INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Motor de Inferência aplicado à deteção de incidentes de segurança no ciberespaço de uma Organização

Rodrigo Pinto Valente

Mestrado em Informática e Gestão

Orientador:
Doutor João Carlos Marques Silva, professor auxiliar,
ISCTE - Instituto Universitário de Lisboa

Coorientadora:
Doutora Maria Cabral Diogo Pinto Albuquerque, professora auxiliar,
ISCTE - Instituto Universitário de Lisboa

Novembro, 2022

TECNOLOGIAS
E ARQUITETURA

Departamento de Ciências e Tecnologias da Informação

Motor de Inferência aplicado à deteção de incidentes de segurança no ciberespaço de uma Organização

Rodrigo Pinto Valente

Mestrado em Informática e Gestão

Orientador:
Doutor João Carlos Marques Silva, professor auxiliar,
ISCTE - Instituto Universitário de Lisboa

Coorientadora:
Doutora Maria Cabral Diogo Pinto Albuquerque, professora auxiliar,
ISCTE - Instituto Universitário de Lisboa

Novembro, 2022

Agradecimentos

Realizar um projeto de investigação e desenvolvimento é um irregular e inconstante progresso, com avanços e retornos, oscilando entre a criatividade e o desespero, com uma permanente revisão da literatura e de trabalhos semelhantes. Manter a trajetória e o foco no cumprimento dos objetivos, só é possível com muito suporte familiar.

À Mariana e aos meus pais, agradeço o constante contributo para o equilíbrio das emoções ao longo de todo o processo e pela incessante confiança nas minhas capacidades e no meu potencial, alimentando diariamente a minha força de vontade e o meu desejo de terminar.

Toda esta investigação e desenvolvimento jamais seria possível sem a forte contribuição e partilha de ideias com o Mário da Costa. Na amizade, no emprego e na universidade, presente nos momentos mais importantes e mais decisivos, juntos abraçámos com confiança e determinação a realização de uma investigação individual que no fim harmonizará um projeto conjunto, robusto e fundamentado.

Por último, mas não menos importante, um agradecimento especial a toda a estrutura que balizou, encaminhou e auxiliou permanentemente esta investigação, os orientadores do ISCTE, professor João Carlos Silva e a professora Maria Pinto Albuquerque e os orientadores na *Outsystems*, o SOC *Manager* Luís Paulino e o *Head of Security Architecture and Assurance* Igor Antunes.

Resumo

As capacidades tecnológicas evoluem exponencialmente todos os dias, assim como os ataques informáticos às organizações, é crucial melhorar as capacidades de monitorização, deteção e resposta a potenciais ameaças. Mecanismos tecnológicos de segurança por meio de automação de processos, Inteligência Artificial (AI) e *Machine Learning* (ML) são os que qualquer organização vai querer ter como arma e escudo para enfrentar os desafios emergentes no mundo da cibersegurança.

Num Centro de Operações de Segurança (SOC), um analista precisa de monitorizar e investigar centenas de potenciais ameaças diariamente, a utilização de procedimentos automáticos para classificação de ameaças é imperativo para uma proficiência na resposta a esses incidentes de segurança de informação (ISIs).

Para poder automatizar totalmente respostas a ISIs, um SOC precisa de ser capaz de classificar os indicadores de compromisso (IoCs) e percorrer uma árvore de decisão baseada nessas classificações, de forma automatizada. O objetivo é fornecer uma solução de classificação de IoCs que se possa adequar a uma organização e possa servir de decisão sobre avançar ou não com um fluxo automatizado. Assumindo que já existe um sistema de monitorização e gestão de eventos de segurança (SIEM), este motor de inferência, é capaz de automatizar a atribuição de um grau de ameaça a cada IoC, quantitativamente e qualitativamente. É uma solução que utiliza *Cyber Threat Intelligence* (CTI) e possibilita decisões automatizadas e personalizadas ao nível da ameaça para a organização, nomeadamente se uma contenção específica deve ou não ser executada com base nesse nível de ameaça identificada.

Palavras-chave: Automação; Inteligência Artificial; Machine Learning; cibersegurança; SOC; Cyber Threat Intelligence, Incidentes de Segurança de Informação

Abstract

Technological capabilities evolve exponentially every day, as well as cyberattacks at organizations, it is crucial to improve monitoring, detection and response capabilities to potential cyber threats. Thus, technological security mechanisms through automation, Artificial Intelligence (AI) and Machine Learning (ML), are what any organization will want as a weapon and shield to face, protect and react to the challenges in the cybersecurity world.

Daily, in a Security Operations Center (SOC) an analyst needs to deal and investigate hundreds of potential threats. Using automatic procedures for threats identification and classification is imperative to improve performance on cyber threats containing, mitigating and responding to information security incidents (ISIs).

To be able to fully automate the information incident responses, a SOC needs to be capable of classifying the Indicators of Compromise (IoCs) and automatically run through a Decision Tree based on such classification. The goal is to develop an inference engine that classifies the IoCs, centering on its characteristics. If there's already a Security Information and Event Management (SIEM), this inference engine is capable of automatically attributing a threat / risk score to each IoC, both quantitively and qualitatively, using Cyber Threat Intelligence (CTI) to allow for personalized automatic decisions according to the organization needs, namely if a specific containment should be executed or not.

Key words: Automation, Artificial Intelligence; Machine Learning; cybersecurity; SOC; Cyber Threat Intelligence; Information Security Incident

Lista de abreviaturas

AI: Artificial Intelligence

ANN: Artificial Neural Network

APIs: Application Programming Interface

AV: Antivirus

BPM: Business Process Management

BPMN: Business Process Model and Notation

CAPEC: Common Attack Pattern Enumerations and Classifications

CIA: Confidentiality, Integrity, Availability

CSC: Critical Security Control

CTI: Cyber Threat Intelligence

CVSS: Common Vulnerability Scoring System

CWE: Common Weakness Enumeration

DoS: Denial-of-Service

DDoS: Distributed Denial of Service

D-S: Dempster-Shafer

DSR: Design Science Research

DT: Decision Tree

HTTP: Hypertext Transfer Protocol Secure

IDE: Integrated Development Environment

IDS: Intrusion Detection System

IoC: Indicator of Compromise

IoT: Internet-of-Things

IP: Internet Protocol

IPS: Intrusion Prevention System

ISI: Information Security Incident

JSON: JavaScript Object Notation

K-NN: K-Nearest Neighbors

MitM: Man-in-the-Middle

ML: Machine Learning

MFA: Multi-Factor Authentication

MRQ: Main Research Question

NB: Naıve Bayes

NIST: Instituto Nacional de Padrões e Tecnologia

OWASP: Open Web Application Security Project

LR: Logistic Regression

LR-DS Logistic Regression and Dempster-Shafer Theory

QA: Quality Assurance

RF: Random Forest

RQ: Research Question

SIEM: Security Information and Event Management

SOAPA: Security Operations and Automation Platform Architecture

SOAR: Security Orchestration, Automation and Response

SOC: Security Operations Center

SDLC: Software Development Life Cycle

SLR: Systematic Literature Review

SQL: Structured Query Language

SSRF: Server-Side Request Forgery

STIX: OASIS Structured Threat Information Expression

SVM*:* Support Vector Machine

TAXII: OASIS Trusted Automated Exchange of Indicator Information

TTD: Time to Detection

TTI: Time to Investigate

TTQ: Time to Qualify

TTR: Time to Respond

TTPs: Tactics, Techniques and Procedures

UML: Unified Modeling Language

VPN: Virtual Private Network

VS Code: Visual Studio Code

VT: Virus Total

XSS: Cross Site Scripting

WEB: World Wide Web

Glossário

Termo	Descrição	Fonte
Evento (de Segurança de Informação)	Um evento pode ser descrito como qualquer processo legal ou ilegal concluído no sistema de informação, identificado e registado pelo sistema.	[57]
Incidente de Segurança de Informação	Um incidente de segurança da informação é entendido como a ocorrência de um ou mais eventos de segurança da informação indesejados ou inesperados, com os quais há uma probabilidade significativa de comprometer as operações do sistema (da organização) e criar uma ameaça.	[58]
IoC	Indicadores de compromisso (IoC) são incidentes de segurança identificados após intrusões nos sistemas organizacionais, especificam condições que podem existir para indicar a presença de uma ciberameaça (informação contextual). IoCs, ao serem partilhados providenciam informações valiosas sobre sistemas que foram comprometidos (CTI).	[82, 85]
Ciberespaço	A rede interdependente de infraestruturas de tecnologia da informação incluindo a *Internet*, as redes de telecomunicações e os sistemas de computador.	[74]
Ciberataque	Um ataque, via ciberespaço, com a finalidade de interromper, desabilitar, destruir ou controlar maliciosamente um ambiente/infraestrutura de computação; ou destruir a integridade dos dados ou roubar informações.	[74]
Ciberameaça	Qualquer circunstância ou evento com potencial para impactar adversamente as operações organizacionais (incluindo missão, funções, imagem ou reputação), ativos organizacionais, indivíduos ou organizações, por meio de um sistema de informação comprometido, por acesso não autorizado, destruição, divulgação, modificação de informações, e/ou negação de serviço.	[74]
Cibersegurança	Prevenção de danos, proteção e restauração de computadores, sistemas de comunicações eletrónicas, serviços de comunicações eletrónicas, comunicações por fio e comunicações eletrónicas, incluindo as informações neles contidas, para garantir sua disponibilidade, integridade, autenticação, confidencialidade e não repúdio.	[74]
Atacante / *Hacker*	Utilizador não autorizado que tenta ou obtém acesso a um sistema de informação.	[74]
Logs	Um registo dos eventos que ocorreram nos sistemas e redes de uma organização.	[84]
Input	Fornecer dados de uma fonte externa.	[67]

Output	Pertencente a dados transmitidos para um destino externo.	[67]
Cloud	É um modelo para permitir acesso de rede ubíquo e conveniente, providenciando acesso a um conjunto compartilhado de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços *web*) que podem ser rapidamente provisionados e disponibilizados com um esforço mínimo de gestão ou interação do provisor de serviços.	[86]
HTTP request	HTTP é o protocolo subjacente aos serviços de navegação na Internet, ou seja, é um protocolo de solicitação-resposta, que possibilita aos utilizadores uma forma de interagir com recursos da *Web*, como arquivos HTML, transmitindo mensagens entre clientes e servidores. Existem diferentes tipos de *HTTP requests,* neste projeto serão abordados o GET e o POST. GET: Solicita um recurso específico na sua totalidade POST: Adiciona conteúdo, mensagens ou dados a uma nova página num recurso da *web* existente	[66, 73]
Software	Programas de computador, procedimentos, documentação e dados relativos à operação de um sistema de computador.	[67]
Malware	*Software* destinado a executar um processo não autorizado que terá impacto adverso na confidencialidade, integridade ou disponibilidade de um sistema de informação.	[74]
Aplicação *Web*	Um programa de *software* hospedado por um sistema de informação.	[85]
Breach	A perda de controlo, comprometimento, divulgação não autorizada, aquisição não autorizada ou qualquer ocorrência semelhante em que: uma pessoa que não seja um utilizador autorizado aceda a informações de identificação pessoal; ou um utilizador autorizado aceda a informações de identificação pessoal para outro propósito que não o autorizado.	[85]
(ciber) Risco	O risco é definido como a exposição provável de uma ameaça, devido à exploração das vulnerabilidades relevantes que impactam na confidencialidade, integridade e disponibilidade dos bens.	[82]
Whitelist / Blacklist	*Whitelist* é uma lista ou um registo daqueles que possuem um determinado privilégio, serviço, mobilidade, acesso ou reconhecimento. Somente os presentes na lista serão aceites, aprovados ou reconhecidos. Por outro lado, uma *blacklist,* é composta por aqueles que são identificados, como maliciosos, não reconhecidos, negados e proibidos.	[72]
Ataque dia-zero	É um ataque executado contra uma ou mais vulnerabilidades previamente desconhecidas	[47]

Índice

Agradecimentos iii
Resumo vii
Abstract ix
Lista de abreviaturas xi
Glossário xv
Capítulo 1 - Introdução 1
1.1 Motivação 2
1.1.2 Organização: Outsystems 3
1.3 Objetivos 3
1.4 Conteúdo 4
Capítulo 2 - Revisão da Literatura 7
2.1 Centro de Operações de Segurança (SOC) 7
2.2 Segurança da Informação: Evento vs Incidente 7
2.3 Ciclo de Vida da Resposta a um Incidente 9
2.4 Vulnerabilidades 10
2.5 Ferramentas de Cibersegurança 14
2.5.1 Cyber Threat Intelligence (CTI) 14
2.5.2 Sistema de Deteção de Intrusões (IDS) 15
2.5.3 Security Information and Event Management (SIEM) 15
2.5.4 Security Orchestration, Automation and Response (SOAR) 16
2.6 O novo paradigma: AI/ML aplicadas à cibersegurança 18
2.7 Trabalhos Relacionados: Soluções de AI/ML 22
2.8 Automatização e Orquestração sem SOAR 26
Capítulo 3 - Metodologia 27
3.1 Questões de Pesquisa 27
3.2 Metodologia de Investigação: *Design Science Research* 28
3.3 Metodologia de Desenvolvimento de *Software* 28
Capítulo 4 - Arquitetura Proposta 31
4.1 Introdução à plataforma 31
4.1.1 Análise de requisitos 31

4.1.2 System Input: Evento 31
4.1.3 System Output: JSON 32
4.1.4 Classificação 33
4.1.5 Score incremental: *Fibonacci* 33
4.1.6 Tabela de classificação de ameaças 35
4.2 System Overview 36
4.3 Arquitetura de processos 37
4.3.1 Macro: Modelo BPMN 37
4.3.2 Micro: Modelo UML 39
Capítulo 5 - Desenvolvimento da Solução 43
5.1 Tecnologias utilizadas 43
5.1.1 OpenCTI 43
5.1.2 Virus Total 44
5.1.3 GraphQL 44
5.1.4 Python 45
5.1.5 Docker 45
5.1.6 AWS Cloud 45
5.1.7 Slack 46
5.2 Investigação e Implementação 47
5.3 Vulnerabilidades 53
5.4 Perspetiva do Analista de um SOC 54
5.5 Resposta a Incidentes de Segurança: *Playbooks* 55
Capítulo 6 - Testes e Avaliação dos Resultados 59
6.1 Teste ao Software: Script Python 59
6.2 Métricas de Teste de Eficiência 62
6.2.1 Tempo para Qualificação (TTQ) 62
6.3 Confusion Matrix 66
6.4 Discussão de Resultados 67
Capítulo 7 - Conclusões 71
7.1 Limitações e Trabalho futuro 72
Referências Bibliográficas 75

CAPÍTULO 1

Introdução

As ameaças à segurança da informação são cada vez mais diversificadas e mais frequentes, têm crescido a um ritmo exponencial nos últimos anos [1-2]. As organizações estão cada vez mais dependentes da tecnologia, *hardware*, *softwares* e *internet*, e com isso mais suscetíveis e mais vulneráveis a ciberataques [3-4]. Esta evolução foi ainda mais intensificada pela pandemia Covid-19 através do consequente crescimento abrupto do trabalho remoto e das constantes mudanças do local de trabalho entre confinamentos [5].

Este crescente e diversificado aumento das ameaças bem como o aumento da velocidade da *internet*, tem intensificado e dificultado o trabalho do analista de um *Security Operations Center* (SOC) [6-7]. Por outro lado, de forma geral, as médias e grande empresas tendem a possuir diversas ferramentas de combate à cibersegurança, de diferentes vendedores e com diferentes capacidades de solução, resultando também numa maior complexidade no tratamento / gestão da informação [8-9].

Assim, torna-se pertinente e fundamental acompanhar o escalonamento das ameaças com novas tecnologias, automatizando e orquestrando os processos, conectando várias ferramentas e usando APIs (*Application Programming Interface*) específicos de um só vendedor, de forma a capacitar os analistas a uma tomada de decisão mais eficiente na resposta aos incidentes de segurança (ISI) [10-11].

O custo médio por exposição de dados numa organização foi, em 2022, em média, 4,35 milhões de dólares, o valor mais alto de sempre. Todavia, os custos foram significativamente mais baixos para algumas organizações com uma postura de segurança mais madura e superior tecnologicamente, quando comparado com organizações que não investiram em áreas e tecnologias de automação. As organizações com uma segurança baseada em Inteligência Artificial (AI) e automação tiveram custos de cibersegurança de 3,15 milhões de dólares, em comparação com organizações sem segurança AI e automação que tiveram em média, custos de 6,20 milhões de dólares [32].

Assim, partindo do pressuposto que uma organização está ciente das vantagens da aplicação de tecnologias de automação num SOC e possui ferramentas de cibersegurança com essas capacidades, o objetivo deste trabalho é desenvolver um mecanismo inteligente que tenha a capacidade de classificar ciberameaças, atribuindo um valor de risco e um impacto para a organização.

O primeiro passo na construção de um sistema seguro é entender, identificar e classificar as ciberameaças, permitindo às organizações compreender e avaliar os respetivos impactos, de forma a

ser possível desenvolver estratégias para a sua prevenção e mitigação, bem como ser capaz de responder, atempadamente, de forma adequada e em conformidade [80].

1.1 Motivação

Com o aumento do tráfego / fluxo de utilização da *internet*, a cibersegurança enfrenta diversos desafios, em particular o elevado número de eventos de segurança, tornando a análise manual impraticável [1, 6-9]. A mais recente literatura [1, 3-4, 12-14] sustenta a necessidade da adaptação de tecnologias, com a capacidade de automação de processos, às reais necessidades de proteção da segurança da informação nas empresas. As principais formas de deteção e prevenção de ciberataques nas empresas são os sistemas IDS (*Intrusion Detection System*), IPS (*Intrusion Prevention System*), Firewalls, *Antivirus, Antimalware* e SIEM (*Security Information and Event Management*) [8, 14] que servem para monitorizar e categorizar eventos potencialmente maliciosos, para posteriormente os analistas do SOC investigarem esses eventos / alertas e decidirem se são realmente maliciosos ou não [2]. No entanto, geralmente o número de alertas é esmagador, sendo que a maioria deles são falsos positivos [3-4, 12-14], o que consequentemente resulta numa tarefa ineficiente por parte do analista e por vezes ineficaz, uma vez que potenciais ataques maliciosos podem passar despercebidos e conseguirem consumar o seu objetivo. Assim, ferramentas com a capacidade de automatizar tarefas e processos permitem construir abordagens viáveis e eficazes para a redução da taxa de falsos positivos e para uma melhoria da produtividade e eficiência dos analistas do SOC, bem como proteger a organização e a sua capacidade / resiliência na resposta a IoCs futuros [76].

Os sistemas de cibersegurança com recurso à automação de tarefas, processos e resposta a incidentes de segurança serão fundamentais para responder ao crescimento contínuo em número e ao aumento da complexidade das ameaças [14]. Por exemplo, os sistemas de defesa que possuam AI / ML têm a capacidade de analisar grandes conjuntos de dados e de identificar anomalias e padrões suspeitos instantaneamente [3]. Análises sofisticadas em tempo real podem prevenir ciberataques de grande escala [14]. As organizações têm aplicado aos gestores de correio eletrónico técnicas de AI para evitar imagens indesejáveis, detetar *phishing*, *malware* e pagamentos fraudulentos, bem como a utilização de ANN (*Artificial Neural Network*) para deteção e classificação de *phishing* e *malware* [10]. Outra abordagem comum de ML e de modelos preditivos é a utilização de técnicas baseadas em árvores, vastamente utilizadas para construir *Intrusion Detection Systems* (IDSs), com a finalidade de prever e detetar ataques em redes de comunicação [15].

As organizações com segurança AI e automação totalmente implantada, demoraram em média 181 dias para identificar uma falha de segurança e 68 dias para a conter, perfazendo um ciclo de vida de 249 dias. Em contrapartida, as organizações sem AI / automação ao nível da segurança

implementado, demoram em média 235 dias para identificar uma falha de segurança e 88 dias para a conter, perfazendo um ciclo de vida de 323 dias [32].

1.1.2 Organização: Outsystems

Esta investigação foi desenvolvida em parceria com a *Outsystems,* referenciada como "Organização" ao longo deste trabalho. Trata-se de uma empresa nacional de programação *low-code*, fundada em 2000, líder de mercado em Portugal e já com uma forte presença a nível internacional. Em 2016, sentiu a necessidade de criar um SOC de forma a reforçar o seu departamento de segurança de operações, tornando-o mais robusto, maduro, eficiente e capaz de dar resposta aos novos desafios da segurança informática e proteção de dados da Organização e dos seus clientes. Assim, neste seguimento, na convergência dos interesses da Organização, numa aposta na automação de processos e inteligência artificial, à nossa motivação em investigação & desenvolvimento de mecanismos automatizados que contribuam para um SOC mais robusto, ao nível da cibersegurança e mais eficiente, do ponto de vista do analista, deu-se início a um projeto de investigação que fosse capaz de auxiliar de forma eficiente e inteligente um analista de um SOC.

1.3 Objetivos

O principal objetivo desta investigação é desenvolver um motor de inferência com a capacidade de analisar e inferir o risco para a organização de determinado IoC, atribuindo-lhe uma classificação quantitativa e qualitativa, para que com base nessa informação seja possível responder, de forma adequada, a esse IoC. A fim de concretizar o objetivo principal para o desenvolvimento da presente solução considerámos os sub-objetivos seguintes:

- **Realizar uma análise automática de IoCs** - Realizar uma análise automática em diferentes plataformas de CTI para determinar se é um IoC conhecido (já identificado), bem como recolher toda a informação relacionada com esse IoC.
- **Classificação quantitativa do nível de ameaça do IoC** – Obter uma classificação quantitativa (*score* de 0 a 100) para determinar o nível de ameaça (nesse momento) desse IoC.
- **Classificação qualitativa do nível de impacto do IoC** - Aplicar uma classificação qualitativa (Baixo, Moderado ou Elevado) do impacto desse IoC para a organização;
- **Reporte Final** – Devolver ao analista um reporte final com os resultados das classificações obtidas, informação contextual do IoC e número de vezes que esse IoC já foi identificado como suspeito e/ou ciberameaça;

- **Repositório dos IoCs observados** – Guardar toda a informação recolhida sobre os IoCs observados, para que seja possível consultar a qualquer momento e gerir o número de vezes que foi observado na Organização.
- **Sugerir um *playbook* de resposta a incidentes**– Sugerir a construção de um mecanismo de resposta a incidentes (*playbook*) que servirá de apoio à decisão / próximos passos a serem dados de como proceder à resposta a um incidente, com base na classificação do IoC obtida.

1.4 Conteúdo

A investigação está organizada em 7 capítulos.

No primeiro capítulo é realizada uma introdução ao contexto deste trabalho, são apresentadas as principais motivações para a respetiva investigação e são enumerados os objetivos específicos do trabalho.

O segundo capítulo descreve o estado da arte, através de uma revisão da literatura abrangendo os principais conceitos, fatores e conhecimentos inerentes à investigação. Inicia-se com uma contextualização geral da problemática e das potenciais soluções e enumeram-se fatores que compõe o problema. Descreve-se que fatores dificultam a solução da problemática em investigação. Segue-se a descrição de algumas ferramentas e conceitos que podem fazer parte da solução e abordam-se técnicas e potenciais soluções inovadoras e complementares à solução.

No terceiro capítulo, inicia-se com a enumeração das questões de investigação, seguindo-se do tipo de metodologia de investigação utilizado, o *Design Science Research* (DSR), bem como a metodologia utilizada no desenvolvimento de *Software*: *Agile*.

No quarto capítulo, iniciamos a abordagem à arquitetura da ferramenta do software que propomos, abordando os pré-requisitos necessários e a arquitetura de integração proposta. Apresentamos uma visão generalista, simples e de fácil compreensão de como e onde se integrará a nossa solução, bem como os principais componentes que a compõem. Posteriormente, são apresentadas duas visões das trocas de mensagens envolvidas no motor de inferência, por meio de um esquema *Business Process Modeling Notation* (BPMN) com as atividades e as trocas de mensagens, bem como os principais objetivos, características e finalidade da ferramenta proposta; e um diagrama de sequência em *Unified Modeling Language* (UML), com uma explicação detalhada de todas as mensagens trocadas durante a execução do software, do ponto de vista do analista, desde o input até ao output final.

No quinto capítulo, apresentamos todo o processo de desenvolvimento da solução, um processo ágil de investigação & desenvolvimento, com vários avanços e retornos ao encontro da melhor solução. São abordadas todas as tecnologias utilizadas e é demonstrada uma execução do

motor de inferência, do ponto de vista do analista. No final do capítulo é apresentado uma sugestão de como realizar um playbook de auxílio à tomada de decisão / próximos passos a serem dados na resposta a um incidente de segurança com base na classificação obtida.

No sexto capítulo, executamos alguns testes de eficiência na classificação de IoCs, utilizando a métrica *Time to Qualify* (TTQ) e realizando uma matriz de confusão ("*confusion matix*") com base num teste efetuado com 240 IoCs. Por fim, realizamos uma discussão de resultados, respondendo às questões de investigação, assim como apresentamos uma reflexão critica dos resultados obtidos e das suas respetivas potencialidades e limitações.

Por último, no sétimo capítulo, apresentamos as principais conclusões da investigação e do trabalho desenvolvido, assim como abordamos as suas limitações, contribuições e oportunidades para trabalho futuro.

CAPÍTULO 2

Revisão da Literatura

2.1 Centro de Operações de Segurança (SOC)

Um Centro de Operações de Segurança / *Security Operations Center* (SOC) é composto por uma equipa de cibersegurança com habilitações especificas para monitorizar em tempo real todo o tráfego de uma rede, garantindo a sua segurança, com especial enfoque na identificação / classificação de potenciais ameaças, gestão de incidentes e reporte [16-17]. Um SOC representa um aspeto organizacional da estratégia de segurança de uma organização, combinando processos, tecnologias e pessoas de forma a garantir a segurança da operação nos processos operacionais [1].

Um SOC emprega múltiplas medidas de segurança automatizadas, como monitores de tráfego, *firewalls, scanners* de vulnerabilidade e sistemas de deteção / prevenção de intrusão (IDS/IPS). Além disso, o SOC depende fortemente de analistas de cibersegurança para investigação dos dados recolhidos e identificação de potencias ciberameaças de forma a conectarem toda a informação recolhida e classificarem os eventos, por exemplo, se a rede está sob ataque; o que é que foi atacado; e quais serão os próximos passos do atacante [6].

Os Centros de Operações de Segurança (SOCs) estão a tentar adaptar-se rapidamente às tecnologias emergentes como aplicações analíticas baseadas em AI e ML, de forma a permitir uma maior eficiência na defesa de ameaças e ciberataques mais sofisticados. Assim, o aprimoramento da automação num SOC está a tornar-se numa necessidade cada vez maior [1, 19, 26], torna-se imperativo que o SOC automatize os seus processos e as suas técnicas de análise de forma a aumentar a velocidade, o ritmo e a precisão na identificação de ameaças e na respetiva resposta [40, 43].

2.2 Segurança da Informação: Evento vs Incidente

Segurança da Informação refere-se aos processos e metodologias que são projetados e implementados para proteger informações impressas, eletrónicas ou qualquer outra forma de informação ou dados confidenciais, privados e sensíveis contra acesso não autorizado, uso indevido, divulgação, destruição, modificação ou interrupção [70]. O principal objetivo da cibersegurança é garantir a segurança da informação, isto é, é o valor da informação que deve ser protegido. Podemos segmentar este valor em três dimensões, confidencialidade, integridade e disponibilidade (CIA) [51,

70, 71]. De acordo com a ISO/IEC[1], a preservação destas três dimensões constitui o principal objetivo da cibersegurança, representando assim as principais "chaves" para a construção de um *software* seguro (Figura 2.1).

A confidencialidade, presume que somente entidades autorizadas tenham acesso a informações e atividades sensíveis [51], ou seja, traduz-se na ausência de divulgação não autorizada de informação [47]; a integridade, pressupõe que somente pessoas e meios autorizados podem alterar, atualizar ou remover materiais e funções confidenciais [51], ou seja, diz respeito à ausência de alterações não autorizadas ao sistema ou à informação em causa [47]; e a disponibilidade, é a prontidão do sistema para fornecer um serviço ou para aceder a determinada informação [47], no entanto apenas a quem está autorizado [51].

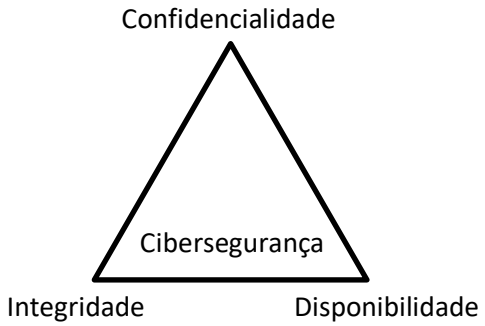

Figura 2.1 *- Triângulo da Cibersegurança (CIA). Fonte: Adaptado de [71]*

O *National Institute of Standards and Technology* (NIST)[2] [72] define um evento como qualquer ocorrência observável num sistema ou numa rede. Porém, podemos obter uma definição consensual mais completa no ISO/IEC 27035:2011 [69], onde se define um evento de segurança de informação como:

> *Uma ocorrência identificada de um sistema, serviço ou estado de rede, indicando uma possível violação de segurança da informação, política ou falha de controles,*

[1] ISSO/IEC: "A ISO é uma organização internacional não governamental independente, com 167 membros de organismos nacionais de normalização. Por meio de seus membros, reúne especialistas para compartilhar conhecimento e desenvolver Normas Internacionais voluntárias, baseadas em consenso e relevantes para o mercado que apoiam a inovação e fornecem soluções para os desafios globais". IEC representa a *international electrotechnical commission* (IEC) que coopera com a ISO.
Página oficial: https://www.iso.org.

[2] NIST: foi fundado em 1901 e faz parte do Departamento de Comércio dos EUA. O NIST é um dos laboratórios de ciências físicas mais antigos e conceituados do país, com especial enfoque na constante investigação, inovação, construção de padrões e desenvolvimento de políticas de segurança económica. Página oficial: https://www.nist.gov.

ou uma situação previamente desconhecida que pode ser relevante para a segurança.

Por outro lado, quando ocorre um ou mais eventos de segurança da informação indesejados ou inesperados que têm uma probabilidade significativa de comprometer um sistema, ameaçando a segurança da informação, trata-se de um incidente de segurança [69]. Assim, NIST [73], define um incidente de segurança da informação (ISI) como:

Uma ocorrência que realmente ou potencialmente coloque em risco a confidencialidade, integridade ou disponibilidade de um sistema de informações ou das informações que o sistema processa, armazena ou transmite ou que constitua uma violação ou ameaça iminente de violação de políticas de segurança, procedimentos de segurança ou políticas de uso aceitável.

2.3 Ciclo de Vida da Resposta a um Incidente

Um ciclo de vida de resposta a incidentes (Figura 2.2) é um modelo concetual que representa as diferentes fases durante a vida útil de um incidente de segurança. Para responder a incidentes de forma eficaz, os analistas precisam de seguir uma abordagem estruturada e organizada com funções e responsabilidades claramente definidas [68].

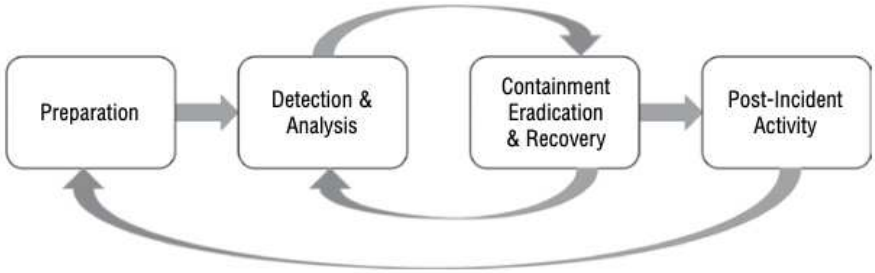

Figura 2.2 *-Ciclo de Vida de um Incidente. Fonte: [68]*

Este ciclo de vida reflete a sequência típica de estágios e atividades funcionais durante o processo de resposta a incidentes e é um modelo de referência para organizações de todos os tamanhos seguirem [68]. Nesta investigação, é utilizado este ciclo de vida como ponto de partida para a criação de um *playbook* para a gestão de respostas a incidentes de segurança, numa organização, onde iremos descrever com detalhe cada passo do ciclo de vida de um incidente de segurança.

2.4 Vulnerabilidades

Uma vulnerabilidade é uma característica / defeito de um sistema que o torna sensível a certos ataques / violação das políticas de segurança [46-47]. Segundo Shirey, R. (2000) citado por [47] as vulnerabilidades podem ser classificadas com base em três categorias:

- **Vulnerabilidade de projeto**: introduzida durante a obtenção de requisitos e desenho (Tabela 2.1, A4:2021-*Insecure Design*), um exemplo é a decisão de utilizar mecanismos de autenticação fracos ou não considerar que a comunicação (troca de mensagens) poderá ser observada na rede.
- **Vulnerabilidade de codificação**: introduzida durante a programação do *software*, ou seja, um defeito com implicações de segurança. Neste projeto a linguagem utilizada para a produção do *software* foi *Python*. A vulnerabilidade mais frequente / encontrada nos pacotes de *Python* (PyPI) é a *Cross Site Scripting: XSS* [38, 39] (vulnerabilidade incluída, em 2021, na Tabela 2.1, no grupo A3:2021-*Injection*).
- **Vulnerabilidade operacional**: causada pelo ambiente no qual o *software* é executado ou pela sua configuração. Um exemplo é a existência de contas sem senhas (*passwords*) num sistema de gestão de base de dados (vulnerabilidade incluída na Tabela 2.1, "A7:2021-I*dentification and Authentication Failures*").

No que concerne às vulnerabilidades da Web, a OWASP[3] apresenta-nos as 10 vulnerabilidades mais críticas, alerta para as suas consequências, assim como apresenta sugestões para evitar essas vulnerabilidades. Consultar o "OWASP Top 10" poderá ser um dos primeiros passos, mais eficazes, para mudar a cultura de desenvolvimento de *software* numa organização, com o principal fundamento de produzir um código mais seguro [47]. A Tabela 2.1 apresenta a lista das 10 vulnerabilidades mais críticas publicadas pela OWASP em 2021. Podemos observar (Tabela 2.1) três novas entradas relativamente ao anterior (2017), de enaltecer a vulnerabilidade *A4:2021-Insecure Design*, esta nova categoria, ampla, representa diferentes pontos fracos, expressos como "*missing or ineffective control design*"[4], isto é, vulnerabilidades associadas a falhas ao nível do *design* do *software* e consequentemente no discernimento / capacidade de validar qual o nível de segurança necessário. Este tipo de vulnerabilidade pode, por exemplo, perpetuar um armazenamento desprotegido de

[3] OWASP - *Open Web Application Security Project* é uma comunidade aberta dedicada a permitir que as organizações desenvolvam, adquiram e mantenham aplicações e APIs confiáveis
[4] Página Oficial: https://owasp.org/www-project-top-ten/

credenciais. Neste sentido, a OWASP recomenda uma melhor utilização da modelagem de ameaças, padrões de projeto seguros e arquiteturas de referência. De enaltecer ainda a passagem da *Injection* da primeira posição em 2017 para a terceira posição em 2021, com as vulnerabilidades *Broken Access Control* e *Cryptographic Failures* a ocuparem a primeira e segunda posição, respetivamente.

Tabela 2.1 - Vulnerabilidades OWASP Top 10. Fonte: Elaboração do Autor baseado no "OWASP top 10", em https://owasp.org/www-project-top-ten/.

Vulnerabilidade	Descrição
A1:2021-*Broken Access Control*	Vulnerabilidade mais vezes encontrada nas aplicações testadas pela OWASP. Acontece quando o controlo de acesso de um utilizador é comprometido. O Controlo de acesso implica que os utilizadores não realizem tarefas nem tenham acesso a informação fora das suas permissões previstas.
A2:2021-*Cryptographic Failures*	Falhas na criptografia de dados / registos, nomeadamente de dados que se enquadram na política de proteção de dados (LGPD). Este tipo de vulnerabilidade pode resultar na exposição de dados sensíveis ou comprometimento do sistema.
A3:2021-*Injection*	Vulnerabilidades ao nível da injeção de código (vulnerabilidade de codificação), pode permitir que um invasor injete *malware* para executar comandos remotos, que podem ler ou modificar uma base de dados, ou alterar dados de um site. De acordo com a OWASP, a revisão do código fonte é o melhor método para detetar se as aplicações são vulneráveis a injeções.
A4:2021-*Insecure Design* (new)	Vulnerabilidades de design inseguro (Vulnerabilidade de Projeto), surgem quando programadores ("*developers*"), *Quality Assurance* (QA) e/ou equipas de segurança não conseguem antecipar e avaliar ameaças durante a fase de *design* de código. A mitigação de vulnerabilidades de *design* requer uma modelagem consistente de ameaças para evitar métodos de ataque conhecidos.
A5:2021-*Security Misconfiguration*	Vulnerabilidade associada a uma frágil configuração de segurança, permitindo a um atacante explorar os pontos fracos da configuração do aplicativo Web, permitindo-lhe, por exemplo, ignorar métodos de autenticação e obter acesso a informações confidenciais e/ou obter privilégios elevados.
A6:2021-*Vulnerable and Outdated Components*	Um componente de software é parte de um sistema ou aplicativo que estende a

	funcionalidade do aplicativo, como um módulo, pacote de *software* ou API. Vulnerabilidades baseadas em componentes ocorrem quando um componente de software não tem suporte, está desatualizado ou está vulnerável a uma exploração conhecida, podendo assim resultar numa ameaça ao sistema de que é componente. A organização deve assegurar um plano contínuo de monitorização, triagem e aplicação de atualizações ou mudanças de configuração durante a vida útil das aplicações.
A7:2021-*Identification and Authentication Failures*	A confirmação da identidade do utilizador, autenticação e gestão da sessão são fundamentais para proteger a aplicação (Vulnerabilidade Operacional). Quando implementadas incorretamente, pode permitir que o atacante aceda a passwords, chaves, *tokens* de sessão, ou explore outras falhas da implementação que lhe permitam assumir a identidade de outros utilizadores.
A8:2021-*Software and Data Integrity Failures* (new)	As falhas de integridade de *software* e dados, acontecem quando é utilizado um software / aplicação sem aderir às melhores práticas de verificação ou autenticação, nomeadamente, quando se utilizam *pipelines* de CI/CD inseguros, possibilitando ao atacante de introduzir código malicioso, comprometendo o sistema.
A9:2021-*Security Logging and Monitoring Failures*	Vulnerabilidades relativas a insuficiente registo de *logs* e monitorização em conjunto com uma resposta a incidentes também insuficiente ou inexistente pode originar, consequente, uma potencial não deteção de intrusões e/ou ataques.
A10:2021-*Server-Side Request Forgery (SSRF)* (new)	As falhas de SSRF ocorrem sempre que uma aplicação *web* executa um pedido remoto sem validar a *URL* fornecida pelo utilizador. O servidor é iludido, levando-o a fazer uma requisição a uma rede externa, de forma a que pareça estar a realizar uma requisição para uma rede interna. Assim, os atacantes conseguem acesso a áreas que não deveriam ser acedidas externamente, como *intranets* ou páginas administrativas. Este tipo de ataque pode expor arquivos internos com informações sensíveis.

As vulnerabilidades, ao nível da cibersegurança, encontram-se nos quatro principais riscos para a estabilidade global. A identificação oportuna de violações de dados / ciberataques e a

consequente identificação e classificação podem evitar que uma organização sofra grandes perdas [20]. Comprar um antivírus ou algum *software* semelhante já não é suficiente [16].

As principais ciberameaças / ciberataques que as organizações enfrentam diariamente são [26]:

- ***Denial of Servise (DoS):*** Tentativa de sobrecarregar o sistema computacional da vítima através de um envio infindo de processos num curto período de tempo. Este ataque pode escalar para um ataque de negação de serviço distribuído (DDoS), desviando o tráfego normal de um servidor, rede ou serviço, sobrecarregando a infraestrutura alvo com um influxo esmagador de tráfego.
- ***Man-in-The-Middle (MiTM):*** Consiste na interseção de duas comunicações legitimas, colocando-se no meio de "A" e "B", e distorcendo / manipulando a conversa entre os dois sem que eles percebam que não estão a comunicar diretamente um com o outro.
- **Phishing:** Consiste em enviar (ex: via correio eletrónico) aparentemente legítimos com o intuito de os utilizadores, ingenuamente, clicarem em *links* maliciosos ou inserirem dados pessoais, informações e/ou passwords na janela apresentada (aparentemente legitima). Este tipo de ataque tem, por vezes, alguma engenharia social, uma vez que o objetivo é fazer acreditar o utilizador que a mensagem é legitima.
- ***Password attacks***: Estes ataques podem ser realizados através de "*keylogs*", isto é, monitorização do teclado da vítima, ou através de "*brute force"*, isto é, um método de ataque sistemático que utiliza combinações de letras, números e símbolos com o intuito de descobrir *passwords*, tentando o maior número de combinações possíveis até acertar na combinação correta. Uma vez que, por vezes, o processo de descoberta é longo e demorado, mais recentemente, começam a surgir ataques de "*brute force"* mais sofisticados, utilizando técnicas de AI [26, 50].
- ***Structured Query Language (SQL) injection:*** Este tipo de ataque explora vulnerabilidades em linguagem SQL de uma página *web* que contenha campos de preenchimento, injetando código "*SQL query"* nesses campos, e quando executada com sucesso, consegue aceder à base de dados e usurpar dados como nomes de utilizador e palavras-chave.
- ***Cross-site scripting (XSS):*** Este ataque consiste na injeção de um script malicioso numa página *web* vulnerável, que quando executada por um utilizador, dependendo do objetivo do atacante pode ser mais ou menos lesivo, isto é, pode ser um ataque simplesmente para descobrir um nome ou por outro lado conseguir acesso remoto ao computador.
- ***Eavesdropping:*** Este ataque consiste na escuta ilícita de comunicações com intuito de usurpar dados / informação. Ao contrário do MiTM este ataque não manipula a informação para a enviar de volta a um utilizador, apenas interceta e armazena informação.

2.5 Ferramentas de Cibersegurança

Nesta secção iremos abordar algumas das principais ferramentas de cibersegurança, que são partes fulcrais na resposta a incidentes de segurança, nomeadamente as plataformas CTI para enriquecimento e partilha de conhecimento sobre determinado IoC e os sistemas IDS, SIEM e SOAR, como os principais mecanismos de monitorização, deteção e resposta às potenciais ciberameaças numa organização.

2.5.1 Cyber Threat Intelligence (CTI)

Gartner define *Threat intelligence* como:

> *"A tarefa de reunir conhecimento baseado em evidências, incluindo contexto, mecanismos, indicadores, implicações e conselhos acionáveis, sobre uma ameaça ou perigo existente ou emergente para ativos que podem ser utilizados para informar decisões sobre potencias respostas a essa ameaça ou perigo"*[5].

Cyber Threat Intelligence (CTI) está a revolucionar o ecossistema da cibersegurança, permitindo que as organizações compartilhem registos de incidentes / ciberameaças e possibilitando uma defesa proactiva contra tentativas sofisticadas de intrusão [20]. CTI é uma base de dados / conjunto de evidências e conhecimento que contém conselhos e sugestões para mitigar uma ciberameaça [21], ou seja, o objetivo é fornecer informações sobre os indicadores técnicos, contexto, motivação e mecanismos acionáveis relacionados com a ameaça existente e emergente [22, 49] e dessa forma auxiliar as equipas de resposta a incidentes, com informações sobre a ameaça que está a ocorrer, bem como os métodos já testados e conhecidos para evitá-la [14].

Combinando as definições de trabalho do governo dos EUA, a indústria e estudos empíricos, MITRE[6] propõe a seguinte definição de CTI [49]:

> *"CTI refers to the collection, processing, organizing, and interpreting of data into actionable information or products that relate to capabilities, opportunities, actions, and intent of adversaries in the cyber domain to meet a specific requirement determined by and informing decision-makers".*

[5] Fonte: https://www.gartner.com/doc/2487216/definition-threat-intelligence

[6] Empresa sem fins lucrativos, pioneira no interesse público, MITRE trabalha em parceria com um ecossistema de inovação do governo (EUA), setor privado e académico ao nível da cibersegurança.

Se um SOC quiser evoluir para uma defesa menos reativa e mais proactiva, dinâmica e antecipatória, produzir, consumir e automatizar conexões / processos com plataformas de CTI é essencial [49], assim como a aplicação de técnicas e algoritmos de ML às propriedades de CTI pode melhorar a precisão da análise de previsão de ameaças e formulação de hipóteses [70].

2.5.2 Sistema de Deteção de Intrusões (IDS)

Um Sistema de Deteção de Intrusões (IDS) é um dispositivo de *software* e/ou *hardware* que monitoriza tráfego e outros dados, de modo a detetar ataques e intrusões num sistema informático [47]. O objetivo do IDS é identificar acessos incomuns ou ataques à segurança da rede interna [23]. É um mecanismo de controlo de qualidade da conceção e da administração das políticas e mecanismos de segurança de uma organização [46]. Uma analogia simples é a dos alarmes dos automóveis ou de casas [46], que detetam indícios / suspeitas de intrusão. Porém, uma das grandes limitações deste instrumento é o elevado número de falsos positivos ou, por outro lado, quando excessivamente configurado, um elevado número de falsos negativos [46, 63, 64]. Torna-se, desta forma, pertinente e fundamental desenvolver complementos tecnológicos inteligentes para reduzir estes falsos positivos sem comprometer o reconhecimento das reais ameaças, isto é, sem aumentar os falsos negativos [43, 64].

2.5.3 Security Information and Event Management (SIEM)

O termo SIEM foi usado pela primeira vez em 2005 pela *Gartner* (*Mark Nicolette e Amrit Williams*) [12], para definir uma tecnologia / ferramenta de cibersegurança que coleta, armazena e correlaciona dados de eventos de cibersegurança em tempo real e, posteriormente, em caso de necessidade, comunica com um sistema de alerta para informar o analista de segurança [12, 16, 19, 70], ou seja, o SIEM é uma solução que analisa eventos / *logs* (Figura 2.3), por exemplo o tráfego de um endereço de IP específico pode não significar muito para a *firewall*, no entanto o SIEM consegue associar esse IP a um *log* que indica que esse IP já foi utlizado previamente num ataque de *Brute Force* a uma *password,* por exemplo [16].

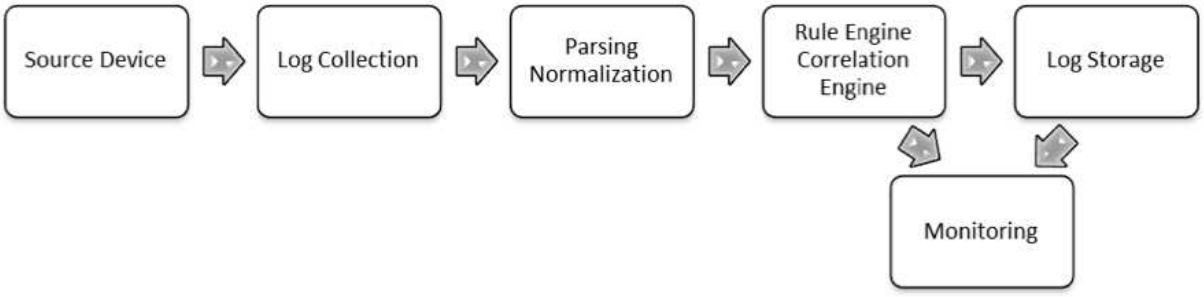

Figura 2.3 *– Componentes básicos de um SIEM. Fonte: [2]*

No entanto, embora a nova geração de SIEMs forneça recursos poderosos em termos de correlação, armazenamento, visualização e desempenho, bem como a capacidade de automatizar o processo de reação selecionando e configurando as respetivas contramedidas, podem ser tomadas medidas para tornar o sistema SIEM mais útil para os analistas de segurança e até mesmo minimizar a possibilidade de os analistas cometerem erros durante a tomada de decisões. Uma das muitas ações que podem ser tomadas nesse sentido é a utilização de ferramentas de CTI [19]. Outro aspeto limitativo, são as regras de correlação de eventos / logs [16], as regras atuais do SIEM são fracas, a maioria deles usa um encadeamento booleano básico de eventos que verificam um caminho de ataque específico (um em milhares de possibilidades). Muito poucas soluções SIEM têm um mecanismo de correlação avançado integrado capaz de realizar o desvio e a correlação histórica útil, por exemplo, para verificar os registos após a primeira deteção do ataque [2].

2.5.4 Security Orchestration, Automation and Response (SOAR)

O termo "SOAR" foi apresentado por *Gartner* no final de 2017 para se referir a tecnologias de segurança que ajudam as equipas de um SOC a padronizar, gerir e automatizar processos entre diversas ferramentas / produtos [35]. Em 2021, [10] apresentaram a seguinte definição de SOAR:

> *Planeamento, coordenação, cooperação e integração de ponta a ponta das atividades de serviços, processos, aplicativos e ferramentas de segurança díspares, juntamente com a equipa do SOC, para automatizar ações necessárias em resposta a incidentes de segurança, incorporando processos e tecnologias de segurança corporativa.*

Os principais casos de uso para ferramentas SOAR centram-se nas operações de segurança e resposta a incidentes. A natureza de uso geral dessas ferramentas, no entanto, levou a casos de uso emergentes em orquestração de segurança na *cloud* [27].

Os sistemas SOAR são uma classe especial de produtos de *software* que podem ser usados para coletar dados sobre ISI de várias fontes, processá-los e configurar uma resposta automatizada usando cenários de resposta típicos [28]. O foco principal desta tecnologia é automatizar diversos processos de segurança como auditoria de segurança de rede, gerando senhas privilegiadas e coordenando e executando diversas ferramentas e grupos de segurança [14]. Um sistema tecnológico como o SOAR é capaz de automatizar diversas tarefas comuns geralmente realizadas por analistas de SOCs, que possuem muitas tarefas de administração, incluindo escrever relatórios e documentar procedimentos de segurança, conseguindo desta forma reduzir esse trabalho administrativo e melhorar os recursos do relatório, agregando informação ("*intelligence*") de várias fontes (internas e externas) e posteriormente exibindo-a num painel visual ("*dashboard*") [29].

Tecnologias SOAR, como parte integral de um SOC, permite uma rápida e mais eficiente resposta aos desafios operacionais digitais [18], bem como podem ajudar na rápida deteção e resposta a incidentes, no entanto sem negligenciar a utilização de recursos humanos qualificados. O uso do SOAR, englobando uma combinação de operações de ML com recursos humanos, é uma solução comprovada e eficaz na execução de operações de segurança em fluxos de trabalho de proteção complexos e na resposta às atuais ameaças que enfrentamos [18], o que consequentemente se começa a refletir em organizações, com SOC, que começam a apostar num único analista, substituindo muitas das tarefas manuais dos analistas por um SOAR [30].

Assim, conforme ilustrado na Figura 2.4, o SOAR engloba as seguintes aplicações, *Network Forensics, Workflow Management, Incident Management, Threat Intelligence e Compliance Management* [24, 35].

Figura 2.4 *– Aplicações Tecnológicas de um SOAR. Fonte: [24]*

2.6 O novo paradigma: AI/ML aplicadas à cibersegurança

AI assenta na premissa de que a inteligência das pessoas (o potencial inato / capacidade de um indivíduo consciente concluir sobre uma determinada informação) pode ser descrita com tanta precisão que é passível de ser simulada por uma máquina [44], ou seja, em como as máquinas conseguem pensar ou apresentar soluções corretamente [52]. Quando se exploram as capacidades da AI, procura-se maximizar o quão perto as máquinas conseguem pensar e reagir como se de humanos se tratassem [53] e por vezes superar as capacidades humanas, automatizando milhares de processos.

Após várias décadas de pesquisa, a AI deixou de ser apenas um objeto de estudo e incorpora agora diversas soluções, mais complexas e interdependentes. As principais formas de AI exploradas incluem raciocínio, apresentação de conhecimento, automação de processos, ML, processamento de linguagem natural, visão computacional, robótica e inteligência comum [44]. No que concerne à cibersegurança, as aplicações de AI mais relevantes têm-se verificado no desenvolvimento de sistemas de deteção de intrusões [54]. As soluções de cibersegurança concentram sinergias na análise de tráfego na rede e na classificação desse mesmo tráfego: legítimo ou malicioso. Com base nestas classificações, são aplicados métodos de contenção baseados em regras estabelecidas pelas equipas de segurança. Porém, a evolução tecnológica não trouxe só benefícios para a segurança, mas também muitos desafios à integridade das organizações, os ataques informáticos são cada vez mais sofisticados [55], atacantes / criadores de *malware* recorrem a técnicas, táticas e procedimentos (TTPs) utilizando tecnologias de AI e ML, não só para propagar ataques como para evitar a sua deteção [40].

Nos dias que correm, as soluções de cibersegurança baseadas em AI e ML são cada vez mais procuradas, no sentido da automação de processos e na resposta a incidentes de segurança detetados. O principal objetivo é uma resposta mais rápida e mais eficiente aos incidentes registados através da AI, bem como uma deteção / reconhecimento mais rápido de um ataque, num conjunto de milhares de registos, através da ML [3].

Na última década, os sistemas de cibersegurança mais utilizados têm sido o IDS e o SIEM, no entanto, estes sistemas estão limitados em termos de proatividade nos avisos (*warnings*) e na capacidade de resposta a eventos de segurança complexos. O SIEM gera centenas ou milhares de alertas por dia, requerendo uma complexa e difícil configuração para reduzir falsos positivos, o que consequentemente faz com que os analistas só consigam analisar uma parte dos alertas gerados, uma vez que possuiu uma reduzida capacidade de orquestrar e automatizar processos [12]. No sentido de melhorar os recursos de deteção, correlação e reação, a próxima geração de SIEMs deve integrar tecnologias de AI / ML nos seus principais mecanismos [77]. A adoção destas tecnologias em SIEMs ofereceriam recursos preditivos particularmente úteis para a análise de comportamento anormal do

tráfego de rede, ferramentas e utilizadores. No entanto, um passo à frente para deteção, mitigação e prevenção de ciberataques é considerar a utilização de AI / ML em soluções SOAR que seriam idealmente integradas em plataformas SIEM [2]. Os sistemas de defesa baseados em AI / ML são capazes de analisar uma grande quantidade de dados e identificar padrões suspeitos em tempo real. Os principais alvos para aplicações de AI / ML incluem deteção de intrusão (ataques baseados em rede), *phishing* e *spam* (*e-mails*), deteção e caracterização de ameaças e análise comportamental [10].

Sistemas de segurança baseados em AI / ML, demonstraram ser capazes de criar automaticamente perfis de comportamento, orquestrar e automatizar respostas a incidentes, a capacidade de detetar num número elevado de dados atividades anómalas, deteção em tempo real de assinaturas de ataques (deteção de *malware*), bem como interagir com outros sistemas de informação de segurança de forma a reduzir os incidentes, resultando num número muito mais reduzido de eventos para o analista a analisar e reduzindo drasticamente os falsos positivos [3, 12-14]. Por outro lado, sistemas baseados em algoritmos ML não conseguem determinar ações maliciosas, no sentido literal da questão, uma vez que não sabem distinguir o bem do mal. O papel destes sistemas é enaltecer, classificar e prever situações anómalas, baseado em características observadas [3].

O *Machine Learning*, como já referido, é parte integrante de um sistema da Inteligência Artificial, convencionalmente os métodos / algoritmos de ML podem ser classificados em duas categorias: aprendizagem supervisionada e não supervisionada. No ML supervisionado, as amostras de dados são rotuladas de acordo com a sua classe (por exemplo, malicioso ou legítimo) [26]. Os dados para treinar o algoritmo são, por norma, introduzidos manualmente, exigindo que os analistas detetem padrões e agrupem, rotulem e organizem os dados por classes, para que com esses dados seja possível criar um modelo matemático capaz de reconhecer novas amostras de dados introduzidas e agrupar na respetiva categoria. Por outro lado, no ML não supervisionado, não há rotulagem ou treino do algoritmo, o algoritmo determina o grau de coerência / dispersão entre amostras de dados, sistematicamente agrupa os dados por classes e classifica essas amostras de acordo com a coerência dos dados dentro da classe [26].

No que concerne ao ML existem vários modelos que podem ser aplicados à cibersegurança e especificamente à monitorização de ameaças com automação de processos utilizados neste projeto, com um grau elevado de sucesso e proficiência, nomeadamente, o *Support Vector Machine* (SVM), as árvores de decisão, Redes Neurais Artificiais (ANNs), *K-Nearest Neighbors* (K-NN), entre outros.

1) ***Support Vector Machine*** **(SVM)** assenta no conceito de minimização de risco estrutural, que oferece vantagens de velocidade e escalabilidade. A principal ideia por de trás do SVM é descobrir um limite de decisão num espaço multidimensional que

divide padrões invisíveis em grupos distintos [42]. O principal objetivo deste modelo é agrupar dados em duas classes principais distintas [26], por exemplo, aplicando a este projeto, dividindo os IoCs em malignos / ameaças ou benignos / potenciais ameaças (exemplo na Figura 2.5).

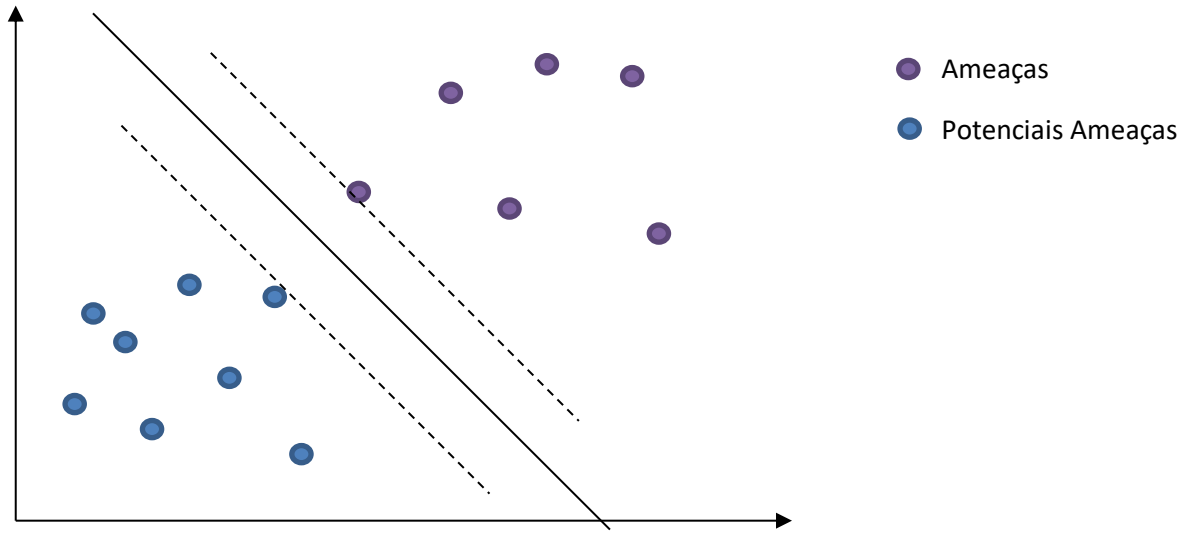

Figura 2.5 *- Exemplo de um output de um modelo SVM. Fonte: Elaboração do Autor*

2) **Árvore de decisão (*Decision Tree*)** é um algoritmo que encontra iterativamente um recurso que melhor categoriza as amostras de dados, dividindo e segmentando os dados constantemente até formar uma estrutura semelhante a uma árvore [13,26]. A Figura 2.6 mostra um exemplo de árvore de decisão que classifica o impacto dos IoCs numa organização usando regras que levam a classificações de baixo impacto, moderado e severo. Na literatura, há um número considerável de estudos que propõem modelos de deteção de intrusão baseados em árvores levando em consideração a classificação e seleção de recursos de segurança. Este processo pode melhorar a precisão da previsão e minimizar a complexidade da computação [15].

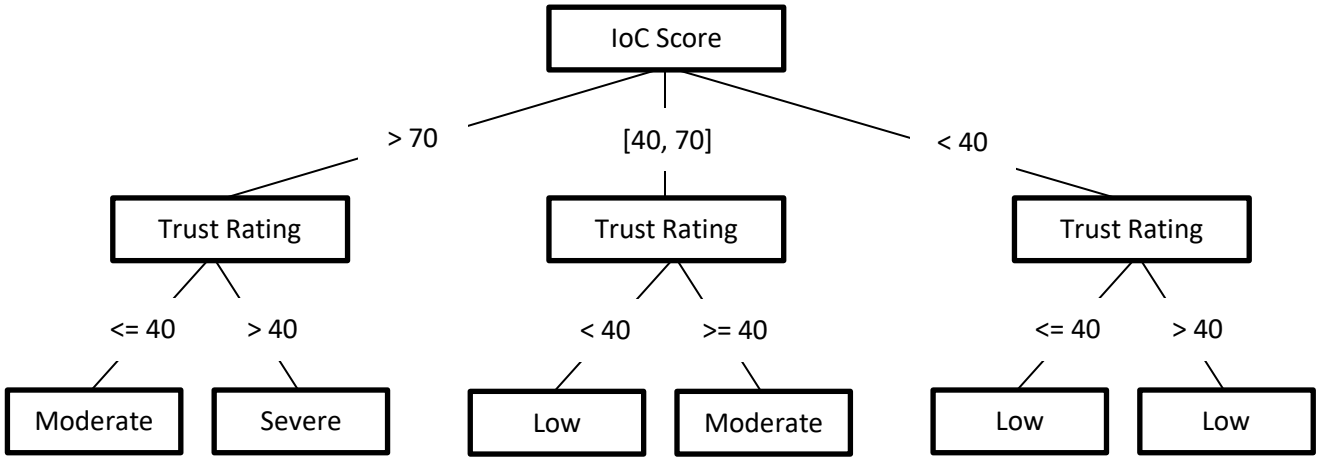

Figura 2.6 *- Exemplo de um modelo de Árvore de Decisão. Fonte: Elaboração do Autor*

3) **Redes Neurais Artificiais (ANNs)** são baseadas no processo do cérebro humano, o seu objetivo é simular o comportamento humano para tomar decisões, aprender e adaptar-se à informação recebida. As ANNs são interessantes pela sua capacidade de aprender, mesmo que as informações fornecidas estejam incompletas. As ANNs podem ser usadas quando há pouco conhecimento das relações entre atributos e classes, são adequadas para entradas e saídas de valor contínuo, ao contrário da maioria dos algoritmos, são bem-sucedidas numa ampla variedade de problemas do mundo real [41]. No entanto, as regras para detetar, prevenir, remediar, recuperar ou responder a incidentes de segurança ainda é, em grande parte, determinado manualmente por analistas do SOC e podem envolver configurações irreais na modelagem dos ambientes dinâmicos da cibersegurança; e na maioria dos casos depende muito da lógica imposta pelos analistas do SOC [10].

4) **K-Nearest Neighbors (K-NNs)** é o algoritmo mais simples entre todas as abordagens de ML, onde o *output* é calculado com base em "K" vizinhos mais próximos ou num "K" pré configurado / treinado. O cálculo do *output* difere com base no treino manual do programador [42]. Este método terá um *output* semelhante ao verificado, anteriormente, no SVM. O algoritmo irá dividir os dados em duas classes, assim poderá ser utilizado em conjunto com o SVM de forma a validar a segmentação e proporcionar um algoritmo mais robusto e mais eficiente [45], nomeadamente ao nível da distinção do que é uma ameaça ou não. Tanto este método como o SVM podem ser uma excelente mais-valia na deteção de ataques a partir do "dia zero" [26].

Com cada vez mais frequência *hackers* utilizam aplicações baseadas em AI / ML para automatizar ataques, como por exemplo, replicar vozes conhecidas de pessoas para extorquir ativos financeiros [12]. Neste sentido, seria interessante e pertinente explorar as capacidades da AI / ML de forma a implementar sistemas IDS de nova geração, com capacidades de automação inteligente, para a rápida deteção e consequente impedimento de ciberataques [3].

Analistas / equipas de cibersegurança estão constantemente sobrecarregados por revisões manuais de alertas de segurança e respetivas coordenações de múltiplos sistemas de segurança [6-7]. A triagem exaustiva de falsos positivos é um desperdício de recursos, no entanto sem análise dos alertas há o risco de ataques serem consumados. Por outro lado, os atuais modelos de cibersegurança começam a ficar obsoletos para fazer frente à adoção de novas tecnologias de AI, utilizadas nos novos ciberataques. Os sistemas de AI e ML são, assim, ambos uma bênção e uma maldição no que concerne à cibersegurança [3].

Urge a necessidade da adoção de novos mecanismos de defesa, ao nível da cibersegurança [3-4, 11-12]. Os sistemas baseados em AI conseguem responder a ciberataques autonomamente e através da ML agilizar a execução das tarefas associadas à resposta do respetivo incidente [24].

O futuro da segurança da informação depende fortemente da aplicação de métodos de AI / ML, de forma a melhorar situações de reconhecimento, envio de alertas urgentes em tempo real e gestão de crises em incidentes de segurança complexos, por meio da automação da deteção de ameaças e da respetiva ativação de contramedidas apropriadas [25, 44]. Um sistema, focado na automação de resposta a incidentes de segurança, com a capacidade de coletar, analisar, correlacionar, validar eventos, e orquestrar investigações e processos de resposta, por meio de inteligência artificial: *SOAR* (*Security Orchestration, Automation and Response*) [27, 30].

Eventualmente, a automação alcançará o que a *Deloitte* e a *Google Cloud* [33] descrevem no seu artigo "*Future of the SOC*", envolvendo a automação na tomada de decisão através de modelos baseados em ML e AI, permitindo que os analistas de um SOC sejam mais proficientes, e tenham mais tempo livre para se concentrarem nas "tarefas mais difíceis".

2.7 Trabalhos Relacionados: Soluções de AI/ML

Nesta secção apresentamos alguns trabalhos relacionados, onde foi utilizado automação de processos, inteligência artificial e/ou ML, com o objetivo de classificar ameaças e/ou prevenir ciberataques. A maioria dos projetos de investigação relacionados focam-se na aplicação de inteligência artificial, na forma de ML, com o objetivo de detetar intrusões / ciberataques [79]. No entanto, como se pode observar na Tabela 2.2, em 2013 [77] desenvolveram um modelo de ML com o objetivo de complementar um SIEM, identificando os eventos que são potenciais ameaças e os que poderão ser

"falsos positivos", de forma a otimizar e reduzir significativamente a intervenção dos analistas de um SOC. Em 2015 [78] desenvolveram um modelo de ML (K-NN) com o objetivo de filtrar eventos num IDS, medindo o risco de cada evento ("não critico" a "muito critico") para a organização e reduzindo os falsos alertas. Em 2018 [76] desenvolveram um modelo ML (*Decision Tree*) para auxiliar o analista de um SOC a classificar eventos como "ameaças" ou "falsos positivos", atribuindo a cada IoC um "*threat score*", que funciona como um "*trust rating*", ou seja, com o nível de confiança que o modelo tem na avaliação se os alertas são uma ameaça. Em 2020 [75] desenvolveram um modelo de ML, baseado em 3 algoritmos diferentes, com especial enfoque em dois deles, K-NN e SVM, com o objetivo de classificar IoCs, ordenando-os por grau de criticidade para a organização ("baixa" a "critica"). Em 2021 [7] desenvolveram uma estrutura integrada, por meio de um modelo de ML (*clustering K-means* e ANN), com *inputs* de CTI e *scores* CVE (*Common Vulnerability Exploits*), com o principal objetivo de reduzir falsos positivos, identificando com precisão as reais ciberameaças e os falsos alertas, apresentando uma taxa de 99% de correção ("*accuracy*") nessa identificação. Também em 2021, [81] desenvolveram um mecanismo de ML para classificação de IoCs, como sendo ou não ameaças reais, com um grau de precisão elevado, utilizando uma abordagem baseada em *Logistic Regression*[7] (LR) em conjunto com a Teoria de *Dempster-Shafer*[8](D-S). Os autores começaram por utilizar / explorar vários algoritmos de ML de forma a apurar o mais preciso (*Feature Selection*) e lhes permitisse construir o modelo mais útil e eficiente, tendo, no final, concluído que o método mais eficiente na classificação de eventos como ameaças ou falsos alertas é o método LR-DS (*Logistic Regression e a teoria de Dempster-Shafer*). Em 2022 [82] desenvolveram uma estrutura integrada de gestão de riscos de cibersegurança, utilizando modelos de ML para prever os tipos de risco de cada IoC, para que as organizações possam tomar as medidas proactivas necessárias para responder a esses incidentes. A estrutura também inclui uma ferramenta de suporte para automatizar algumas das atividades de gestão de risco, integrando CTI e consequentemente tornando as atividades de gestão mais eficazes.

Tabela 2.2 *- Trabalhos Relacionados. Fonte: Elaboração do Autor*

Ref No	Algoritmo	Objetivo	Descrição e *Performance*
[77] 2013	ANN e Genetic Programming (GP)	Modelo de ML para categorizar eventos como ciberameaças ou falsos positivos, com o intuito de complementar um SIEM e	O modelo divide-se em duas partes uma primeira parte onde recorre a um algoritmo de ANN, classifica os eventos de acordo com o seu

[7] LR: Modelo estatístico, frequentemente utilizado para classificação e análise preditiva, uma vez que estima a probabilidade de um evento ocorrer [81].

[8] D-S: Semelhante à probabilidade *Bayesiana*, o *Dempster-Shafer* (D-S) ou teoria da evidência é um modelo para resolver problemas de inferência, congregando evidências de múltiplas fontes heterogéneas [81].

		reduzir o trabalho de análise de eventos por parte de um analista.	contexto (*tags*[9] associadas e extraídas do SIEM, pelo analista) como "positivo" ou "negativo". Posteriormente, com base nessa classificação, utiliza uma metodologia GP[10], uma técnica de ML inspirada na evolução biológica, para correlacionar a classificação do algoritmo anterior e classificar os eventos como "ameaças", "falsos positivos" ou "falsos negativos". Durante o período de testes, 11 dias, apresentou uma taxa de 86,89% na identificação de alertas reais. No entanto, apenas 8,82% dos "falsos negativos" foram reportados por este modelo.
[78] 2015	K-NN	Modelo de ML com o objetivo de classificar incidentes de segurança / IoCs filtrados num IDS, pelo risco que representam para a organização ("nada crítico" a "crítico").	O modelo consiste em três componentes principais: base de dados (BD), classificação e filtro. Na BD armazena os alertas classificados e utiliza-o para treinar o algoritmo K-NN. Na classificação, os alertas introduzidos pelo IDS são agrupados em *clusters* apropriados e atribuem-lhes uma *label*. O filtro de alertas é usado para reduzir alertas indesejados. O conhecimento do analista é muito importante no filtro, de forma a classificar corretamente os alertas no processo de treino do algoritmo. Os resultados experimentais indicam que o filtro de alertas pode alcançar bons resultados de filtragem, reduzindo os falsos alertas.
[76] 2018	Decision Tree	Modelo de *ML* para auxiliar o analista de um SOC a identificar ciberameaças, atribuindo a cada evento uma das seguintes *tags*: "Ameaça" ou "Falso Positivo", com um "*trust rating*" associado (nível de confiança na *tag* atribuída).	O modelo foi treinado, durante 6 meses, com base nos *tags* que o analista atribuiu a cada evento / alerta detetado num SOC: "Ameaça" ou "Falso Alerta" e um "*threat score*" associado (nível de confiança associado à respetiva atribuição da *tag*). Ao longo de 3 meses, testaram a eficiência da classificação autónoma do modelo, apresentando uma taxa de sucesso de 87% na classificação de "Ameaça". Por outro lado, não

[9] Tag: É uma palavra-chave que representa a natureza de uma determinada descrição de evento. Cada evento é, portanto, definido por uma tag [77].

[10] GP: *Genetic Programming*, aplicado a um mecanismo de correlação SIEM. GP é uma técnica de ML inspirada na evolução biológica, e já foi aplicada em abordagens de deteção de intrusões / ciberataques [77].

			conseguiu classificar IoCs como "Falsos Positivos" em 69% dos casos.
[75] 2020	K-NN, SVM e Bayesian model	Modelo de ML para classificação de IoCs, ordenando-os por grau de criticidade ("Low" até "Critical").	Os autores testaram diversos métodos de ML (*linear regression*, *decision tree*, K-NN, SVM e *Bayesian models*), no entanto ao realizarem vários testes de eficiência na classificação de ameaças optaram por utilizar apenas os 3 mais precisos, K-NN, SVM e o *Bayesian model*. Os resultados da classificação de ameaças foram correlacionados com um antivírus e uma *firewall*, apresentando resultados semelhantes.
[7] 2021	CTI & ML (*clustering* com *k-means* e *ANN*)	Modelo de ML, com *inputs* de CTI, com o principal objetivo de reduzir falsos positivos, identificando com precisão as reais ameaças e os falsos alertas	Um modelo que propõe uma estrutura integrada, com um mecanismo automatizado de CTI, integrado com o SIEM, capaz de: - Prevenir e identificar *malware;* - Filtrar "falsos positivos"; - Pesquisar e analisar automaticamente ciberameaças. Nos testes efetuados ao modelo ANN de identificação de ciberameaças, classificando como "true positive" or "true negative", ou seja, ciberameaça ou falso alerta, apresentou uma taxa de 99% ("*accuracy*")
[81] 2021	*Logistic Regression e* Teoria de *Dempster-Shafer*	Modelo de ML para apurar / classificar eventos como "ciberameaças" e "falsos positivos"	Algoritmo de ML, LR-DS, com base em *Logistic Regression e na teoria de Dempster-Shafer,* é capaz detetar com precisão se um IoC se trata de uma ciberameça, falsos negativos e falsos positivos. Os testes foram realizados com o auxílio do *Canadian Institute for Cybersecurity,* que forneceram um conjunto de dados de um IDS, composto por dados benignos e instâncias maliciosas. Foram testados 8 modelos de ML, *Analysis of Variance* (ANOVA*), Decision Tree* (DT), *Extra Trees* (ET), L1 *regularisation* (Lasso), *Random Forest* (RF), LR-DS, *Principal Component Analysis* (PCA) e *Permutation Importance* (PI), tendo no fim optado pelo algoritmo que apresentou os resultados mais corretos, com uma percentagem de cerca de 99% ("*accuracy"*): LR-DS.
[82] 2022	CTI & ML	Plataforma de gestão de riscos dos incidentes de segurança	A plataforma proposta utiliza vários repositórios de vulnerabilidades e

		numa organização, utilizando um processo sistemático baseado em ML e CTI com o objetivo de classificar ameaças, por tipo de ameaça e impacto para a organização. No fim, sugere ainda soluções de resposta a incidentes de segurança com base no tipo de incidente.	controlo de riscos: CWE[11], CAPEC[12], CSC[13]. Recorre ainda a CTI e modelos ML para classificação das ameaças (tipo de ameaça), tais como: *K-nearest neighbours* (KNN), *Naıve Bayes* (NB), artificial neural network (ANN), *decision tree* (DT), *random forest* (RF) e *logistic regression* (LR). Nos testes realizados, o modelo que obteve a melhor precisão na classificação das ameaças foi o DT, com uma taxa ("*accuracy*") de 92,92%.

2.8 Automatização e Orquestração sem SOAR

Existem variadas formas de obter uma forte automação e orquestração para triagem de alertas num SOC, bem como para respostas a incidentes de segurança. De forma a alcançar resultados semelhantes aos anunciados por produtos SOAR, de forma gratuita ou menos dispendiosa, pode ser configurado num SOC, através de *scripts*[14] simples usando uma linguagem popular como *Python* e APIs de produtos documentados [49]. Se uma organização utilizar uma plataforma de resposta a incidentes para automatizar processos, os analistas podem pré-configurar alguns atributos, como autor, notificações automáticas ou acionamento de outros processos. Algumas plataformas também podem calcular automaticamente a gravidade do incidente com base na urgência e nos fatores de impacto que os analistas inserem ou pré-configuram como parte do modelo [68]. Nesta investigação, procurámos iniciar estes mecanismos, com automação de processos na classificação de potenciais ameaças e sugerindo *playbooks* / *use cases* a aplicar de resposta, de acordo com essa classificação. Em aplicações futuras será possível aplicar automação de processos, dentro das ferramentas utilizadas nesta investigação, para ativar *use cases*, automatizando as respostas aos incidentes de segurança verificados.

[11] CWE: Common Weakness Enumeration, é uma lista de tipos de vulnerabilidades de software e hardware, serve como uma linguagem comum e um ponto de partida para a identificação, mitigação e prevenção de ameaças. Página oficial: https://cwe.mitre.org/

[12] CAPEC: Common Attack Pattern Enumerations and Classifications, fornece um dicionário abrangente de padrões conhecidos de ataques utilizados / conhecidos que exploraram vulnerabilidades conhecidas. Página oficial: https://capec.mitre.org/

[13] CSC: Critical Security Control, uma organização sem fins lucrativos orientada para a comunidade, que partilha as melhores práticas reconhecidas globalmente para proteger dados e sistemas de informação. Página oficial: https://www.cisecurity.org/controls

[14] Script: programa ou sequência de instruções que é interpretada ou executada por outro programa e não pelo processador do computador

CAPÍTULO 3

Metodologia

3.1 Questões de Pesquisa

De forma a elaborar as questões de pesquisa é necessário perceber de que se trata o problema. Como já enunciado, o SOC, na sua área de operações, é composto por analistas, no entanto, nunca em número suficiente para o elevado número de eventos de segurança, falsos positivos para analisar, bem como um elevado tempo despendido em tarefas administrativas. Um analista de um SOC necessita de consultar diferentes fontes externas, analisar e correlacionar toda a informação. Tendo por referência estas problemáticas, a principal questão de pesquisa (MRQ) a responder nesta investigação é:

Será uma solução automática, com a capacidade de classificar eventos de segurança e medir o impacto das potenciais ciberameaças para a organização, uma solução eficiente na gestão de eventos ou incidentes de segurança, num SOC?

No entanto, de forma a segmentar a MRQ, é importante definir as questões de pesquisa (RQ) segundo um ciclo regulativo, ou seja, questões relacionadas ao problema, às potenciais soluções e à avaliação (adequando a solução ao problema) [36]:

RQ1 – (Problema) Quais os problemas associados à gestão de incidentes de segurança num SOC?

RQ2 – (Solução) Quais as potenciais soluções para melhorar a eficiência da gestão de incidentes num SOC, ao nível da medição / classificação do risco das potenciais ciberameaças?

RQ2.1 – (Solução) De que forma estas soluções poderão ser implementadas e automatizadas?

RQ3 – (Avaliação) Até que ponto os problemas típicos associados à gestão de incidentes num SOC podem ser resolvidos com uma classificação automática de eventos ou incidentes de segurança?

Estas questões constituem o enquadramento desta investigação, e as suas respostas darão uma opinião objetiva sobre quais devem ser as principais regras a ter em consideração na classificação de uma ameaça e de que forma um motor de inferência automatizado poderá contribuir para a gestão automatizada de incidentes de segurança e melhoria de um SOC.

3.2 Metodologia de Investigação: *Design Science Research*

A metodologia utilizada nesta investigação e que contribuirá para dar resposta às questões de pesquisa enumeradas anteriormente é a *Design Science Research* (DSR). Esta metodologia é amplamente utilizada em investigações de desenvolvimento de sistemas de informação, e visa criar uma solução inovadora para o problema, que na maioria dos casos se baseia nos componentes existentes de uma solução e combina, revisa e expande o conhecimento de *design* existente [98].

DSR trata-se de um paradigma de resolução de problemas [98] que procura aprimorar o conhecimento humano com a criação de artefactos inovadores, construindo um design de conhecimento por meio de soluções inovadoras para problemas do mundo real [99].

A literatura identifica 6 etapas na metodologia DSR [98, 99], que foram guiando e balizando este processo de investigação. Iniciou-se com a identificação do problema e da motivação; de seguida a definição dos objetivos para a solução proposta; o desenho e o desenvolvimento do artefacto; uma demonstração da execução da solução; e por fim a avaliação; e a consequente comunicação dos resultados.

3.3 Metodologia de Desenvolvimento de *Software*

Para o desenvolvimento deste projeto foi utilizado uma metodologia de desenvolvimento *Agile* de *Software* (Figura 3.1), ou seja, uma metodologia de desenvolvimento que se adapta às situações em que os requisitos mudam com frequência [37]. Na medida em que os requisitos técnicos sofreram várias alterações ao longo da investigação / produção do *software* optou-se por adotar esta metodologia de desenvolvimento, capaz de fornecer flexibilidade para lidar com a mudança de requisitos, comunicação aprimorada, mecanismos de coordenação e melhoria da qualidade [37], bem como os vários testes de investigação e o *feedback* da organização, proporcionaram um aperfeiçoamento continuo do produto final. Por fim, o objetivo posterior à implementação, será analisar / perceber se os resultados da experiência desenvolvida são capazes de responder à *Main Research Question* (MRQ) e ao problema apresentado, à qual iremos dar resposta na Discussão e Conclusão final, nos capítulos 6 e 7, respetivamente.

Figura 3.1 *- Ciclo de Vida de Desenvolvimento de Software Agile (Agile SDLC). Fonte: Elaboração do Autor baseado no SDLC de https://snyk.io/learn/sdlc-software-development-life-cycle/*

Requisitos: De forma a ser possível iniciar uma investigação é necessário conhecer os requisitos iniciais, para este projeto esses requisitos são o desenvolvimento de um sistema automatizado, capaz de, através da introdução de um IoC, realizar uma análise em plataformas de CTI; recolher informações sobre esse IoC, classificar quantitativamente e qualitativamente o nível de ameaça desse IoC e no final devolver um reporte final ao analista do SOC. O reporte deverá incluir um valor (*score*) que represente o risco dessa ameaça, um valor de confiança nesse *score* (*trust rating*) e um impacto desse IoC para a organização. No entanto, ao longo do ciclo de desenvolvimento (Figura 3.1), o número de requisitos vai crescendo e/ou sofrendo adaptações / alterações consoante o desenvolvimento da investigação. Esses requisitos / novas *"features"* introduzidas irão ser detalhadas ao longo do documento, no decorrer da investigação.

Design: O desenho da solução é uma das partes fulcrais da investigação, uma vez que terão de ser testadas e investigadas várias potenciais soluções para o desenvolvimento e hospedagem / "*deployment*" do *software.* Relativamente à linguagem de programação a utilizar, foi-nos solicitado pela organização a utilização do *Python*, uma vez que no SOC da organização, trabalham diariamente nessa linguagem. No sentido de uma implementação futura desta solução, é preferível que o projeto desenvolvido se encontre numa linguagem de fácil leitura para a organização. No que diz respeito à hospedagem da solução, o SOC da organização trabalha em rede, na *cloud* da *Amazon Web Services* (AWS), pelo que sugeriram que a nossa solução pudesse, de alguma maneira, ser acionada via AWS e de preferência hospedada também na AWS (todos estes termos serão explicados com detalhe ao longo

da investigação). Ao nível da plataforma de CTI a utilizar para partilha, arquivo e apresentação dos resultados, foi sugerido pela organização a utilização da plataforma "OpenCTI".

Investigação e Desenvolvimento: Etapa que se avizinha longa e trabalhosa, na medida em que requer uma investigação minuciosa a todas as ferramentas a utilizar e respetivas documentações, bem como inúmeros testes até encontrar a melhor solução de interconexão entre todas.

Testes: Desde perceber se o *output* de determinado processo é o esperado, passando por tentativas de conexão a APIs e trocas de mensagens entre aplicações, até aos testes finais dos algoritmos e posteriormente da solução final. Todos estes testes e tantos outros serão realizados ao longo da investigação até à obtenção do motor de inferência.

Deployment: No que concerne à ativação, disponibilização e hospedagem do *software* os esforços serão no sentido de ser via AWS, no entanto, inicialmente, numa primeira fase, este processo irá ser realizado localmente, para efeitos de investigação e testes, utilizando o *Docker* e o *Visual Studio Code* (VS Code). Ao longo da investigação iremos demonstrar a evolução do processo de desenvolvimento e "*deployment*" e respetiva migração da versão "local" para a "*cloud".*

Revisão: No fim de cada ciclo, após efetuar as etapas anteriores, é realizada uma revisão ao trabalho efetuado e ao *output* gerado, com base nessa informação, inicia-se um processo de *brainstorming* relativamente aos próximos passos a dar e de que forma se pode melhorar, para que se possam identificar novos requisitos e próximas etapas a seguir.

CAPÍTULO 4

Arquitetura Proposta

4.1 Introdução à plataforma

Este projeto pretende desenvolver um motor de inferência de classificação automática de IoCs / potenciais ameaças quantitativamente e qualitativamente. Neste sentido, será desenvolvido um *software* que se relacionará com uma plataforma de *Cyber Threat Intelligence* principal (OpenCTI), no sentido de utilizar essa plataforma para criar ou consultar IoCs observados na Organização, bem como recorrer a outras plataformas de CTI para enriquecer a informação sobre determinado IoC. Assim, através do motor desenvolvido neste projeto, um analista de um SOC, poderá apenas introduzir (*input*) um IoC que considere suspeito (ex: IPv4 addr) e obterá (*output*) uma classificação quantitativa (*score* e *trust rate*) e qualitativa (impacto) do nível de ameaça / risco para a sua organização. Ao longo dos próximos capítulos, será explicado com mais detalhe a solução desenvolvida.

4.1.1 Análise de requisitos

Para o desenvolvimento deste projeto, os requisitos fundamentais a possuir e a configurar são, conta AWS e conta OpenCTI. Por outro lado, opcionalmente, no entanto aconselhado, configurar um ambiente em *Docker* em conjunto com o *Portainer*[15] tornou-se vantajoso (não comprometendo o ambiente de desenvolvimento / trabalho da organização[16]) pelas capacidades de realização de testes e visualização de *output,* no início da investigação. Todas estas tecnologias iremos abordar com mais detalhe no capítulo seguinte, nas tecnologias utilizadas.

4.1.2 System Input: Evento

Este projeto foi configurado para receber como *input* um endereço IP[17] (*Internet Protocol*), do tipo IPv4[18], no entanto, é possível à posteriori adaptar o *software,* de forma a receber outros tipos de *input* (ex: IPv6, *Domain*, *URL*, etc).

[15] A plataforma de gestão de *containers* mais popular do mundo. Página oficial: https://www.portainer.io
[16] Outsystems
[17] IP: *Internet Protocol*, Protocolo da camada de rede sem ligação que implementa os serviços de endereçamento, fragmentação e agrupamento de pacotes, encaminhamento, filtragem, etc. [48]
[18] Versão 4 do protocolo IP, considera endereços lógicos com 32 *bits* divididos em quatro grupos de 8 *bits*.

O protocolo IP é um conjunto de regras que define como um computador envia dados para outro computador [65]. Um endereço IP, versão 4, é a quarta versão do *Internet Protocol.* Este protocolo é amplamente desenvolvido como o primeiro protocolo de *Internet*. O comprimento do endereço é de 32 *bits*, o que permite 4.294.967.296 endereços únicos. Cada endereço é escrito em notação decimal pontuada, onde cada valor decimal do *byte* de quatro endereços é separado por "pontos". O intervalo de cada *byte* é de 0 a 255, ou seja, um endereço IP pode variar entre 0.0.0.0 e 255.255.255.255 [48, 65].

4.1.3 System Output: JSON

O *output* final, ou seja, o reporte final ao analista, de um determinado IoC, é em formato *JSON* (*JavaScript Object Notation* - Notação de Objetos JavaScript, é uma formatação leve de troca de dados. Para seres humanos, é fácil de ler e escrever, para máquinas, é fácil de interpretar e gerar) [19].

Em JSON, os dados são apresentados com "**{**"(*chave de abertura)* e termina com "**}**" *(chave de fechamento)*. Cada nome é seguido por "**:**" *(dois pontos)* e os pares nome/valor são seguidos por "**,**" *(vírgula).* Na Figura 4.1 podemos observar o *output* (via AWS), de uma solicitação de um determinado IPv4 (atribuído aleatoriamente, apenas para efeito de demonstração).

O *output* / relatório final pode ser visualizado em diferentes plataformas, dependendo de qual o analista usou para realizar o pedido, ou seja, se o analista solicitar via AWS irá poder analisar o *output* na *Cloudwatch* da AWS, se por outro lado, solicitar via *Slack,* o *output* aparecerá diretamente no *Slack*. Assim como se utilizar a linha de comandos (ex: *Windows Porwershell*) também irá aparecer diretamente o resultado final. No capítulo 5 iremos abordar com mais detalhe as tecnologias utilizadas e o ponto de vista (POV) do analista.

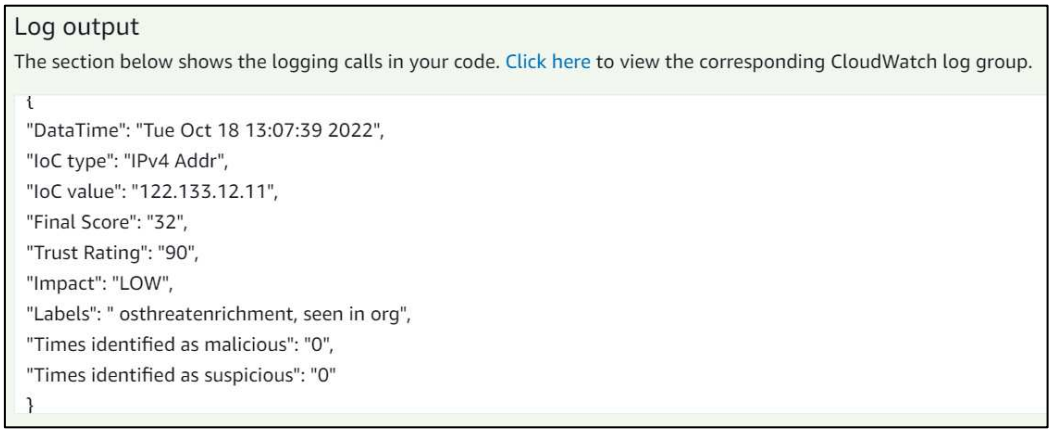

Figura 4.1 *– Output em formato JSON na plataforma AWS. Fonte: Snapshot do Autor.*

[19] Página Oficial: https://www.json.org/json-pt.html

4.1.4 Classificação

As classificações quantitativas e qualitativas são partes fulcrais do projeto. Em termos quantitativos, apresentamos um *score* e um *trust Rating* associado a esse *score*.

O *score* será determinado com base na *Threat Intelligence* recolhida pelos conectores de enriquecimento da OpenCTI e o *trust rating* é determinado com base no número de plataformas em que há informação acerca do IoC em questão, ou seja, o *score* determina o risco do IoC para a Organização e o *trust rating* fundamenta / suporta o valor do *score* (mais detalhes no subcapítulo 4.2.5). Porém, na medida em que é importante ter em consideração a frequência com que cada IoC surge / persiste na Organização, como por exemplo, um IoC que está a ser constantemente detetado pelo sistema de monitorização e/ou deteção de ISIs na Organização, o nosso motor de inferência irá ter esse fator em consideração e incrementar o *score* atual do IoC sempre que um IoC é persistente na Organização. Este processo de incrementação é realizado com base na sequência de *Fibonacci* (abordado no subcapítulo seguinte e aprofundado ao longo do capítulo 5).

4.1.5 Score incremental: *Fibonacci*

O *score*, como já referido, é sobretudo fundamentado através dos conectores de enriquecimento da plataforma OpenCTI, no entanto, sempre que um IoC é reincidente / persistente na organização é incrementado *à priori* o valor do *score*, no sentido de alertar o analista para o facto de existir uma potencial ameaça em crescimento dentro da organização. Para tal, na solução proposta, este processo incremental, foi baseado na sequência matemática de *Fibonacci*. Diversas investigações, na literatura, nomeadamente [90, 94, 95, 96], utilizam a sequência de *Fibonacci* ou adaptam uma sequência matemática com base na de *Fibonacci* para a incrementação de valores. É uma sequência onde, por norma, é possível realizar adaptações para tornar a computação eficiente [90], ou simplesmente, utilizá-la definindo um início e um fim.

A sequência de *Fibonacci* é definida como um conjunto de intervalos distribuídos aperiodicamente de acordo com o seguinte procedimento recursivo (equação 4.1), começando com dois elementos, F (0) = 0 e F (1) = 1, os números de *Fibonacci* { $F(x)$ } são obtidos pela regra iterativa [90]:

$$F(x) = \begin{cases} 0, & x = 0 \\ 1, & x = 1 \\ F(x-1) + F(x-2), & x > 1 \end{cases} \qquad (4.1)$$

$$F(x) = \{0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...\} \quad (4.2)$$

Neste projeto, como pretendemos que o *score* se encontre num intervalo [0, 100], quando superior a 89, o próximo valor será 100, uma vez que a sequência de *Fibonacci* é infinita (equação 4.2). Neste sentido, optámos por definir um valor final como objetivo, o 100. Outra particularidade da sequência é o valor 1, que se repete. Assim, quando se trata de um incidente reincidente na organização, o motor de inferência verifica primeiro, quantas vezes é que esse IoC já foi referenciado pela organização, especificado no Algoritmo 4.1 como "count". Quando o valor do contador é igual a 1 e o valor do score também é 1, o score de *Fibonacci* atribuído é igualmente 1, posteriormente é sempre um valor incrementado, de acordo com a seguinte sequência: list_fibo = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89], utilizada na nossa função de incrementação (Algoritmo 4.1).

ALGORITMO 4.1: FUNÇÃO FIBONACCI (PYTHON)

```
1  def fibonacci (score, count):
2      if score >= 89:
3          new_score = 100
4          return new_score
5      if score ==1 and count == 1:
6          new_score = 1
7          return new_score
8      else:
9          for i in range(len(list_fibo[i])):
10             if list_fibo[i] > score:
11                 new_score = list_fibo[i]
12                 return new_score
```

Este algoritmo (Algoritmo 4.1) é apenas aplicado quando um IoC é reincidente na organização e serve como *score* base ("new_score") de referência para o conector de enriquecimento "*osthreatenritchment*", ou seja, terá um peso na fórmula de atribuição desse *score,* não representando o *score* final do IoC. No capítulo 5 será explicado com mais detalhe o que é este conector, qual o seu objetivo e qual a sua pertinência para esta investigação.

4.1.6 Tabela de classificação de ameaças

A tabela de classificação de ameaças tem como principal objetivo atribuir, a cada ameaça ou potencial ameaça identificada, uma qualificação qualitativa (*Impact*): *Low* (Baixa), *Medium* (Moderada), *High* (Elevada).

A Tabela 4.1 é consultada pelo algoritmo 4.2 (função que atribui o impacto para a Organização de determinado IoC), após obter a informação atualizada do *score* e *trust rating* do IoC submetido na plataforma OpenCTI. No final, quando o analista do SOC recebe o relatório / "*output*", é informado do impacto que esse IoC tem para a Organização, e com base nessa qualificação atribuída poderá ser despoletada uma resposta em conformidade, pelo analista.

Tabela 4.1 *- Tabela de classificação qualitativa do impacto de um IoC para uma organização. Fonte: Elaboração do Autor.*

Score	Trust rating	Impact
>= 70	> 40	High
>= 70	<= 40	Medium
[40, 69]	>= 40	Medium
[40, 69]	< 40	Low
< 40	[0, 100]	Low

A validação / formalização da atribuição deste "*trust rating"* (nível de confiança no *score* atribuído) está a ser desenvolvido numa dissertação de mestrado em paralelo a esta investigação. De forma simples, o valor do *trust rating* é mais elevado consoante o número de plataformas externas em que existe informação sobre esse IoC e ainda sobre o peso que cada plataforma tem, na ótica do Autor dessa investigação. A barreira ("*threshold*") que foi escolhida para o "*trust rating"* foi no valor 40, uma vez que para se obter esse nível de confiança significa que o IoC está em pelo menos duas plataformas externas ou na plataforma externa que apresentou maior grau de confiança. Relativamente aos "*thresholds*" dos valores do *score*, foram baseados na escala qualitativa de classificação de gravidade *CVSS*[20] *Base Score* [87], um grupo de métricas com uma pontuação de 0 a 10, em que o 4 separa o "*Low*" do "*Medium*" e o 7 o "*Medium*" do "*High*", assim, uma vez o nosso score é de 0 a 100, foi utilizado o 40 para separar o baixo do médio e o 70 para separar o médio do alto.

[20] CVSS: *Common Vulnerability Scoring System*, é uma estrutura de suporte e comunicação de gestão do risco de vulnerabilidades de um software [87]. NIST criou o CVSS em colaboração com *first.org* como um avanço nas suas responsabilidades sob a Lei Federal de Gestão de Informações, em 2002, com o objetivo de ser um padrão para entidades do setor público e privado, no âmbito da padronização e pontuação de vulnerabilidades, permitindo que as organizações priorizem o risco [88].

ALGORITMO 4.2: FUNÇÃO ATRIBUIÇÃO DE IMPACTO (PYTHON)

```
1  def getImpact (score, trust_rate):
2      if trust_rate == “Not atributed yet”:
3          trust_rate = 0
4      if score >= 70 and trust_rate > 40:
5          impact = “HIGH”
6      elif score >=70 and trust rate <= 40 or (score >= 40 and score <= 69 and trust_rate >= 40):
7          impact = “MEDIUM”
8      else:
9          impact = “LOW”
10     return impact
```

4.2 System Overview

O motor de inferência proposto integra sobretudo quatro grandes componentes, recolha de CTI automática; filtragem em plataformas de antivírus (AV); cálculo do risco de IoCs com um grau de confiança associado e respetivo impacto para a organização; e um reporte final.

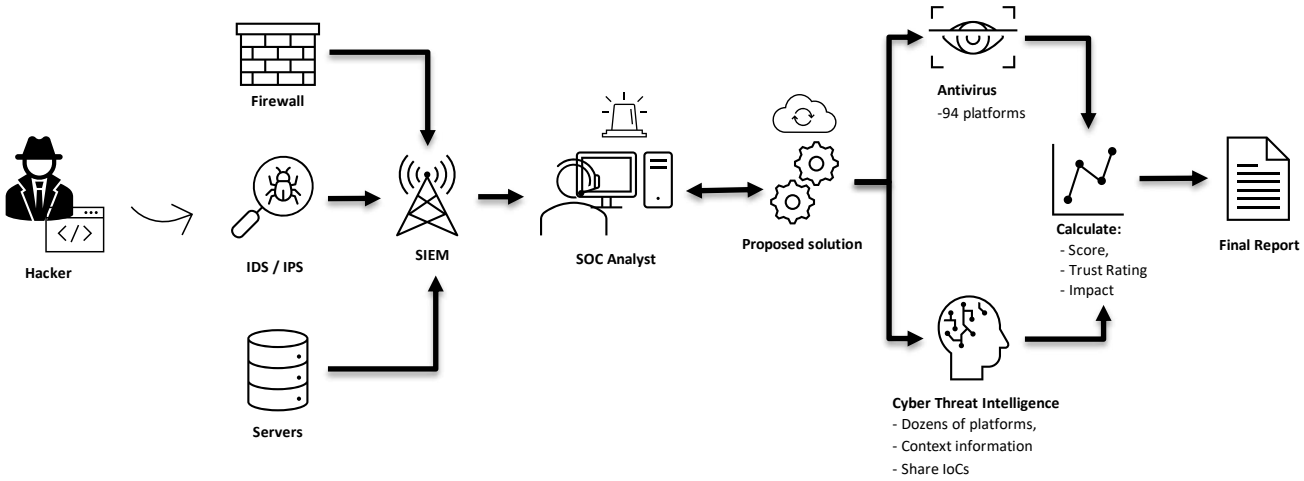

Figura 4.2 *- System Overview. Fonte: Elaboração do Autor.*

Como observável na Figura 4.2, a solução desenvolvida poderá ser integrada com um SIEM de uma organização e ser configurada para ser automatizada a partir da chegada de “*tickets*”, uma vez

que o motor de inferência é despoletado a partir de uma AWS *lambda function*, permitindo a invocação a partir de qualquer plataforma com capacidades para despoletar reações, como é o caso do SIEM. No entanto, existe sempre a possibilidade ativar o motor de inferência de forma manual, por um analista, invocando a função lambda, e será esta a forma de demonstração trabalhada neste projeto. Ao longo dos próximos capítulos, irá ser explicado com detalhe cada componente que integra a solução proposta e algumas demonstrações.

4.3 Arquitetura de processos

Um processo é um conjunto de tarefas que interagem para atingir um propósito comum através da transformação de *inputs* em *outputs*, e são realizadas por pessoas ou máquinas [60]. Para demonstrar a arquitetura de processos, optámos por desenhar duas demonstrações uma macro, para uma visão mais ampla da solução, e uma micro, para uma demonstração mais fechada nos processos intrínsecos aos componentes principais do motor de Inferência.

4.3.1 Macro: Modelo BPMN

Business Process Management (BPM) é uma abordagem sistemática para alcançar a melhoria contínua dos processos de negócios, integrando tecnologia da informação e metodologias de processo e governança. O ciclo de vida do BPM inclui fases para modelagem e análise de processos de negócios, automação de processos e monitorização de processos [60, 61]. A automação de processos de negócios é frequentemente abordada com soluções de BPM, para melhor interpretação e análise, com o intuito de conhecer a arquitetura de processos de um negócio e consequentemente melhorá-la [61]. O BPMN (*Business Process Model and Notation*) tem sido utilizado tanto para modelar processos de negócio e para especificar a integração de sistemas sob a Arquitetura SOA (*Service Oriented Architecture*), como também para modelar o fluxo de atividades presentes nos sistemas de informação [62]. O principal objetivo do BPMN[21] é fornecer uma notação que seja facilmente compreensível por todos os utilizadores, desde os analistas de negócio que criam os rascunhos iniciais dos processos, até os desenvolvedores técnicos responsáveis pela implementação da tecnologia que executará esses processos e, finalmente, ao utilizador final que irá gerir e monitorizar esses processos.

[21] Página oficial: http://www.omg.org/spec/BPMN/2.0

Modelo BPMN

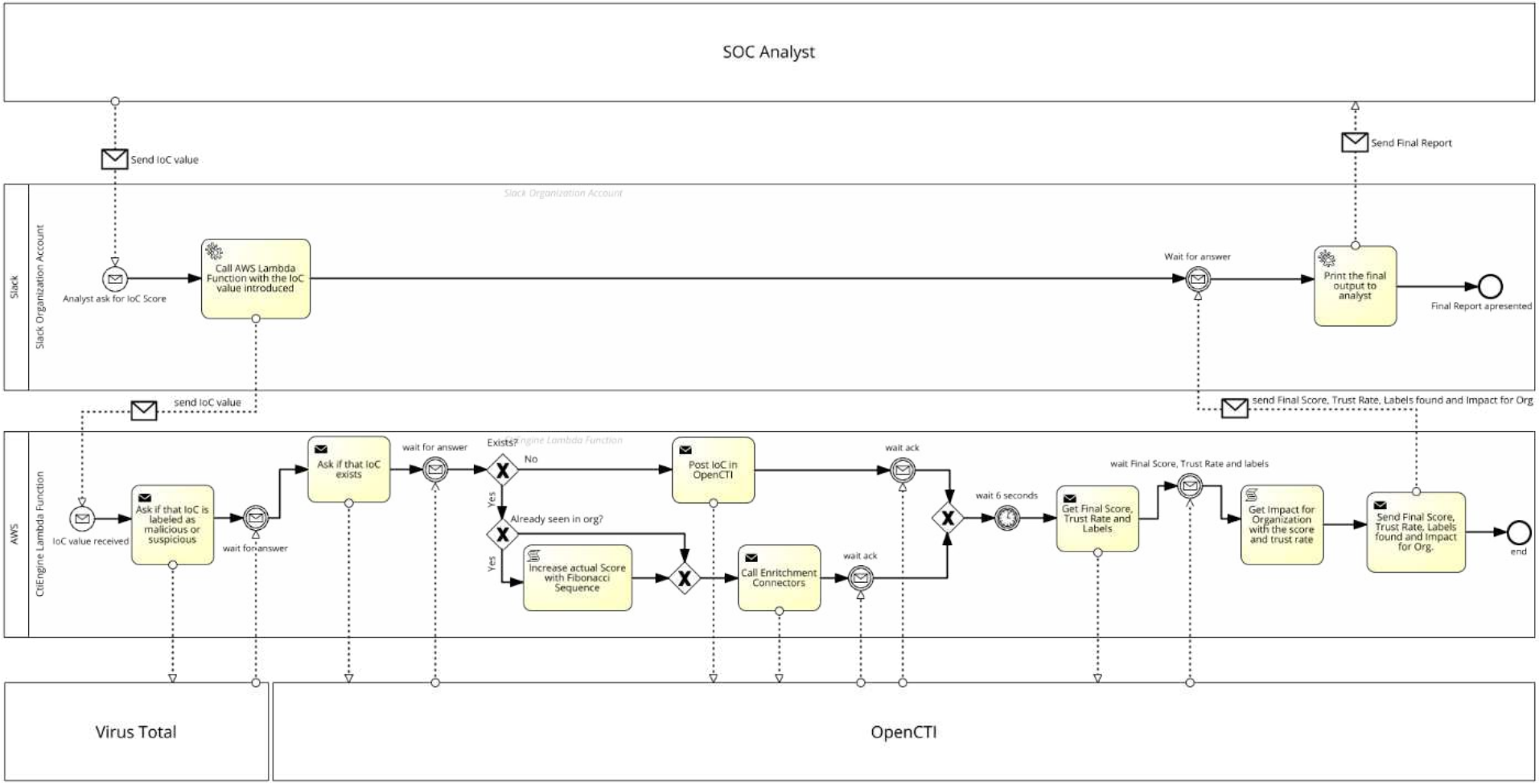

Figura 4.3 *- Modelo BPMN. Fonte: Elaboração do Autor*

O modelo observado na Figura 4.3 foi desenhado com vista à demonstração da utilização do motor de inferência via *Slack*, num SOC de uma organização. O processo inicia-se com a introdução de um IoC no *Slack* (com recurso ao *Chatbot* configurado), que por sua vez despoleta a AWS *lambda function*, onde os processos principais do motor de inferência ocorrem (explicados com mais detalhe no subcapítulo seguinte, na arquitetura micro). Porém, nesta arquitetura macro é possível observar que antes de solicitar um *score* final e o *trust rating* associado, é necessário aguardar seis segundos, para garantir que o conector "*osthreatenritchement*" termina os seus processos, de forma a ser possível obter um *score* atualizado e um grau de confiança correspondente. Com essa informação é obtido um impacto desse IoC para a organização. No fim, a informação é toda agregada num relatório e devolvido ao analista, pela mesma via (*Slack*). No entanto todos os *logs* e relatórios ficam guardados na AWS *Cloudwatch* para consultar sempre que necessário.

4.3.2 Micro: Modelo UML

Para demonstrar a arquitetura da solução proposta com todas as mensagens trocadas pelo motor de inferência, utilizámos um diagrama de sequência UML. O objetivo de uma arquitetura UML é demonstrar as interações entre objetos na ordem sequencial em que essas interações ocorrem [83].

A troca de mensagens, será realizada através do protocolo HTTP[22], utilizando os métodos "*GET Request*" e "*POST request*". Na arquitetura proposta, o analista de um SOC irá receber informação por parte de um sistema de monitorização que disponha no seu SOC (ex: SIEM), e irá enviar esse IoC (ex: via *Slack* ou diretamente na AWS) e partir desse momento inicia-se o motor de inferência desenvolvido.

O processo proposto, na Figura 4.4, inicia-se com a realização de um HTTP *GET request* à plataforma externa, *Vírus Total* (VT), de forma a recolher informação de se esse IoC já alguma vez foi identificado como "malicioso" ou "suspeito", por algum dos 94 antivírus a que a VT recorre. Posteriormente, existe a primeira conexão com a OpenCTI, através de uma solicitação (*GET Request*), de forma a verificar se aquele IoC já é conhecido ou não na plataforma ("Verify if Exists Ioc"), cujas respostas poderão ser:

1. Sim, esse IoC já foi visto na sua organização (tem o *label* "seen in org"), o *Score* é "x", *Trust Rate* é "y" e as *labels* encontradas são: {"xpto", "seen in org", etc};
2. Sim, no entanto, esse IoC nunca foi visto na Organização, o *Score* é "x" e as *labels* encontradas são: {"xpto", "yzv", etc};

[22] HTTP: *Hypertext Transfer Protocol*, é um protocolo de aplicação para sistemas de informação entre cliente e servidor, permitindo aos utilizadores comunicar na *World Wide Web* (WEB).

3. Não, esse IoC nunca foi visto / inserido nesta plataforma;

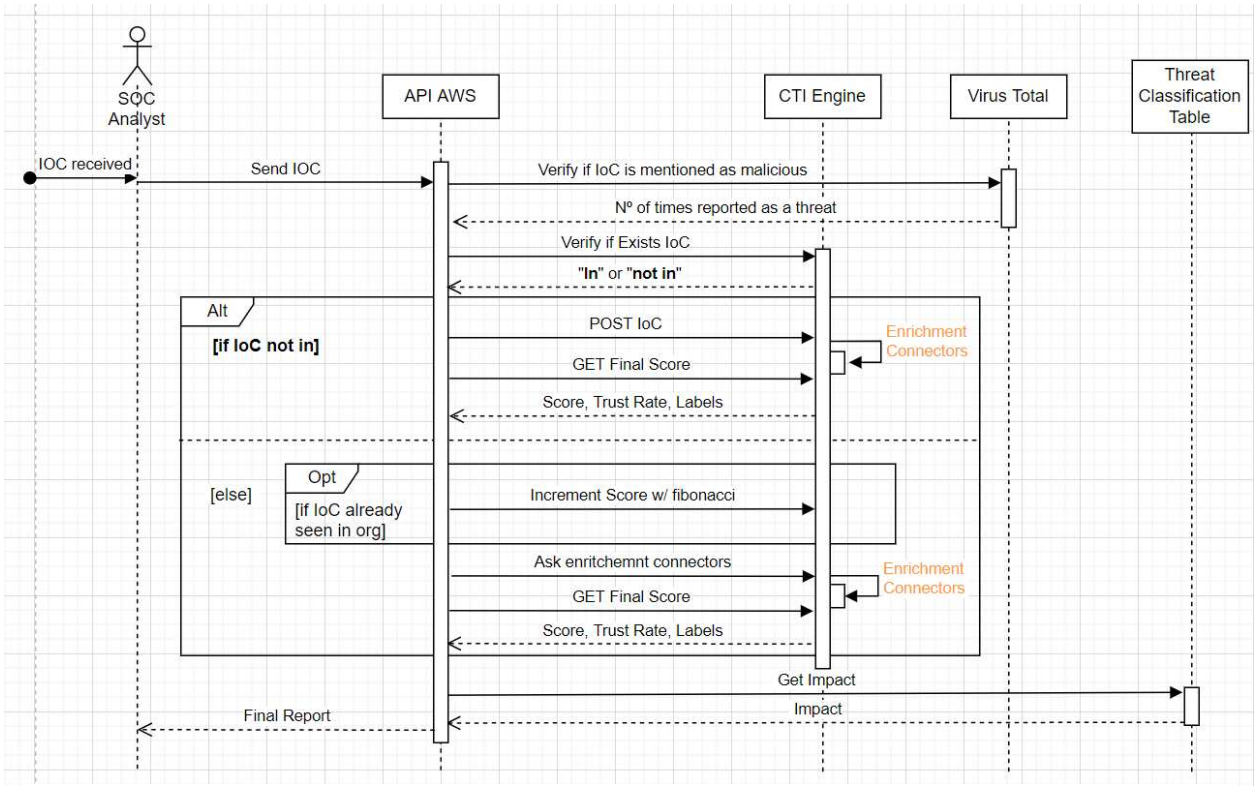

Figura 4.4 *- Modelo UML. Fonte: Elaboração do Autor*

Assim, como existem 3 possíveis respostas e as mensagens seguintes dependem dessas respostas, no diagrama colocámos uma "*box*" "Alt" (de alternativamente), ou seja:

Se a resposta recebida for a nº 1, o *request* seguinte será incrementar o *score* recebido, seguindo a sequência de *Fibonacci* ("box Opt", de "*option*"), uma vez que se trata de um IoC reincidente, e após o incremento do *score*, executa dois *requests*, um primeiro para ativar / forçar os conectores de enriquecimento a executarem uma nova investigação sobre aquele IoC (sem este *request* os conectores de enriquecimento mantêm-se inativos) e seguidamente um outro "GET request" com o novo *score*, o *trust rate* associado e as *labels* encontradas.

Se a resposta recebida for a nº 2, adicionamos a *label* "seen in org", uma vez que agora já foi visto na Organização, ao adicionar esta *label* o conector de enriquecimento ("osthreatenritchment") está programado para automaticamente iniciar o seu processo de investigação sobre aquele IoC (apenas a primeira vez que um IoC é visto na organização é que este processo é automático), e assim executamos o "GET request" para obter o novo *score*, *trust rate* associado e as *labels* encontradas.

Se a resposta recebida for a nº 3, ou seja, esse IoC nunca foi introduzido na OpenCTI, executamos de imediato um "*POST request*" com os dados desse IoC, o tipo, o valor, a *label* "seen in org", *score* a zero e o autor (do POST, nós). Como é a primeira vez que esse IoC é introduzido na plataforma, os conectores de enriquecimento correm automaticamente e posto isso executamos o "*GET request*" para obter o *score* final, *trust rate* associado e as *labels* encontradas.

Posto estas alternativas, independentemente de qual fluir a comunicação, com o *score* final obtido e o *trust rate* associado, é automaticamente consultada uma tabela de classificação de ameaças (conhecida pelo nosso motor de inferência) que irá devolver ao analista o impacto daquele IoC para a organização.

Terminados os processos / troca de mensagens, no final, o analista receberá um "reporte final", no qual estará contido, o dia e a hora a que foi realizado o pedido; o tipo e o valor do IoC introduzido; o *score* final obtido e o grau de confiança correspondente; o impacto para a organização; as *labels* de contexto encontradas; e quantas vezes esse IoC já foi identificado como malicioso e suspeito (entre 0 e 94). No final, com a classificação obtida, o analista poderá desencadear / despoletar os mecanismos pertinentes e em conformidade para a resposta a esse incidente de segurança.

CAPÍTULO 5

Desenvolvimento da Solução

5.1 Tecnologias utilizadas

Nesta secção iremos abordar as principais tecnologias utilizadas no desenvolvimento da solução, nas diversas fases da investigação.

5.1.1 OpenCTI

Uma plataforma de CTI deve ser capaz de consultar, correlacionar e interagir com outros plataformas de CTI e outras ferramentas relevantes, incluindo arquitetura(s) analítica(s) do SOC (ex: SIEM, SOAR, IDS, etc). Isto significa que a ferramenta de CTI deve suportar tanto os padrões CTI abertos (STIX[23]/TAXII[24]) quanto as APIs das ferramentas que o SOC utiliza, como o seu SIEM ou SOAR [49]. Neste sentido, indo de encontro ás necessidades e interesses da organização, foi escolhida para plataforma principal de trabalho, a OpenCTI. Esta, é uma plataforma de código aberto que permite que as organizações façam a gestão da sua *cyber threat intelligence* (CTI), dos seus IoCs / potencias ameaças observadas (denominados na plataforma como "*observables*") e configurem / programem novos conectores de enriquecimento, assim como possibilita a configuração de conectores de enriquecimento ou de *input* na plataforma já desenvolvidos. A OpenCTI fornece ainda uma poderosa base de dados de gestão de conhecimento, com um esquema especialmente adaptado para CTI e operações de cibersegurança, bem como diversas ferramentas e recursos de visualização.

Conector "osthreatenritchement": Paralelamente a esta investigação, encontra-se em desenvolvimento outra investigação que utiliza também a plataforma OpenCTI e desenvolve um conector denominado "osthreatenritchement", com a finalidade de apurar um "*threat* score" e atribuir um valor de confiança nesse "*score*" ("*trust rating*"). Este conector será utilizado nesta solução desenvolvida sempre que é introduzido um IoC no nosso motor de inferência.

[23] STIX: *OASIS Structured Threat Information eXpression*. Uma linguagem para modelação e representação de *cyber threat intelligence.* Fonte: https://docs.oasis-open.org/cti/stix/v2.1/csprd01/stix-v2.1-csprd01.html
[24] TAXII: *OASIS Trusted Automated eXchange of Indicator Information*. Um protocolo para partilha e análise de *cyber threat intelligence.* Fonte: http://docs.oasis-open.org/cti/taxii/v2.0/cs01/taxii-v2.0-cs01.html

5.1.2 Virus Total

A plataforma *Virus Total* (VT) é um serviço online que analisa arquivos, domínios, IP *Adresses* e URLs suspeitos, com o objetivo de averiguar se o IoC introduzido já foi identificado como malicioso ou suspeito e respetivo tipo de *malware* / conteúdo malicioso associado, entre outras informações acerca do IoC[25].

A VT utiliza / recorre a 94 plataformas antivírus (AV) para obter a informação de cada um, ou seja, para cada pesquisa efetuada devolve o número de vezes em que esse IoC foi encontrado, um valor que poderá variar entre 0 (nunca identificado em nenhum AV) até 94 (identificado em todos os AV).

A VT fornece uma API que nos permitiu uma conexão direta e uma eficiente e rápida acessibilidade para a troca de mensagens (HTTP). Esta troca de mensagens tem o único objetivo de recolher a informação se o IoC já foi identificado como suspeito ou ciberameaça, de forma a auxiliar e facilitar o processo de decisão do analista na escolha para responder a um IoC, de baixo impacto, com o mesmo *score*, ou seja, no caso de o analista estar indeciso por qual dos dois IoCs começar, escolher com base no número de vezes em que esse IoC já foi identificado como suspeito ou ameaça. Importante, assim, enaltecer que a consulta na VT não pretende substituir o *score* produzido pelo conector de enriquecimento "osthreatenritchement". A informação obtida pela VT, será pertinente, apenas para se o IoC apresentar um impacto baixo ("LOW") e o analista estiver em dúvida a qual (Ioc) responder primeiro, podendo assim optar pelo que já alguma vez (ou mais vezes) tiver sido identificado como suspeito ou referenciado como ciberameaça.

5.1.3 GraphQL

GraphQL (*Graph Query Language*) pode ser segmentada em duas partes para melhor entendimento, "*Graph*" de gráfico, uma vez que permite a construção de "*schemas*", ou seja, dicionários com dados definidos e modelados para poderem ser consultados dessa forma; e "QL" de "*Query Language*" que representa o formato como os dados definidos podem ser consumidos, por meio de "*queries*", possuindo a particularidade de na realização das "*queries*" solicitar apenas os dados que necessitam, aprimorando substancialmente a performance em grandes quantidades de dados e tipos de informação.

GraphQL é uma linguagem de consulta, cuja prioridade é fornecer exatamente os dados que os clientes solicitam e nada mais[26]. Porém, de forma a ser utilizada esta linguagem, necessita de existir

[25] Página oficial: https://www.virustotal.com
[26] Página oficial: https://graphql.org

tecnologias que a implementem. A *OpenCTI* possui essas tecnologias e disponibiliza a sua utilização, acrescentado no *URL* da sua API "/graphql"[27]. Neste sentido, utilizou-se a linguagem GraphQL para comunicar de forma ágil e eficiente com a plataforma OpenCTI.

5.1.4 Python

Python é uma linguagem orientada a objetos de alto nível ("*Very High Level Language*") [28], isto é, uma linguagem mais perto da linguagem humana do que da linguagem de máquina.

Python é muito utilizado como linguagem *script* em *softwares*, permitindo automatizar tarefas [56] e servir de conexão entre diferentes plataformas. Na solução apresentada, foi desenvolvido um *script Python* para realizar a conexão entre a AWS e a OpenCTI.

As fortes capacidades ao nível da automação de processos e a possibilidade de integrar o *Python* com outras linguagens, como a GraphQL, tornou a escolha desta linguagem para o desenvolvimento do projeto, uma decisão simples.

5.1.5 Docker

O *Docker* é um projeto de *software* de livre acesso, que visa automatizar o "*deployment*" de aplicativos, como *containers*[29] autossuficientes e portáteis que podem ser executados na *cloud* ou localmente. O *Docker* elimina as tarefas de configuração repetitivas e mundanas e é usado em todo o ciclo de vida do desenvolvimento de software (SDLC) [30], permitindo trabalhar em rede através de um determinado *container*.

Neste projeto foi utilizado o *Docker*, numa fase inicial, de investigação e desenvolvimento e respetivos testes de produção, para clonar o ambiente de comunicação com a plataforma OpenCTI da Organização. Assim, desta forma, foi possível executar vários testes sem comprometer a plataforma que a Organização utiliza.

5.1.6 AWS Cloud

A *Amazon Web Services* (AWS) é a plataforma *cloud* mais adotada e mais abrangente do mundo, oferecendo mais de 200 serviços completos de *data centers* em todo o mundo. Milhões de clientes,

[27] Documentação oficial OpenCTI: https://www.notion.so/GraphQL-API-cfe267386c66492eb73924ef059d6d59
[28] Página oficial: http://www.python.org/
[29] *Container* é uma unidade padrão de *software* que empacota o código e todas as suas dependências, de forma a ser executado de forma rápida e confiável. Fonte: https://www.docker.com/resources/what-container/
[30] Página oficial: https://www.docker.com

grandes empresas e os maiores órgãos governamentais, utilizam a AWS para reduzirem custos, ficarem mais ágeis e inovarem mais rapidamente[31].

A AWS foi projetada para ser um dos ambientes de computação *cloud* mais flexíveis e seguros atualmente disponíveis. Possui uma infraestrutura central desenvolvida para satisfazer os requisitos de segurança militares, de bancos globais e de outras organizações que lidam com informações estritamente confidenciais[32].

As principais ferramentas AWS utilizadas ao longo do projeto foram:

- ***Identity and Access Management (IAM),*** onde é necessário configurar as regras de utilizador, papel do utilizador (*role*), políticas de utilização e as respetivas permissões.
- ***System Manager – Session Manager****,* é um *hub* de operações para as aplicações e recursos guardados na AWS, nomeadamente por meio de uma consola partilhada entre os utilizadores com permissão, dentro da Organização. Através desta consola é possível observar e configurar os *containers* utilizados para incorporar os conectores (da OpenCTI) da Organização.
- ***AWS Lambda****,* é um serviço de computação sem servidor e orientado a eventos que permite executar código, ou seja, tratou-se da conexão desenvolvida neste projeto entre o utilizador (ex: analista de um SOC) e a plataforma OpenCTI. Foram desenvolvidas várias funções lambdas ao longo da investigação até ao resultado final obtido. A função lambda, neste projeto, serve para fazer executar o *Script Python* desenvolvido.
- ***Cloudwatch****,* coleta dados de monitorização e operações na forma de *logs*, métricas e eventos, e permite uma visualização através de painéis automatizados para uma visão personalizada (*dashboard*). Esta ferramenta tornou-se muito útil ao longo do desenvolvimento de projeto, nomeadamente, como interpretação do *output* e *debug*.
- ***Chatbot,*** é um sistema interativo que facilita a monitorização, a operação e a solução de problemas ao nível da gestão de tarefas. Neste projeto configurámos um *AWS Chatbot* para iniciar um fluxo de trabalho, via *Slack* (Figura 5.1).

5.1.7 Slack

O *Slack* é uma aplicação de mensagens para empresas, conectando as pessoas às informações de que elas precisam. Reúne pessoas e estimula uma produção em equipa, fomentando a partilha de informação e a coesão entre equipas.[33]

[31] Página oficial: https://aws.amazon.com/pt/
[32] Página oficial: https://aws.amazon.com/pt/
[33] Página oficial: https://slack.com/

Neste projeto foi utilizado o *Slack* para, em conjunto com a AWS, criar um *Chatbot* para agilizar a execução da função lambda AWS desenvolvida.

Figura 5.1 *- AWS chatbot via Slack. Fonte: https://aws.amazon.com/pt/chatbot/*

5.2 Investigação e Implementação

O processo de investigação teve início após a análise dos requisitos da organização e dos requisitos inerentes à solução proposta, nomeadamente, a utilização da plataforma OpenCTI e a capacidade de despoletar a solução via AWS. Por outro lado, é ainda pertinente a Organização possuir um SIEM, um IDS, ou outro sistema que desempenhe funções semelhantes, de forma a identificar IoC's suspeitos.

Primeiramente, foi-nos atribuído um *user* e um *role* na conta AWS da organização, de forma a termos acesso ao *container* onde corre a OpenCTI. Contudo, após todas as configurações na AWS e garantia de todas as permissões para conseguir manipular o conteúdo do *container* e criar novos *containers* para interagir com a *OpenCTI*, a capacidade de testar código com *feedback* para *debug* era muito limitada e o progresso no desenvolvimento estava a ser lento. Neste sentido, optou-se por mudar de estratégia e demos início ao processo de clonagem do ambiente de trabalho dessa plataforma para a nossa máquina (localmente). Para isso, são necessários / uteis alguns pré-requisitos, como possuir o *Docker Desktop* e instalar o *Portainer* (é possível fazer também através da linha de comandos, no entanto para se acompanhar melhor e visualizar as alterações e configurações, optámos por utilizar o *Docker Desktop* e o *Portainer*). Posteriormente é necessário realizar todas as configurações para que a OpenCTI consiga correr localmente através do *Docker* e do *Portainer*. Após instalar o *Docker Desktop* é necessário criar o volume que o *Portainer Server* irá utilizar para guardar os dados (Figura 5.2), e de seguida fazer *download* e instalar o *container* do *Portainer Server* (Figura 5.3).

```
docker volume create portainer_data
```

Figura 5.2 *- Criar o volume Portainer para guardar os dados. Fonte: https://docs.portainer.io/v/ce-2.9/start/install/server/docker/linux*

```
docker run -d -p 8000:8000 -p 9443:9443 --name portainer \
    --restart=always \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v portainer_data:/data \
    portainer/portainer-ce:2.9.3
```

Figura 5.3 *- Download e Instalação do servidor do Portainer. Fonte: https://docs.portainer.io/v/ce-2.9/start/install/server/docker/linux*

Depois de instalado e a correr o *Portainer* é necessário configurar um ficheiro do tipo "YML" com o *Docker-compose* para conectar à OpenCTI, e posteriormente configurar um ficheiro "ENV" com as variáveis de ambiente para completar a conexão à OpenCTI. Para fazer *login* no *Portainer Server* é necessário abrir o *web browser* e digitar: https://localhost:9443, colocar o *user* e *password* e temos acesso aos *containers* da OpenCTI que se encontram a correr localmente no *Docker*, como se pode observar na Figura 5.4. Através do *Docker Desktop* também é possível visualizar o *container* do *Portainer* e os da *OpenCTI* a correrem (Figura 5.5).

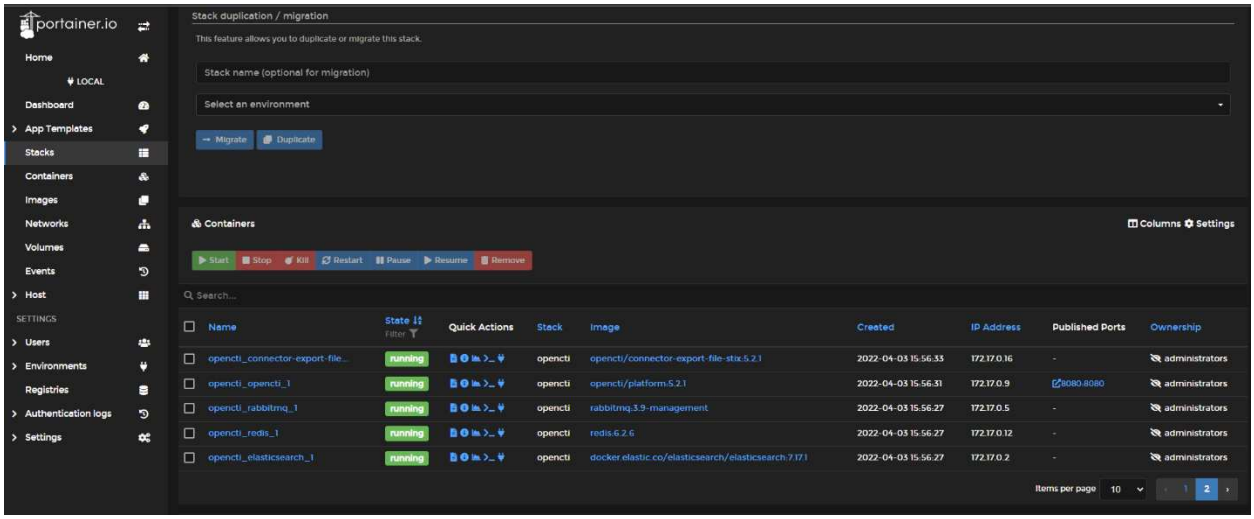

Figura 5.4 *– Containers da OpenCTI a correr no Portainer. Fonte: Snapshot do Autor.*

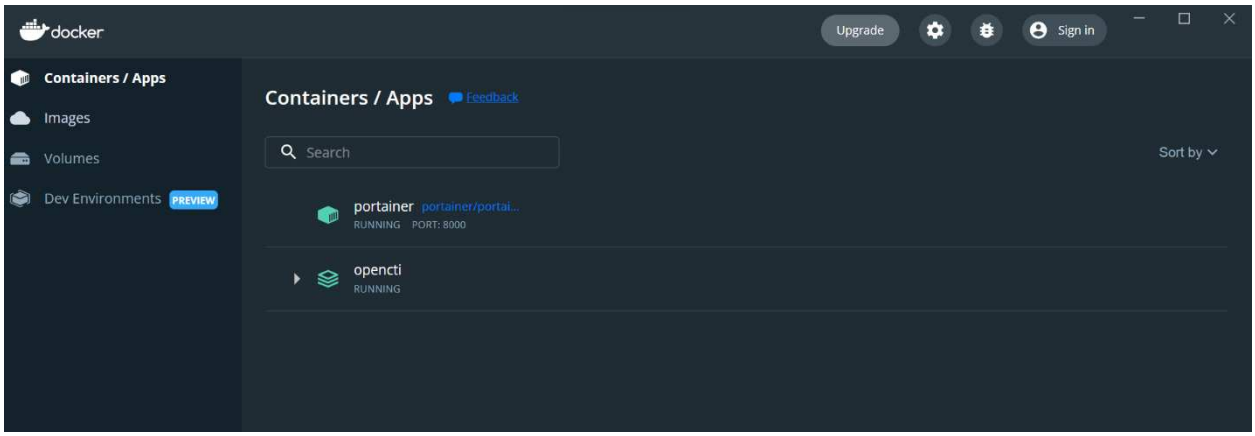

Figura 5.5 *- Docker Desktop: Portainer e OpenCTI a correrem com sucesso. Fonte: Snapshot do Autor*

Após as configurações para ambiente de desenvolvimento iniciou-se então o processo de exploração das bibliotecas *Python* que interagem com a *OpenCTI* de forma a conseguir produzir uma classe que fosse capaz de criar um novo *observable* na plataforma (localmente). O objetivo desta classe é que fosse capaz de, para além de criar o *observable* na plataforma, esse *observable* possuísse alguns atributos personalizados como um *label* identificativo de que este IoC foi visto na organização: "seen in org", para que seja facilmente distinguido dos outros *observables* que estão na plataforma e sobretudo para que saibamos quais são os da nossa organização.

Nesta fase, por uma questão de se tornar mais prática a investigação e desenvolvimento da mesma, iremos ter apenas em consideração os IoC's "IPv4-Addr". Assim, o *observable* criado por nós tem os seguintes atributos: (Observable type: "Ipv4-Addr", Value: "xxx.x.x.x", Label: "**seen in org**", Author: "My Org", Creator: "my user name"), observável nas Figuras 5.6 e 5.7.

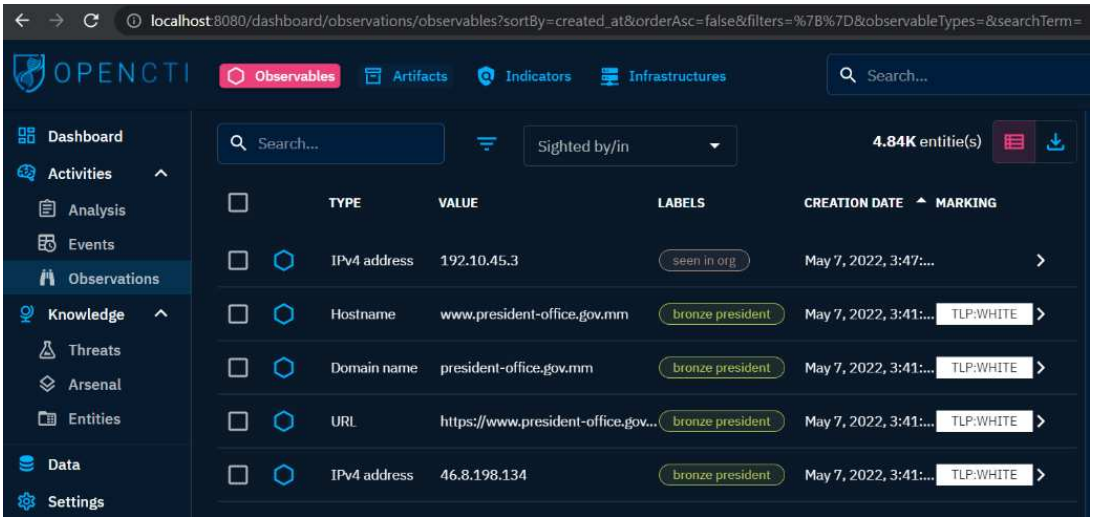

Figura 5.6 *- Observable introduzido na plataforma OpenCTI (localmente). Fonte: Snapshot do Autor*

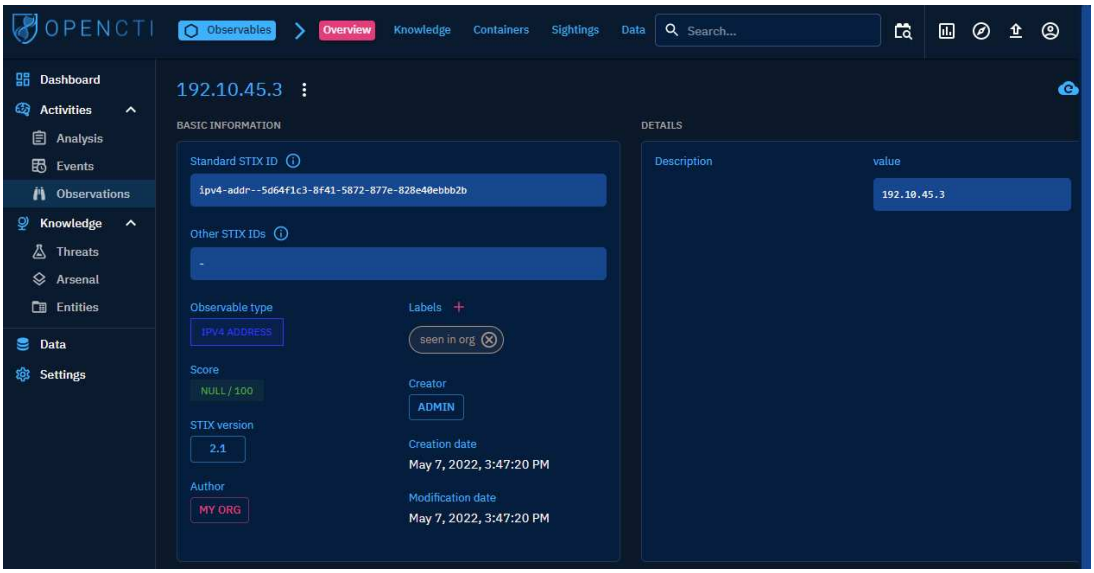

Figura 5.7 *- Características do observable introduzido na plataforma OpenCTI (localmente). Fonte: Snapshot do Autor*

Após termos um *script Python*, capaz de criar um *observable* localmente, iniciámos o processo de tentativa de migração para a *cloud AWS*. Começámos por conectar o *script python* criado à *API* da *OpenCTI* e invocar o *url* público da organização na plataforma, juntamente o com a *API Key* da *OpenCTI* (chave privada do utilizador).

Assim que conseguimos conectar o *script Python* ao *endpoint* da *OpenCTI* da organização, o objetivo seguinte foi colocar o *script* numa função AWS *Lambda,* que esteja conectada a uma *REST API* na *AWS* da organização, de forma a estar sempre "à escuta" e pronta a receber como atributo uma *String* com um IPv4. No entanto, para isso, é necessário alguns pré requisitos, como ter conta na AWS e um respetivo *user* com as permissões necessárias (mais uma vez é possível fazer através da linha de comandos, no entanto para uma melhor visualização e organização é preferível ter conta AWS, e é gratuito). Uma vez, criada e configurada a conta AWS, iniciámos a construção de uma função lambda, que passa por uma reestruturação do código, adaptando as novas necessidades, e no final, compactar todas as dependências e o *script Python* num ficheiro "ZIP" e realizar o *deployment* na AWS. Contudo, após diversas tentativas de *deployment* e muito *debug* não conseguimos superar o erro no módulo "_regex" (módulo em C++, rejeitado pela AWS, após configuração da AWS *Lambda Function* em *Python* 3.9), erro observável na Figura 5.8. Este módulo aparece nas dependências utilizadas pelo *script Python* desenvolvido para se conectar à OpenCTI, trata-se da biblioteca "pycti", sugerida na documentação da OpenCTI. No entanto apesar de funcionar quando executado via IDE e na *AWS cloud* da organização, por via de execução através de funções lambda, tal não foi possível, rejeitando sempre a utilização desta biblioteca.

Figura 5.8 *- Erro lançado na configuração da AWS lambda Function ao utilizar a biblioteca "pycti". Fonte: Snapshot do Autor*

Assim, para resolver / contornar esta questão, foi necessário iniciar nova investigação para encontrar métodos alternativos para correr um *script Python* na AWS *Lambda Function*. Ao explorar outros pontos de vista na documentação da OpenCTI percebemos que a OpenCTI utiliza a API GraphQL como método de comunicação, isto é, possibilita a realização de HTTP *requests (POST e GET)* à plataforma OpenCTI, através desta linguagem. Assim, iniciámos o processo de desenvolvimento de código em GraphQL, construindo funções, "*querys" e "mutations*", em GraphQL.

Começámos por criar um *observable* em GraphQL (localmente) e fomos explorando e percebendo como poderíamos fazer as comunicações todas através desta linguagem. Após uma continua e prolongada investigação conseguimos construir as primeiras "*queries*" e "*mutations*" necessárias, como a verificação se um determinado "*observable*" existe ou não na plataforma OpenCTI e caso exista, atualizar o *Score* desse "*observable*". Caso não exista, adicionar o "*observable*" à plataforma.

Adicionámos a "*query*" de verificar se um IoC existe ao *script Python* e estruturámos uma classe que recebe como input um IoC (ex: IPv4 addr) e verifica se esse IOC já existe na Base de Dados da Open CTI. Se existir, devolve o valor do IOC recebido, o "*Score*" e o "ID", se não existir devolve apenas o valor do IoC recebido. Desta forma conseguíamos saber se o IoC já existe na plataforma, logo é reincidente e o *score* é para incrementar segundo a sequência de *Fibonacci* apresentada anteriormente, ou se se trata de IoC novo e é necessário adicioná-lo à plataforma OpenCTI.

Posteriormente, construímos outra classe *Python*, que recebe como "*input*" o valor do IoC, "*score*" e "ID" ou apenas o valor do IoC. Se receber, valor do IoC, "*score*" e "ID", incrementa o "*score*" segundo a sequência de *Fibonacci*, ou seja, se recebe por exemplo 30 de "Score" deverá ser atualizado para o número da sequência *Fibonacci* que se segue (neste caso seria: 34). De seguida deverá invocar o conector de enriquecimento: "osthreatenritchement" (de forma a atualizar o "*score*" com base nas ferramentas externas). Por outro lado, esta classe deve também estar preparada para receber como *input* apenas um valor de IoC (ex: IPv4 addr), para casos em que o "*observable*" com esse IoC ainda não exista. Neste caso, deverá criar um "*observable*" na OpenCTI, com o valor de IoC recebido, uma "*label*" a informar que foi visto na organização ("seen in org") e o "*score*" igual a zero (ao ser criado

pela primeira vez ativa automaticamente os conectores de enriquecimento da OpenCTI). No entanto, quando um IoC / *observable* já existe na plataforma a simples atualização de score não despoleta a ativação do conector de enriquecimento, assim torna-se imperativo a sua chamada, isto é, uma nova "*mutation*" em GraphQL capaz de forçar a chamada de conectores de enriquecimento, de forma que o *score* possa ser atualizado. Caso contrário apenas uma vez, no momento da criação, o IoC receberia *updates* das plataformas externas de CTI.

Por fim, precisávamos agora de uma *query* final, também em GraphQL, que devolvesse os atributos de um determinado IoC, de forma que no fim de tudo seja possível obter o *score* final de um IoC. Construímos uma nova classe *Python*, que recebe como input um valor de um IoC e devolve o "*score*" e o "*trust rating*" final desse IoC. Assim, com o "*score*" e o "*trust rating*" obtidos e com uma tabela de classificação de impacto para a organização construída, automatizou-se o resultado do impacto de determinado IoC para a organização.

Chegando novamente à fase final, integração das classes *Python*, na AWS. O objetivo era construir duas funções *AWS Lambda*, uma para verificar se o IoC existe e em caso afirmativo incrementar o *score* e ativar o conector de enriquecimento, e em caso negativo adicionar o IoC à plataforma OpenCTI. A outra função lambda, seria executada posteriormente para obtermos o *score* final, o *trust rating* associado e o impacto para a organização. Desta vez, o erro anterior foi suprimido com sucesso. Surgiram assim as duas primeiras funções lambda, capazes de cumprir com os objetivos pretendidos. No entanto, havia algo que ainda podia ser melhorado, tanto do ponto de vista da eficiência para o analista como em termos de organização e estruturação do código e dos algoritmos associados.

Neste cenário, com estas duas funções lambda, o analista teria primeiro de introduzir um IoC suspeito, aguardar que o IoC fosse adicionado ou atualizado e passado alguns segundos, executar uma segunda função lambda para obter o reporte final.

Ao investigar sobre formas de otimizar este cenário, começámos a explorar a possibilidade de construir funções lambda em cadeia, isto é, funções lambda capazes de chamarem a função seguinte, podendo assim, tornar-se mais eficiente para o analista, uma vez que desta forma, o analista precisaria apenas de introduzir o IoC (ex: IPv4 Addr) e no fim receberia o *score* e *trust rating* final. Assim, o objetivo é que a primeira AWS Lambda invoque a seguinte. Neste sentido, foi necessário proceder a alguns ajustes nas classes *Python* anteriormente referidas, de forma a tal invocação ser possível, nomeadamente configurar no final da classe o "*invoke*" com o "ARN" (*Amazon Resource Name*) da função lambda seguinte e parâmetros a enviar. É importante acrescentar na última classe *Python* um tempo de espera de cerca de 6 segundos, de forma a permitir que a atualização do "score" seja efetuada com sucesso.

Nova revisão do *software*, vantagens de trabalhar utilizando o método ágil, o software funciona apenas com um *input* por parte do analista, no entanto, ainda há margem para otimização, novamente, tanto ao nível da estruturação do código e processos, como do ponto de vista do analista. Assim, com as funções já trabalhadas e a funcionar em cadeia, começámos a explorar a hipótese de pôr tudo numa única função lambda, com todos os "HTTP *requests"* na mesma função. Este processo implicava um melhor conhecimento de como é realizada a troca de mensagens entre cliente e servidor, neste caso, nós e a API da OpenCTI (via AWS), o qual tem vindo a ser adquirido e enriquecido ao longo da investigação. Ao fim de alguns testes, tornámos possível a colocação de todas as mensagens de comunicação dentro da mesma função lambda. Por outro lado, do ponto de vista do analista, existiam duas otimizações que tencionávamos otimizar, a primeira ao nível da ativação / execução da função lambda construída. Nesta fase, ao investigar sobre hipóteses de execução da função lambda, tivemos conhecimento que na organização era utilizado o *Slack* para comunicação dentro da organização, bem como ativações de mecanismo. Assim, para otimizar a colocação do *input* (IoC), por parte do analista, construímos um *Bot* no *Slack* para fazer a comunicação entre a AWS e a OpenCTI, realizando "ele" a chamada da função lambda. Assim, o analista não precisa de aceder à AWS, pode utilizar o *Slack* que utiliza na organização para realizar o pedido de "reporte" acerca de um determinado IoC.

A segunda otimização que pretendíamos disponibilizar na nossa solução, prendia-se precisamente com o reporte final apresentado ao analista. Sentimos a necessidade de enriquecer esse reporte, para que sustentasse da melhor forma possível a ativação dos "*use cases*" de resposta a determinado IoC. Neste sentido, melhorámos o código com novas funções, trocas de mensagens e novas conexões, de forma a apresentar um reporte final mais rico e com o máximo de informação classificativa e contextual ao analista. Toda a informação contextual relacionada ao IoC introduzido será devolvido ao analista (ex: "*Bot*", "*email spam*", "*phising*", etc). Realizamos, ainda, uma nova conexão a uma plataforma externa (*Virus Total*) que por sua vez se encontra conectada a outras 94 plataformas de AV, de forma a obter quantas vezes é que um determinado IoC já foi identificado como suspeito ou malicioso. Assim, é apresentado ao analista de um SOC um reporte final com: um "*score*", um "*trust rating*", o impacto para a organização, toda a informação contextual obtida, e quantas vezes foi identificado como suspeito ou malicioso.

5.3 Vulnerabilidades

Como apresentado na revisão da literatura, a segurança de *software* diz respeito aos seguintes três tipos vulnerabilidades:

- **Vulnerabilidade de projeto**: Neste projeto, optámos por utilizar uma comunicação / troca de mensagens (HTTP) via AWS, que possui uma autenticação robusta com políticas de

privacidade e permissões especiais para o utilizador, bem como uma autenticação MFA (Multifator).

- **Vulnerabilidade de codificação**: Para suprimir esta possibilidade, neste projeto, foram implementadas algumas medidas de prevenção de injeção, nomeadamente a impossibilidade de introduzir como *input* algo que não seja o pretendido (ex: se o input pretendido é um IPv4 *Address*, a função só aceitará *inputs* com essa configuração).
- **Vulnerabilidade operacional**: Neste projeto foram utilizados sempre sistemas de *passwords* e autenticação em todas as plataformas, incluindo nos *requests* ao servidor (sempre com autenticação no cabeçalho).

5.4 Perspetiva do Analista de um SOC

Na perspetiva de um analista, o nosso software começará por receber como input um IoC, por exemplo por meio de um *Slack Bot*, diretamente na *AWS lambda function*, na linha de comandos da *PowerShell*, etc. No exemplo seguinte, iremos demonstrar utilizando o *Slack Bot* (@aws). O analista realiza uma invocação da função lambda, ao introduzir o valor do IoC (ex. IPv4 Addr), o nome da função lambda (neste caso "ctiEngine) e a região *AWS* associada à função lambda (neste caso "eu-central-1"), como se pode observar na Figura 5.9.

```
@aws lambda invoke --payload {"value": "192.0.194.206"} --function-name ctiEngine --
region eu-central-1
```

***Figura 5.9**- Invocação do motor de inferência via Slack. Fonte: Snapshot do Autor*

Ao fim de alguns segundos, o analista irá observar no *Slack* o *output* (reporte) do seu *request,* com a *thread intelligence* relacionada com o IoC introduzido, nomeadamente o *score* de risco, o *trust rating*, as *labels* encontradas, o impacto para a organização e quantas vezes foi identificado como suspeito ou malicioso (Figura 5.10).

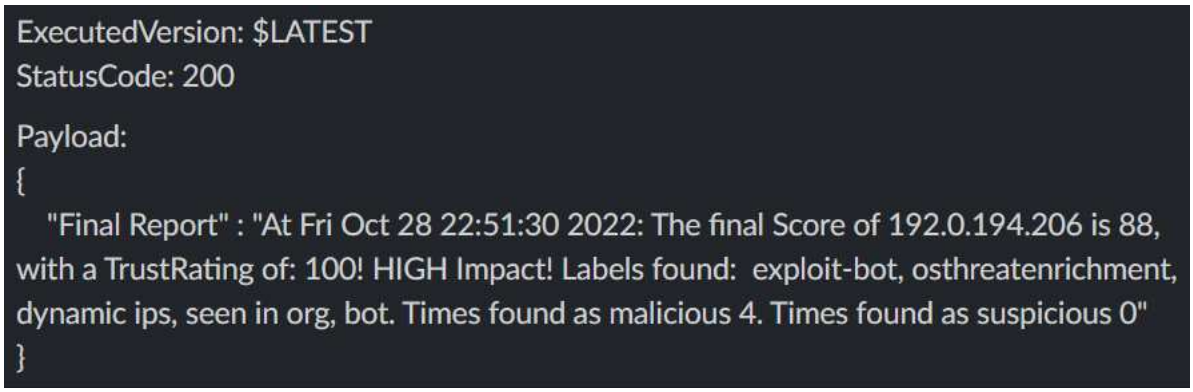

Figura 5.10 *- Reporte Final do Motor de Inferência no Slack. Fonte: Snapshot do Autor*

5.5 Resposta a Incidentes de Segurança: *Playbooks*

Um *playbook* de resposta a incidentes é uma série predefinida de etapas que uma equipa de resposta a incidentes executa ao responder a um determinado tipo de incidente. Essa abordagem permite que as organizações agilizem os seus processos e lidem com incidentes semelhantes ou rotineiros de forma eficiente e consistente [68]. Esta seção discute o conteúdo genérico e as recomendações que as organizações podem optar por incluir nos seus *playbooks* de resposta a incidentes. O conteúdo é organizado pelos estágios do ciclo de vida da resposta a incidentes (Figura 2.2, do Cap. 2), e apresenta-se de acordo com a Figura 5.11, com as seguintes etapas, adaptado de [68]:

Triagem *(Triage)***:** Consiste em perceber se o evento encontrado / despoletado se trata de um incidente de segurança ou não. Para obter essa confirmação, o analista deverá começar por solicitar ao *software* desenvolvido neste projeto informações acerca do evento encontrado.

Classificação *(Classification)***:** Os analistas devem estabelecer fatores de impacto e urgência para determinar o nível de gravidade de um incidente. Assim, os analistas ao solicitarem ao *software* informações acerca do evento, não só registam o evento encontrado, como obtêm uma classificação quantitativa e qualitativa e *labels* com características, já identificadas, desse incidente de segurança (ex: *Phishing, VPN, Bot, Malware*, etc).

Documentação *(Documentation)***:** Após a triagem e classificação de um incidente, os analistas devem documentar todas as informações necessárias num registo de incidentes.

Escalável *(Escalation)***:** Alguns incidentes podem ser relativamente simples de resolver e não requerem conhecimentos e habilidades avançadas. Outros incidentes podem exigir um conjunto de habilidades especializadas ou atenção da administração. Com base no impacto estimado para a organização e/ou nas características recebidas (*labels*), o analista terá de decidir como pretender responder ao incidente

e se é capaz de o fazer sozinho ou se tem de recorrer a alguém hierarquicamente superior e / ou com mais *skills* / experiência.

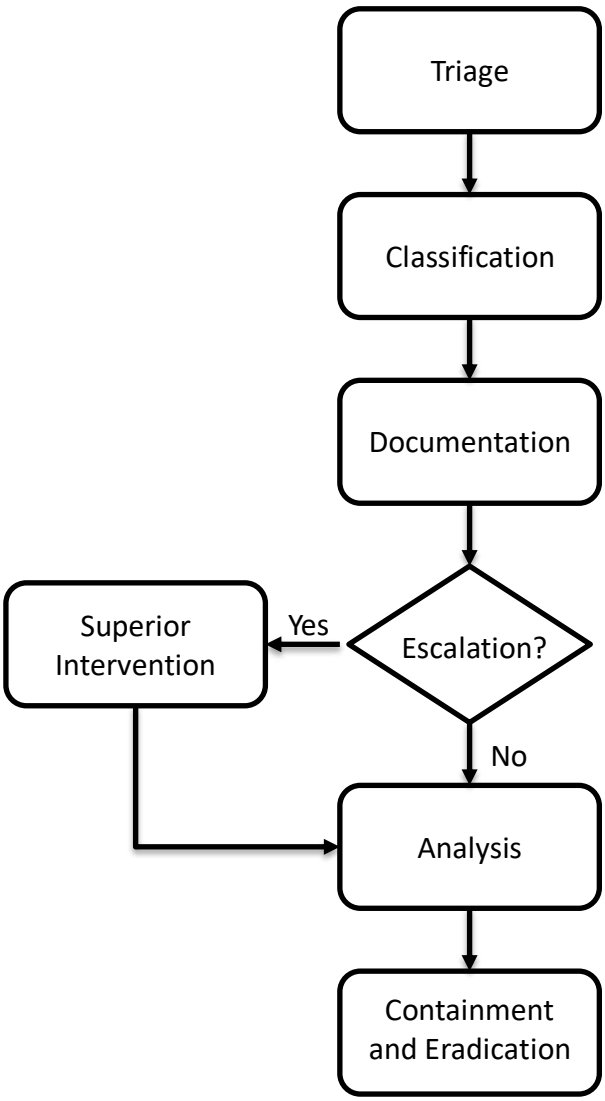

Figura 5.11 *– Etapas no processo de resposta a Incidentes de Segurança. Fonte: Elaboração do Autor adaptado de [68]*

Análise *(Analysis)*: Dependendo da natureza de um incidente, os analistas podem adquirir e examinar diferentes tipos de dados. O objetivo da fase de análise é determinar a atividade do atacante no ambiente comprometido e entender por completo o incidente. As etapas típicas nesta fase incluem, a aquisição e preservação de dados (ex: coletar dados de rede, imagens forenses, etc), análise técnica (consoante o tipo de incidente), construir uma linha do tempo, partilhar / obter informações via plataformas de CTI (já realizado anteriormente, ao utilizar a plataforma proposta neste projeto) e por

fim, ajustar a gravidade / impacto do incidente (com base em informações novas e significativas relacionadas ao incidente investigado). É crucial enfatizar que a análise é um processo iterativo e muitas vezes incremental.

Contenção e Erradicação *(Containment and Eradication):* As informações que os analistas compilam durante as etapas de análise orientam as atividades de contenção e erradicação. A primeira atividade na definição das etapas de contenção e erradicação como parte de uma definição de um *playbook* é desenvolver uma estratégia e, em seguida, selecionar ferramentas e tecnologias para implementar essa estratégia. Esta estratégia deverá incluir, aplicação de mecanismos de controle de acesso, correção de vulnerabilidades, redefinir credenciais, implementar ferramentas de segurança e ajustar as políticas e permissões do sistema. Por outro lado, como medidas diretas de contenção e erradicação, e adaptando ao impacto gerado, o analista deverá, quando aplicável, bloquear o *hash*, conter / bloquear o IoC, procurar *Malware*, geolocalizar o IP ou o Domínio, procurar e extrair os *process files* da memoria RAM, adicionar à *watchlist*, terminar a conta de utilizador (se for o caso), terminar o processo suspeito (se ainda se encontrar a correr), gerar um ticket e realizar um reporte final [89].

CAPÍTULO 6

Testes e Avaliação dos Resultados

6.1 Teste ao Software: Script Python

Testar *software* é uma tarefa repetitiva, demorada e tediosa. Neste sentido, surgiram várias ferramentas para automatizar testes em *scripts python*, nomeadamente o *doctest* e o *unittest* [56]. Ao nosso software iremos aplicar o módulo *doctest,* que usa "*Doc Strings"* presentes no *script Python* para definir os testes do código. O *doctest* procura no *script* algo que seja semelhante a uma sessão interativa*,* executa a sequência de comandos, analisa o *output* e produz um relatório dos testes que falharam, com os erros encontrados [56]. No caso de insucesso apresenta um relatório com os erros encontrados, no caso de sucesso, responde com "ok" a cada teste individual e no final informa "*Test passed*". Assim, aplicámos o modulo *doctest* (Algoritmo 6.1) ao nosso *software*, sobretudo de forma a testar os Algoritmos 4.1 e 4.2, descritos anteriormente no capítulo 4, de forma a garantir que não existiriam falhas na incrementação da sequência baseada em *Fibonacci* e na atribuição no nível de impacto dos IoCs para a organização.

ALGORITMO 6.1 - APLICAÇÃO DO *DOCTEST*

```
1  if __name__ == "__main__":
2      import doctest
3      doctest.testmod()
```

Ao aplicar o *doctest* para testar o Algoritmo 4.1, a incrementação de acordo com a sequência de *Fibonacci,* o objetivo do teste foi analisar, para diferentes *inputs*, se a atribuição do valor seguinte na sequência estaria sempre correto, nomeadamente para os valores que se encontram na fronteira dos intervalos da sequência bem como na atribuição repetida do valor 1. Por outro lado, testar também valores "incorretos", como negativos ou acima de 100. A Figura 6.1 apresenta um exemplo dos resultados dos testes efetuados, a classe tem o nome de "cti" e o algoritmo a ser testado "Fibonacci". Como referido anteriormente, a função recebe como "input", respetivamente, o valor do "score" anterior (quando já existente na plataforma) ou score igual 0 quando é a primeira vez que é inserido, e recebe ainda o número de vezes que já foi identificado na organização, iniciando-se em 1. Todos os testes foram efetuados com sucesso.

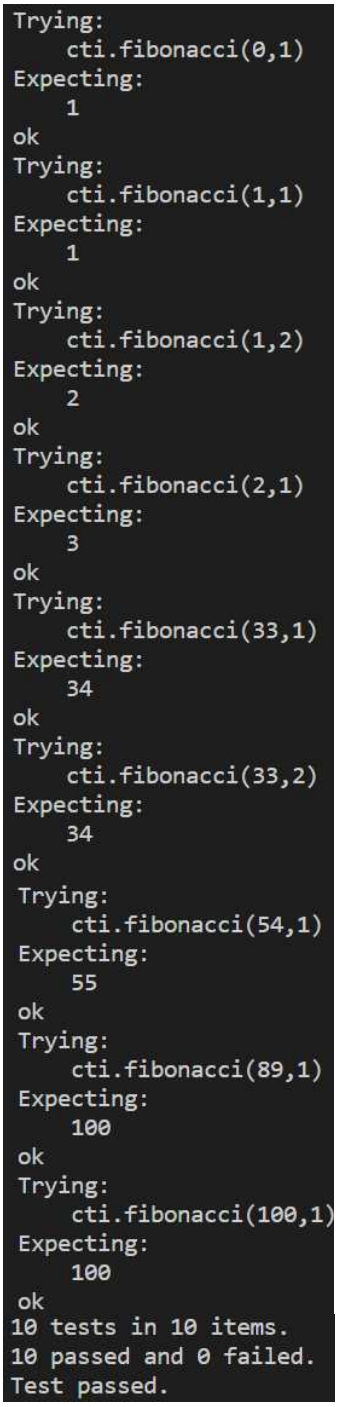

Figura 6.1 *- Teste "doctest" efetuado ao algoritmo 4.1 (Fibonacci). Fonte: Snapshot do Autor*

Posteriormente procedemos ao teste da atribuição do nível de impacto de um IoC para a organização. Aqui, os testes foram muito importantes, ao detetarem uma situação, em que ainda não existia um *trust rate* associado ao *score* atual, isto é, por algum motivo o conector que atribui um *trust rate* ao *score* ainda não tivesse sido invocado com sucesso (como por exemplo algum problema técnico

com o conector, estar em manutenção, ou a ser reiniciado). Quando não existe um valor para o *trust rate*, o nosso *software* identifica-o com uma *label "not atributed yet".* Assim, ao realizarmos os testes evidenciou-se que quando o *trust rate* tinha a *label* referida, não se tratando de um valor inteiro ("*Integer*"), o código gerava o erro "*Value Error*" e não atribuía o impacto para a organização. Esta situação nunca se tinha colocado, nem acontecido, uma vez que o conector que atribui o *trust rate* sempre foi invocado com sucesso e sempre funcionou na perfeição. Neste sentido, alterámos a função para quando um score não tiver um *trust rate* associado, atribuir o valor zero, na medida em que não sabemos se podemos confiar naquele *score* ou não.

A Figura 6.2 representa os testes efetuados ao algoritmo do cálculo do impacto para a organização, como referido anteriormente a função recebe como *input*, respetivamente, um *score* e um *trut rating* associado. Após a referida adaptação, a função passou em todos os testes, como se pode observar.

```
Trying:
    cti.getImpact(39,41)
Expecting:
    'LOW'
ok
Trying:
    cti.getImpact(40,41)
Expecting:
    'MEDIUM'
ok
Trying:
    cti.getImpact(70,100)
Expecting:
    'MEDIUM'
ok
Trying:
    cti.getImpact(71,20)
Expecting:
    'MEDIUM'
ok
Trying:
    cti.getImpact(71,"Not atributed yet")
Expecting:
    'MEDIUM'
ok
Trying:
    cti.getImpact(71,41)
Expecting:
    'HIGH'
ok
```

Figura 6.2 *- Resultado dos testes "doctest" efetuados ao Algoritmo 4.2 do cálculo do impacto de um IoC para a Organização. Fonte: Snapshot do Autor*

6.2 Métricas de Teste de Eficiência

As principais métricas utilizadas em testes de eficiência de *softwares* incluem o Tempo Médio para Deteção (MTTD), o Tempo Médio para Reparação / Resolução (MTTR), Tempo Médio para Investigação (MTTI) e Tempo para Qualificação (TTQ). Neste projeto iremos testar o TTQ do nosso *software*, na medida em que a sua principal funcionalidade é a qualificação / classificação de eventos de segurança. Estas métricas são muito importantes serem do conhecimento da organização, nomeadamente pela equipa que compõe o SOC, como ingrediente chave para a mitigação de incidentes de segurança. Quanto mais baixos forem os valores destas métricas, melhor desempenho um SOC terá na deteção, investigação e resposta a ciberameaças [10].

6.2.1 Tempo para Qualificação (TTQ)

Time to Qualify (TTQ) mede a quantidade de tempo desde que soou o alarme para uma potencial ameaça (ex: *ticket* do SIEM) até ser totalmente inspecionada e qualificada. TTQ irá ter início na introdução de um IoC no *Slack Bot,* iniciar o processo de investigação e no fim aferir por via um reporte final (*output* no *Slack* ou na plataforma OpenCTI) a classificação final desse IoC. O processo termina com a qualificação do grau de ameaça e respetivo impacto para a organização desse IoC.

Teste 1: Neste primeiro teste, será utilizado um IoC (IPv4 Addr) já identificado pela *Snort*[34] como uma ciberameaça, escolhido com o propósito de consultar uma plataforma completamente independente das que compõem esta solução, bem como, por outro lado, ainda não fosse do conhecimento da OpenCTI, de forma a ser possível verificar se o score obtido no final corresponde a um *score* de uma potencial ciberameaça.

Assim, começámos por inserir manualmente o IoC (“91.109.182.12”) na plataforma OpenCTI, ativando automaticamente os conectores de enriquecimento da OpenCTI, à exceção do conector “osthreatenritchement) que só executa verificações quando recebe a “*label”*: “*seen in org*” atribuída pela nossa solução. O objetivo desta primeira parte do teste é apenas introduzir o IoC na plataforma e verificar / demonstrar que não existe nenhuma informação sobre ele. Neste sentido, verificamos na Figura 6.3 que nenhum conector identificou este IoC como uma potencial ameaça e o *score* é zero.

[34] Snort: É o principal Sistema de Prevenção de Intrusão de Código Aberto (IPS) do mundo. O Snort IPS usa uma série de regras que ajudam a definir a atividade maliciosa da rede e usa essas regras para encontrar pacotes que correspondam a eles e gera alertas para os utilizadores. Fonte: https://www.snort.org/

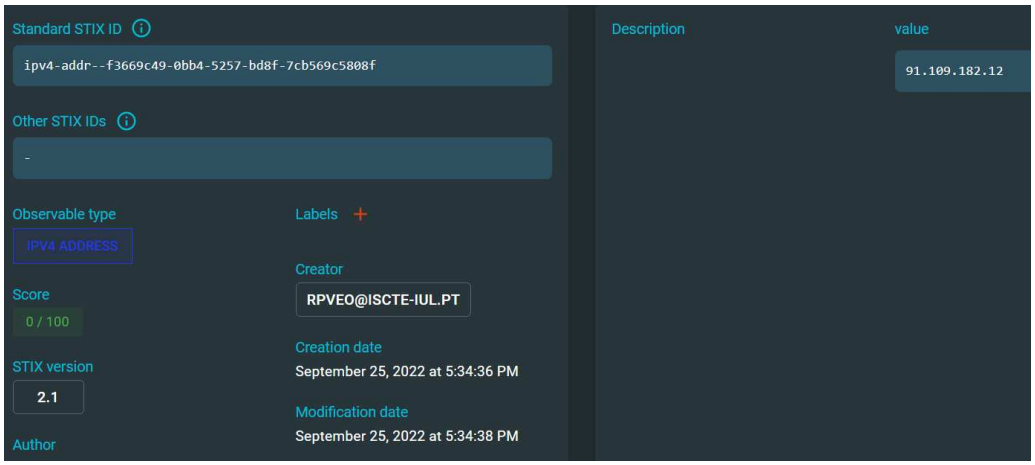

Figura 6.3 *- Introdução de um IoC malicioso na OpenCTI sem label "seen in org". Fonte: Snapshot do Autor*

Posteriormente, quando aplicada a nossa solução, o IoC é imediatamente reconhecido como uma forte possibilidade de se tratar de uma ciberameaça, com um score de 72 (Figura 6.4), comprovando que se trata de uma ciberameaça e com um risco e um potencial impacto elevado para a organização.

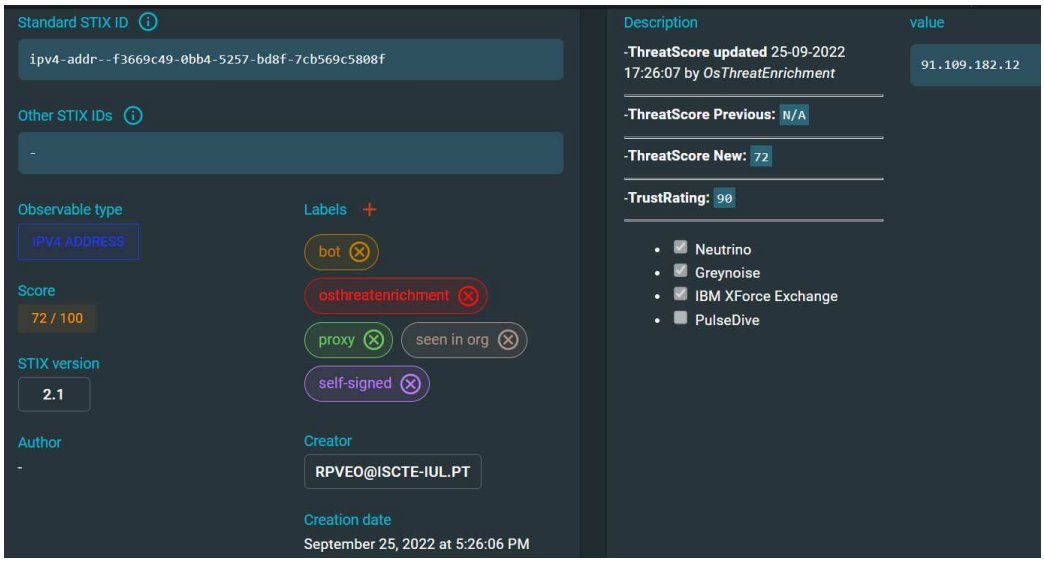

Figura 6.4 *- Execução do motor de inferência. Fonte: Snapshot do Autor*

Resultado do Teste 1: 01:06 min para ser confirmada a ciberameaça, uma vez que inclui o processo manual de introdução do IoC na plataforma e só depois a execução da nossa solução. Representando assim o cenário mais lento possível para TTQ.

Teste 2: No segundo teste, decidimos manter ainda as premissas do primeiro teste, no entanto sem a introdução manual na OpenCTI, de forma a testar a real eficiência do nosso motor de inferência. Foi utilizada outra fonte externa, independente a esta solução, no caso uma *blacklist*, "*blockedservers*", que está constantemente a atualizar a partilha de IPv4 *addr* maliciosos, escolhemos o mais recente (últimas 24h), o IPv4: 92.255.85.201. Executámos o nosso *software*, e o output revelou, com clareza e confiança, que o IoC introduzido se trata de uma ciberameaça, com um grau de confiança de 100% (Figura 6.5).

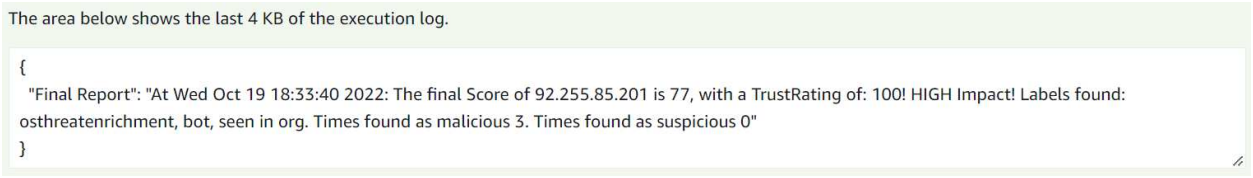

Figura 6.5 *- Reporte final da execução do motor de inferência para IoC malicioso. Fonte: Snapshot do Autor*

Resultado do teste 2: Este teste, como executado diretamente, via AWS, precisou apenas de 16.28 segundos para apurar que o IoC introduzido se trata de uma ciberameaça, de risco elevado, com 100% de confiança. Representando, assim, o TTQ médio mais rápido possível.

Teste 3: Para o teste 3 optámos por introduzir um IoC (*IPv4 Addr*) de uma *whitelist*, potencialmente benigno, tentando perceber, quanto tempo demoraria a considerar tratar-se de uma ciberameaça para a organização. Simulando um ciberataque em que estivesse a utilizar este IoC pela primeira vez ("dia zero"). Testando a sequência de *Fibonacci* em conjunto com as métricas do conector de enriquecimento ("*osthreatenritchement"*), utilizados nesta solução. Executámos o motor de inferência, via AWS, obtivemos um primeiro *log output* (Figura 6.6), a informar que este era um IoC novo para a OpenCTI, e que nunca foi observado em nenhum AV, seguidamente, obtivemos um primeiro *score,* 32, baixo impacto (Figura 6.7). Corresponde ao esperado, um potencial IoC benigno.

Figura 6.6 *- Log output na AWS, ao introduzir um IoC novo na OpenCTI. Fonte: Snapshot do Autor*

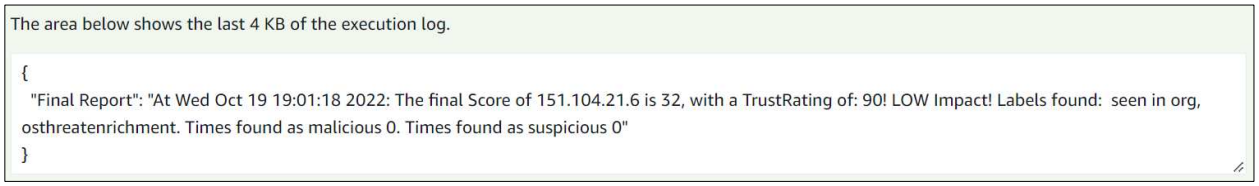

Figura 6.7 *- Reporte Final na AWS do IoC potencialmente benigno introduzido. Fonte: Snapshot do Autor*

Assim, decidimos continuar com o teste, e persistir, executando novamente o *software*, solicitando uma nova pesquisa sobre aquele IoC, simulando a receção de um novo *ticket* por parte de um SIEM, IDS, IPS ou outro. O *output* seguinte revela um primeiro incremento de *score*, consequente da sua reincidência e consequente aplicação do incremento via sequência de *Fibonacci* (Figura 6.8). E foram precisas mais duas execuções, Figuras 6.9 e 6.10, respetivamente, até ser considerada uma ciberameaça com elevado impacto para a organização, devido à sua persistência nos sistemas organizacionais, no mesmo dia.

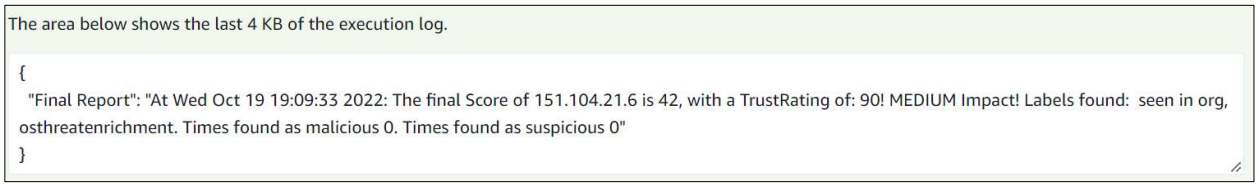

Figura 6.8 *– Segundo Reporte Final na AWS do IoC potencialmente benigno introduzido, com o primeiro incremento devido a reincidência na organização. Fonte: Snapshot do Autor*

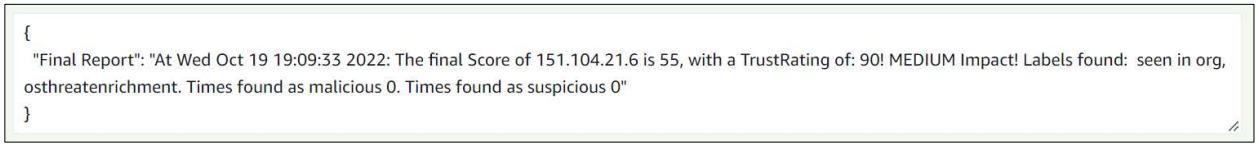

Figura 6.9 *- Terceiro Reporte Final na AWS do IoC potencialmente benigno introduzido, com o segundo incremento devido a reincidência na organização. Fonte: Snapshot do Autor*

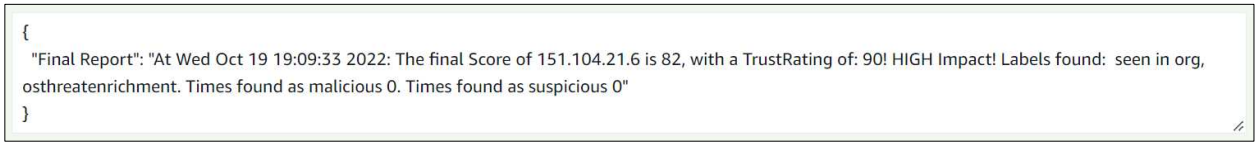

Figura 6.10 *- Quarto Reporte Final na AWS do IoC potencialmente benigno introduzido após o terceiro incremento devido a reincidência na organização. Fonte: Snapshot do Autor*

Resultado do teste 3: Este teste consistiu em 4 execuções da solução proposta, perfazendo um total de 64.74 segundos, no entanto, importante enaltecer que um sistema de deteção de intrusões, tem por norma, um tempo de intervalo definido entre reportes *(tickets)*, que pode ser de por exemplo 15 ou 30 min, ou outro intervalo. Assim, se por exemplo, o intervalo de reportes fosse de 15min, demoraria cerca de 1 hora a classificar um IoC como uma ciberameaça para a Organização, no caso de ser um ataque executado pela primeira vez ("dia zero") com aquele IoC.

6.3 Confusion Matrix

Como teste final, e no sentido de apurar resultados concretos, de forma global e generalizada, da nossa solução, realizámos um teste com 240 IPv4 *Addrs,* todos eles identificados nas últimas 48h antes do teste, de forma a poder apurar, mais do que o *score* final, a eficiência dos modelos de classificação, utilizados na presente solução. Considerámos esta premissa crucial para a validação do resultado, uma vez que se utilizados IoCs registados em *blacklists* há mais de 48h, haverá tendencialmente já mais informação partilhada em diversas fontes sobre esses IoCs, correndo o risco de enviesar o resultado, com uma taxa de *accuracy* superior.

Para o teste efetuado, foram utilizados 120 IPs identificados como malignos em 4 fontes de "*Blacklists"* distintas (*BlockedServers.com, Blocklist.de, rbl.InterServer.net, ProjectHoneyPot.org*), em que nenhuma destas fontes tem contribuição na formulação da CTI da solução presente, e 120 outros retirados de uma *whitelist* (*gooddata.com*). Como forma de apresentar os resultados obtidos, construímos uma "*Confusion Matrix*" (Tabela 6.1).

Tabela 6.1 *- Confusion Matrix. Fonte: Elaboração do Autor*

		ATRIBUIÇÃO	
		SIM	**NÃO**
REAL	**SIM**	Verdadeiro Positivo (VP) 105	Falso Negativo (FN) 15
	NÃO	Falso Positivo (FP) 0	Verdadeiro Negativo (VN) 120

Legenda: *VP: Verdadeiro Positivo: Valores previstos corretamente: FP: Falsos Positivos: Valores negativos previstos como positivos; FN: Falso Negativo: Valores positivos previstos como negativos; VN: Verdadeiro Negativo: Valores previstos corretamente como um negativo real*

Um dos principais objetivos desta matriz é apurar o nível de correção de uma solução. Esse nível é denominado de "*accuracy*", uma métrica para identificar a percentagem de previsões que foram classificados corretamente e calcula-se através da equação 6.1.

$$Accuracy = \frac{VP + VN}{\mathrm{VP} + \mathrm{VN} + \mathrm{FP} + \mathrm{FN}} \qquad (6.1)$$

Resultado do teste: O teste efetuado contundiu na matriz apresentada anteriormente, e consistiu no discernimento entre "ciberameaça" e "benigno". A premissa utilizada para ser considerada "ciberameaça" foi: o *score* obtido ser superior a 40 e o *trust rating* associado ser maior ou igual a 40 ou estar identificado pelo VT, pelo menos, uma vez, ou seja, ((*score* > 40 **and** *trust rating* >= 40) **or** *malicious from VT* > 0). Assim, ao analisar a matriz, observamos que a nossa solução apurou com uma ciberameaça 105 das 120 identificadas e 120 em 120 dos IoCs identificados em *whitelists*, assim, segundo a equação 6.1, apresentou como resultado deste teste, uma taxa de "*accuracy*" de 93.75%.

6.4 Discussão de Resultados

A avaliação de ameaças é uma componente essencial na avaliação de risco da segurança da informação [80]. A solução desenvolvida nesta investigação revelou-se muito eficaz na rápida classificação de um evento de segurança, recorre a CTI e devolve ao analista um nível de risco associado à potencial ciberameaça, um nível de confiança nesse valor, o impacto para a organização com base nesses indicadores e ainda outras informações contextuais associadas ao IoC.

A abordagem presente nesta investigação contribui para a avaliação do risco de determinados eventos de segurança, bem como para a classificação de potenciais ameaças, o que, de acordo com [80], possibilita a identificação e organização de ciberameaças em classes, avaliação dos seus impactos e impulsiona a criação de estratégias para prevenir e mitigar os impactos das ciberameaças no sistema organizacional, assim como no emprego de métodos de resposta em conformidade.

Num centro de operações de segurança, os analistas realizam a triagem de eventos de segurança, e como de ciberameaças se podem tratar, exige a sua total atenção. Um grande desafio que enfrentam é o número elevado de alertas "falsos positivos". Os analistas precisam de investigar esses alertas benignos, de baixo impacto, mas de alto volume, reduzindo a capacidade do SOC e corroendo a moral do analista [76]. Os analistas precisam de soluções que possam ajudá-los a quantificar os riscos de cada incidente de segurança detetado [93]. Assim, os principais fatores a

colmatar na gestão de eventos ou incidentes de segurança, com a solução proposta, é precisamente a redução do tempo em investigar "falsos positivos" e sobretudo reduzir o tempo de pesquisa / investigação, por parte do analista, em cada evento de segurança suspeito.

Os resultados desta investigação são comparáveis com algumas investigações / trabalhos relacionados descritos anteriormente. Comparado com outros trabalhos, [78] desenvolveu um modelo de ML (K-NN) com o objetivo de classificar eventos de segurança, ao nível do impacto para a organização. Os autores utilizaram, na fase experimental, de forma a treinarem o modelo, "IP *adresses*" e um *label* com a descrição do evento e outro identificador, colocado manualmente pelo analista, com o nível de prioridade para dar resposta ao IoC. A classificação atribuída pelo modelo varia entre "nada crítico" a "muito crítico". Os autores concluíram que o modelo poderá alcançar bons resultados de filtragem e que é capaz de reduzir a carga de trabalho de um analista de segurança, filtrando efetivamente muitos alarmes indesejados. Por outro lado, [76] desenvolveram um modelo de ML para auxiliar o analista de um SOC a identificar ciberameaças, atribuindo a cada evento uma das seguintes *tags*: "Ameaça" ou "Falso Positivo", bem como um "trust rating" associado, correspondendo ao nível de confiança na *tag* atribuída. No entanto, como se tratam de modelos de previsão, baseados em modelos matemáticos, por vezes erram e identificam como "falso positivo" uma ciberameaça [76], assim, os autores do modelo sentiram necessidade de recorrer, além da previsão do algoritmo de ML, a uma plataforma externa, no caso, o "*Vírus Total*" (VT), de forma a tentar colmatar algumas lacunas do algoritmo matemático através de evidências reais. Este tem sido um dos principais problemas evidenciados na literatura [76, 78, 91], no que se refere à classificação de ameaças e à respetiva precisão ("*Accuracy*"). A precisão de uma métrica é, por definição, dependente da precisão das medidas que compõem a métrica. As organizações enfrentam atualmente vários problemas relacionados com a precisão de medidas, uma vez que, as medidas são muitas vezes definidas de forma imprecisa [91].

Na nossa solução, aqui proposta, também foi utilizado a plataforma VT para auxiliar na priorização dos IoC identificados com a *Label* "Low Impact", de forma a discernir com mais contexto os falsos alertas das potenciais ameaças. Porém, tal como [76] afirmaram, não é suficiente para validar se é ou não um falso alerta (os autores observaram casos em que o IoC não se encontrava na plataforma VT e veio a ser identificado como ciberameaça). Conscientes dessa realidade, na nossa solução, apenas consideramos a informação recolhida na plataforma VT, como uma fonte de suporte à priorização dos IoCs de "*Low Impact*", ou seja, se um analista tiver 2 incidentes de segurança para dar resposta, ambos com um "score" de 30 e "trust rating" 90 (por exemplo) e num deles o valor do "*times identified as malicious*" ou o "*times identified as suspicious*" for superior, o analista responde primeiro a esse. Porém, um exemplo, da plataforma VT não ser suficiente para discernir se um IoC é ou não uma ciberameaça, resultou num teste aleatório a um IPv4 *addr* (Figura 6.11), em que o IoC foi

identificado como ciberameaça pelo conector de enriquecimento ("*osthreatenrichment"*) e na plataforma VT não foi encontrado. Por outro lado, demonstra a qualidade da CTI, por trás, do mecanismo de classificação aqui proposto e valida a qualidade da classificação obtida com esta solução de automatização da classificação de ameaças, assim como o sistema de pontuação e confiança nessa pontuação. Os sistemas de pontuação, de forma geral, são amplamente utilizados para suportar e validar opções, fazer comparações e apoiar decisões [92], assim como as soluções de automação, que incluem ferramentas de *software* personalizadas para lidar, tratar e classificar um número crescente de eventos, sem depender de um crescimento proporcional no número de funcionários [34], contribuem diretamente para a equipa do SOC, ao diminuir a "fadiga do alerta", tornando os analistas de segurança mais produtivos e permitindo que toda a equipa se concentre em decisões importantes e não em tarefas mundanas [1, 7, 31, 34]. Assim, é neste sentido que, esta solução proposta, é uma mais-valia na gestão de incidentes num SOC, com uma solução robusta, flexível, dinâmica e eficiente na classificação e priorização de ciberameaças, fornecendo um suporte fundamental / critico, em tempo útil, no momento de resposta aos incidentes de segurança.

Figura 6.11 *– Output (formato Json) de um teste a um IoC aleatório, reconhecido como ciberameaça pelo motor de inferência e nunca identificado com malicioso ou suspeito em plataformas de AV*

Durante os testes efetuados à solução (TTQ), aqui proposta, observámos uma importante eficiência e rapidez na classificação de uma ciberameaça já identificada em plataformas CTI, com recolha das características desse IoC, por meio de *labels* e com nível de priorização para a resposta, por meio do "impacto" para a Organização. Por outro lado, através da penalização atribuída em *score* sempre que um evento persiste / reaparece nos sistemas da Organização, via sequência de *Fibonacci* conseguimos ser rápidos na identificação de potenciais ciberameaças ("dia zero"), ou seja, IoCs nunca observados e que poderão estar a atacar pela primeira vez. Neste sentido, os resultados evidenciaram que uma solução automática, com a capacidade de classificar eventos de segurança e medir o impacto das potenciais ciberameaças para uma organização, é uma solução eficiente na gestão de eventos ou incidentes de segurança.

CAPÍTULO 7

Conclusões

Muitas vezes, os analistas precisam de consultar diversas ferramentas e bases de dados para obter informações contextuais de determinado evento de segurança, porém, se aplicados mecanismos automáticos as estas tarefas, o tempo de triagem e classificação, por parte dos analistas, irá ser substancialmente menor, contribuído também para uma redução critica, do tempo médio de resolução (MTTR) geral [34].

Os analistas de um SOC são frequentemente inundados com um elevado volume de alertas de baixa prioridade [1, 7]. Este processo de automação agilizará a análise de alertas e permitirá que os analistas do SOC dediquem mais tempo a problemas mais difíceis, aumentando a eficiência dos analistas e reduzindo a sua carga de trabalho.

Proporcionar classificações de risco a eventos de segurança pode contribuir para as organizações desenvolverem estratégias eficientes ao nível da gestão do risco [93]. No entanto, antes de criar regras de automação para responder aos incidentes, os analistas precisam de ter garantias que podem confiar num modelo [76]. A solução proposta permite uma melhor compreensão da natureza das ciberameaças, contribuindo para o desenvolvimento de estratégias adequadas e metodologias de resposta a incidentes de segurança, bem como uma contribuição para a prevenção e mitigação de ciberameaças.

Este trabalho pretendeu tratar e mitigar os problemas ao nível da gestão de incidentes de segurança num SOC, nomeadamente ao nível da classificação de ciberameaças, com o objetivo de propor uma solução genérica, flexível, dinâmica e robusta, aplicável a qualquer organização. A presente solução foi apresentada, de forma genérica e como um produto final, na Conferência Internacional de Cibersegurança[35] (agregando os dois projetos de investigação desenvolvidos em paralelo), tendo obtido *feedbacks* muito motivadores e foi galardoada como a terceira melhor solução apresentada na conferência. O "*Head of Technology*" da *Altice Labs*, Miguel Biscaia, um dos membros do júri [97], deixou claro que "cibersegurança sem automação não é eficiente", as métricas onde se baseia essa automação têm de ser válidas e de confiança, enaltecendo ainda a extrema importância de uma rápida deteção, identificação e classificação de uma ciberameaça para uma consequente eficiente resposta.

[35] "Cybersecurity as the Foundation of the Digital World" (2022), in Inncyber Innovation Hub, Digital transformation, Cyber & IoT, International Conference, 3rd Edition, Altice Labs HQ, Aveiro.

O motor de inferência proposto, pode tornar-se uma parte importante em aplicações futuras dentro de uma Organização, e pode acelerar significativamente o processo de tomada de decisão.

7.1 Limitações e Trabalho futuro

As ameaças são exponencialmente crescentes a cada dia que passa, assim como a sua complexidade e sofisticação associada, tornando as oportunidades de desenvolvimento, investigação e melhoria contínua um desafio constante e um universo de possibilidades. Como constatado ao longo deste trabalho, são inquestionáveis as vantagens da automação de processos, AI e de *Machine Learning*. No seguimento deste princípio, em trabalhos futuros seria pertinente conectar este sistema de classificação aos sistemas de monitorização e deteção de incidentes de segurança, SIEM, IDS, IPS, ou outros, de forma que a introdução do IoC no nosso motor de inferência não precisasse de ser sempre realizado manualmente. Uma das limitações desta investigação passa pelo único tipo de *input* abordado, o IPv4 *Address*. Esta decisão foi consciente e assumida desde início, como forma de não acrescentar mais complexidade inicial à elaboração da solução, de forma a conseguirmos focar-nos exclusivamente nos melhores métodos de trabalhar o problema e encontrar a melhor solução. Porém, todo o sistema desenvolvido é flexível à adaptação de novos *inputs*, com pequenas alterações, e nesse sentido seria pertinente acrescentar a possibilidade de introduzir outros formatos de IoC como por exemplo IPv6 *Addr*, domínios, *URLs*, *Hostnames*, entre outros.

Depois de ampliar as opções de *input* e automatizar a introdução na nossa solução, a sugestão seguinte é trabalhar / melhorar a taxa de "*accuracy*" apurada, uma vez que, como explorado na literatura, antes de automatizar respostas é necessário confiar fortemente na classificação. Por outro lado, a "*accuracy*" foi determinada com base em resultados de "*blacklists*", representando algumas limitações que as podem tornar pouco confiáveis, por exemplo, a falta de capacidade em conter *malware* recém-gerado [13], no entanto, de forma a minimizar essas limitações consideramos apenas endereços de IP listados em "*blacklists*" nas últimas 48h.

É sempre difícil estarmos 100% certos, mas podemos e devemos tentar estar mais perto desses valores. Nesse sentido, apostar em ML para complementar a classificação baseada em CTI é um caminho com fortes possibilidades de ser bem-sucedido, uma vez que, como observado na literatura, os progressos nessa área têm-se verificado manifestamente bem-sucedidos e com forte potencial de crescimento e desenvolvimento, nomeadamente ao nível da redução dos falsos negativos (FN) e ciberameaças recém geradas. Por outro lado, para a CTI ser uma ferramenta útil e poderosa, é necessário que exista uma compartilha de informações em tempo útil com outras organizações. No entanto, isso nem sempre acontece, sobretudo devido à gestão da privacidade de determinadas organizações, que optam por manter os ataques ou tentativas de, no anonimato [14].

Por fim, seria importante finalizar o processo de automação, com uma resposta adequada ao respetivo incidente de segurança. Esta automação de resposta poderá ser efetuada por meio de uma ferramenta SOAR ou também com funções AWS lambda.

Referências Bibliográficas

[1] M. Vielberth, F. Böhm, I. Fichtinger and G. Pernul (2020), "Security Operations Center: A Systematic Study and Open Challenges", in IEEE Access, vol. 8, pp. 227756-227779, doi: 10.1109/ACCESS.2020.3045514.

[2] G. González-Granadillo, S. González-Zarzosa, and R. Diaz (2021), "Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures", Sensors (Basel, Switzerland), 21(14), 4759, [online], https://doi.org/10.3390/s21144759.

[3] F. Kamoun, F. Iqbal, M. A. Esseghir and T. Baker (2020), "AI and machine learning: A mixed blessing for cybersecurity", 2020 International Symposium on Networks, Computers and Communications (ISNCC). pp. 1-7, doi: 10.1109/ISNCC49221.2020.9297323.

[4] A. Salih, S. T. Zeebaree, S. Ameen, A. Alkhyyat and H. M. Shukur (2021), "A Survey on the Role of Artificial Intelligence, Machine Learning and Deep Learning for Cybersecurity Attack Detection", *2021 7th International Engineering Conference "Research & Innovation amid Global Pandemic" (IEC)*, pp. 61-66, doi: 10.1109/IEC52205.2021.9476132.

[5] P. Baroni et al. (2021), "Self-Aware Effective Identification and Response to Viral Cyber Threats", 2021 13th International Conference on Cyber Conflict (CyCon), pp. 353-370, doi: 10.23919/CyCon51939.2021.9468294.

[6] C. Zhong, J. Yen, P. Liu and R. F. Erbacher (2019), "Learning From Experts' Experience: Toward Automated Cyber Security Data Triage", in IEEE Systems Journal, vol. 13, no. 1, pp. 603-614, doi: 10.1109/JSYST.2018.2828832.

[7] A. Perera, S. Rathnayaka, N. D. Perera, W. W. Madushanka and A. N. Senarathne (2021), "The Next Gen Security Operation Center", *2021 6th International Conference for Convergence in Technology (I2CT)*, pp. 1-9, doi: 10.1109/I2CT51068.2021.9418136.

[8] C. Islam, M. Ali Babar, and S. Nepal (2019), "Automated Interpretation and Integration of Security Tools Using Semantic Knowledge", (pp. 513–528). https://doi.org/10.1007/978-3-030-21290-2_32.

[9] C. Islam, M. Ali Babar, and S. Nepal (2019), "A Multi-Vocal Review of Security Orchestration", *ACM Comput. Surv.*, *52*(2). https://doi.org/10.1145/3305268.

[10] J. Kinyua and L. Awuah (2021), “AI/ML in security orchestration, automation and response: Future research directions”, *Intelligent Automation and Soft Computing*, *28* (2), 527–545. https://doi.org/10.32604/iasc.2021.016240.

[11] C. Acarturk, M. Ulubay and E. Erdur (2020), “Continuous improvement on maturity and capability of Security Operation Centres”, *IET Information Security*, doi: 15. 10.1049/ise2.12005.

[12] C. Nilă, I. Apostol and V. Patriciu (2020), "Machine learning approach to quick incident response", *2020 13th International Conference on Communications (COMM)*, pp. 291-296, doi: 10.1109/COMM48946.2020.9141989.

[13] S. Soni and B. Bhushan (2019), “Use of Machine Learning algorithms for designing efficient cyber security solutions", *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pp. 1496-1501, doi: 10.1109/ICICICT46008.2019.8993253.

[14] R. Vast, S. Sawant, A. Thorbole and V. Badgujar (2021), "Artificial Intelligence based Security Orchestration, Automation and Response System", *2021 6th International Conference for Convergence in Technology (I2CT)*, pp. 1-5, doi: 10.1109/I2CT51068.2021.9418109.

[15] M. Al-Omari, M. Rawashdeh, F. Qutaishat, M. Alshira’H, and N. Ababneh (2021), “An Intelligent Tree-Based Intrusion Detection Model for Cyber Security. Journal of Network and Systems Management”, 29(2), 20. https://doi.org/10.1007/s10922-021-09591-y.

[16] O. Podzins and A. Romanovs (2019), "Why SIEM is Irreplaceable in a Secure IT Environment?", *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pp. 1-5, doi: 10.1109/eStream.2019.8732173.

[17] D. Mihindu and F. Khosrow-shahi (2020), “Collaborative Visualisation embedded Cost-efficient, Virtualised Cyber Security Operations Centre”, https://doi.org/10.1109/IV51561.2020.00078.

[18] K. Ferencz, J. Domokos and L. Kovács (2021), "Review of Industry 4.0 Security Challenges", *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pp. 245-248, doi: 10.1109/SACI51354.2021.9465613.

[19] U. Ünal, C. N. Kahya, Y. Kurtlutepe and H. Dağ (2021), "Investigation of Cyber Situation Awareness via SIEM tools: a constructive review", *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pp. 676-681, doi: 10.1109/UBMK52708.2021.9558964.

[20] U. Noor, Z. Anwar, J. Altmann and Z. Rashid (2020), “Customer-oriented ranking of cyber threat intelligence service providers”, *Electronic Commerce Research and Applications*, 41, 100976. https://doi.org/https://doi.org/10.1016/j.elerap.2020.100976.

[21] T. D. Wagner (2019), "Cyber Threat Intelligence for “Things””, *2019 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (Cyber SA)*, pp. 1-2, doi: 10.1109/CyberSA.2019.8899384.

[22] Yeboah-Ofori, S. Islam and E. Yeboah-Boateng (2019), "Cyber Threat Intelligence for Improving Cyber Supply Chain Security", *2019 International Conference on Cyber Security and Internet of Things (ICSIoT)*, pp. 28-33, doi: 10.1109/ICSIoT47925.2019.00012.

[23] M. A. M. Carrasco and C. Wu (2020), “Review: Deep Learning Methods for Cybersecurity and Intrusion Detection Systems”, CoRR, abs/2012.02891. https://arxiv.org/abs/2012.02891.

[24] S. De Dutta and R. Prasad (2020), "Cybersecurity for Microgrid", *2020 23rd International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pp. 1-5, doi: 10.1109/WPMC50192.2020.9309494.

[25] P. Perrone, F. Flammini and R. Setola (2021), "Machine Learning for Threat Recognition in Critical Cyber-Physical Systems", *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pp. 298-303, doi: 10.1109/CSR51186.2021.9527979.

[26] S. Zeadally, E. Adi, Z. Baig and I. A. Khan (2020), "Harnessing Artificial Intelligence Capabilities to Improve Cybersecurity", in IEEE Access, vol. 8, pp. 23817-23837, doi: 10.1109/ACCESS.2020.2968045.

[27] Cortex XSOAR (2019), “The State of SOAR Report”. [Online]. Available: https://start.paloaltonetworks.com.

[28] R. F. Gibadullin and V. V. Nikonorov (2021), "Development of the System for Automated Incident Management Based on Open-Source Software", *2021 International Russian Automation Conference (RusAutoCon)*, pp. 521-525, doi: 10.1109/RusAutoCon52004.2021.9537385.

[29] Brewer, R. (2019). “Could SOAR save skills-short SOCs?”, *Computer Fraud & Security*. https://doi.org/10.1016/S1361-3723(19)30106-X.

[30] E. Agyepong, Y. Cherdantseva, P. Reinecke and P. Burnap (2020), "Towards a Framework for Measuring the Performance of a Security Operations Center Analyst", *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1-8, doi: 10.1109/CyberSecurity49315.2020.9138872.

[31] S. Muppidi, L. Fisher and G. Parham (2022), "AI and automation for cybersecurity. How leaders succeed by uniting technology and talent (IBM)".

[32] IBM. (2022), "Cost of a Data Breach: Report 2022". From https://www.ibm.com/reports/data-breach?mhsrc=ibmsearch_a&mhq=data%20breach

[33] Google Cloud & Deloitte (2020), "Future of the SOC, Forces shaping modern security operations". White Paper.

[34] Trull, J. (2017), "Top 5 best practices to automate security operations". [Online]. Available: https://cloudblogs.microsoft.com/microsoftsecure/2017/08/03/top-5-best-practices-to-automate-security-operations/.

[35] Swimlane ebook (2020), "Security Orchestration, Automation and Response (SOAR) capabilities".

[36] Knauss, E. (2020), "Constructive Master's Thesis Work in Industry: Guidelines for Applying Design Science Research", CoRR, abs/2012.04966. https://arxiv.org/abs/2012.04966

[37] A. Rauf and M. AlGhafees (2015), "Gap Analysis between State of Practice and State of Art Practices in Agile Software Development", *2015 Agile Conference*, pp. 102-106, doi: 10.1109/Agile.2015.21.

[38] M. Alfadel, D. E. Costa and E. Shihab (2021), "Empirical Analysis of Security Vulnerabilities in Python Packages", *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 446-457, doi: 10.1109/SANER50967.2021.00048.

[39] J. Ruohonen (2018), "An Empirical Analysis of Vulnerabilities in Python Packages for Web Applications", *2018 9th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, pp. 25-30, doi: 10.1109/IWESEP.2018.00013.

[40] C. Onwubiko (2015), "Cyber security operations centre: Security monitoring for protecting business and supporting cyber defense strategy", *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pp. 1-10, doi: 10.1109/CyberSA.2015.7166125.

[41] I. Ortiz Garces, M. F. Cazares and R. O. Andrade (2019), “Detection of Phishing Attacks with Machine Learning Techniques in Cognitive Security Architecture”, *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. doi:10.1109/csci49370.2019.000

[42] R. O. Ogundokun, S. Misra, A. N. Babatunde and S. Chockalingam (2022), "Cyber Intrusion Detection System based on Machine Learning Classification Approaches," *2022 International Conference on Applied Artificial Intelligence (ICAPAI)*, pp. 1-6, doi: 10.1109/ICAPAI55158.2022.9801566.

[43] A. Mishra and P. Yadav (2020), "Anomaly-based IDS to Detect Attack Using Various Artificial Intelligence & Machine Learning Algorithms: A Review," *2nd International Conference on Data, Engineering and Applications (IDEA)*, pp. 1-7, doi: 10.1109/IDEA49133.2020.9170674.

[44] R. Trifonov, O. Nakov and V. Mladenov (2018), “Artificial Intelligence in Cyber Threats Intelligence”. *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*. doi:10.1109/iconic.2018.860123

[45] Rong Li et al (2007), “Support Vector Machine combined with K-Nearest Neighbors for Solar Flare Forecasting”, Chin. J. Astron. Astrophys. doi: https://doi.org/10.1088/1009-9271/7/3/15

[46] A. Zúquete (2018). “Segurança em Redes Informáticas”. 5ª Edição. FCA.

[47] M. Correia and P. Sousa (2017). “Segurança no Software”. 2ª Edição. FCA

[48] M. Véstias (2019). “Redes Cisco – Para Profissionais”. 7ª Edição. FCA

[49] K. Knerler, I. Parker and C. Zimmerman (2022), “11 Strategies of a World-Class Cybersecurity Operations Center”. MITRE

[50] N. Krabbe (2017), “Detecting Brute Force Attacks with Splunk”. From https://www.splunk.com/en_us/blog/partners/detecting-brute-force-attacks-with-splunk.html

[51] A. Mehra and S. Badotra (2021), "Artificial Intelligence Enabled Cyber Security", *2021 6th International Conference on Signal Processing, Computing and Control (ISPCC)*, pp. 572-575, doi: 10.1109/ISPCC53510.2021.9609376.

[52] S. Russel and P. Norvig (2010), “Artificial Intelligence: A Modern Approach”. 3rd Edition. Pearson Education.

[53] A. Turing (1950), “Computing machinery and intelligence” in “The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life: Plus The Secrets of Enigma”

[54] C. Tsai et al. (2009), “Intrusion detection by machine learning: A review”. doi: https://doi.org/10.1016/j.eswa.2009.05.029

[55] P. Arntz (2018), “How artificial intelligence and machine learning will impact cybersecurity”, from https://www.malwarebytes.com/blog/news/2018/03/how-artificial-intelligence-and-machine-learning-will-impact-cybersecurity

[56] Borges, L. 2009. “Python para desenvolvedores”. Edição do Autor. 1ª Edição.

[57] R. F. Gibadullin and V. V. Nikonorov (2021), "Development of the System for Automated Incident Management Based on Open-Source Software", *2021 International Russian Automation Conference (RusAutoCon)*, pp. 521-525, doi: 10.1109/RusAutoCon52004.2021.9537385.

[58] I. A. Tøndel, M. B. Line and M. G. Jaatun (2014), “Information security incident management: Current practice as reported in the literature. Computers & Security”, 45, 42–57. doi:10.1016/j.cose.2014.05.003

[59] T. Takemura (2008), "Formal Semantics and Verification of BPMN Transaction and Compensation", *2008 IEEE Asia-Pacific Services Computing Conference*, pp. 284-290, doi: 10.1109/APSCC.2008.208.

[60] C. Olariu and C. C. Aldea (2014), "Managing processes for virtual teams - a {BPM} approach", *Procedia - Social and Behavioral Sciences*, vol. 109, no. 0, pp. 380 - 384, *2nd World Conference on Business, Economics and Management*. doi: https://doi.org/10.1016/j.sbspro.2013.12.476

[61] O. Demirörs and F. Çelik (2011), "Process modeling methodologies for improvement and automation", *2011 IEEE International Conference on Quality and Reliability*, pp. 312-316, doi: 10.1109/ICQR.2011.6031732.

[62] L. B. G. Gomes, P. P. M. Farias, A. Bessa Albuquerque and A. Herden (2016), "Software measure based on BPMN activity points", *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1-6, doi: 10.1109/CISTI.2016.7521492.

[63] E. Hooper (2006), "Experimental Validation of An Intelligent Detection and Response Strategy for Complex Infrastructure Attacks and False Positives Using Firewalls," *Proceedings 40th Annual 2006*

International Carnahan Conference on Security Technology, pp. 252-256, doi: 10.1109/CCST.2006.313458.

[64] E. Hooper (2006), "An intelligent detection and response strategy to false positives and network attacks: operation of network quarantine channels and feedback methods to IDS," *Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU'06)*, pp. 6 pp.-21, doi: 10.1109/SECPERU.2006.5.

[65] A. Shiranzaei and R. Z. Khan (2015), "Internet protocol versions — A review," *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 397-401.

[66] R. Fielding and J. Reschke (2014), "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231". [online] https://httpwg.org/specs/rfc7231.html

[67] "IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries", in IEEE Std 610, vol., no., pp.1-217, 18 Jan. 1991, doi: 10.1109/IEEESTD.1991.106963.

[68] A. Gorecki (2020), "Cyber Breach Response That Actually Works (Organizational Approach to Managing Residual Risks) || Crafting an Incident Response Plan", 10.1002/9781119679349(), 143–194. doi:10.1002/9781119679349.ch4

[69] ISO/IEC 27035:2011 "Information technology - Security techniques - Information security incident management". 2011.

[70] A. Rezakhani, A. Hajebi and N. Mohammadi (2011), "Standardization of all Information Security Management Systems", in Int. Journal of Computer Applications. doi: 10.5120/2307-2592

[71] A. Al-Far, A. Qusef and S. Almajali (2018), "Measuring Impact Score on Confidentiality, Integrity, and Availability Using Code Metrics", *2018 International Arab Conference on Information Technology (ACIT)*, pp. 1-9, doi: 10.1109/ACIT.2018.8672678.

[72] P. Cichonski, T. Millar, T. Grance and K. Scarfone (2012), "NIST SP 800-61: Computer Security Incident Handling Guide".

[73] K. Stouffer, J. Falco and K. Scarfone (2013), "Guide to Industrial Control Systems (ICS) Security", *Special Publication (NIST SP)*, National Institute of Standards and Technology, Gaithersburg, MD, [online], https://doi.org/10.6028/NIST.SP.800-82r1.

[74] C. Dukes (2015), “Committee on National Security Systems (CNSS) Glossary: CNSSI 4009-2015”.

[75] J. C. Sancho, A. Caro, M. Ávila and A. Bravo (2020), “New approach for threat classification and security risk estimations based on security event management”, *Future Generation Computer Systems*, 113, 488–505, [online], https://doi.org/10.1016/j.future.2020.07.014.

[76] A. Sopan, M. Berninger, M. Mulakaluri and R. Katakam, "Building a Machine Learning Model for the SOC, by the Input from the SOC, and Analyzing it for the SOC", *2018 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1-8, doi: 10.1109/VIZSEC.2018.8709231.

[77] G. Suarez-Tangil et al. (2013), “Providing SIEM systems with self-adaptation”, Informat. Fusion, [online], http://dx.doi.org/10.1016/j.inffus.2013.04.009

[78] W. Meng, W. Li and L.-F Kwok (2015), “Design of intelligent KNN-based alarm filter using knowledge-based alert verification in intrusion detection”, *Security and Communication Networks*, 8(18), 3883–3895. doi:10.1002/sec.1307

[79] M. Chakraborty, M. Singh, V. E. Balas and I. Mukhopadhyay (2021), “The “Essence” of Network Security: An End-to-End Panorama”, *Lecture Notes in Networks and Systems*. doi:10.1007/978-981-15-9317-8

[80] B. Gupta, D. P. Agrawal and S. Yamaguchi (2016), “Threats Classification: State of the Art. In Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security”, *IGI Global: Hershey*, PA, USA, 2016; pp. 368–392.

[81] M. Beechey, K. G. Kyriakopoulos and S. Lambotharan (2021), “Evidential classification and feature selection for cyber-threat hunting”, *Knowledge-Based Systems*, 226, 107120. doi:10.1016/j.knosys.2021.10712

[82] H. I. Kure, S. Islam and H. Mouratidis (2022), “An integrated cyber security risk management framework and risk predication for the critical infrastructure protection”, *Neural Comput & Applic* 34, 15241–15271, [online], https://doi.org/10.1007/s00521-022-06959-2

[83] D. Bell (2004), “Explore the UML sequence diagram”, IBM from https://developer.ibm.com/articles/the-sequence-diagram/

[84] K. Kent and M. Souppaya (2006), “Guide to Computer Security Log Management. Recommendations of the National Institute of Standards and Technology (NIST)”. *Special Publication 800-92*.

[85] JOINT TASK FORCE (2020), “Security and Privacy Controls for Information Systems and Organizations”, *NIST Special Publication 800-53 Revision 5*, [online], https://doi.org/10.6028/NIST.SP.800-53r5.

[86] P. Mell and T. Grance (2011), “The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology (NIST)”, *Special Publication 800-145*

[87] CVSS v3.1 Specification Document - Revision 1, from https://www.first.org/cvss/

[88] S. Rizvi, N. McIntyre and J. Ryoo (2019), "Computing Security Scores for IoT Device Vulnerabilities", *2019 International Conference on Software Security and Assurance (ICSSA)*, pp. 52-59, doi: 10.1109/ICSSA48308.2019.00014.

[89] Five Automation Use Cases for Splunk SOAR (2021), from https://www.splunk.com/en_us/form/5-automation-use-cases-for-splunk-soar.html

[90] Y. A. Liu (2000), “Efficiency by Incrementalization: An Introduction. Higher-Order and Symbolic Computation”, 13, 289–313, [online], https://doi.org/10.1023/A:1026547031739

[91] P. Black, K. Scarfone and M. Souppaya (2009), “Cyber Security Metrics and Measures”, *John Wiley & Sons, Inc., Hoboken, NJ*, doi: 10.1002/9780470087923.hhs440

[92] D. J. Bodeau, R. D. Graubart, R. M. McQuaid and J. Woodill (2018), “Cyber resiliency metrics, measures of effectiveness, and scoring: Enabling systems engineers and program managers to select the most useful assessment methods”, *Mitre Corp Bedford Ma Bedford United States*.

[93] M. Francesca Carfora and A. Orlando (2019), "Quantile based risk measures in cyber security", *2019 International Conference on Cyber Situational Awareness, Data Analytics and Assessment* (Cyber SA), pp. 1-4, doi: 10.1109/CyberSA.2019.8899431.

[94] M. S. Makowski (2010), "On systematic modeling of switched capacitor DC-DC converters: Incremental graph approach", *2010 IEEE 12th Workshop on Control and Modeling for Power Electronics (COMPEL)*, pp. 1-6, doi: 10.1109/COMPEL.2010.5562408.

[95] S. Baldoni, F. Battisti, M. Carli and F. Pascucci (2021), "On the Use of Fibonacci Sequences for Detecting Injection Attacks in Cyber Physical Systems", in IEEE Access, vol. 9, pp. 41787-41798, doi: 10.1109/ACCESS.2021.3065228.

[96] S. Hashemi (2001), “Incremental Case-Based Reasoning for Classification”, In: Stroulia, E., Matwin, S. (eds) Advances in Artificial Intelligence. Canadian AI 2001. Lecture Notes in Computer Science, vol 2056. Springer, Berlin, Heidelberg, [online], https://doi.org/10.1007/3-540-45153-6_36

[97] “Cybersecurity as the Foundation of the Digital World” (2022), in Inncyber Innovation Hub, Digital transformation, Cyber & IoT, International Conference, 3rd Edition, Altice Labs HQ, Aveiro.

[98] J. Brocke, A. Hevner and A. Maedche (2020). “*Introduction to Design Science Research*”. Doi: 10.1007/978-3-030-46781-4_1.

[99] A. Hevner, et al. (2004). “*Design Science in Information Systems Research. Management Information Systems Quarterly*”.