchestration of an initial computer security incident response by introducing centralized incident tracking system. *Journal of Information Processing*, 27:564–573, 2019.

- [58] OutSystems. Homepage. <https://www.outsystems.com/>. [Online; accessed Aug 14, 2022].
- [59] OutSystems. Outsystems on aws. <https://www.outsystems.com/partners/technology-alliance/aws/>. [Online; accessed Aug 14, 2022].
- [60] OutSystems. Outsystems security. <https://www.outsystems.com/security/>. [Online; accessed Aug 14, 2022].
- [61] OutSystems. Outsystems sentry. <https://www.outsystems.com/sentry/>. [Online; accessed Aug 14, 2022].
- [62] OutSystems. Pricing. <https://www.outsystems.com/pricing-and-conditions/>. [Online; accessed Aug 14, 2022].
- [63] Carl Adam Petri. Kommunikation mit Automaten. Dissertation, Schriften des IIM 2, Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, Bonn, 1962.
- [64] Rabix. Rabix composer github repository. <https://github.com/rabix/composer>. [Online; accessed Aug 14, 2022].
- [65] Mangey Ram and J. Paulo Davim. *Mathematics applied to engineering*. Academic Press, first edition, 2017.
- [66] Parikshit Savjani. Deploying apache airflow in azure to build and run data pipelines. <https://azure.microsoft.com/en-us/blog/deploying-apache-airflow-in-azure-to-build-and-run-data-pipelines/>. [Online; accessed Aug 14, 2022].
- [67] Markus Schmitt. Airflow vs. luigi vs. argo vs. mlflow vs. kubeflow. <https://www.datarevenue.com/en-blog/airflow-vs-luigi-vs-argo-vs-mlflow-vs-kubeflow>. [Online; accessed Feb 25, 2023].
- [68] Dave Shackleford. SANS 2022 Cloud Security Survey. Technical report, SANS, 03 2022.
- [69] Benjamin Smith. Orchestrating a security incident response with aws step functions. <https://aws.amazon.com/blogs/compute/orchestrating-a-security-incident-response-with-aws-step-functions/>. [Online; accessed Aug 14, 2022].
- [70] Verizon. 2022 Data Breach Investigations Report. Technical report, Verizon, 2022.

- 
- [71] Weijia Wang, Xiaofeng Qiu, and Rui Sun, Li; Zhao. A data driven orchestration framework in software defined security. *IEEE International Conference on Communications Workshops (ICC)*, -:34–39, 2016.



# Appendix A

## AWS Pricing Calculator quote

Table A.1: AWS Pricing Calculator quote

Description	Service	Cost/mo	Configuration summary
Alerts payload	Standard topics	\$0.27	DT Inbound: Not selected (0 TB per month), DT Outbound: Internet (3 GB per month), Requests (10000 per month), SQS Notifications ( million per month)
Playbook calls	AWS Lambda	\$0.32	Architecture (x86), Architecture (x86), Number of requests (300000 per month), Amount of ephemeral storage allocated (512 MB)
Playbooks Manager	AWS Lambda	\$0	Architecture (x86), Architecture (x86), Amount of ephemeral storage allocated (512 MB), Number of requests (10000 per month)
Keep SOAR warm	AWS Lambda	\$0.36	Architecture (x86), Architecture (x86), Number of requests (14600 per month), Amount of ephemeral storage allocated (512 MB)
Keep SOAR warm calls	AWS Lambda	\$3.04	Architecture (x86), Architecture (x86), Number of requests (14600000 per month), Amount of ephemeral storage allocated (512 MB)
Secrets	Secrets Manager	\$40.82	Number of secrets (102), Average duration of each secret (30 days), Number of API calls (102 per day)

Playbooks	Step Functions	\$12.4	Workflow requests (10000 per month), State transitions per workflow (50)
CI/CD	CodePipeline	\$1	Number of active pipelines used per account per month (2)
IaC	CloudFormation	\$0	Number of third-party extensions managed (100), Average duration per operation (30 seconds), Total number of operations per extension (0 per day)
IaC	CodeBuild	\$2.7	Number of builds in a month (180), Average build duration (minutes) (3), Operating system (Linux), Compute instance type (general1.small)
(DEV) IaC	CodeBuild	\$1.35	Number of builds in a month (90), Average build duration (minutes) (3), Operating system (Linux), Compute instance type (general1.small)
(DEV) CI/CD	CodePipeline	\$1	Number of active pipelines used per account per month (2)

# Appendix B

## SOAR Modules and Actions

Table B.1: SOAR Modules and Actions

Module	Action	Description
AbuseIPDB	report_ip	Report the IP to AbuseIPDB.
	check_ip	Checks the reputation of the IP.
	check_subnet	Checks the reputation of the subnet.
EmailRep	check_email	Checks the reputation of the email.
SANS ISC	get_asnum_ips	Get a summary report about the AS.
	get_ip_details	Get a detailed report about the IP.
	get_ip_summary	Get a summary report about the IP.
NetTools	dns_lookup	Do a DNS lookup.
	whois	Query the WHOIS database.
OpenCTI	search	Search for a given entity.
	add_observable	Add an observable ( <i>e.g.</i> IP, hash, etc.).
	check_vuln	Get the details about an CVE.
PhishTank	check_url	Checks if the URL is in PhishTank.
Splunk Cloud	search	Creates a search job.
	get_results	Get the results of a search.
SSL Labs	analyze_host	Checks the SSL/TLS config/health.
SSLMate	check_ct	Checks a domain in the CT logs.
unshorten.me	unshort_url	Unshort the URL.
urlscan.io	analyze_url	Do a scan against the URL.
VirusTotal	check_file	Checks if the file is in VT.
	check_url	Checks if the URL is in VT.
	check_ip	Checks if the IP is in VT.
	check_domain	Checks if the domain is in VT.
Directory System	get_user	Get the details about the user.
Messaging	add_to_chn	Add users to channel.
	archive_chn	Archive a channel.
	create_chn	Creates a channel.
	get_user_details	Get the details about a user.
	remove_from_chn	Removes users from channel.
	rename_chn	Removes users from channel.

	send_message	Sends a message to a channel/users.
	set_topic	Sets the topic for a channel.
Wiki	search	Search for pages matching the query.
Issue Tracking	get_issue	Get the details about the issue.
	search	Search for issues matching the query.
Endpoint Protection	add_indicator	Add an indicator ( <i>e.g.</i> IP, hash, etc.).
Ticketing	add_tags	Add tags to the ticket.
	remove_tags	Remove tags from the ticket.
	search	Search for a ticket matching the query.
	create_ticket	Creates a ticket.
	get_ticket	Get the details about a ticket.
	post_comment	Post a comment to the ticket.
	update_status	Update the ticket's status.