ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

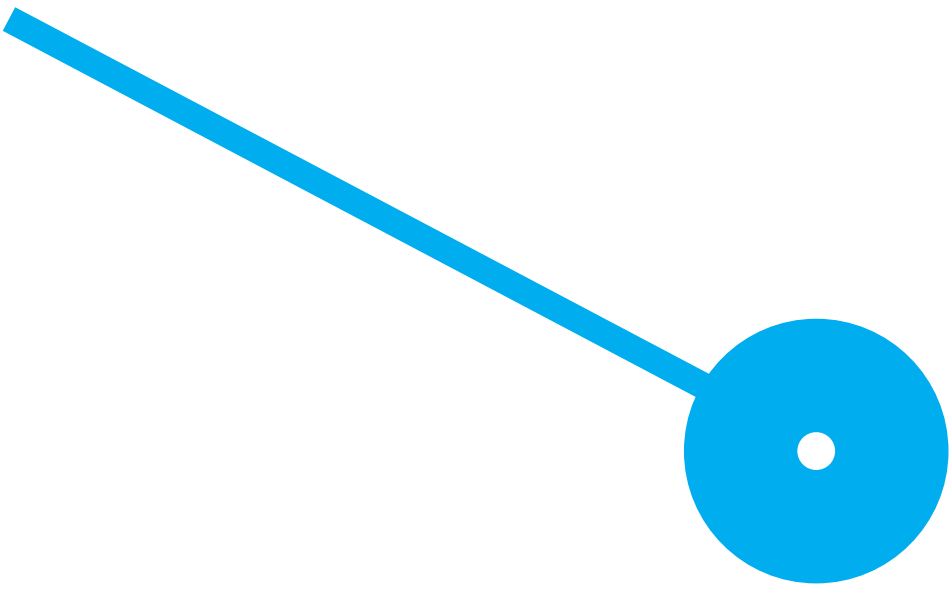
M MESTRADO
ENGENHARIA INFORMÁTICA

Deteção de intrusões na rede recorrendo a deteção de anomalias

André Manuel da Silva Macedo

João Paulo Magalhães

11/2023



ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

MESTRADO
ENGENHARIA INFORMÁTICA

Deteção de intrusões na rede recorrendo a deteção de anomalias

André Manuel da Silva Macedo
8170429

Orientador

Professor Doutor João Paulo Magalhães

Dissertação apresentada para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática pela Escola Superior de Tecnologia e Gestão do Instituto Politécnico do Porto.

11/2023

Declaração de Integridade

Eu, André Manuel da Silva Macedo, estudante nº 8170429, do Mestrado de Engenharia de Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico do Porto, declaro que não fiz plágio nem auto-plágio, pelo que o trabalho intitulado “Deteção de intrusões na rede recorrendo a deteção de anomalias” é original e da minha autoria, não tendo sido usado previamente para qualquer outro fim. Mais declaro que todas as fontes usadas estão citadas, no texto e na bibliografia final, segundo as regras de referenciação adotadas na instituição.

Agradecimentos

Gostaria de agradecer a todas as pessoas que tornaram este momento possível e que ajudaram e contribuíram para conseguir obter esta etapa, nomeadamente:

À minha família por todo o apoio e motivação para a conclusão.

Ao Professor Doutor João Paulo Magalhães pelo acompanhamento ao longo da realização de todo o projeto, pela disponibilidade e ajuda prestada, que me permitiu avançar perante as dificuldades encontradas.

A todos os meus amigos e colegas que estiveram ao meu lado ao longo destes anos.

E por fim, mas não menos importante, à Escola Superior de Tecnologia e Gestão pela formação.

Resumo

A proliferação e uso generalizado das novas tecnologias trouxeram consigo, inevitavelmente, um aumento da ocorrência de ataques informáticos. A problemática dos ataques informáticos assume relevância extrema no quotidiano de qualquer pessoa ou organização. Neste âmbito a segurança dos equipamentos informáticos que processam e armazenam a informação é cada vez mais um ponto central que requer soluções inovadoras e adequadas.

Nesta dissertação propõe-se uma solução que visa ajudar a mitigar o problema da segurança informática, tendo como objetivo auxiliar as organizações a identificar, detetar, gerir e responder às ameaças de segurança de forma assertiva e rápida. De forma mais específica, o projeto contempla a modelação do comportamento dos dispositivos ligados em rede usando *machine learning* e a deteção de forma automática desvios de comportamento que possam evidenciar o comprometimento, em termos de cibersegurança, desses dispositivos.

Palavras-chaves: Security Operations Center, Inteligência Artificial, Segurança, Incidente, Automatização, Recuperação

Abstract

The proliferation and widespread use of new technologies has inevitably brought with it an increase in the occurrence of cyber attacks. The problem of computer attacks is extremely relevant in the daily life of any person or organization. In this context, the security of computer equipment that processes and stores information is increasingly a central point that requires innovative and appropriate solutions.

This dissertation proposes a solution that aims to help mitigate the problem of computer security, with the objective of helping organizations to identify, detect, manage, and respond to security threats assertively and quickly. More specifically, the project includes modeling the behavior of networked devices using machine learning and the automatic detection of behavioral deviations that may indicate the compromise, in terms of cybersecurity, of these devices.

Keywords: Security Operations Center, Artificial Intelligence, Security, Incident, Automation, Recovery

Conteúdo

Lista de Figuras

Abreviaturas

APT - Advanced Persistent Threat

CRISP-DM - Cross-Industry Standard Process for Data Mining

DDoS - Distributed Denial of Service

DMARC - Domain-based Message Authentication

DNS - Domain Name System

ETL - Extract, transform, and load

HIPS - Host Intrusion Prevention System

IA - Inteligência artificial

IDS - Intrusion Detection Systems

IPS - Intrusion Prevention Systems

IP - Protocolo de Internet

IT - Information Tecnology

IoT - Internet of Things

LSTM - Long short-term memory

MAC - Media Access Control

MSSP - Managed security service providers

MX - Mail exchanger

NOC - Network Operations Centers

OSI - Open Systems Interconnection

RNN - Recurrent neural network

SIEM - Security information and event management

SOC - Security Operations Center

SPF - Sender Policy Framework

SQL - Structured query language

SSH - Secure Shell

SSL - Secure Sockets Layer

SVM - Support vector machines

TCP - Protocolo de Controlo de Transmissão

UDP - User Datagram Protocol

Capítulo 1

Introdução

Com o aumento da digitalização e o surgimento de novas tecnologias impulsionadoras do crescimento de negócio, para as empresas é imperativo que elaborem um plano estratégico abrangente a todas as áreas incluindo, por exemplo, a automatização de processos, *upgrades* de infraestrutura local, migração de aplicações para a *cloud* e mudanças na cultura corporativa.

Mas torna-se obrigatório também que como parte deste plano a área de cibersegurança seja um dos pilares da estratégia. Contudo, devem garantir que as suas equipas de cibersegurança têm todas as ferramentas que lhes garantam total visibilidade sobre as suas infraestruturas, com mecanismos eficazes de proteção, deteção e resposta para que em caso de existir um possível ataque possam também saber responder a outro tipo de questões, por exemplo: Qual foi o programa que despoletou o ataque? Qual a máquina infetada? Como o ataque aconteceu?

Além de mais, com o aumento do número de ataques informáticos ao longo dos anos, cada vez mais, faz sentido às empresas dedicar parte do seu orçamento anual à segurança informática. As perdas associadas a uma fuga de dados tem aumentado de forma consistente, assim como refere um relatório [36] dizendo que as perdas, em média, sofreram um aumento de $3.86m para $4.24m.

Hoje em dia, a deteção de ciberataques assenta maioritariamente em indicadores de compromisso (por exemplo, um domínio ou *digest*) ou em casos de uso definidos pelos engenheiros de segurança. Porém, esses casos só detetam ataques conhecidos, deixando escapar os restantes. Os atores maliciosos têm conhecimento dos processos atuais de cibersegurança

e aproveitam para explorar novas técnicas, ferramentas e procedimentos para contornar as barreiras de segurança implementadas. Algumas empresas já apresentam algumas soluções que envolvem *machine learning*, nomeadamente, e a mais conhecida, Darktrace.

Face ao exposto, continuam a ser necessários sistemas avançados, capazes de detectar ataques num fase inicial, melhorando, assim, o *Mean-Time-to-Detection* e *Mean-Time-to-Resolution*. A par do que o artigo [11] refere relativamente à melhoria dos tempo aplicados a AIOps (*i.e. handle the exponential growth of IT operations and the complexity of new technology through the application of artificial intelligence*), os mesmos benefícios podem e devem ser aplicados ao contexto da cibersegurança. A aplicação de inteligência artificial tem vindo a aumentar, com o objetivo de melhorar a capacidade de deteção e resolução autónoma dos ciberataques. Nesse âmbito, há ainda trabalho de fundo a realizar no sentido de melhorar as tecnologias e a capacidade das mesmas atuar, ajudando o ser humano nas suas decisões e acelerando o processo de deteção e mitigação de ciberataques. É de salientar que, tal como referido no relatório [36], as empresas com soluções de deteção de incidentes baseadas em Inteligência Artificial em conjunto com automatização da resposta a incidentes de segurança conseguem uma média de $2.90m (ao invés dos $4.24m) de perdas devido a maior rapidez na atuação.

1.1 Motivação e Objetivos

A solução proposta nesta dissertação visa a implementação de um sistema de deteção de intrusões tirando partido da *Machine Learning* recorrendo mais especificamente a algoritmos de deteção de anomalias, técnica esta que consiste em detetar anomalias nos dados, como por exemplo, aumento do volume de tráfego na rede. No caso deste projeto, passa por reconhecer anomalias no tráfego de rede e, assim, alertar o analista de segurança.

A solução pretende, em conjunto com uma equipa de monitorização de eventos de segurança SOC, mitigar e tratar incidentes de segurança da forma mais eficaz na medida em que irá suportar as atividades dos operadores, automatizando o processo de análise e deteção de desvios a um comportamento normal de rede, associando esses desvios a potenciais ciberataques.

O principal objetivo do trabalho é a criação de um sistema de deteção de intrusão baseado em *machine learning*. Na persecução deste objetivo, há um conjunto de sub-objetivos a atingir,

nomeadamente:

- Criar um cenário de rede e eventos através do qual seja possível simular um ambiente o mais realista possível;
- Ser capaz de recolher um grande número de eventos gerados na rede;
- Derivar um *dataset* representativo dos eventos que permita criar um padrão do comportamento da rede;
- Modelar o comportamento normal através de *machine learning*;
- Injetar atividade maliciosa (realística) e verificar se a mesma é detetada, com eficácia e em tempo útil, pela deteção de anomalias.

1.2 Metodologia

A metodologia adotada para esta dissertação compreende em três grandes etapas. Na primeira etapa é feita a revisão da literatura cinzenta, informação recolhida de *web sites* e livros, fontes essas devidamente referenciadas no final do presente documento. Foram consideradas também as informações colhidas de conversas, debates e sessões de *brainstorming* com colegas e orientador.

Numa segunda etapa é feita a recolha e análise de dados que envolve a utilização da *Machine Learning* com algoritmos de deteção de anomalias e ferramentas de recolha de eventos de rede, de processamento e também de armazenamento.

Numa terceira etapa é feita a análise de resultados com a execução da solução desenvolvida e posteriormente as conclusões obtidas.

1.3 Estrutura da dissertação

Este documento está estruturado da seguinte forma: no segundo capítulo é exposto o estado atual da arte com uma introdução aos diferentes tipos de *machine learning*, os vários métodos de captura de tráfego de rede, uma introdução ao *Spark Streaming*, tal como, ao *Kafka*, uma

pequena análise ao processo de exploração e análise de dados no ambiente de *machine learning*; no terceiro capítulo é feita uma descrição das tarefas realizadas e ferramentas utilizadas durante a elaboração da dissertação; o quarto capítulo detém um descrição técnica sobre o tema, como *snippets* de código e eventos, tal como, a descrição do ambiente; o último capítulo aborda algumas conclusões sobre o tema e o futuro do mesmo.

Capítulo 2

Estado da arte

Neste capítulo é apresentado o estado da arte relacionado com o projeto. São apresentadas as redes informáticas e alguns possíveis ataques, os tipos de *Machine Learning*, como podem ser feitas as capturas de tráfego, como é feito o processamento dos eventos, como esses eventos são consumidos e o processo de análise exploratória, casos de uso de *machine learning* na cibersegurança e, por fim, uma breve explicação sobre o funcionamento de um SOC e os seus principais desafios.

2.1 Rede Informática e Ciberataques

Uma rede informática é uma série de nós interligados que podem transmitir, receber e trocar dados. As redes informáticas funcionam utilizando um conjunto variável de hardware e software. Todas as redes comutadas por pacotes utilizam o Protocolo de Controlo de Transmissão/Protocolo de Internet (TCP/IP) para estabelecer um meio de comunicação padrão. Cada nó de uma rede tem um identificador único que é utilizado para indicar a origem ou destino da transmissão. Os identificadores incluem o endereço IP do nó ou o endereço MAC (Media Access Control). Os nós, que são utilizados para fins de encaminhamento, incluem switches e routers, servidores, computadores pessoais, telefones, impressoras em rede e outros dispositivos informáticos periféricos, bem como sensores. O modelo Open Systems Interconnection (OSI) define a forma como os dados são transferidos entre computadores.

As redes informáticas podem ser utilizadas para vários fins, como por exemplo:

- Partilha de ficheiros;
- Partilha de hardware, que permite aos utilizadores de uma rede partilhar dispositivos de hardware, tais como impressoras e sistemas de ficheiros;
- Modelo cliente-servidor, que permite que os dados sejam armazenados em servidores, onde os dispositivos do utilizador final – ou clientes – podem aceder a esses dados;
- Voz sobre IP (VoIP), o que permite aos utilizadores enviar dados de voz através de protocolos de Internet;
- Comunicação, que pode incluir vídeo, texto e voz.

Com a digitalização massiva das empresas, a criação de redes informáticas tornou-se imperativa. Esse crescimento leva a um aumento da preocupação de ciberataques nas mesmas. Infra-mencionado, são enumerados alguns dos ataques mais comuns:

- **Ataque de Força Bruta** - Quando um atacante tenta adivinhar a palavra-passe tentando várias diferentes.
- ***Malware*** - Software malicioso usado por atores maliciosos. Incluí vírus, *worms*, entre outros.
- ***Ransomware*** - É um tipo de *malware* que cifra ficheiros e pede resgate monetário para decifrar.
- **Exfiltração de dados** - Quando um atacante, ou por exemplo, um colaborador descontente transfere dados confidenciais para um local da sua posse.
- **Negação de Serviço** - Quando um atacante causa interferência no sistema, ou por alguma vulnerabilidade que afete a disponibilidade do serviço ou com envio de um enorme volume de tráfego.
- ***Botnets*** - Uma botnet é uma rede de computadores infetados com software malicioso. O atacante usa a máquina da vítima para executar ataques de DDoS ou outras atividades maliciosas (*Command and Control*, Exfiltração de dados, etc).
- **Enumeração de Rede (*Scanning*)** - Quando um atacante enumera os ativos de rede e portos expostos, com o intuito, por exemplo, de encontrar vulnerabilidades em serviços.

2.2 Machine Learning e Cibersegurança

Machine Learning [21] é um subconjunto da Inteligência Artificial, que permite aprender automaticamente com os dados, melhorar o desempenho de experiências passadas, e fazer previsões. A aprendizagem contém um conjunto de algoritmos que funcionam com uma enorme quantidade de dados. Os dados são alimentados por estes algoritmos para os treinar, e com base na formação, constroem o modelo e executam uma tarefa específica.

2.2.1 Tipos de Machine Learning

Com base nos métodos e na forma de aprendizagem, o *Machine Learning* está dividido principalmente em quatro tipos, que são:

- **Aprendizagem supervisionada**

  A aprendizagem supervisionada significa que treinamos os modelos utilizando o conjunto de dados "classificados", e com base na formação, o modelo prevê a solução. Aqui, os dados classificados especificam que algumas das entradas já estão mapeadas para a saída. Mais precisamente, podemos dizer: primeiro, treinamos o modelo com a entrada e a saída correspondente, e depois pedimos ao modelo que preveja a saída utilizando o conjunto de dados de teste.

- **Aprendizagem não supervisionada**

  A aprendizagem não supervisionada é diferente da técnica de aprendizagem supervisionada, como o seu nome sugere, não há necessidade de supervisão. Significa que, na aprendizagem não supervisionada, o modelo é treinado utilizando o conjunto de dados sem supervisão, e prevê a saída sem qualquer supervisão.

  Na aprendizagem não supervisionada, os modelos são treinados com os dados que não são classificados nem rotulados, e o modelo atua sobre esses dados sem qualquer supervisão.

  O principal objetivo do algoritmo de aprendizagem não supervisionada é o de agrupar ou categorizar o conjunto de dados não ordenado de acordo com as semelhanças, padrões, e diferenças. O modelo é instruído a encontrar os padrões ocultos do conjunto de dados de entrada.

- **Aprendizagem semi-supervisionada**

  Aprendizagem Semi-Supervisionada é um tipo que se situa entre a Aprendizagem Supervisionada e a Aprendizagem Não Supervisionada. Representa o terreno intermédio entre os algoritmos de aprendizagem supervisionada (com dados de treino classificados) e não supervisionada (sem dados de treino classificados) e utiliza a combinação de conjuntos de dados classificados e não classificados durante o período de treino.

  Embora a aprendizagem semi-supervisionada seja o meio termo entre a aprendizagem supervisionada e não supervisionada e funcione com base nos dados que consistem em algumas classificações, constitui principalmente dados não classificados.

  Para superar os inconvenientes da aprendizagem supervisionada e dos algoritmos não supervisionados, é introduzido o conceito de aprendizagem semi-supervisionada. O principal objetivo da aprendizagem semi-supervisionada é utilizar eficazmente todos os dados disponíveis, em vez de apenas os dados classificados, como na aprendizagem supervisionada. Inicialmente, os dados semelhantes são agrupados juntamente com um algoritmo de aprendizagem não supervisionada e, além disso, ajudam a classificar os dados não classificados em dados classificados. Fazendo uma analogia, a aprendizagem supervisionada é quando um estudante está sob a supervisão de um instrutor em casa e na faculdade. Além disso, se esse estudante estiver a analisar por si próprio o mesmo conceito sem qualquer ajuda do instrutor, fica sob aprendizagem não supervisionada. Sob aprendizagem semi-supervisionada, o aluno tem de se rever a si próprio após analisar o mesmo conceito sob a orientação de um instrutor na faculdade.

- **Aprendizagem por reforço**

  O reforço da aprendizagem funciona num processo baseado no *feedback*, no qual um agente de inteligência artificial (um componente de software) explora automaticamente dados mais próximos de si através de *hitting & trail*, tomando medidas, aprendendo com as experiências, e melhorando o seu desempenho. O agente é recompensado por cada boa ação e punido por cada má ação; daí que o objetivo do agente de reforço da aprendizagem seja o de maximizar as recompensas.

  Na aprendizagem do reforço, não há dados classificados como aprendizagem supervisionada, e os agentes aprendem apenas com as suas experiências.

  O processo de aprendizagem de reforço é semelhante a um ser humano; por exemplo, uma criança aprende várias coisas através de experiências na sua vida quotidiana.

Com o *Machine Learning* é possível desenvolver aplicações de software de visão computacional, reconhecimento de voz, processamento de linguagem natural, controlo de robôs, e outras aplicações [22]. Grandes empresas, como a Amazon, Google e Facebook utilizam *Machine Learning* para melhorar a experiência do utilizador final, como sugerir compras e promover ofertas especiais. A *Machine Learning* tem sido explorada por uma série de indústrias com problemas de dados intensivos como a cibersegurança.

Os dispositivos de segurança de última geração tipicamente fornecem sistemas proprietários com alertas e eventos baseados em assinaturas de monitorização [29]. Estes eventos, normalmente, identificam atividades maliciosas e geram alertas na rede através de *Intrusion Prevention Systems*/*Intrusion Detection Systems* (IPS/IDS), dispositivos de proteção de rede (firewalls) ou proteção do ponto final (Anti-Vírus, *Host Intrusion Prevention System* (HIPS), Host Firewall, etc.). Dispositivos de segurança que confiam apenas na deteção de assinaturas sofrem com duas limitações: 1) ataques novos ou desconhecidos passam despercebidos e 2) a deteção/proteção é limitada e normalmente não é partilhada [29].

A maioria dos proprietários de sistemas estão principalmente preocupados com sistemas e redes de defesa contra ataques conhecidos [29]. No entanto, as organizações avançadas e maduras estão igualmente preocupadas em identificar ataques de dia zero e *Advanced Persistent Threat* (APT) [7]. Estes tipos de ataques são mais difíceis de detetar, pois são ataques "baixos e lentos" que facilmente se perdem no "ruído" de milhões de outros eventos.

A fim de fazer face à explosão dos eventos de segurança detetados entre organizações, estratégias de gestão de eventos de segurança devem ser desenvolvidas numa abordagem adaptativa a fim de responder a novos e únicos ataques, enquanto continua a abordar o conhecer os ataques [6]. Novas estratégias de gestão de eventos precisam de tornar-se conscientes da situação e aprender ao longo do tempo [6]. Hoje em dia, os Centros de Operação de Segurança (SOC) e Centros de Operação de Rede (NOC) reagem a eventos detetados desenvolvendo o contexto dos ataques. Os analistas de ciberdefesa montam alertas/incidentes com informações de eventos forenses de ataque com informações de fontes como captura de pacotes, dados de fluxo de rede e registos do dispositivo. Automatizar a gestão de eventos de segurança exige elementos fundamentais tais como contextualmente e informação de deteção semanticamente rica que reúne ontologias e lógica que podem ser partilhadas através de uma infraestrutura de segurança [6]. Infelizmente, os analistas NOC/SOC são necessários para montar, correlacionar e explorar eventos de segurança manualmente através de dispositivos

heterogéneos que fornecem diversos informações sobre eventos [5]. Um empresa recente/moderna tipicamente faz uso de várias soluções de segurança de deteção e proteção de forma independente. As novas soluções devem reunir estes sistemas e partilhar informação para promover a aprendizagem adaptativa para os eventos de segurança [5].

2.2.2 Casos de Uso de Machine Learning na Segurança

É possível classificar os casos de uso de *machine learning* em duas categorias [4]: reconhecimento de padrões e deteção de anomalias. A linha que distingue o reconhecimento de padrões e a deteção de anomalias é por vezes imprecisa, mas cada tarefa tem um objetivo claramente distinto. No reconhecimento de padrões, o objetivo é descobrir características explícitas ou latentes escondidas nos dados. Estas características, quando destiladas em conjuntos de características, podem ser utilizadas para ensinar um algoritmo a reconhecer outras formas dos dados que exibem o mesmo conjunto de características. A deteção de anomalias aproxima-se da descoberta de conhecimento a partir do outro lado da mesma moeda. Em vez de aprender padrões específicos que existem dentro de certos subconjuntos de dados, o objetivo é estabelecer uma noção de normalidade que descreve a maioria (mais de 95%) de um dado conjunto de dados. Posteriormente, os desvios desta normalidade de qualquer tipo serão detetados como anomalias.

É comum pensar erroneamente na deteção de anomalias como o processo de reconhecer um conjunto de padrões normais e diferenciá-lo de um conjunto de padrões anormais. Os padrões extraídos através do reconhecimento de padrões devem ser rigorosamente derivados do dados observados utilizados para treinar o algoritmo. Por outro lado, na deteção de anomalias pode haver um número infinito de padrões anómalos que se encaixam na composição de um *outlier*, mesmo aqueles derivados de dados hipotéticos que não existem nos conjuntos de dados de treino ou de teste.

A deteção de spam é talvez o exemplo clássico de reconhecimento de padrões porque o spam tem tipicamente um conjunto de características amplamente previsíveis, e um algoritmo pode ser treinado para reconhecer essas características como um padrão pelo qual se podem classificar os emails. No entanto, também é possível pensar na deteção de spam como um problema de deteção de anomalias. Se for possível derivar um conjunto de características que descreva suficientemente bem o tráfego normal para tratar desvios significativos desta

normalidade como spam. Na realidade, contudo, a deteção de spam pode não ser adequada ao paradigma de deteção de anomalias, porque não é difícil convencer-se de que na maioria dos contextos é mais fácil encontrar semelhanças entre mensagens de spam do que dentro do vasto conjunto do tráfego normal.

A deteção de *malware* e a deteção de *botnet* são outras aplicações que se enquadram claramente na categoria de reconhecimento de padrões, onde a *machine learning* se torna especialmente útil quando os atacantes empregam o polimorfismo para evitar a deteção.

Para autenticação do utilizador e análise do comportamento, a delimitação entre o reconhecimento de padrões e a deteção de anomalias torna-se menos clara. Para casos em que o modelo de ameaça é claramente conhecido, poderá ser mais adequado abordar o problema através da lente de reconhecimento de padrão. Noutros casos, a deteção de anomalias pode ser a resposta. Em muitos casos, um sistema pode fazer uso de ambas as abordagens para conseguir uma melhor cobertura. A deteção de anomalias na rede é um exemplo clássico de deteção de anomalias porque a maioria do tráfego de rede segue protocolos rigorosos e o comportamento normal corresponde a um conjunto de padrões na forma ou sequência. Qualquer atividade maliciosa da rede que não consiga disfarçar bem, imitando o tráfego normal, será apanhada por algoritmos de deteção de anomalias. Outros problemas de deteção relacionados com a rede, tais como a deteção de URLs maliciosos, também podem ser abordados do ângulo da deteção de anomalias.

O controlo de acesso refere-se a qualquer conjunto de políticas que regem a capacidade dos utilizadores do sistema para acederem a determinadas informações. Frequentemente utilizadas para proteger informação sensível de exposição desnecessária, as políticas de controlo de acesso são frequentemente a primeira linha de defesa contra violações e roubo de informação. A *machine learning* tem gradualmente encontrado o seu caminho para soluções de controlo de acesso devido às dores sentidas pelos utilizadores do sistema à mercê de políticas rígidas e implacáveis de controlo de acesso. Através de uma combinação de aprendizagem não supervisionada e deteção de anomalias, tais sistemas podem inferir padrões de acesso à informação para certos utilizadores ou papéis numa organização e envolver-se em ações de retaliação quando é detetado um padrão não convencional.

Imagine, por exemplo, um sistema de armazenamento de registos de pacientes de um hospital, onde enfermeiros e técnicos médicos precisam frequentemente de aceder a dados individuais de pacientes mas não precisam necessariamente de fazer correlações entre pacientes.

Os médicos, por outro lado, consultam e agregam frequentemente os registos médicos de múltiplos pacientes para procurarem semelhanças de casos e histórias de diagnóstico. Não se quer impedir que enfermeiros e técnicos médicos consultem múltiplos registos de pacientes porque pode haver casos raros que justifiquem tais ações. Um sistema rigoroso de controlo de acesso baseado em regras não seria capaz de proporcionar a flexibilidade e adaptabilidade que os sistemas de *machine learning* podem proporcionar.

2.2.3 Limitações do Machine Learning na Segurança

A percepção de que *machine learning* dará sempre bons resultados em diferentes casos de utilização é categoricamente falsa [4]. Em cenários do mundo real existem normalmente fatores a otimizar para além da *precision*, *recall*, ou *accurancy*.

Como exemplo, a explicabilidade dos resultados de classificação pode ser mais importante em algumas aplicações do que noutras. Pode ser consideravelmente mais difícil extrair as razões de uma decisão tomada por um sistema de *machine learning* em comparação com uma regra simples. Alguns sistemas de *machine learning* podem também ser significativamente mais intensivos em recursos do que outras alternativas, o que pode ser um obstáculo para execução em ambientes constrangidos, tais como sistemas embutidos/integrados.

Não existe nenhum algoritmo de *machine learning* que funcione bem em todos os espaços problemáticos. Os diferentes algoritmos variam enormemente na sua adequação a diferentes aplicações e diferentes conjuntos de dados. Embora os métodos de *machine learning* contribuam para a noção de inteligência artificial, as suas capacidades ainda só podem ser comparadas à inteligência humana ao longo de determinadas dimensões.

O processo de tomada de decisão humano é informado por um vasto corpo de contexto extraído do conhecimento cultural e experimental. Este processo é muito difícil de simular para os sistemas de aprendizagem automática. Tomemos como exemplo palavras *blacklist* para a filtragem de spam. Quando uma pessoa avalia o conteúdo de um e-mail para determinar se é fidedigno ou *spam*, o processo de tomada de decisão nunca é tão simples como procurar a existência de certas palavras. O contexto em que uma palavra da *blacklist* está a ser utilizada pode resultar na sua inclusão razoável em correio eletrónico não spam. Além disso, os *spammers* podem utilizar sinónimos de palavras incluídas na *blacklist* em futuros e-mails para transmitir o mesmo significado, mas uma *blacklist* simplista não se adaptaria adequadamente.

O sistema simplesmente não tem o contexto que um humano tem - não sabe que relevância uma palavra em particular tem para o leitor. A atualização contínua da *blacklist* com novas palavras suspeitas é um processo trabalhoso, e de forma alguma garante uma cobertura perfeita.

Mesmo que o seu modelo de *machine learning* possa funcionar perfeitamente num conjunto de treino, poderá descobrir que tem um mau desempenho num conjunto de testes. Uma razão comum para este problema é que o modelo tem os seus limites de classificação *overfitted* aos dados de formação, características de aprendizagem do conjunto de dados que não se generalizam bem através de outros conjuntos de dados nunca antes vistos. Por exemplo, o seu filtro de spam pode aprender com um conjunto de treino que todas as mensagens de *e-mail* contendo as palavras "herança" e "Nigéria" podem receber imediatamente uma pontuação elevada de suspeito, mas não sabe da discussão legítima da cadeia de *e-mails* sobre heranças e seguros agrícolas nigerianos.

Com todas estas limitações em mente, devemos abordar o *machine learning* com partes iguais de entusiasmo e cautela, lembrando que nem tudo pode ser melhorado instantaneamente com a Inteligência Artificial.

2.2.4 Algoritmos de Deteção de Anomalias

Desde já, existe uma distinção importante entre deteção de *outliers* (*outlier detection*) e deteção de "novidades" (*novelty detection*) [4]. A tarefa de deteção de uma novidade envolve aprender uma representação de dados "regulares" utilizando dados que não contem quaisquer *outliers*, enquanto a tarefa de deteção de *outliers* envolve aprender com dados que contem tanto dados regulares como *outliers*. Tanto a deteção de novidade como a deteção de *outliers* são formas de deteção de anomalias.

Depois de ter desenvolvido um conjunto de *features* de um conjunto de eventos para gerar um intervalo temporal, é altura de utilizar algoritmos para gerar impressões a partir desses dados. A deteção de anomalias tem tido um estudo académico, mas como todas as outras áreas de aplicação na análise de dados, não há algoritmo *one-size-fits-all* que funcione para todos os tipos de séries temporais. Assim, deve ser esperado que o processo de encontrar o melhor algoritmo para a aplicação específica seja uma viagem de exploração e experimentação. Antes de selecionar um algoritmo, é importante pensar sobre a natureza e qualidade da fonte de dados. Se os dados estão significativamente poluídos por anomalias, afetará a metodologia

de deteção.

Como definido anteriormente, se os dados não contiverem anomalias (ou se tiverem anomalias assinaladas para que as possamos remover), referimo-nos à deteção de novidades . Caso contrário, referimo-nos à tarefa como deteção de *outliers*. Na deteção de *outliers*, o algoritmo escolhido precisa de ser resistente a pequenos desvios que prejudicarão a qualidade do modelo treinado. Muitas vezes, determinar qual a abordagem a adotar é uma decisão não trivial. Limpar um conjunto de dados para remover anomalias é trabalhoso e, por vezes, completamente impossível.

Nesta análise, é sintetizado uma grande variedade de métodos de deteção de anomalias a partir da literatura e da indústria num esquema de categorização baseado nos princípios fundamentais de cada algoritmo. No esquema, cada categoria contém um ou mais algoritmos específicos, e cada algoritmo pertence a um máximo de uma categoria. As categorias são as seguintes [4]:

- *Forecasting* (aprendizagem supervisionada)
- Métricas estatísticas
- Testes *Goodness-of-fit*
- Aprendizagem não supervisionada
- Métodos baseados na densidade

2.2.4.1 *Forecasting* (aprendizagem supervisionada)

Forecasting é uma forma intuitiva de realizar a deteção de anomalias: aprendemos com dados anteriores e fazemos uma previsão sobre o futuro. Podemos considerar qualquer desvio substancial entre as previsões e as observações como anómalo. Tomando o tempo como exemplo, se não tivesse chovido durante semanas, e não houvesse nenhum sinal visível de chuva próxima, a previsão previria uma baixa probabilidade de chuva nos próximos dias. Se chovesse nos próximos dias, seria um desvio em relação à previsão.

Esta classe de algoritmos de deteção de anomalias utiliza dados passados para prever dados atuais, e mede quão diferentes são os dados atualmente observados da previsão.

- **ARIMA**

  A utilização da família de funções ARIMA (*autoregressive integrated moving average*) [31] é uma forma poderosa e flexível de realizar previsões em séries temporais. Os modelos auto-regressivos são uma classe de modelos estatísticos que têm resultados que dependem linearmente dos seus próprios valores anteriores, em combinação com um fator estocástico.

  Ao escolher um modelo de *forecasting* apropriado, é importante procurar sempre os seus dados para identificar tendências, se a sazonalidade é uma característica forte da série, considera-se modelos com ajustes sazonais tais como SARIMA e métodos HoltWinters sazonais. Os métodos de *forecasting* aprendem as características das séries temporais, analisando pontos anteriores e fazendo previsões sobre o futuro. Ao explorar os dados, uma métrica útil a aprender é a auto-correlação, que é a correlação entre a série e ela própria num ponto do tempo anterior. Uma boa previsão da série pode ser pensada como os pontos futuros tendo uma elevada auto-correlação com os pontos anteriores.

- **Artificial Neural Networks**

  Outra forma de realizar *forecasting* [12] [13] sobre dados de séries temporais é utilizar redes neuronais artificiais. Em particular, as redes de *long short-term memory* (LSTM) são adequadas para esta aplicação. As LSTM são uma variante das redes neuronais recorrentes (RNNS) que são desenhadas exclusivamente para aprender tendências e padrões na inserção de séries temporais para efeitos de classificação ou previsão.

De notar que o *forecasting* não funciona bem normalmente para a deteção de *outliers* [4]; isto é, se os dados de treino do modelo incluírem anomalias que não possa ser filtradas facilmente, o modelo adequar-se-á tanto aos *outliers* como aos *inliers*, o que dificultará a deteção de *outliers* futuros. É bem adequado para a deteção de novidades, o que significa que as anomalias só estão contidas nos dados de teste e não nos dados de treino. Se a série temporal for altamente irregular e não seguir qualquer tendência observável, ou se a amplitude das flutuações variar muito, não é provável que a previsão tenha um bom desempenho. A previsão funciona melhor em séries uni-dimensionais de métricas de valor real, por isso, se o seu conjunto de dados contém vetores de características multi-dimensionais ou variáveis categóricas, será melhor utilizar outro método de deteção de anomalias.

2.2.4.2 Métricas estatísticas

Pode ser utilizado testes estatísticos para determinar se um novo evento é semelhante aos dados anteriormente vistos. Podem também ser utilizadas médias variáveis de dados de séries temporais como uma métrica adaptativa que indica a conformidade dos pontos de dados com uma tendência a longo prazo. Especificamente, a média variável é o ponto de referência para comparações estatísticas, e os desvios significativos em relação à média serão considerados anomalias.

As métricas estatísticas é uma forma muito simples de realizar a deteção de anomalias, e pode não ser considerada por muitos como sendo uma técnica de *machine learning*. No entanto, verifica muitas das características de um detetor de anomalias: os alertas de anomalias são reprodutíveis e fáceis de explicar, os algoritmos adaptam-se às tendências de mudança dos dados, pode ser muito performante devido à sua simplicidade, e é relativamente fácil de afinar e manter. Devido a estas propriedades, a comparação métrica estatística pode ser uma escolha ótima para alguns cenários em que as medidas estatísticas podem funcionar com precisão, ou para os quais um nível de precisão inferior pode ser aceite. Devido à sua simplicidade, as métricas estatísticas têm capacidades de aprendizagem limitadas, e muitas vezes desempenham pior do que algoritmos de *machine learning* mais poderosos.

- **Median absolute deviation**

  O desvio padrão [26] de uma série de dados é frequentemente utilizado no *threshold* adaptativo para detetar anomalias. Por exemplo, uma definição razoável de anomalia pode ser qualquer ponto que se encontre a mais de dois desvios-padrão da média. Assim, para um conjunto de dados padrão normal com uma média de 0 e desvio padrão de 1, quaisquer pontos de dados situados entre -2 e 2 serão considerados regulares, enquanto que um ponto de dados com o valor 2,5 seria considerado anómalo. Este algoritmo funciona se os seus dados estiverem perfeitamente limpos, mas se os dados contiverem valores anómalos, a média calculada e os desvios padrão serão desequilibrados.

- **Grubbs outlier test**

  O teste *Grubbs* [33] é um algoritmo que encontra um único *outlier* num conjunto de dados normalmente distribuído, considerando o valor mínimo ou máximo atual da série. O algoritmo é aplicado iterativamente, removendo o *outlier* previamente detetado entre

cada iteração. Uma forma comum de utilizar o teste de *Grubbs* para detetar anomalias é calcular a estatística do teste de *Grubbs* e o valor crítico de *Grubbs*, e marcar o ponto como *outlier* se a estatística do teste for maior do que o valor crítico. Esta abordagem só é adequada para distribuições normais, e pode ser ineficiente porque só deteta uma anomalia em cada iteração.

2.2.4.3 Testes *Goodness-of-fit*

A goodness-of-fit [4] [44] é uma técnica estatística e é aplicada para medir "de que maneira os dados reais (observados) se encaixam num modelo de machine learning". Resume a diferença entre dados reais (observados) e dados esperados no contexto de um modelo estatístico ou de machine learning. Os testes de Goodness-of-fit são testes estatísticos para determinar se um conjunto de valores observados reais corresponde aos previstos pelo modelo.

Alguns dos testes mais comuns para quantificar o quanto similar duas distribuições contínuas são:

- Chi-squared test
- Kolmogorov-Smirnov test
- Cramér-von Mises criterion

Estes testes são maioritariamente válidos apenas para conjuntos de dados com uma dimensão, o que, pode ser bastante limitador a sua utilidade.

2.2.4.4 Aprendizagem não supervisionada

A aprendizagem não supervisionada são uma classe de soluções para o problema de deteção de anomalias que surgem de modificações de modelos típicos de aprendizagem supervisionada. Os classificadores da aprendizagem supervisionada são utilizados de forma genérica para resolver problemas que envolvem duas ou mais classes. No entanto, quando utilizados para a deteção de anomalias, as modificações destes algoritmos conferem-lhes características de aprendizagem não supervisionada.

- **One-class support vector machines**

  Podemos utilizar one-class SVM [1] para detetar anomalias, ajustando o SVM com dados pertencentes apenas a uma única classe. Estes dados (que se assume não conterem anomalias) são utilizados para treinar o modelo, criando um limite de decisão que pode ser utilizado para classificar futuros pontos de dados de entrada. Não existe um mecanismo de robustez incorporado em implementações SVM de uma classe padrão, o que significa que a formação do modelo é menos resiliente a *outliers* no conjunto de dados. Como tal, este método é mais adequado para a deteção de novidades do que a deteção de *outliers*; ou seja, os dados de treino devem, idealmente, ser cuidadosamente limpos e não conter anomalias.

- **Isolation Forest**

  Isolation Forest [28] é semelhante em princípio à *Random Forest* e é construída com base em árvores de decisão. A *Isolation Forest*, no entanto, identifica anomalias em vez de traçar o padrão dos dados normais. *Isolation Forest* isola observações selecionando aleatoriamente uma *feature* e depois selecionando aleatoriamente um valor dividido entre os valores máximo e mínimo dessa *feature* selecionada. Esta divisão depende de quanto tempo leva a separar os pontos.

  A partição aleatória produz caminhos visivelmente mais curtos para as anomalias. Quando uma floresta de árvores aleatórias produz coletivamente percursos mais curtos para determinadas amostras, é muito provável que sejam anomalias.

2.2.4.5 Métodos baseados na densidade

Os métodos de *clustering* como o algoritmo *k-means* são conhecidos pela sua utilização em classificação e regressão não supervisionada. Podemos utilizar métodos semelhantes baseados na densidade no contexto da deteção de anomalias para identificar *outliers*. Os métodos baseados na densidade são bem adequados para conjuntos de dados de alta densidade, que podem ser difíceis de lidar com a utilização de outras classes de métodos de deteção de anomalias. Vários métodos diferentes baseados na densidade foram adaptados para utilização na deteção de anomalias. A ideia principal subjacente a todos eles é formar uma representação em *cluster* dos dados de treino, sob a hipótese de os *outliers* ou anomalias se localizarem em regiões de baixa densidade desta representação em *cluster*. Esta abordagem tem a propriedade conveniente de ser resiliente a *outliers* nos dados de formação porque tais casos serão

provavelmente encontrados também em regiões de baixa densidade.

- **Local outlier factor**

  Local Outlier Factor [37] classifica as anomalias utilizando densidade local em torno de uma amostra. A densidade local de um dado ponto refere-se à concentração de outros pontos na região circundante imediata, onde a dimensão desta região pode ser definida por um limiar de distância fixo ou pelos n pontos vizinhos mais próximos. *Local Outlier Factor* mede o isolamento de um único ponto de dados em relação aos seus n vizinhos mais próximos. Os pontos de dados com uma densidade local significativamente inferior à dos seus n vizinhos mais próximos são considerados como anomalias.

2.3 Ciência de Dados

A ciência dos dados [18] combina matemática e estatística, programação especializada, análise avançada, inteligência artificial (IA), e *machine learning* com conhecimentos específicos da matéria, para desvendar conhecimentos acionáveis escondidos nos dados de uma organização. Estes conhecimentos podem ser utilizados para orientar a tomada de decisões e o planeamento estratégico.

O ciclo de vida da ciência dos dados envolve vários papéis, ferramentas, e processos, o que permite aos analistas obterem conhecimentos acionáveis. Tipicamente, um projeto de ciência de dados passa pelas seguintes fases:

- **Ingestão de dados:** O ciclo de vida começa com a recolha de dados - tanto dados brutos estruturados como não estruturados de todas as fontes relevantes, utilizando uma variedade de métodos. Estes métodos podem incluir entrada manual, *web scraping*, e *streaming* de dados em tempo real a partir de sistemas e dispositivos. As fontes de dados podem incluir dados estruturados, tais como dados de clientes, juntamente com dados não estruturados como ficheiros de registo, vídeo, áudio, imagens, a *Internet of Things* (IoT), redes sociais, e muito mais.
- **Armazenamento de dados e processamento de dados:** Uma vez que os dados podem ter diferentes formatos e estruturas, é necessário considerar diferentes sistemas de armazenamento com base no tipo de dados que precisam de ser capturados. As

equipas de gestão de dados ajudam a estabelecer padrões em torno do armazenamento e estrutura de dados, o que facilita os fluxos de trabalho em torno da análise, *machine learning* e modelos de aprendizagem profunda. Esta fase inclui a limpeza dos dados, a deduplicação, a transformação e a combinação dos dados utilizando trabalhos ETL (extrair, transformar, carregar) ou outras tecnologias de integração de dados. Esta preparação de dados é essencial para promover a qualidade dos dados antes de carregar num *data warehouse*, *datalake*, ou outro repositório.

- **Análise de dados:** Os cientistas de dados conduzem uma análise exploratória de dados para examinar tendências, padrões, gamas, e distribuições de valores dentro dos dados. Também permite aos analistas determinar a relevância dos dados para utilização no âmbito dos esforços de modelação para análise preditiva, *machine learning*, e/ou aprendizagem profunda. Dependendo da precisão de um modelo, as organizações podem tornar-se dependentes destes conhecimentos para a tomada de decisões empresariais, o que lhes permite conduzir a uma maior escalabilidade.
- **Comunicar:** Finalmente, os conhecimentos são apresentados como relatórios e outras visualizações de dados que tornam os conhecimentos - e o seu impacto nos negócios - mais fáceis de compreender pelos analistas de negócios e outros decisores. Uma linguagem de programação de ciência de dados como o R ou Python inclui componentes para gerar visualizações; alternadamente, os cientistas de dados podem utilizar ferramentas de visualização dedicadas.

2.3.1 Spark

O Apache Spark [40] é um motor de análise unificado para o processamento de dados em grande escala. Fornece APIs de alto nível em Java, Scala, Python e R, e um motor optimizado que suporta gráficos de execução geral. Também suporta um rico conjunto de ferramentas de nível superior incluindo Spark SQL para SQL e processamento de dados estruturados, pandas API on Spark para cargas de trabalho de pandas, MLlib para modelos de *machine learning*, GraphX para processamento de gráficos, e Structured Streaming para computação incremental e processamento de fluxos.

2.3.1.1 Structured Streaming

O *Structured Streaming* [41] é escalável e tolerante a falhas, construído sobre o *Spark SQL Engine*. A forma de expressar a computação em *streaming* é mesma do que em dados estáticos. O *Spark SQL Engine* trata de o executar de forma incremental e contínua e de atualizar o resultado final à medida que os dados de *streaming* continuam a chegar. Ainda é disponibilizado o *Dataset/DataFrame API* em *Scala*, *Java*, *Python* ou *R* para executar agregações de *streaming*, janelas de tempo de evento, uniões de *stream-to-batch*, etc. O cálculo é executado no mesmo *Spark SQL Engine* otimizado. Finalmente, o sistema assegura garantias de tolerância a falhas de ponta a ponta através de *checkpointing* e *Write-Ahead Logs*. Em suma, o *Structured Streaming* proporciona um processamento de fluxo rápido, escalável, tolerante a falhas, de ponta a ponta exatamente no mesmo instante.

Internamente, por defeito, as consultas de *Structured Streaming* são processadas utilizando um motor de processamento de micro-lotes, que processa fluxos de dados como uma série de pequenos trabalhos de lote, atingindo assim latências de ponta a ponta tão baixas como 100 milissegundos e garantias de tolerância exata a falhas.

2.3.2 CRISP-DM - Framework para mineração de dados

CRISP-DM, que significa *Cross-Industry Standard Process for Data Mining* [17] [2] é uma forma testada pela indústria para orientar os seus esforços de mineração de dados.

Como metodologia, inclui descrições das fases típicas de um projeto, as tarefas envolvidas em cada fase, e uma explicação das relações entre estas tarefas. Como modelo de processo, o CRISP-DM fornece uma visão geral do ciclo de vida da mineração de dados.

O ciclo de vida consiste em seis fases com setas indicando as dependências mais importantes e frequentes entre as fases (consultar Figura 1). A sequência das fases não é rigorosa. De facto, a maioria dos projetos deslocam-se de um lado para o outro entre fases, conforme necessário.

O modelo CRISP-DM é flexível e pode ser facilmente personalizado. Por exemplo, se a sua organização tiver como objetivo de detetar o branqueamento de capitais, é provável que faça uma filtragem através de grandes quantidades de dados sem um objetivo específico de

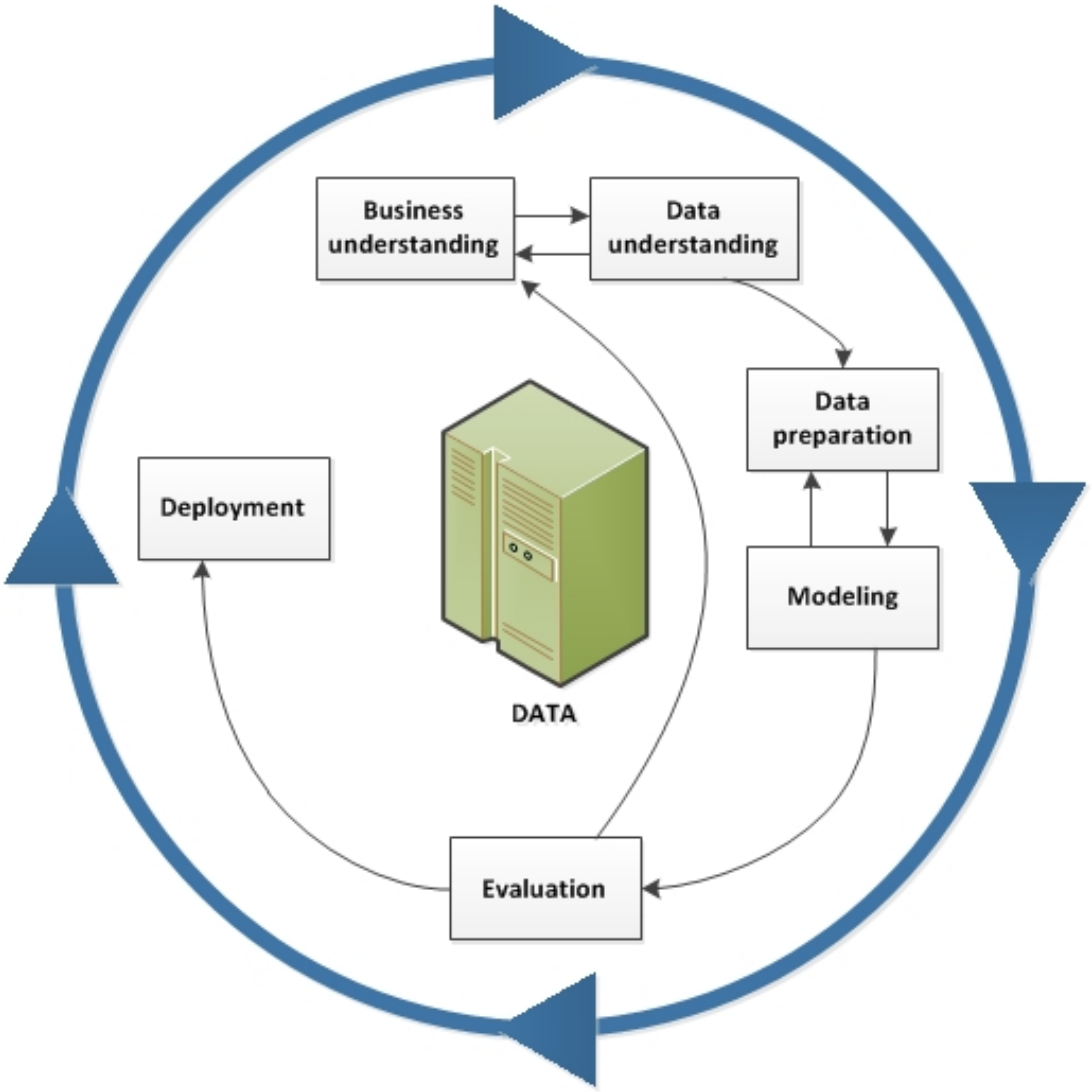


Figura 1: Ciclo de vida CRISP-DM. [17]

modelação. Em vez da modelação, o seu trabalho concentrar-se-á na exploração e visualização de dados para descobrir padrões suspeitos nos dados financeiros. O CRISP-DM permite-lhe criar um modelo de mineração de dados que se adapta às suas necessidades particulares.

Em tal situação, as fases de modelação, avaliação e implementação podem ser menos relevantes do que as fases de compreensão e preparação dos dados. Contudo, é ainda importante considerar algumas das questões levantadas durante estas fases posteriores para o planeamento a longo prazo e objetivos futuros de extração de dados.

O modelo CRISP-DM, além das fases definidas na Figura 1, possui também em cada fase, tarefas específicas a realizar (consultar Figura 2).

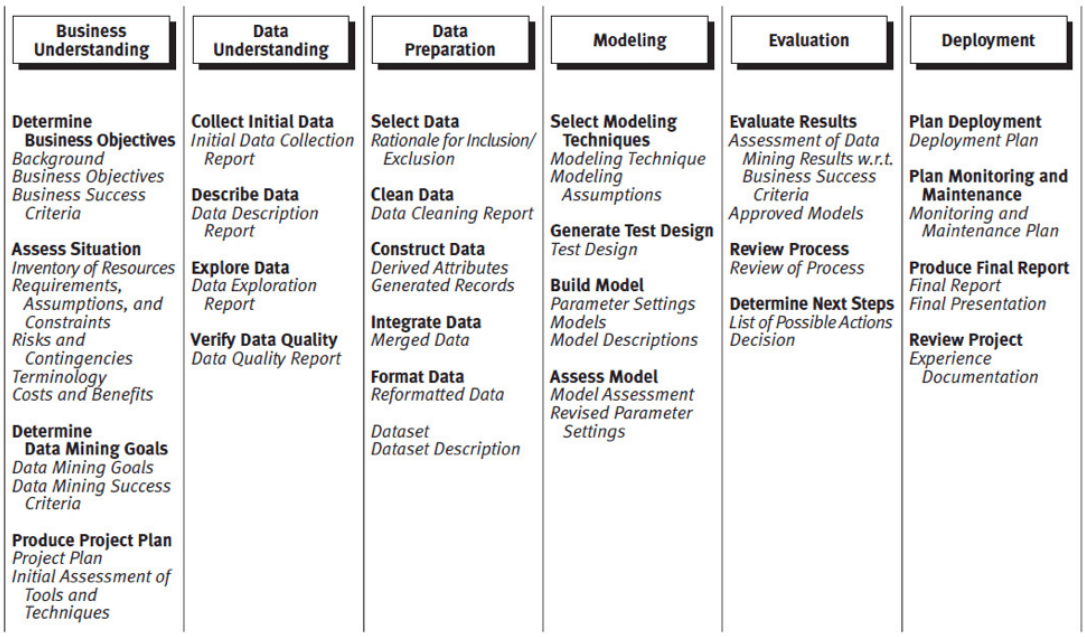


Figura 2: Tarefas CRISP-DM. [2]

2.4 Captura de tráfego

Existem várias formas de realizar a captura de tráfego de rede, nomeadamente, através da ferramenta *tcpdump* [43] e a biblioteca *libpcap*, sendo capaz de captar o tráfego que passa pela máquina. Funciona a nível de pacotes, o que significa que capta os pacotes reais que entram e saem do computador. Pode ainda guardar os pacotes num ficheiro e guardar os pacotes inteiros ou apenas os cabeçalhos. Mais tarde pode "reproduzir" o ficheiro gravado e aplicar filtros diferentes nos pacotes, utilizando os filtros do *tcpdump* para ignorar os pacotes que não lhe interessa ver.

A segunda forma de capturar tráfego, pode ser, através do o *ssldump* [42] que é um analisador do protocolo de rede SSL/TLS. Identifica as ligações TCP na interface de rede e tenta interpretá-las como tráfego SSL/TLS. Quando identifica o tráfego SSL/TLS, descodifica os registos e apresenta-os sob a forma de texto para stdout. Se for fornecido com o material de codificação apropriado, descodificará também as ligações e exibirá o tráfego de dados da aplicação.

A terceira forma de capturar tráfego, por exemplo através do *Wireshark* que, segundo a [47] é o maior e mais utilizado analisador de protocolos de rede do mundo. Permite ver o que está a acontecer na rede a um nível microscópico e é padrão em muitas empresas comerciais e sem fins lucrativos, agências governamentais, e instituições educacionais. O desenvolvimento da *Wireshark* prospera graças às contribuições voluntárias em todo o mundo e é a continuação

de um projeto iniciado por *Gerald Combs* em 1998. O *Wireshark* tem um rico conjunto de características que inclui o seguinte: inspeção profunda de centenas de protocolos, captura ao vivo e análise *offline*, lê/escreve em muitos formatos diferentes de ficheiros de captura: *tcpdump (libpcap)*, *Pcap NG*, *Catapult DCT2000*, *Cisco Secure IDS iplog*, *Microsoft Network Monitor*, entre outros.

Por último, e forma utilizada para efetuar a recolha do tráfego de rede, o *Scapy* que é uma biblioteca *Python* que permite ao utilizador enviar, capturar e forjar pacotes de rede. Esta funcionalidade permite a construção de ferramentas que podem sondar ou atacar redes.

O *Scapy* é uma biblioteca interativa de manipulação de pacotes. É capaz de forjar ou descodificar pacotes de um vasto número de protocolos, enviá-los e capturá-los, combinar pedidos e respostas, e muito mais. O *Scapy* pode facilmente lidar com a maioria das tarefas clássicas como *traceroutes*, sondagens à redes, testes unitários, ataques ou descoberta de redes. Pode até substituir ferramentas como *hping*, *arpspoof*, *arp-sk*, *arping*, *p0f* e até algumas partes do *Nmap*, *tcpdump*, e *tshark*.

Na Figura 3 é apresentado um excerto de código recorrendo à biblioteca *scapy* para a captura de 5 pacotes numa interface de rede.

```
>>> from scapy.all import sniff
>>> sniff(iface='[redacted]', prn=lambda x:x.summary(), count=5)
Ether / IP / TCP [redacted].64:39614 > [redacted].145:https PA / Raw
Ether / IP / TCP [redacted].64:39612 > [redacted].145:https PA / Raw
Ether / IP / TCP [redacted].64:38086 > [redacted].216:https PA / Raw
Ether / IP / TCP [redacted].145:https > [redacted].64:39614 PA / Raw
Ether / IP / TCP [redacted].64:39614 > [redacted].145:https A
<Sniffed: TCP:5 UDP:0 ICMP:0 Other:0>
```

Figura 3: Exemplo de captura com *scapy* em linguaguem *python*.

2.5 Plataforma de transmissão de eventos

A transmissão de eventos [24] é o equivalente ao sistema nervoso do corpo humano. É a base tecnológica para o mundo *always-on*, onde as empresas são cada vez mais definidas e automatizadas por software, e onde o utilizador de software é mais software.

De um ponto de vista mais técnico, a transmissão de eventos é a prática de capturar dados em tempo real a partir de fontes de eventos como bases de dados, sensores, dispositivos móveis, serviços de nuvem e aplicações de software sob a forma de transmissões de eventos; armazenar estas transmissões de eventos de forma durável para posterior recuperação; manipular, processar e reagir ás transmissões de eventos em tempo real, bem como retrospetivamente; e encaminhar as transmissões de eventos para diferentes tecnologias de destino conforme necessário. O fluxo de eventos assegura assim um fluxo contínuo e uma interpretação dos dados para que a informação certa esteja no lugar certo, no momento certo.

A transmissão de eventos é aplicada a uma grande variedade de casos de utilização através de uma infinidade de indústrias e organizações. Por exemplo: processar pagamentos e transações financeiras em tempo real, tais como em bolsas de valores, bancos e seguradoras, monitorizar carros, camiões, frotas e expedições em tempo real, tais como na logística e na indústria automóvel, capturar e analisar continuamente dados de sensores de dispositivos IoT (*Internet of Things*) ou outros equipamentos, tais como em fábricas e parques eólicos.

O Apache Kafka [34] é um exemplo de uma plataforma distribuída de transmissão de eventos que é capaz de publicar, subscrever, armazenar e processar fluxos de registo em tempo real (consultar Figura 4). Esta plataforma foi desenvolvida para processar fluxos de dados provenientes de diversas fontes e entregá-los a vários clientes. Em suma, o Apache Kafka transporta grandes quantidades de volumes de dados não apenas do ponto A ao ponto B, mas também de A a Z e para qualquer outro local que você precisar simultaneamente. Pode ser implementado em *bare-metal hardware*, máquinas virtuais, e *containers* em ambientes *on-premise*, bem como em ambientes de nuvem.

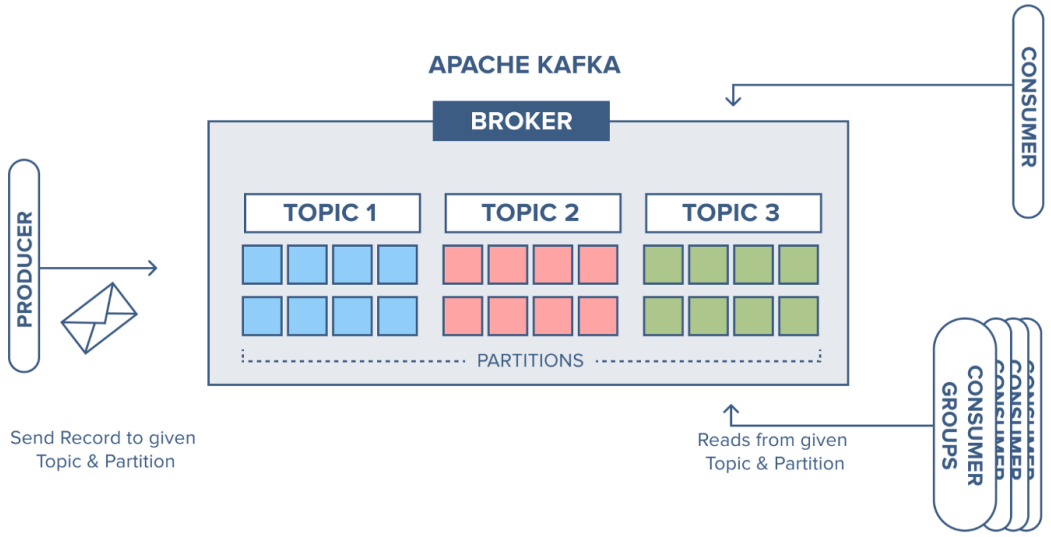


Figura 4: Arquitetura Kafka. [34]

2.6 Centro de Operação de Segurança

Segundo [23], um SOC é um centro de monitorização que ajuda as organizações a que presta serviços a identificar, detetar, gerir e responder às ameaças de segurança. O principal objetivo da equipa SOC é descobrir antecipadamente ameaças ou ataques antes que possam influenciar negativamente o negócio. SOC's podem ser implementados [9] in-house de forma tradicional, em que a organização acarreta com todas as responsabilidades da sua gestão. Outra opção é na nuvem, em que fica o cargo do MSSP (managed security service providers) coletar e agregar os eventos no SIEM, sendo a gestão feita pela empresa. No modelo SOC híbrido, a empresa terá de comprar o hardware e software para o MSSP implementar o SIEM e todas as suas capacidades. O último modelo aqui apresentado será o SOC-as-a-Service, em que o MSSP fica encarregue de tudo e a empresa apenas precisa de se preocupar com os processos de segurança relacionados com o SIEM.

2.6.1 Papel de um Centro de Operação de Segurança

O principal papel da equipa SOC é a monitorização contínua da infraestrutura e resposta a incidentes, cumprindo os requisitos de forma a combater ameaças internas e externas, neutralizando possíveis impactos negativos no negócio, aumentando assim o trio: Confidencialidade, Integridade e Disponibilidade. De acordo com o estudo [20], alguns aspetos primários que diferenciam equipas de SOC relacionam-se com a sua capacidade em: 1) recolher logs

heterogéneos; 2) reter de logs e proceder ao seu armazenamento (por exemplo, o tempo limite de retenção); 3) estabelecer regras para a criação de alertas; 4) analisar um incidente de segurança, distinguindo-o de falsos positivos. Outras funcionalidades que cabem numa equipa SOC são, por exemplo, a análise e a gestão de vulnerabilidades.

2.6.2 Hierarquia de um Centro de Operação de Segurança

Segundo os artigos da Exabeam [9] [8] [10] e ATTCyberSecurity [3], a hierarquia de uma equipa SOC está divida em, pelo menos, 5 camadas. Estas camadas são apresentadas a seguir.

- **Primeira Linha:**
  - **Descrição:** Linha de Triagem
  - **Responsabilidades:** é a primeira camada da equipa SOC, sendo que a linha de triagem tem como função analisar potenciais ameaças determinando quais devem seguir para avaliação mais profunda na segunda linha.
- **Segunda Linha:**
  - **Descrição:** Linha de Resposta a Incidentes
  - **Responsabilidades:** tratam dos incidentes reencaminhados pela primeira linha, analisando efusivamente a causa do incidente correlacionando, por exemplo, com *threat intelligence* (IOC's (indicators of compromise), regras atualizadas ou procedimentos) de forma a encontrar a origem do ataque e os ativos afetados. No final, esta linha ainda determina uma estratégia para a mitigação e recuperação do acontecimento.
- **Terceira Linha:**
  - **Descrição:** Linha de Threat Hunter (segurança informática proativa)
  - **Responsabilidades:** a terceira linha explora novas formas de identificar novas ameaças que ainda não foram encontradas com ajuda de *threat intelligence*. Também realiza testes de penetração nos sistemas de produção com o intuito de testar a sua resiliência e identificar áreas que devam ser corrigidas. Esta linha ainda é responsável por resolver incidentes críticos que a segunda não possui capacidade.

- **SecOps (Security Operations):**
  - **Descrição:** Apoio na plataforma SIEM
  - **Responsabilidades:** equipa responsável pela configuração e ajustes de regras associadas ao SIEM, como por exemplo, colocar um utilizador em *white-list* para não gerar mais incidentes.
- **DevOps (Development Operations):**
  - **Descrição:** Plataformas de apoio ao SOC
  - **Responsabilidades:** desenvolver ferramentas para auxiliar o SOC a automatizar as operações. Esta equipa está integrada dentro do SOC ou ligada a outra de desenvolvimento.

2.6.3 Principais problemas

Um estudo do instituto Ponemon [19], baseado na resposta de 637 profissionais de cibersegurança que trabalham e conhecem mais do que ninguém o funcionamento de um SOC, e que são responsáveis por várias tarefas de deteção e mitigação de ameaças e gestão de operações de segurança, abordou o verdadeiro custo do SOC, tanto a nível económico, como a nível de gestão de recursos humanos.

Um dado interessante nas equipas SOC, como mostra a Figura 5, cerca de 70% dos inquiridos aponta o SOC como um ambiente de alta pressão e alta carga de trabalho.

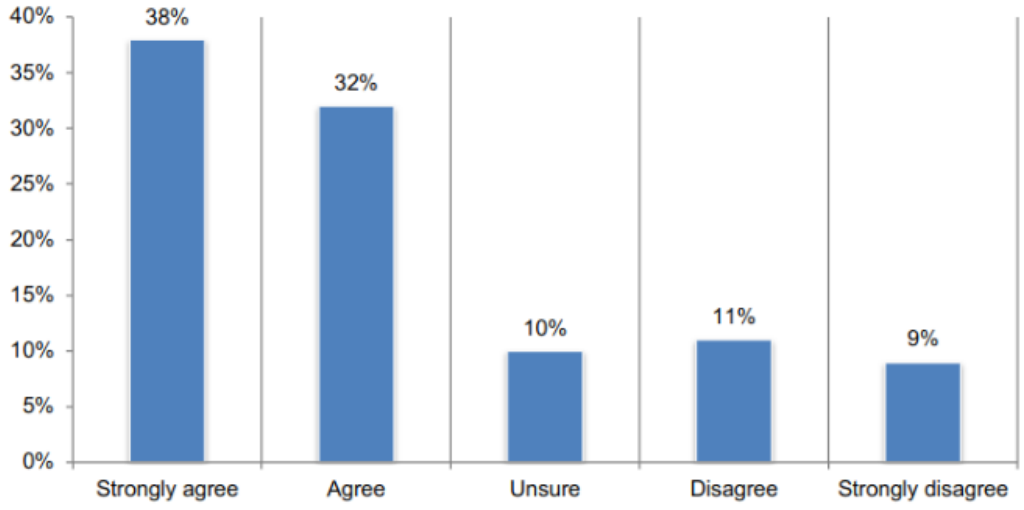


Figura 5: Respostas em relação ao ambiente de alta pressão no SOC. [19]

Foi pedido aos inquiridos que classificassem a “dor” da experiência do pessoal de segurança do SOC em satisfazer as suas necessidades diárias de trabalho de uma escala de 1 (nenhuma) dor a 10 (muito dolorosa). Setenta por cento dos inquiridos afirmam que trabalhar no SOC é muito doloroso e a razão número um é o aumento da carga de trabalho que provoca o esgotamento seguido de uma falta de visibilidade na rede e nas infra-estruturas informáticas. Citam também estar de serviço 24/7/365 e ter demasiados alertas para perseguir, como mostra a Figura 6.

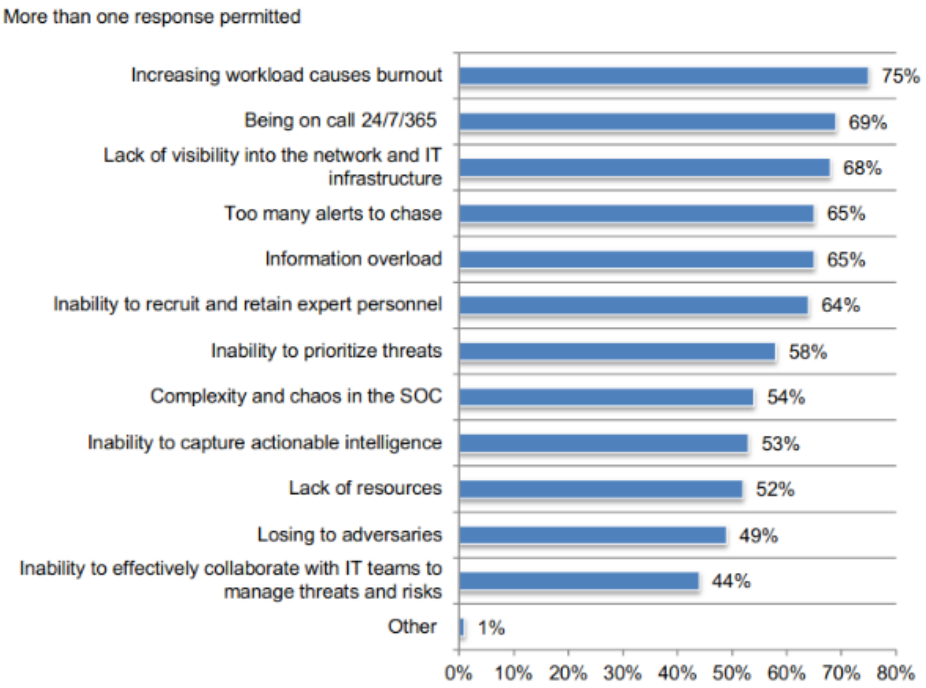


Figura 6: Principais razões pelas quais é doloroso trabalhar no SOC. [19]

Dos inquiridos que classificam o seu SOC como ineficaz (58% dos inquiridos), as principais razões são a falta de visibilidade do tráfego na rede e a remediação atempada (69% e 63% dos inquiridos, respetivamente), como mostra a Figura 7.

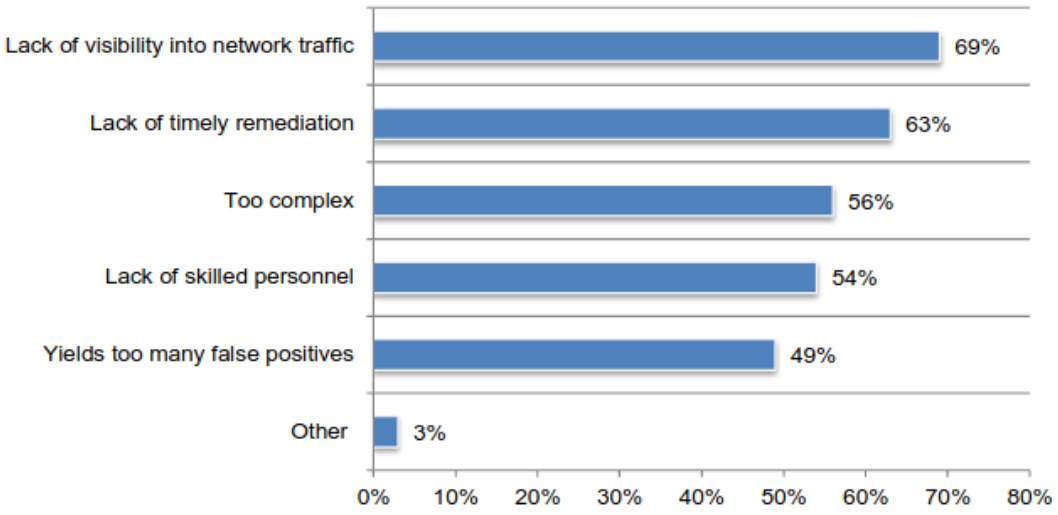


Figura 7: Principais razões pelas quais o SOC não é eficiente. [19]

A maioria dos SOCs não tem alta interoperabilidade com a inteligência de segurança da organização ferramentas. A Figura 8 apresenta seis fatores que podem influenciar a capacidade do SOC de prevenir, detetar, analisar e responder a incidentes de ciber-segurança. No entanto, a maioria dos organizações não estão a garantir que os seus SOC incorporaram estas ferramentas.

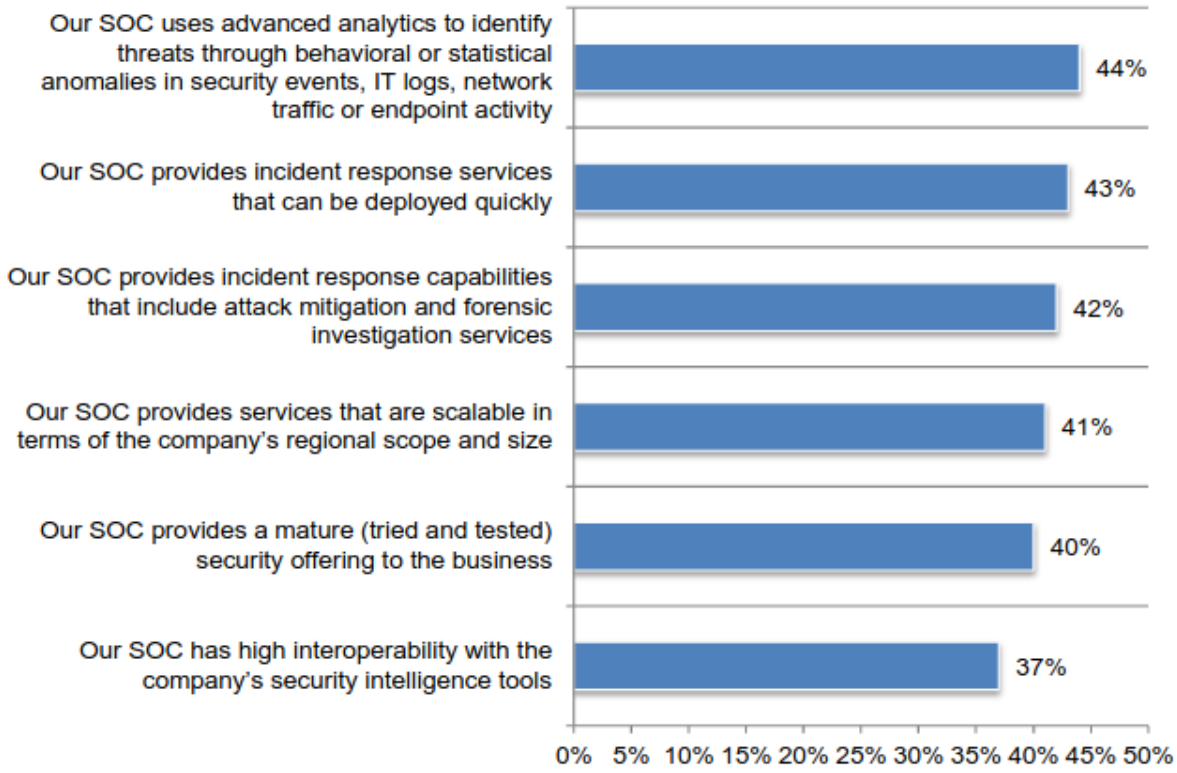


Figura 8: Desafios para um SOC eficiente. [19]

Em suma, com os dados apresentados acima, verifica-se que o SOC é um ambiente bastante propício à alta pressão e cargas de trabalho elevadíssimas, tanto pelo grande número de alertas e incidentes que normalmente estas equipas tem de responder, tal como, pela falta de visibilidade da infra-estrutura de IT e falta de engenheiros competentes para criar os casos de uso de ciber-segurança, estando a inteligência limitada ao conhecimento dele. Normalmente, a análise destes casos de uso é manual e muito demorada devido à grande quantidade de eventos que necessitam de ser verificados antes de se declarar o alerta como verdadeiro-positivo ou falso-positivo, claramente, que irá sempre depender do tipo de alerta (p.e *scanning* à rede interna vs autenticações de um pais suspeito).

Capítulo 3

Metodologia do Trabalho - Arquitetura

Nesta capítulo é apresentada a arquitetura relativa ao projeto de deteção de intrusões. Este projeto utiliza tráfego de rede capturado em tempo real recorrendo a técnicas de captura, processamento e armazenamento descritas neste capítulo. Por fim, para a deteção de anomalias são adotados algoritmos de *machine learning* onde o tráfego capturado é modelado e treinado com objetivo principal de detetar intrusões na rede em que se insere.

3.1 Captura de tráfego

Como se pode verificar na Figura 9 o processo de captura de tráfego passa por várias etapas das quais a captura de tráfego na interface de rede, posteriormente, o encaminhamento para a plataforma distribuída de eventos *Kafka* que envia para o *Spark Structured Streaming* responsável por algumas operações, como por exemplo a agregação e soma de eventos. Em quarto lugar os eventos passam por um processo de enriquecimento de dados e, por fim, são armazenados numa base de dados *MySQL*.

Figura 9: Processo de captura de tráfego.

3.2 Processamento de eventos de tráfego

Para o processamento dos eventos de tráfego é possível processá-los com recurso à biblioteca *scapy*. O formato recebido da interface de rede é o mostrado na Figura 10 e Figura 11. De salientar que, a solução apenas recolhe tráfego IPv4 ou IPv6 e protocolo de transporte TCP ou UDP.

```
###[ Ethernet ]###
  dst       = [redacted]3
  src       = [redacted]9
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 10879
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xd53e
     src       = [redacted].64
     dst       = [redacted].90
     \options   \
###[ TCP ]###
        sport     = 50382
        dport     = https
        seq       = 282385744
        ack       = 772636592
        dataofs   = 5
        reserved  = 0
        flags     = FA
        window    = 501
        chksum    = 0x3b2d
        urgptr    = 0
        options   = ''
```

Figura 10: Pacote de demonstração TCP.

```
###[ Ethernet ]###
  dst       = [redacted]3
  src       = [redacted]9
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 102
     id        = 19080
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x2c15
     src       = [redacted].64
     dst       = [redacted]
     \options   \
###[ UDP ]###
        sport     = 44950
        dport     = domain
        len       = 82
        chksum    = 0xc44d
###[ DNS ]###
           id        = 60292
           qr        = 0
           opcode    = QUERY
           aa        = 0
           tc        = 0
           rd        = 1
           ra        = 0
           z         = 0
           ad        = 0
           cd        = 0
           rcode     = ok
           qdcount   = 1
           ancount   = 0
           nscount   = 0
           arcount   = 1
           \qd        \
            |###[ DNS Question Record ]###
            |  qname     = '[redacted].com.'
            |  qtype     = AAAA
            |  qclass    = IN
           an        = None
           ns        = None
           \ar        \
            |###[ DNS OPT Resource Record ]###
            |  rrname    = '.'
            |  type      = OPT
            |  rclass    = 512
            |  extrcode  = 0
            |  version   = 0
            |  z         = 0
            |  rdlen     = None
            |  \rdata     \
```

Figura 11: Pacote de demonstração UDP.

Com o formato recebido da Figura 10, são extraídos os campos:

- **MAC Address de origem e destino:** é um identificador único atribuído a um controlador de rede (NIC) para utilização como endereço de rede em comunicações dentro de um determinado segmento de rede. Um exemplo de um endereço MAC é: 00-B0-D0-63-C2-26.
- **IP de origem de destino:** é uma designação numérica (por exemplo, 192.168.1.1) que está ligada a uma rede informática que utiliza o Protocolo Internet para a comunicação. Um endereço IP serve duas funções principais: identificação da interface de rede e endereçamento de rede.
- **Protocolo:** No *Internet Protocol* versão 4 (IPv4) existe um campo chamado "Protocolo" para identificar o protocolo da camada seguinte. Este é um 8 campo de bits. Na versão 6 do Protocolo Internet (IPv6), este campo é chamado *Next Header* (Próximo cabeçalho). Em [15] pode ser consultada a lista de protocolos.
- **Tamanho do pacote:** Campo responsável por identificar o tamanho de pacote em bytes.
- **Portos origem e destino:** Campo responsável por identificar o porto de origem/destino do equipamento em questão que está a iniciar/receber o pacote.
- **Flags TCP:** Numa ligação TCP, as *flags* são utilizadas para indicar um determinado estado de ligação ou para fornecer alguma informação útil adicional, como por exemplo, para fins de resolução de problemas ou para lidar com um controlo de uma determinada ligação.

  Tipos de flags:

  - **Sincronização (SYN)** - É utilizada na primeira etapa da fase de estabelecimento da ligação ou processo de aperto de mão de 3 vias entre os dois sistemas.
  - **Reconhecimento (ACK)** - É utilizado para reconhecer pacotes que são recebidos com sucesso pelo sistema. A *flag* é colocada se o campo do número de *acknowledgement* contiver um número de *acknowledgement* válido.
  - **Finish (FIN)** - É utilizado para solicitar a conclusão da ligação, ou seja, quando não há mais dados do remetente, solicita a conclusão da ligação.
  - **Reset (RST)** - É utilizado para terminar a ligação se o remetente RST considerar que algo está errado com a ligação TCP ou que a ligação não deve existir.

  - **Push (PSH)** - A camada de transporte por defeito espera durante algum tempo que a camada de aplicação envie dados suficientes equivalentes ao tamanho máximo do segmento, para que o número de pacotes transmitidos na rede minimize o que não é desejável por alguma aplicação como aplicações interativas (*chatting*).
  - **Urgente (URG)** - Os dados dentro de um segmento com URG = 1 são imediatamente enviados para a camada de aplicação, mesmo que haja mais dados a serem fornecidos à camada de aplicação.

Com o formato recebido da Figura 11, são extraídos os campos acima mencionados com exceção das *flags TCP* e a adição do domínio:

- **Domínio:** um domínio é um nome de fácil memorização e que serve para localizar e identificar computadores na *Internet*.

3.3 Enriquecimento de dados

Nesta secção são descritos alguns parâmetros que são recolhidos através da captura de rede e que enriquecimento de dados é feito mediante os mesmos.

Pacotes de rede transmitidos num intervalo de tempo

Com o número e tamanho dos pacotes transmitidos é possível definir outras variáveis, como:

- Número total de pacotes
- Número de pacotes que saíram da origem
- Número de pacotes que chegaram à origem
- Média, soma e máximo do tamanho dos pacotes
- Média, soma e máximo do tamanho dos pacotes que saíram da origem
- Média, soma e máximo do tamanho dos pacotes que chegaram à origem
- Total de vezes que cada *flag* aparece

Pesquisa de dados sobre o domínio

- Se está no *ranking* da *Alexa* (1 ou 0)
- Se tem *MX Record* (1 ou 0)
- Se o domínio tem menos de seis meses/entre seis e um ano/mais de um ano (1 ou 0)
- Se tem *SPF Record* (1 ou 0)
- Se tem *DMARC Record* (1 ou 0)

A *feature* de *ranking Alexa* foi escolhida pois permite determinar se o domínio se encontra numa lista de domínios pesquisados amplamente na Internet. Normalmente, se um domínio está nessa lista pode ser um indicativo que é fidedigno.

A *feature* de *MX Record*, *SPF* e *DMARC Record* foi escolhida devido à presença destes *records* poderem indicar que se trata também de um domínio com infraestrutura configurada, o que pode indicar, mais uma vez, que é fidedigno.

A *feature* de criação do domínio pode também indicar que se trata de um domínio criado à pouco tempo para atividades maliciosas ou então um domínio com vários anos de atividade legítima.

Com a pesquisa de dados sobre o domínio, é também possível, agregar o número de acessos a domínios pela mesma origem num determinado intervalo. Nomeadamente:

- Número de acessos a domínios repetidos
- Número de acessos a domínios diferentes
- Número de acessos a domínios que estão na lista da *Alexa*
- Número de acessos a domínios com menos de 6 meses/entre seis meses e um ano/mais de um ano
- Número de acessos a domínios com *MX Record*
- Número de acessos a domínios com *SPF Record*
- Número de acessos a domínios com *DMARC Record*

3.4 Estrutura dos datasets

Nesta secção são apresentados os quatro *datasets* criados para a solução desenvolvida e uma breve descrição de cada variável dos mesmos. Cada *dataset* está organizado em intervalos de tempo de um minuto e, por sua vez, o tráfego que é capturado nesse instante passa pelo processo de enriquecimento de dados, tal como descrito na secção anterior.

Na tabela 3.1 são apresentadas as variáveis do *dataset "packet"* responsável por armazenar maioritariamente tráfego de rede sazonal.

Variável	Descrição
startdate	Representa o início do intervalo de tempo
enddate	Representa o fim do intervalo de tempo
srcmac	Representa o endereço MAC de origem
srcip	Representa o IP de origem
dstmac	Representa o endereço MAC de destino
dstip	Representa o IP de destino
sport	Representa o porto de origem
dport	Representa o porto de destino
proto	Representa o protocolo
count	Representa o número de pacotes transmitidos no intervalo de tempo
total_bytes	Representa o total de bytes no intervalo de tempo
avg_bytes	Representa o mínimo de bytes no intervalo de tempo
pkt_max_bytes	Representa o tamanho do pacote maior (em bytes) no intervalo de tempo
fin_count	Representa a contagem dos pacotes TCP com a flag FIN
syn_count	Representa a contagem dos pacotes TCP com a flag SYN
rst_count	Representa a contagem dos pacotes TCP com a flag RST
psh_count	Representa a contagem dos pacotes TCP com a flag PSH
ack_count	Representa a contagem dos pacotes TCP com a flag ACK
urg_count	Representa a contagem dos pacotes TCP com a flag URG
ece_count	Representa a contagem dos pacotes TCP com a flag ECE
cwr_count	Representa a contagem dos pacotes TCP com a flag CWR

Tabela 3.1: Estrutura do dataset "packet".

Na tabela 3.2 são apresentadas as variáveis do *dataset "packet_dns"* responsável por armazenar maioritariamente tráfego de rede DNS.

Variável	Descrição
startdate	Representa o início do intervalo de tempo
enddate	Representa o fim do intervalo de tempo
srcmac	Representa o endereço MAC de origem
srcip	Representa o IP de origem
sub_domain	Representa o subdomínio do domínio acedido
domain	Representa o domínio acedido
count	Representa o número de acessos ao domínio
total_bytes	Representa o total de bytes no intervalo de tempo
avg_bytes	Representa o mínimo de bytes no intervalo de tempo
pkt_max_bytes	Representa o tamanho do pacote maior (em bytes) no intervalo de tempo
has_alexa	Representa se o domínio está na base de dados da Alexa
less_six_month	Representa se o domínio tem menos de seis meses
six_and_one	Representa se o domínio tem entre seis meses e um ano
plus_one_year	Representa se o domínio tem mais de um ano
has_mx	Representa de o domínio tem MX Record
has_spf	Representa de o domínio tem SPF Record
has_dmarc	Representa de o domínio tem DMARC Record

Tabela 3.2: Estrutura do dataset "packet_dns".

Na tabela 3.3 são apresentadas as variáveis do *dataset destination_ip_agg_inbound* responsável por armazenar maioritariamente tráfego de rede com agregação no destino das comunicações. Por exemplo neste *dataset* é possível validar a quantidade de *bytes* que entraram numa determinada origem.

Variável	Descrição
startdate	Representa o início do intervalo de tempo
enddate	Representa o fim do intervalo de tempo
dstmac	Representa o endereço MAC de destino
dstip	Representa o IP de destino
count	Representa o número de pacotes transmitidos no intervalo de tempo
inbound_total_bytes	Representa o total de bytes transferidos para a máquina no intervalo de tempo
avg_bytes	Representa o mínimo de bytes no intervalo de tempo
pkt_max_bytes	Representa o tamanho do pacote maior (em bytes) no intervalo de tempo

Tabela 3.3: Estrutura do dataset "destination_ip_agg_inbound".

Na tabela 3.4 são apresentadas as variáveis do *dataset "source_ip_agg_outbound"* responsável por armazenar maioritariamente tráfego de rede com agregação na origem das comunicações. Por exemplo neste *dataset* é possível validar a quantidade de *bytes* que saíram de uma determinada origem.

Variável	Descrição
startdate	Representa o início do intervalo de tempo
enddate	Representa o fim do intervalo de tempo
srcmac	Representa o endereço MAC de origem
srcip	Representa o IP de origem
count	Representa o número de pacotes transmitidos no intervalo de tempo
outbound_total_bytes	Representa o total de bytes transferidos da máquina no intervalo de tempo
avg_bytes	Representa o mínimo de bytes no intervalo de tempo
pkt_max_bytes	Representa o tamanho do pacote maior (em bytes) no intervalo de tempo
total_domains_accessed	Representa o número total de domínios acedidos no intervalo de tempo
same_domains	Representa o número de domínios iguais que foram acedidos
diff_domains	Representa o número de domínios distintos acedidos
alexa_domains	Representa o número de domínios que estão na base de dados da Alexa
less_six_month_domains	Representa o número de domínios que tem menos de seis meses
six_and_one_domains	Representa o número de domínios que tem entre seis meses e um ano
plus_one_year_domains	Representa o número de domínios que tem mais de um ano
mx_domains	Representa o número de domínios que tem MX Record
spf_domains	Representa o número de domínios que tem SPF Record
dmarc_domains	Representa o número de domínios que tem DMARC Record

Tabela 3.4: Estrutura do dataset "source_ip_agg_outbound".

3.5 Desenho da solução

Na Figura 12 é apresentado o desenho da solução final em condições de recolher, processar, armazenar e detetar eventos anómalos.

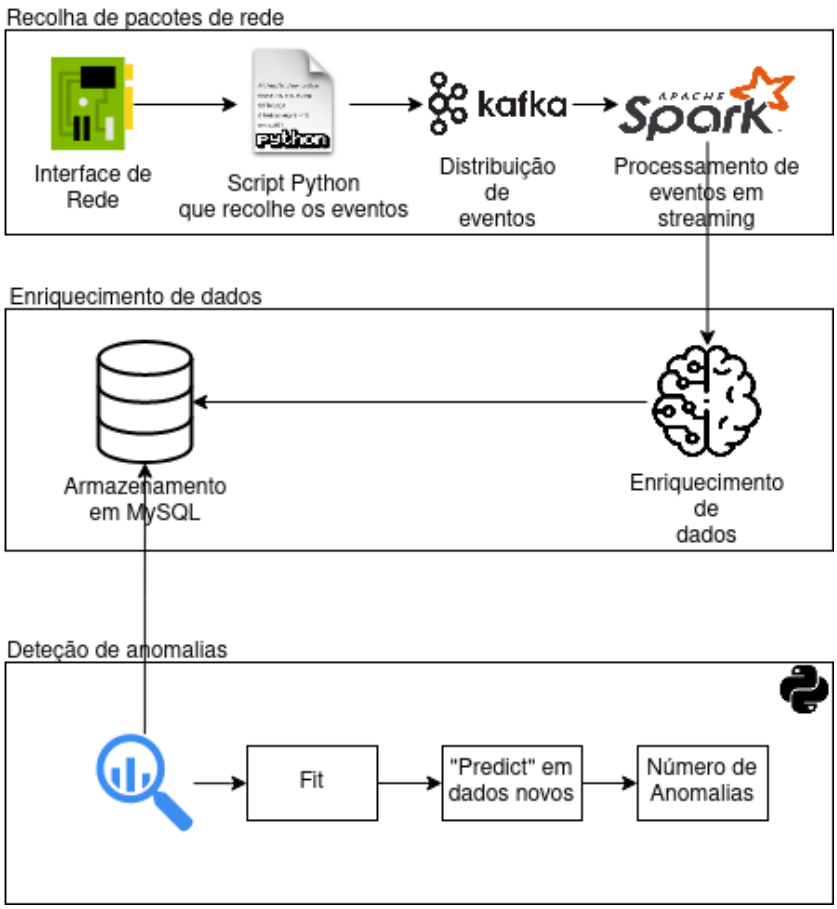


Figura 12: Desenho da solução proposta.

A recolha de pacotes de rede é feita através de um programa em linguagem *python* que captura os pacotes com recurso à biblioteca *scapy* e aplica uma normalização aos eventos (pacotes de rede) personalizada de forma a chegar ao resultado final pretendido (exemplo: 2022-00-00 00:00:00, AA:BB:CC:DD:EE:19, 192.168.50.64, FF:GG:HH:II:JJ:e3, XXX.XXX.XXX.82, 6, 363, unknown, http, 0, 0, 0, 1, 1, 0, 0, 0, NOFILL). Com esta normalização é possível o utilizador escolher os protocolos que a solução suporta, bastando para isso, adequar o filtro nos eventos de forma mais conveniente ao seu objetivo. No fim, esses eventos, são enviados para um tópico do *Kafka* com ajuda da biblioteca *Kafka*.

No *Spark Structured Streaming*, que subscreve o tópico que são esperados os eventos do passo anterior, são normalizados novamente através de um esquema definido e os dados são sujeitos a uma série de *groupBy* e funções de agregação de forma a aplicar as somas,

contagens e máximos necessários para o enriquecimento de dados. Neste caso, o utilizador também pode definir um *timeframe* de acordo com as suas necessidades. Nesta solução foi aplicado um *timeframe* de um minuto. Os dados são assim enviados novamente para um tópico *Kafka* que será subscrito por outro programa que aprofunda o enriquecimento de dados.

Na parte de enriquecimento de dados referida no paragrafo anterior, são feitas as inserções e atualizações na base de dados conforme o estipulado na lógica do funcionamento do parâmetro em si, por exemplo, garantir que na coluna “count” é feita a inserção quando não existe a agregação dos parâmetros em questão e a soma quando a linha já existe, de forma a não serem perdidos nem corrompidos dados.

Por último, na deteção de anomalias é feita a leitura dos eventos armazenados em base de dados da primeira captura, do tráfego assumido normal, e segue-se a captura dos novos eventos que serão testados com a primeira captura. Este processo ficará a correr em ciclo e o modelo é treinado a cada iteração.

Neste capítulo foi apresentado o sistema que faz a captura, tratamento e enriquecimento de dados. No próximo capítulo é apresentado um caso prático de aplicação, incluindo a análise de resultados.

Capítulo 4

Aprendizagem do comportamento da rede e deteção de desvios - Descrição técnica

Neste capítulo são apresentados os resultados do estudo experimental realizado para avaliar o funcionamento da solução desenvolvida.

4.1 Descrição do ambiente

O ambiente é composto pela rede com um computador, um *router* e vários dispositivos ligados a ela. Este computador possui as seguintes características:

- Modelo Lenovo Legion Y5
- Processador Intel Core i5-7300HQ CPU @ 2.50GHz
- Armazenamento SSD de 512GB
- Memória RAM de 16GB
- Placa de rede *wireless* Qualcomm Atheros QCA6174
- Placa de rede *Ethernet* Realtek Semiconductor RTL8111/8168/8411 PCI Express Gigabit

Para os efeitos do estudo experimental a rede de prova de conceito é composta por computadores e *smartphones*. Na Figura 13 é possível representar de forma visual o cenário de rede utilizado:

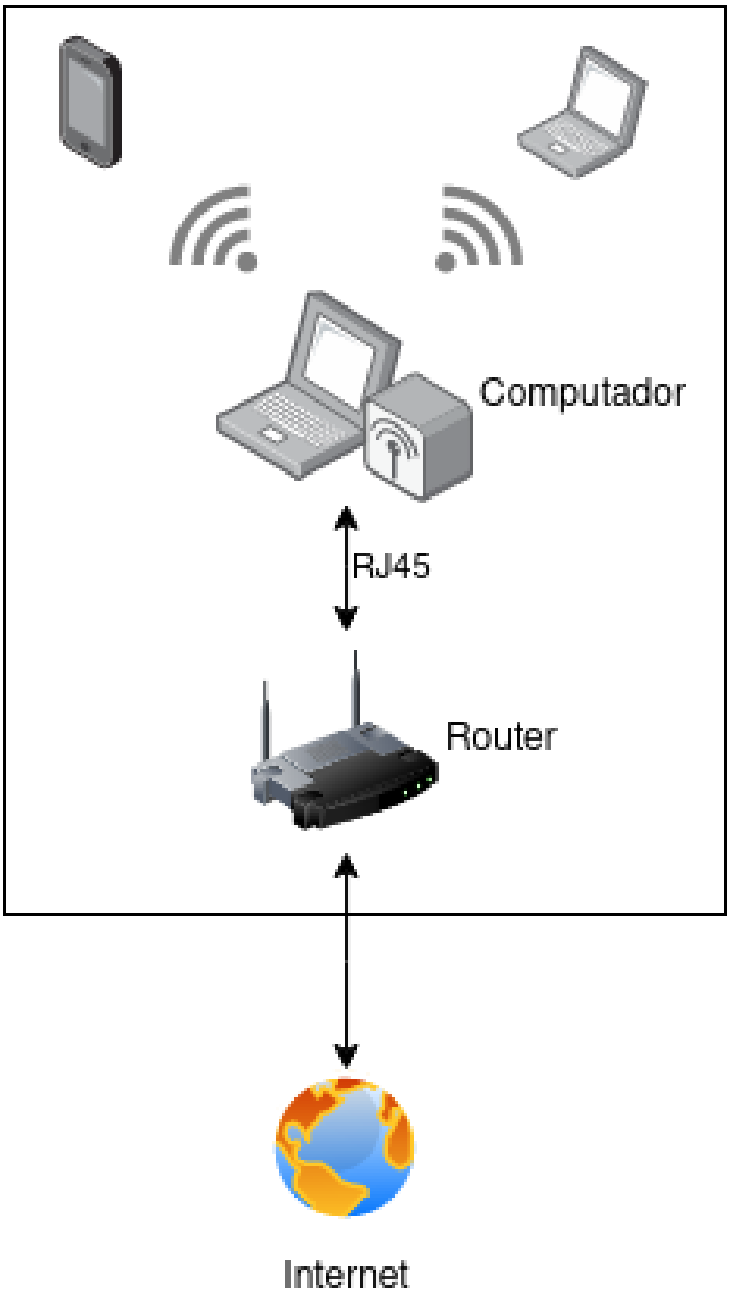


Figura 13: Diagrama do ambiente de rede.

A solução desenvolvida está instalada no computador da Figura 13, permitindo assim a recolha, processamento, armazenamento e analise do tráfego/eventos de rede dos dispositivos nela inseridos.

4.2 Captura de Tráfego

Para o estudo experimental os pacotes de rede capturados são da rede doméstica do autor, sem qualquer restrição ou modificação no seu uso, decorrendo essa captura por sensivelmente dois dias. Esta captura é a que consideramos ser o tráfego normal/comportamento normal da rede.

Após esse tempo de captura, obteve-se os quatros *datasets* já referidos na secção 3.4. O primeiro *dataset* "packet" com 46239 entradas, o segundo "packet_dns" com 19159, o terceiro "source_ip_agg_outbound" com 3633 e por último o "destination_ip_agg_inbound" com 25887 entradas.

Para realizar a captura de tráfego foi usado o programa já apresentado e que permite ler o tráfego na interface *wireless* do computador.

A Figura 14 representa a iniciação do *sniff* no lado do programa *python* responsável por capturar os eventos.

```
def main():
    print("Starting capturing...")
    sniff(iface='      ', prn=custom_action, store=0)

if __name__ == "__main__":
    main()
```

Figura 14: Código que invoca a captura de pacotes de rede.

Como referido anteriormente no ponto 3.2 e 3.5 a solução desenvolvida apenas processa tráfego IPv4 ou IPv6 e protocolo de transporte TCP ou UDP e para este último apenas o tráfego DNS. Esse filtro é aplicado neste ponto no momento da chegada do pacote com o código da Figura 15.

```python
def custom_action(p):
    try:
        if p['Ether'].type == 2048:

            if p.haslayer('TCP'):
                packet = ...

                producer.send('packet', packet)
            elif p.haslayer('UDP') and p.haslayer('DNS') and p['DNSQR'].qname:
                packet = ...

                producer.send('packet', packet)

    except Exception as error:
        print(str(error))
```

Figura 15: Código que filtra a captura de pacotes de rede.

Devido à captura ser feita na mesma máquina que é feito enriquecimento de dados (fase seguinte) encontrou-se muita distorção dos verdadeiros eventos e uma mistura do tráfego de rede, como por exemplo, os pedidos *whois*. Tentou-se no entanto, excluir esse tráfego ao máximo com *whitelisting* de domínios que contivessem "dmarc" e além disso foram descobertos alguns problemas com domínios especiais como por exemplo: "ipv4only.arpa" [16]. De forma a evitar este problema o enriquecimento de dados deverá ser feito numa máquina à parte sobre a qual se não recolhem dados de tráfego. Na Figura 16 representa o resultado final enviado para o *Kafka* que envia para o *Spark Structered Streaming*.

```
date,srcmac,srcip,dstmac,dstip,proto,len,srcport,dstport,fin,syn,rst,psh,ack,urg,ece,cwr,domain
2022-00-00 00:00:00,XX:XX:XX:XX:XX:e3,X.X.X.142,AA:AA:AA:AA:AA:19,BB.BB.BB.64,6,91,https,unknown,0,0,0,1,1,0,0,0,NOFILL
2022-00-00 00:00:00,AA:AA:AA:AA:AA:19,BB.BB.BB.64,XX:XX:XX:XX:XX:e3,X.X.X.142,6,52,unknown,https,0,0,0,0,1,0,0,0,NOFILL
2022-00-00 00:00:00,XX:XX:XX:XX:XX:e3,XX.XX.XX.8,AA:AA:AA:AA:AA:19,BB.BB.BB.64,6,52,https,unknown,1,0,0,0,1,0,0,0,NOFILL
2022-00-00 00:00:00,AA:AA:AA:AA:AA:19,BB.BB.BB.64,XX:XX:XX:XX:XX:e3,XX.XX.XX.8,6,76,unknown,https,0,0,0,1,1,0,0,0,NOFILL
2022-00-00 00:00:00,AA:AA:AA:AA:AA:19,BB.BB.BB.64,XX:XX:XX:XX:XX:e3,XX.XX.XX.8,6,52,unknown,https,1,0,0,0,1,0,0,0,NOFILL
2022-00-00 00:00:00,XX:XX:XX:XX:XX:e3,XX.XX.XX.8,AA:AA:AA:AA:AA:19,BB.BB.BB.64,6,52,https,unknown,0,0,0,0,1,0,0,0,NOFILL
2022-00-00 00:00:00,XX:XX:XX:XX:XX:e3,XX.XX.XX.8,AA:AA:AA:AA:AA:19,BB.BB.BB.64,6,52,https,unknown,0,0,0,0,1,0,0,0,NOFILL
2022-00-00 00:00:00,AA:AA:AA:AA:AA:19,BB.BB.BB.64,XX:XX:XX:XX:XX:e3,XX.XX.XX.12,6,86,unknown,https,0,0,0,1,1,0,0,0,NOFILL
2022-00-00 00:00:00,XX:XX:XX:XX:XX:e3,XX.XX.XX.12,AA:AA:AA:AA:AA:19,BB.BB.BB.64,6,40,https,unknown,0,0,0,0,1,0,0,0,NOFILL
2022-00-00 00:00:00,XX:XX:XX:XX:XX:e3,XX.XX.XX.12,AA:AA:AA:AA:AA:19,BB.BB.BB.64,6,86,https,unknown,0,0,0,1,1,0,0,0,NOFILL
2022-00-00 00:00:00,AA:AA:AA:AA:AA:19,BB.BB.BB.64,XX:XX:XX:XX:XX:e3,XX.XX.XX.12,6,40,unknown,https,0,0,0,0,1,0,0,0,NOFILL
2022-00-00 00:00:00,XX:XX:XX:XX:XX:e3,XX.XX.XX.24,AA:AA:AA:AA:AA:19,BB.BB.BB.64,6,152,https,unknown,0,0,0,1,1,0,0,0,NOFILL
2022-00-00 00:00:00,AA:AA:AA:AA:AA:19,BB.BB.BB.64,XX:XX:XX:XX:XX:e3,XX.XX.XX.24,6,156,unknown,https,0,0,0,1,1,0,0,0,NOFILL
2022-00-00 00:00:00,XX:XX:XX:XX:XX:e3,XX.XX.XX.24,AA:AA:AA:AA:AA:19,BB.BB.BB.64,6,52,https,unknown,0,0,0,0,1,0,0,0,NOFILL
2022-00-00 00:00:00,AA:AA:AA:AA:AA:19,BB.BB.BB.64,XX:XX:XX:XX:XX:e3,XX.XX.XX.82,6,363,unknown,http,0,0,0,1,1,0,0,0,NOFILL
2022-00-00 00:00:00,AA:AA:AA:AA:AA:19,BB.BB.BB.64,XX:XX:XX:XX:XX:e3,XX.XX.XX.140,6,91,unknown,https,0,0,0,1,1,0,0,0,NOFILL
2022-00-00 00:00:00,AA:AA:AA:AA:AA:19,BB.BB.BB.64,XX:XX:XX:XX:XX:e3,XX.XX.XX.82,6,361,unknown,http,0,0,0,1,1,0,0,0,NOFILL
```

Figura 16: Excerto de eventos enviados para o *Kafka*.

Nota: No caso de se tratar de um pacote de UDP (no caso em específico tráfego DNS) os campos referentes às TCP Flags serão igualados a "-1", enquanto que, se for TCP o campo domínio será preenchido com "NOFILL".

4.3 Enriquecimento de eventos

Em primeiro lugar, quando o evento é recebido no *Spark Structured Streaming* é normalizado e de seguida é agregado, num intervalo temporal de um minuto, pelos campos “srcmac”, “srcip”, “dstmac”, “dstip”, “sport”, “dport” e “proto” (no caso de ser tráfego DNS também foi desenvolvido outro *Spark* que agrega inclusive pelo domínio). Com essa agregação ainda são realizadas contagens, somas, médias e valores máximos como representa o excerto de código da Figura 17. Neste excerto ainda foi aplicado uma *watermark* de dez minutos para o caso de algum evento chegar fora de ordem ou chegar atrasado, ou seja, fora seu intervalo de tempo [38].

```
pkt_df4 = pkt_df3 \
        .withWatermark("timestamp", "10 minutes") \
        .groupBy(
                F.window(pkt_df3.timestamp, "1 minutes"),
                "srcmac",
                "srcip",
                "dstmac",
                "dstip",
                "sport",
                "dport",
                "proto"
        ).agg(F.count("srcip"), \
                F.sum("len"), \
                F.avg("len"), \
                F.max("len"), \
                F.sum("fin"), \
                F.sum("syn"), \
                F.sum("rst"), \
                F.sum("psh"), \
                F.sum("ack"), \
                F.sum("urg"), \
                F.sum("ece"), \
                F.sum("cwr"), \
        )
```

Figura 17: Excerto de código responsável pela agregação de pacotes de rede no intervalo de um minuto.

No final deste processo de agregação é enviado o evento para o *Kafka* novamente em modo de saída *update*. O *Spark* suporta três tipos de métodos de saída de eventos o *Append Mode*, que é o modo por defeito quando não é específicado nenhum, o *Complete Mode* e por fim, o que está a ser utilizado nesta solução, o *Update Mode* [39].

Depois dos eventos serem enviados pelo *Kafka* para outro programa *python*, responsável por

inserir, atualizar e armazenar os dados dos *datasets*. Por fim, após os dados serem armazenados em base de dados, o programa responsável por treinar o modelo com os novos dados capturados irá executar em ciclos de um minuto, treinando o modelo novamente a cada iteração.

4.4 Algoritmos de Machine Learning

Nesta secção serão apresentados os algoritmos de deteção de anomalias utilizados para esta solução, tal como, uma breve descrição dos seus parâmetros configuráveis. O primeiro algoritmo abordado, o *One-Class SVM*, encaixa na categoria de deteção de novidades já mencionado anteriormente no capítulo 2.2.4 e o segundo algoritmo, o *Isolation Forest* está inclinado para deteção de *outliers*, também referido no mesmo capítulo.

4.4.1 One-class SVM

Considere-se um conjunto de dados *n* de observações da mesma distribuição descrita por *p* características. Considere-se agora que foi adicionado mais uma observação a esse conjunto de dados. Será a nova observação tão diferente das outras que podemos duvidar da sua regularidade? (ou seja, provém da mesma distribuição?) Ou, pelo contrário, é tão semelhante à outra que não a podemos distinguir das observações originais? Esta é a questão abordada pelos instrumentos e métodos de deteção de novidades.

De forma geral, está prestes a aprender uma fronteira aproximada e próxima que delimita o contorno da distribuição inicial das observações, traçada no espaço *p*-dimensional incorporado. Depois, se houver mais observações dentro do sub espaço delimitado pela fronteira, são consideradas como provenientes da mesma população do que as observações iniciais. Caso contrário, se se situarem fora da fronteira, podemos dizer que são anormais com uma dada confiança na nossa avaliação.

O One-Class SVM foi introduzido por Schölkopf com esse propósito e foi implementado na biblioteca *Support Vector Machines* em “svm.OneClassSVM”. Requer a escolha de um *kernel* e de um parâmetro escalar para definir uma fronteira. O *kernel RBF* é geralmente escolhido embora não exista uma fórmula ou algoritmo exato para definir o seu parâmetro de largura

de banda. Este é o padrão na implementação do *scikit-learn*. O parâmetro "nu", também conhecido como a margem do SVM de uma classe, corresponde à probabilidade de encontrar uma nova observação, mas regular, fora da fronteira [35].

Na Figura 18 representa um excerto de código do algoritmo *One-class SVM* implementado na solução proposta.

```
# fit the model
clf = OneClassSVM(nu=0.3, kernel="rbf", gamma=0.1).fit(x_train)

...

# incoming events
y_pred_test = clf.predict(x_test)
n_outliners = y_pred_test[y_pred_test == -1].size

print("Outliers: " + str(n_outliners))
```

Figura 18: Excerto de código da invocação do algoritmo *One-Class SVM*.

4.4.2 Isolation Forest

A deteção de *outliers* é semelhante à deteção de novidades no sentido em que o objetivo é separar um núcleo de observações regulares de algumas observações contaminantes, chamadas *outliers*. No entanto, no caso da deteção de *outliers*, não temos um conjunto de dados limpo que represente a população de observações regulares que possa ser utilizado para treinar qualquer ferramenta.

O algoritmo *Isolation Forest* isola as observações através da seleção aleatória de uma *feature* e depois selecionando aleatoriamente um valor dividido entre os valores máximo e mínimo da *feature* selecionada.

Uma vez que a partição recursiva pode ser representada por uma estrutura em árvore, o número de divisões necessárias para isolar uma amostra é equivalente ao comprimento do caminho desde o nó de raiz até ao nó terminal. Este comprimento do caminho, medido em média sobre uma floresta de tais árvores aleatórias, é uma medida de normalidade e a função de decisão.

A segmentação aleatória produz caminhos visivelmente mais curtos para as anomalias. Assim, quando uma floresta de árvores aleatórias produz coletivamente percursos mais curtos para

determinadas amostras, é muito provável que sejam anomalias.

Na Figura 19 representa um excerto de código do algoritmo *Isolation Forest* implementado na solução proposta.

```
# fit the model
clf = IsolationForest(n_estimators=100, max_samples="auto", contamination="auto", warm_start=False) \
                    .fit(x_train)

...

# incoming events
y_pred_test = clf.predict(x_test)
n_outliners = y_pred_test[y_pred_test == -1].size
print("Outliers: " + str(n_outliners))
```

Figura 19: Excerto de código da invocação do algoritmo *Isolation Forest*.

4.5 Simulação de ataques

Nesta secção é apresentado duas simulações de ataques, a primeira por exfiltração de dados e a segunda por enumeração de ativos. No final é feito uma análise dos resultados obtidos pela solução.

4.5.1 Exfiltração de dados

Para representação deste ataque, foi simulada uma exfiltração de dados através da transferência de vários ficheiros de múltiplos tamanhos para a plataforma *OneDrive*. Na Figura 20 está representado o resultado do modelo de *machine learning One-class SVM* e na Figura 21 o modelo de Isolation Forest. À medida que foi executado o ataque, e como é possível verificar existe um aumento significativo no número de *outliers*. Infelizmente, devido a não ser implementado um sistema de explicabilidade, não foi possível garantir com cem por cento que o acréscimo foi devido à exfiltração de dados.

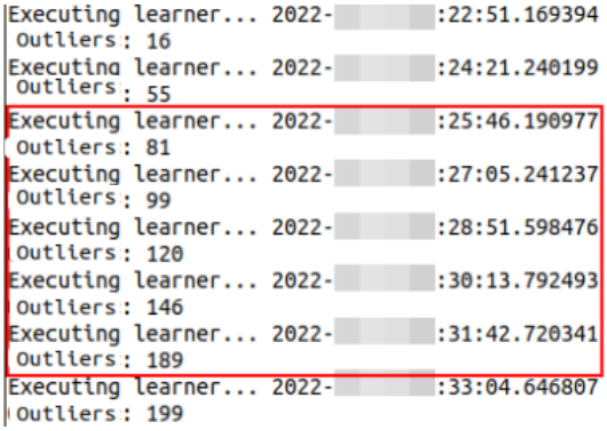


Figura 20: Resultado do teste realizado com ex-filtração de dados com o algoritmo *One-class SVM*.

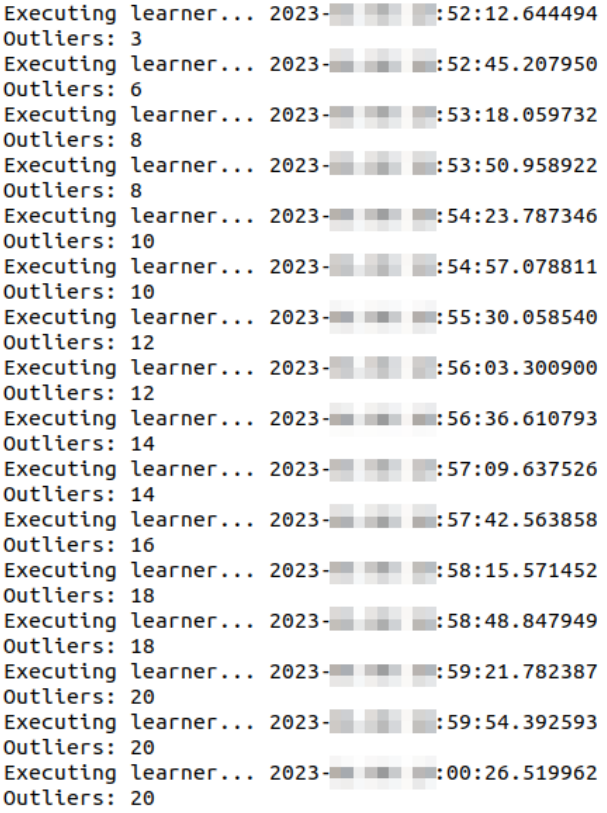


Figura 21: Resultado do teste realizado com ex-filtração de dados com o algoritmo *Isolation Forest*.

Na secção 4.5.3 é apresentado uma análise dos resultados obtidos.

4.5.2 Enumeração de ativos

Para representação deste ataque, foi simulada uma enumeração dos ativos de rede através da ferramenta *nmap*, com a execução do comando *nmap -Pn -p1-65535 -r 192.168.1.1/24*. Infelizmente, devido à arquitetura de rede e a limitações tecnológicas, a recolha de eventos que representam uma enumeração não foi conseguida de forma clara e consolidada. Nos eventos apenas era possível constatar comunicações para o endereço de rede *192.168.1.1* por ser o *default gateway* onde o computador que alojava a solução comunicava para o resto da rede. Para serem apresentadas comunicações tipo de enumeração de ativos, o comportamento que deveria ser apresentado seria um equipamento a realizar várias comunicações a todos os outros equipamentos na rede destino (no caso *192.168.1.1/24*) e não apenas a um endereço (*192.168.1.1*). Contudo, foi na mesma realizado o teste com os dois algoritmos de *machine learning* como representa a Figura 22, com o algoritmo *One-class SVM*, e a Figura 23 com o *Isolation Forest*.

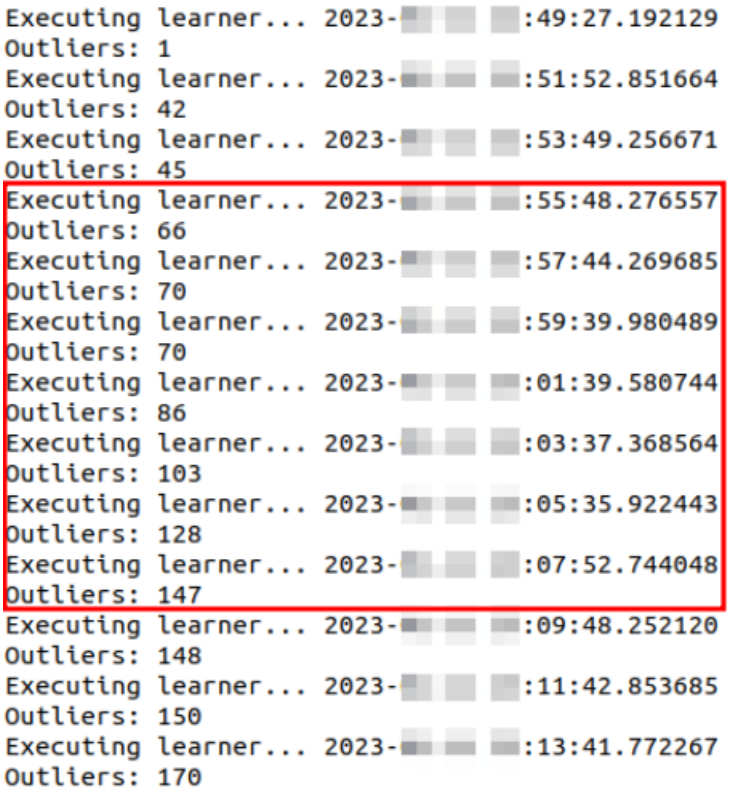


Figura 22: Resultado do teste realizado com enumeração de ativos com o algoritmo *One-class SVM*.

```
Executing learner... 2023-[illegible]:13:37.229730
Outliers: 0
Executing learner... 2023-[illegible]:14:10.395139
Outliers: 8
Executing learner... 2023-[illegible]:14:43.528385
Outliers: 9
Executing learner... 2023-[illegible]:15:16.303601
Outliers: 12
Executing learner... 2023-[illegible]:15:48.991496
Outliers: 11
Executing learner... 2023-[illegible]:16:21.286967
Outliers: 12
Executing learner... 2023-[illegible]:16:53.345485
Outliers: 12
Executing learner... 2023-[illegible]:17:25.427578
Outliers: 12
Executing learner... 2023-[illegible]:17:57.356568
Outliers: 12
Executing learner... 2023-[illegible]:18:29.190964
Outliers: 12
Executing learner... 2023-[illegible]:19:01.261167
Outliers: 11
Executing learner... 2023-[illegible]:19:33.132944
Outliers: 12
Executing learner... 2023-[illegible]:20:04.855032
Outliers: 11
Executing learner... 2023-[illegible]:20:36.665294
Outliers: 12
Executing learner... 2023-[illegible]:21:08.569976
Outliers: 11
```

Figura 23: Resultado do teste realizado com enumeração de ativos com o algoritmo *Isolation Forest*.

4.5.3 Análise de Resultados

É possível concluir que no algoritmo *One-class SVM* foram encontrados mais *outliers* devido à menor resiliência aos mesmos, como referido na secção 2.2.4. No caso do *Isolation Forest* é um algoritmo que encaixa nos mais resilientes indo de encontro também aos resultados obtidos.

Infelizmente falha semântica da explicabilidade do *outlier* é um dos maiores obstáculos para os detetores de anomalias que utilizam o *machine learning*, o que neste ponto não é possível afirmar com clareza qual o melhor algoritmo testado. É expectável que com a aplicação da explicabilidade dos resultados obtidos e a integração com *user feedback* o modelo poderá ser melhorado e afinado de acordo com a infraestrutura analisada.

Capítulo 5

Conclusão

Os atores maliciosos estão constantemente a inovar nas suas capacidades e é crucial que empresas sejam suficientemente inteligentes para reconhecer o valor de um serviço SOC, durante e posteriormente à sua transformação para estar ao nível das grandes tecnológicas. Já uma equipa SOC deve ter especialmente atenção ao possível surgimento de ataques automatizados, baseados em inteligência artificial, o que fomenta a criação de defesas igualmente inteligentes para combater este tipo de ameaças que cada vez mais tem lugar no nosso quadro, devido à velocidade das comunicações (e aqui poderá ser feita referência ao recente 5G/6G, que eleva a sensação de real-time a outro nível).

Não é possível deixar de referir que a criação de uma cultura de segurança informática em volta dos colaboradores é um dos melhores investimentos que uma empresa pode fazer, visto que 85% das fugas de dados ainda estão associados ao fator humano [46].

Ainda existem muitos desafios na utilização de *machine learning* no âmbito de deteção de anomalias. Uma das aplicações mais bem sucedidas do *machine learning* está nos sistemas de recomendação. Utilizando técnicas como a filtragem coletiva, tais sistemas são capazes de extrair as preferências latentes dos utilizadores e atuar como um motor para a geração de procura ativa. E se for feita uma recomendação errada? Se um produto irrelevante for recomendado a um utilizador que navegue por um site de compras *online*, as repercussões são insignificantes. Para além do custo de oportunidade perdida de uma potencial recomendação bem sucedida, o utilizador simplesmente ignora a recomendação desinteressante. Se for cometido um erro no algoritmo de classificação de pesquisa personalizada, o utilizador pode não encontrar o que procura, mas não há nenhuma perda grande e tangível incorrida.

A deteção de anomalias está alicerçada num paradigma fundamentalmente diferente. O custo dos erros na deteção de intrusão ou anomalia é enorme. A classificação incorreta de uma anomalia pode causar uma falha incapacitante no sistema. A criação de alertas falsos positivos tem um impacto menos drástico, mas falsos positivos espúrios podem rapidamente degradar a confiança no sistema, resultando mesmo em alertas totalmente ignorados. Devido ao elevado custo dos erros de classificação, são muito raros os sistemas de deteção de anomalias totalmente automatizados, de ponta a ponta, que são alimentados puramente pelo *machine learning* - há quase sempre um humano no laço para verificar se os alertas são relevantes antes de se tomar qualquer ação sobre eles.

A falta semântica é um problema real com o *machine learning* em muitos ambientes. Em comparação com os conjuntos de regras estáticas, pode por vezes ser difícil explicar porque é que um evento foi assinalado como uma anomalia, levando a ciclos de investigação de incidentes mais longos. Em casos práticos, a interpretabilidade ou explicabilidade dos resultados é muitas vezes tão importante como a exatidão dos resultados. Especialmente para sistemas de deteção de anomalias que evoluem constantemente os seus modelos de decisão ao longo do tempo, vale a pena investir recursos de engenharia em componentes do sistema que possam gerar explicações de leitura humana para alertas gerados por um sistema de aprendizagem de máquinas. Na medida do possível, dada a opacidade de muitos processos de aprendizagem da máquina, será útil gerar explicações sobre a razão pela qual o modelo tomou a decisão que tomou [4].

Além de mais, os atores maliciosos podem (e irão) gastar tempo e esforço para contornar os sistemas de deteção de anomalias se houver uma compensação que valha a pena do outro lado. O efeito da adaptação nos sistemas e algoritmos de aprendizagem de máquinas é real e é uma consideração necessária quando se implementam sistemas num ambiente potencialmente hostil.

A deteção de anomalias é uma área em que as técnicas de *machine learning* têm demonstrado uma grande eficácia. A resposta a um melhor sistema de deteção de anomalias pode não ser a utilização de um algoritmo mais avançado, mas sim a geração de um conjunto de dados mais completo e descritivo. Devido ao grande alcance das ameaças que lhes são exigidas para mitigar, os sistemas de segurança têm uma tendência para crescer incontrolavelmente em complexidade. Ao construir ou melhorar sistemas de deteção de anomalias, manter sempre a simplicidade como uma prioridade máxima [4].

Com a solução desenvolvida prevê-se melhorar a deteção e resposta a incidentes de segurança da forma mais eficaz na medida em que irá suportar as atividades dos operadores, automatizando o processo de análise e deteção de desvios a um comportamento normal de rede, associando esses desvios a potenciais ciberataques reduzindo o número de falsos positivos.

5.1 Trabalho futuro

Nesta secção são apresentados alguns tópicos idealizados para trabalho ainda a realizar na solução desenvolvida.

5.1.1 Integrar Resposta e Mitigação de Incidentes

Os alertas de anomalias podem vir sob a forma de um e-mail ou de uma notificação (*webhook*). Em muitos casos, as organizações que mantêm uma variedade de diferentes sistemas de deteção de anomalias e monitorização de segurança encontram valor na agregação de alertas de múltiplas fontes numa única plataforma conhecida como sistema de Gestão de Informação e Eventos de Segurança (SIEM). Os SIEM podem ajudar na gestão da produção de sistemas de segurança fragmentados, que podem rapidamente crescer fora de controlo em volume. Os SIEMs podem também correlacionar alertas levantados por diferentes sistemas para ajudar os analistas a recolher conhecimentos a partir de uma grande variedade de sistemas de deteção de segurança. Ter um local uniformizado para a notificação e alerta pode também fazer uma diferença notável no valor dos alertas de segurança levantados.

5.1.2 Integração com *human feedback*

Ter um sistema de deteção de anomalias com *feedback* pode fazer um sistema adaptável excelente. Se os analistas de segurança forem capazes de reportar falsos positivos e falsos negativos diretamente no sistema que ajusta os parâmetros do modelo com base neste *feedback*, a capacidade de manutenção e flexibilidade do sistema pode ser vastamente elevada. Em ambientes não confiáveis, contudo, a integração direta do *feedback* humano no modelo de treino pode ter efeitos negativos.

5.1.3 Explicabilidade

Como mencionado anteriormente, a falha semântica da explicabilidade do alerta é um dos maiores obstáculos para os detetores de anomalias que utilizam o *machine learning*. Muitas aplicações práticas valorizam as explicações de resultados. No entanto, a verdadeira explicabilidade de *machine learning* é uma área de investigação que ainda não obteve muitas respostas.

Os simples classificadores de *machine learning* são bastante transparentes nas suas previsões. Por exemplo, um modelo de regressão linear num conjunto de dados bidimensional gera resultados muito explicáveis, mas falta-lhe a capacidade de aprender características mais complexas e variadas. Modelos de *machine learning* mais complexos, tais como redes neuronais e *random forest*, podem encaixar melhor nos dados do mundo real, mas são muito *black-box* - os processos de tomada de decisão são completamente imprecisos para um observador externo. Contudo, existem formas de abordar o problema que podem aliviar a preocupação de que as previsões de *machine learning* são difíceis de explicar, provando que a explicabilidade não está de facto em desacordo com a exatidão. Ter um sistema externo gera explicações simples e legíveis para as decisões tomadas por um classificador *black-box* satisfaz as condições de explicabilidade de resultados, mesmo que as explicações não descrevam as condições reais de tomada de decisão do sistema de *machine learning*. Este sistema externo analisa qualquer saída do sistema de aprendizagem da máquina e efetua uma análise de dados consciente do contexto para gerar as razões mais prováveis para o facto de o alerta ter sido levantado [30] [45].

5.1.4 Adversarial Machine Learning

Adversarial machine learning é o estudo das vulnerabilidades em *machine learning* em ambientes hostis. Os investigadores de segurança e *machine learning* publicaram pesquisas sobre ataques práticos contra mecanismos de antivírus com *machine learning* [48], filtros de spam [32], detetores de intrusão de rede, classificadores de imagem [25] e analise de sentimentos [27] [14]. Esta tem sido uma área de investigação cada vez mais ativa nos últimos tempos, ainda que tais ataques raramente tenham sido observados.

Bibliografia

[1] Mahbubul Alam. Support vector machine (svm) for anomaly detection. https://towardsdatascience.com/support-vector-machine-svm-for-anomaly-detection-73a8d676c331, 2020. [Online; accessed 16-October-2022].

[2] Data Science Process Alliance. Crisp-dm for data science teams: 5 actions to consider. https://www.datascience-pm.com/crisp-dm-for-data-science-teams-5-actions-to-consider/, 2022. [Online; accessed 12-October-2022].

[3] ATT. The security operations center (soc) team: operations and responsibilities. https://cybersecurity.att.com/solutions/security-operations-center/building-a-soc/soc-team, 2022. [Online; accessed 11-September-2022].

[4] Clarence Chio and David Freeman. *Machine Learning and Security*. OReilly, 2018.

[5] Bayer et al. Scalable, behavior-based malware clustering, 2009.

[6] K . Thakur et al. An analysis of information security event managers, 2016.

[7] P. Duessel et al. Detecting zero-day attacks using context-aware anomaly detection at the application-layer., 2016.

[8] Exabeam. Security operations center roles and responsibilities. https://www.exabeam.com/security-operations-center/security-operations-center-roles-and-responsibilities/, 2022. [Online; accessed 11-September-2022].

[9] Exabeam. Siem architecture: Technology, process and data. https://www.exabeam.com/explainers/siem/siem-architecture/, 2022. [Online; accessed 11-September-2022].

[10] Exabeam. The soc, siem, and other essential soc tools. https://www.exabeam.com/siem-guide/the-soc-secops-and-siem/, 2022. [Online; accessed 11-September-2022].

[11] Rahul Gaikwad, Ravindra Vaidya, and Manasi Bhate. Design engineering a framework design for algorithmic it operations (aiops). *No Journal associated*, pages 2037–2044, 2021.

[12] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[14] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Deceiving google's perspective api built for detecting toxic comments, 2017.

[15] IANA. Protocol numbers. https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml. [Online; accessed 10-November-2022].

[16] IANA. Special-use domain names. https://www.iana.org/assignments/special-use-domain-names/special-use-domain-names.xhtml. [Online; accessed 22-November-2022].

[17] IBM. Crisp-dm help overview. https://www.ibm.com/docs/en/spss-modeler/saas?topic=dm-crisp-help-overview, 2021. [Online; accessed 12-October-2022].

[18] IBM. What is data science? https://www.ibm.com/cloud/learn/data-science-introduction, 2022. [Online; accessed 06-September-2022].

[19] Ponemon Institute. The economics of security operations centers: What is the true cost for effective results? https://mandiant.widen.net/s/bqvqvpjwp8/rpt-ponemon-institute-second-annual-study-economics-of-the-soc-2021-1, 2021. [Online; accessed 17-October-2022].

[20] Pierre Jacobs, Alapan Arnab, and Barry Irwin. Classification of security operation centers. In *2013 Information Security for South Africa*, pages 1–7, 2013.

[21] Javatpoint. Types of machine learning. https://www.javatpoint.com/types-of-machine-learning, 2022. [Online; accessed 03-September-2022].

[22] M. Jordan and T. Mitchell. Machine learning: Trends, perspectives, and prospects. *American Association for the Advancement of Science*, pages 255–260, 2015.

[23] Fehér Dávid János and Nguyen Huu Phuoc Dai. Security concerns towards security operations centers. In *2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 000273–000278, 2018.

[24] Apache Kafka. Getting started. https://kafka.apache.org/documentation/#gettingStarted, 2022. [Online; accessed 06-September-2022].

[25] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2016.

[26] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

[27] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, jul 2018.

[28] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest.

[29] W. Lynn. Defending a new domain: the pentagon's cyberstrategy. *Foreign Affairs*, pages 97–108, 2010.

[30] mlsecproject. On explainability in machine learning. https://github.com/mlsecproject/blog.mlsecproject.org/blob/master/posts/2015/03/2015-03-05_bianco_explainability_in_ML.md, 2015. [Online; accessed 06-March-2022].

[31] Robert Nau. Statistical forecasting: notes on regression and time series analysis. https://people.duke.edu/~rnau/411home.htm, 2020. [Online; accessed 15-October-2022].

[32] Blaine Nelson, Marco Barreno, Fuching Chi, Anthony Joseph, Benjamin Rubinstein, Udam Saini, Charles Sutton, J. Tygar, and Kai Xia. Exploiting machine learning to subvert your spam filter., 01 2008.

[33] NIST. Grubbs' test for outliers. https://www.itl.nist.gov/div898/handbook/eda/section3/eda35h1.htm. [Online; accessed 16-October-2022].

[34] Redhat. O que é apache kafka? https://www.redhat.com/pt-br/topics/integration/what-is-apache-kafka, 2022. [Online; accessed 06-September-2022].

[35] scikit learn. Novelty detection. https://scikit-learn.org/stable/modules/outlier_detection.html#novelty-detection. [Online; accessed 03-December-2022].

[36] Ibm Security and Ponemon Institute. *Cost of a Data Breach Report 2021*. IBM Security, 2021.

[37] SKLearn. Unsupervised outlier detection using the local outlier factor (lof). https://scikit-learn.org/dev/modules/generated/sklearn.neighbors.LocalOutlierFactor.html. [Online; accessed 17-October-2022].

[38] Spark. Handling late data and watermarking. https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#handling-late-data-and-watermarking. [Online; accessed 25-November-2022].

[39] Spark. Output modes. https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#output-modes. [Online; accessed 25-November-2022].

[40] Spark. Spark overview. https://spark.apache.org/docs/latest/index.html, 2022. [Online; accessed 07-September-2022].

[41] Spark. Spark streaming overview. https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#overview, 2022. [Online; accessed 11-September-2022].

[42] ssldump. ssldump(1) - linux man page. https://linux.die.net/man/1/ssldump, 2022. [Online; accessed 12-September-2022].

[43] tcpdump. tcpdump for dummies. www.alexonlinux.com/tcpdump-for-dummies, 2022. [Online; accessed 12-September-2022].

[44] Great Learning Team. Understanding goodness of fit test, definition — what is goodness of fit? https://www.mygreatlearning.com/blog/understanding-goodness-of-fit-test/, 2022. [Online; accessed 13-October-2022].

[45] Ryan Turner. A model explanation system. https://www.blackboxworkshop.org/pdf/Turner2015_MES.pdf, 2015. [Online; accessed 25-October-2022].

[46] Verizon. Cybercrime thrives during pandemic: Verizon 2021 data breach investigations report. https://www.verizon.com/about/news/verizon-2021-data-breach-investigations-report, 2021. [Online; accessed 13-May-2022].

[47] Wireshark. Wireshark. https://www.wireshark.org/, 2022. [Online; accessed 12-September-2022].

[48] Weilin Xu, Yanjun Qi, and David Evans. Automatically evading classifiers: A case study on pdf malware classifiers, 01 2016.