Agentes e Sistemas Multi-Agente

Sistema de Deteção de Intrusões

David Teixeira PG55929
Eduardo Cunha PG55939
Jorge Rodrigues PG55966
Tiago Rodrigues PG56013

Maio, 2025

Indice

1. Introdução

Atualmente, os ataques informáticos têm-se tornado cada vez mais frequentes, sofisticados e difíceis de mitigar, representando uma séria ameaça à integridade, confidencialidade e disponibilidade de sistemas informáticos. A proliferação de dispositivos conectados, bem como a crescente adoção de infraestruturas e de ambientes *cloud*, expõe redes a um número crescente de ataques que exploram vulnerabilidades a diferentes níveis da comunicação.

Neste contexto, torna-se fulcral a construção de mecanismos eficazes de deteção e prevenção de intrusões. Os Sistemas de Deteção de Intrusões (IDS) são essenciais para impedir ataques, permitindo a identificação proativa de comportamentos maliciosos em redes. No entanto, abordagens tradicionais, centralizadas ou baseadas apenas num tipo de deteção, revelam-se muitas vezes insuficientes, tanto pela falta de escalabilidade como pelo elevado custo computacional envolvido.

A utilização de Sistemas Multi-Agente (MAS) surge como uma possível solução para colmatar estas limitações. Ao distribuir as tarefas de monitorização e análise por múltiplos agentes inteligentes, é possível conceber uma arquitetura mais flexível, escalável e resiliente, capaz não só de detetar intrusões, mas também de reagir de forma coordenada e eficiente.

Este trabalho propõe o desenvolvimento de um IDS baseado em agentes, para combinar técnicas de deteção por assinaturas e por anomalias. Através da colaboração entre diferentes agentes especializados, o sistema é capaz de detetar uma gama de ataques, incluindo *port scans*, *ping flood* e *SYN flood*, mesmo em situações onde a infraestrutura subjacente apresenta restrições ao nível da comunicação entre nós. Ao explorar a arquitetura modular dos MAS, propomos uma solução viável para ambientes distribuídos, minimizando o *overhead* e maximizando a eficácia na resposta a ameaças em tempo real.

2. Definição de Domínio

Um IDS é um componente de segurança responsável por monitorizar e analisar o tráfego de rede ou atividades de um sistema, para identificar comportamentos suspeitos, violações de políticas de segurança ou tentativas de intrusão. Os IDS podem atuar de forma passiva, apenas alertando o administrador, ou integrados com sistemas de prevenção, permitindo ações automáticas como bloqueio de tráfego ou alteração de regras nas *firewalls*.

Existem diferentes abordagens para a deteção de intrusões, sendo estes sistemas baseados em assinaturas, que detetam ataques conhecidos comparando padrões de tráfego com uma base de dados de assinaturas previamente identificadas. São eficientes contra ameaças já catalogadas, mas ineficazes perante ataques novos ou variantes. Ainda, existem sistemas baseados em anomalias, que utilizam modelos estatísticos, heurísticas ou técnicas de *machine learning* para identificar comportamentos que se desviam do normal. São úteis na deteção de ataques desconhecidos, mas tendem a apresentar uma maior taxa de falsos positivos e requerem assistência externa para resolução de problemas.

Adicionalmente, os IDS podem variar no desenho da arquitetura. Sistemas centralizados apresentam um único ponto de concentração de funcionalidades, enquanto sistemas distribuídos dividem propósitos por partes, permitindo uma maior flexibilidade. A abordagem cooperativa, especialmente em ambientes complexos, como redes heterogéneas ou infraestruturas *cloud*, oferece vantagens significativas ao nível da escalabilidade, resiliência e especialização dos agentes.

Desta forma, definimos os seguintes objetivos para o projeto:

- Desenvolver um IDS que funcione a nível de rede (*network-based*), capaz de inspecionar o tráfego entre diferentes nós da infraestrutura.
- Permitir a instalação modular do sistema em múltiplos nós, estrategicamente posicionados ao longo da rede.
- Suportar múltiplas configurações operacionais, permitindo a combinação de agentes baseados em assinaturas e agentes baseados em anomalias.
- Assegurar um elevado grau de configurabilidade, adaptando o sistema a diferentes cenários e requisitos de segurança.
- Integrar mecanismos de prevenção de intrusões, com controlo direto sobre regras de *firewall*, para mitigar ataques em tempo real.

3. Descrição do Sistema

O sistema proposto consiste num IDS distribuído, baseado em agentes, que combina duas abordagens complementares: a deteção por assinaturas e a deteção por anomalias. Esta arquitetura visa maximizar a capacidade de deteção de ataques conhecidos e desconhecidos, tirando partido da robustez da deteção por assinaturas e da flexibilidade adaptativa da deteção por anomalias. A distribuição dos agentes pelo sistema permite ainda um melhor aproveitamento dos recursos computacionais e uma maior escalabilidade.

O sistema foi concebido para operar num ambiente de rede simulado, utilizando tecnologias como os agentes XMPP (Openfire), a framework SPADE para o desenvolvimento dos agentes inteligentes, e o emulador de redes CORE para simulação da infraestrutura de rede.

3.1. Arquitetura

Durante a fase de definição da arquitetura, várias decisões foram tomadas e ajustadas com base nas limitações e capacidades das tecnologias envolvidas.

Inicialmente, o conceito passava por distribuir os agentes pelos diferentes nós da rede, de modo a equilibrar a carga computacional e aumentar a escalabilidade do sistema. A ideia era que cada agente estivesse localizado num nó distinto. No entanto, esta abordagem revelou-se inviável devido a dificuldades de comunicação via Openfire entre nós distintos no ambiente emulado. Desta forma, apesar de ter sido possível hospedar múltiplos agentes, a comunicação entre agentes alojados em nós diferentes não funcionava corretamente.

Face a estas limitações, optou-se por centralizar a operação num único nó. No entanto, surgiu então um novo desafio: a execução simultânea dos módulos de deteção por assinaturas e por anomalias num só nó. Este modo de operação tornava-se demasiado exigente em termos de recursos, resultando numa degradação significativa do desempenho e numa elevada taxa de falsos negativos.

Como resposta, a arquitetura foi reformulada para permitir separar fisicamente os dois tipos de IDS. A camada de assinaturas e a camada de anomalias foram atribuídos a diferentes *routers*, funcionando como uma camada de defesa antes do tráfego chegar ao nó final (a aplicação alvo da proteção). A ideia inicial era que cada IDS fosse executado num *host* dedicado, mas novamente, devido a limitações técnicas, isso não foi possível. A solução final consistiu em hospedar todos os agentes no mesmo servidor Openfire utilizado pelo IDS baseado em assinaturas. Ou seja, embora os agentes responsáveis pela deteção de anomalias estivessem logicamente atribuídos a outro nó e comunicassem exclusivamente entre si, encontravam-se fisicamente alojados no mesmo servidor da camada de assinaturas.

Outro aspeto importante do refinamento do sistema foi a necessidade de monitorizar múltiplas interfaces de rede. A abordagem inicial consistia em criar um agente de monitorização por interface, mas esta solução revelou-se ineficiente, devido ao elevado *overhead* de comunicação (provavelmente causado por limitações do *hardware*). Como alternativa, optou-se por desenvolver um único agente de monitorização capaz de analisar todas as interfaces simultaneamente, o que se revelou uma solução muito mais eficiente e eficaz no contexto específico do sistema.

A arquitetura final implementada assenta numa estrutura de duas camadas bem definidas:

- **Camada de Assinaturas:** Hospedada no nó principal, é responsável pela deteção de ataques conhecidos com base em regras pré-definidas. Para além das regras de deteção, foram também definidas assinaturas de prevenção que, em caso de alerta, aplicam automaticamente regras na *firewall*, com o objetivo de bloquear e prevenir a continuação do ataque identificado.

- **Camada de Anomalias:** Esta camada é responsável pela deteção de comportamentos anómalos, recorrendo a um modelo de *machine learning*. Para o treino e validação deste modelo, foi utilizado um dataset desenvolvido internamente, composto por tráfego normal da rede complementado com amostras de ataques simulados.

A arquitetura desenvolvida para o sistema descrito alinha-se claramente com um sistema fechado, cooperativo e hierarquizado. Trata-se de um sistema fechado, pois a estrutura e os papéis dos agentes foram previamente definidos, bem como as linguagens de comunicação e os objetivos a atingir, garantindo assim previsibilidade e controlo sobre o comportamento global. Além disso, é um sistema cooperativo, onde os agentes trabalham em conjunto para garantir a deteção e mitigação de ameaças, partilhando informação de forma estruturada para atingir um objetivo comum. Finalmente, a arquitetura adota uma organização hierárquica, com uma comunicação vertical entre agentes, desde os monitores responsáveis pela recolha de dados, passando pelos módulos de análise, até ao coordenador, que toma decisões sobre ações corretivas.

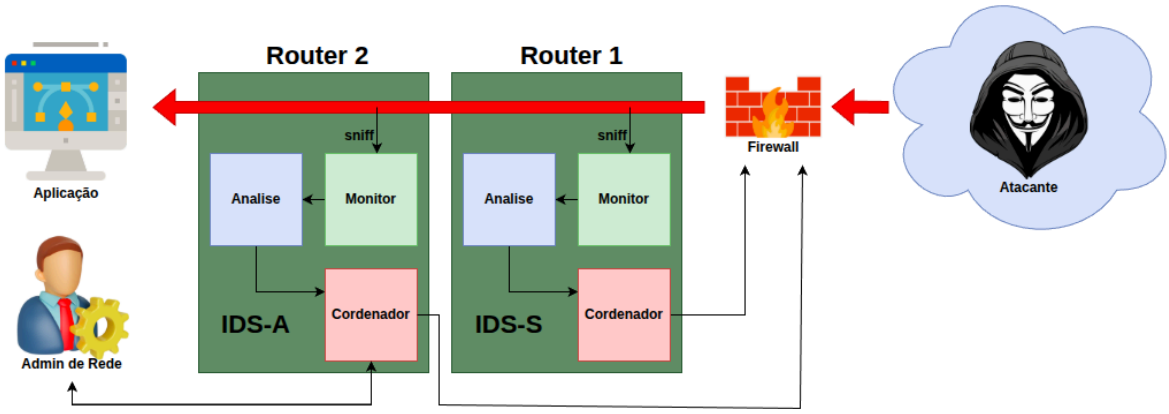

Figura 1: Funcionamento do IDS

3.2. Agentes e Funcionamento

Cada componente do sistema é implementado como um agente especializado, cujo comportamento (*behaviour*) é ativado consoante o modo de funcionamento por assinaturas ou por anomalias. Antecipando a descrição de cada agente e respetivos comportamentos e responsabilidades no sistema, apresentamos a representação da arquitetura em *UML*, através do seguinte diagrama de classes.

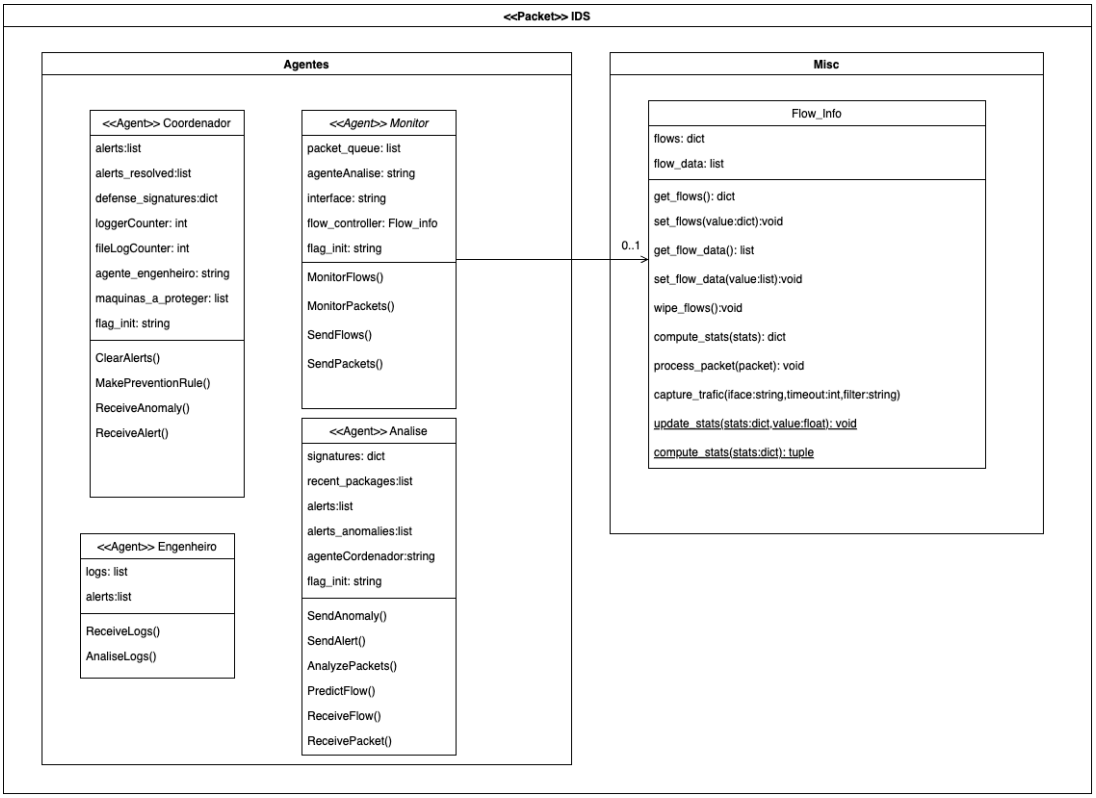

Figura 2: Diagrama de Classes

3.2.1. Agente Monitor

O agente monitor é responsável por recolher o tráfego de rede, através da biblioteca Scapy. O comportamento deste varia consoante o tipo de deteção em curso. No modo de assinaturas, cada pacote capturado é imediatamente encaminhado para o agente de análise, já no modo de anomalias, os pacotes são agregados num *flow* durante um intervalo de tempo definido. No fim desse período, são calculadas várias métricas estatísticas relativas ao tráfego recolhido, que serão posteriormente utilizadas para deteção de padrões anómalos.

Esta lógica é operacionalizada através da definição de diferentes *behaviours* no agente monitor, cada um adaptado ao tipo de deteção pretendido. O agente possui três *behaviours* distintos: dois associados à deteção por assinaturas e um dedicado à deteção por anomalias.

No contexto das assinaturas, destacam-se:

- MonitorBehaviour (CyclicBehaviour): um comportamento cíclico responsável por escutar ("sniffar") o tráfego da rede e armazenar os pacotes numa fila de espera.
- SendPacketBehaviour (CyclicBehaviour): também cíclico, este comportamento retira os pacotes da fila e envia-os para o agente de análise sob a forma de lista.

Para a deteção baseada em anomalias, é utilizado:

- MonitorFlowBehaviour (CyclicBehaviour): um comportamento cíclico que agrega pacotes num *flow* durante um intervalo de tempo pré-definido. Este *flow* é encapsulado numa estrutura `flow_info`, que, além de armazenar os pacotes, é responsável por calcular as estatísticas relevantes. Uma vez concluído o período de captura, o *flow* é enviado diretamente para o agente de análise de anomalias. Para garantir maior eficácia na deteção e minimizar falsos

positivos, os *flows* são organizados por IP de origem, evitando assim, a mistura de tráfego proveniente de diferentes fontes, o que poderia comprometer a fiabilidade da análise.

Este agente, na nossa ótica, é classificado como um agente reativo com representação do mundo, dado que, em modo de assinatura, reage ativamente face aos pacotes recebidos, enquanto no modo de anomalias, requer a acumulação de pacotes, criando um estado interno ao agente com uma representação parcial do ambiente, para a extração de métricas. Assim, vai álem de um simples reflexo, na medida em que apresenta análise e construção de estado.

3.2.2. Agente de Análise

O agente de análise adapta igualmente o seu comportamento ao modo de funcionamento. No caso das assinaturas, os pacotes recebidos são comparados com um conjunto de padrões definidos manualmente. Se for detetada uma correspondência, o agente gera um alerta e notifica o agente de coordenação, especificando o tipo de ataque e o endereço IP do atacante. No modo de anomalias, as estatísticas geradas pelo agente de captura são analisadas por um modelo de *machine learning*, previamente treinado. Quando o modelo classifica um *flow* como anómalo, essa informação é igualmente encaminhada para o agente de coordenação.

Esta lógica é implementada através da definição de diferentes comportamentos no agente, cada um orientado para o tipo de deteção em causa. O agente integra seis comportamentos distintos: três associados à deteção por assinaturas e três dedicados à deteção por anomalias.

No contexto das assinaturas, destacam-se:

- AnaliseBehaviour(CyclicBehaviour): comportamento cíclico responsável por receber pacotes da rede e verificar se correspondem a alguma das assinaturas conhecidas. Caso seja encontrada uma correspondência, é gerado um alerta, representado por um dicionário com informações relativas ao ataque, que é então adicionado a uma lista interna.
- EnviaAlertaBehaviour(CyclicBehaviour): também cíclico, este comportamento monitoriza a lista de alertas e, sempre que existirem novos registos, envia-os ao agente de coordenação.
- ApagaAlertasBehaviour(PeriodicBehaviour): comportamento periódico que elimina, a cada hora, os alertas registados. Este mecanismo foi introduzido com o objetivo de aumentar a eficiência geral do IDS e reduzir tanto o número de mensagens desnecessárias como a quantidade excessiva de alertas gerados por determinados tipos de ataque.

Para a deteção baseada em anomalias, é utilizado:

- FlowReceiveBehaviour(CyclicBehaviour): comportamento cíclico encarregado de receber os *flows* e colocá-los numa fila de espera para análise.
- FlowAnaliseBehaviour(CyclicBehaviour): comportamento cíclico que processa os *flows* da fila utilizando um modelo de *machine learning* para identificar eventuais anomalias. Caso seja detetado um padrão anómalo, é gerado um alerta, representado por um dicionário com informações relevantes extraídas do flow. Este alerta é então adicionado a uma lista de alertas.
- EnviaAnomaliaBehaviour(CyclicBehaviour): comportamento cíclico responsável por notificar o agente de coordenação sempre que é detetada uma anomalia.

Ao contrário do modo de assinaturas, nesta abordagem revelou-se necessária uma separação clara entre a receção e a análise dos *flows*. Esta decisão deveu-se, fundamentalmente, ao facto

de o tempo de análise de um *flow* ser significativamente superior ao tempo necessário para verificar assinaturas. Esta diferença tornou-se ainda mais evidente após a opção de distinguir os *flows* com base no IP de origem, o que aumentou de forma considerável o número total de *flows* processados e impôs uma maior eficiência na gestão paralela das tarefas.

O agente de Análise é classificado como um agente reativo com aprendizagem. No modo de assinaturas, encontramos um agente com regras bem definidas (tipo "if-then"), típico de agente reativo. Já no modo de anomalias, utiliza *machine learning*, o que o classifica como um agente com aprendizagem. Esta capacidade de aprendizagem pode ser ainda desenvolvida com técnicas que serão referidas na secção de trabalho futuro.

3.2.3. Agente Coordenador

O agente de coordenação assume a responsabilidade de tomar decisões com base nos alertas recebidos. No modo de assinaturas, ao receber um alerta, aplica de forma automática regras na *firewall* com o objetivo de mitigar o ataque. Estas regras podem incluir, por exemplo, o *rate limiting* ou o bloqueio total de pacotes provenientes do IP atacante. No modo de anomalias, devido à maior probabilidade de falsos positivos, a atuação do agente depende da validação por parte de um administrador da rede. Assim, sempre que é detetada uma anomalia, é gerado automaticamente um relatório em formato *.txt* contendo os detalhes do alerta, para que possa ser analisado manualmente. A ideia inicial previa o envio deste relatório por email, mas essa solução exigia uma ligação à internet fora do ambiente emulado no CORE, o que se revelou tecnicamente complexo e pouco relevante face ao objetivo principal do projeto.

Esta lógica é implementada através da definição de diferentes comportamentos no agente de coordenação, cada um adaptado ao tipo de deteção em causa. O agente integra cinco comportamentos distintos: três associados à deteção por assinaturas e dois dedicados à deteção por anomalias.

No contexto das assinaturas, destacam-se:

- ReceiveAlertsBehaviour(CyclicBehaviour): Comportamento cíclico responsável por receber alertas do agente de análise e armazenar numa lista interna o IP da ameaça detetada, juntamente com o tipo de ataque identificado.
- PreventionBehaviour(CyclicBehaviour): Comportamento cíclico que verifica a lista interna de alertas e, sempre que deteta entradas, aplica medidas de mitigação na *firewall*, de acordo com as regras definidas para cada tipo de ataque.
- ApagaAlertasBehaviour(PeriodicBehaviour): Comportamento periódico que limpa, a cada hora, os alertas registados. Este mecanismo surgiu da necessidade de evitar a duplicação excessiva de regras durante ataques do tipo flood, nos quais o agente de análise pode gerar múltiplos alertas consecutivos. Embora não houvesse impacto funcional ao adicionar regras repetidas, esta solução permitiu uma abordagem mais eficiente e limpa.

Para a deteção baseada em anomalias, é utilizado:

- ReceiveAnomaliaBehaviour(CyclicBehaviour): Comportamento cíclico responsável por receber alertas de anomalia e registar a informação recebida num ficheiro `.txt`, para posterior análise manual por parte do administrador da rede. A cada 10 registos de logs, é enviado um alerta ao agente que representa o engenheiro de rede, contendo uma referência que lhe

permite saber qual ficheiro de logs correspondente. Os logs são organizados em ficheiros com 10 entradas cada, facilitando a leitura e evitando a necessidade de apagar registos antigos que seria uma prática incorreta.

- ReceiveRequestBehaviour(CyclicBehaviour): Comportamento cíclico que espera por uma mensagem com a performativa *request*, enviada pelo administrador da rede. A mensagem inclui no corpo uma lista de endereços IP associados a logs anómalos. Perante esta informação, o agente coordenador mitiga novos ataques aplicando regras à *firewall*, bloqueando os IPs identificados.

O agente Coordenador pode ser classificado como um agente híbrido. Esta classificação advém do facto de que o seu funcionamento envolve a tomada de decisões com base em alertas recebidos da etapa de análise. No modo de assinaturas, o agente atua autonomamente, aplicando regras na *firewall*. Já no modo de anomalias, devido ao risco acrescido de falsos positivos, a atuação é condicionada à validação manual por parte de um administrador, refletindo um comportamento deliberativo.

3.2.4. Agente Engenheiro

O agente engenheiro é inicializado no sistema IDS apenas quando o modo de deteção por anomalias está ativado. Este agente surge com o objetivo de representar, ou até substituir parcialmente, o papel de um administrador de rede humano. A sua função principal é receber alertas de ativação e, com base nesses alertas, analisar um ficheiro de logs. Através do parsing desses registos, o agente tenta determinar se trata se de uma anomalia real ou de um falso positivo.

Caso identifique uma anomalia real, o agente envia uma mensagem ao agente de coordenação, contendo os endereços IP responsáveis pela anomalia. O agente de coordenação tem então a capacidade de agir sobre essa informação, aplicando medidas para mitigar potenciais ataques futuros.

Esta lógica é implementada através da definição de diferentes comportamentos (*behaviours*) no agente engenheiro. O agente integra dois comportamentos distintos associados à deteção por anomalias:

- ReceiveLogsBehaviour(CyclicBehaviour): Comportamento cíclico responsável por receber alertas. A partir de cada alerta recebido, extrai-se a informação necessária para identificar o ficheiro de logs a analisar. Essa informação é então armazenada numa fila de *logs*.

- AnaliseLogsBehaviour(CyclicBehaviour): Comportamento cíclico que processa os elementos da fila de alertas. Para cada elemento, determina o ficheiro de *logs* correspondente (através da concatenação do nome base com o identificador presente na fila) e inicia a análise do seu conteúdo atravéz do *parse* de cada *log*. Durante a análise, os endereços IP associados a comportamentos anómalos são registados numa lista. No final, essa lista é enviada ao agente de coordenação, permitindo-lhe tomar as medidas adequadas.

O agente engenheiro pode ser classificado como um agente híbrido. Esta classificação advém do facto de que o seu funcionamento envolve a escuta de alertas e a respetiva reação sobre a forma de pedido do bloqueio, sendo esta a componente reativa, e uma componente delibe-

rativa, derivada do uso dos *logs* do sistema como base de conhecimento para a tomada de decisão sobre problemas registados.

3.2.5. Comunicação entre Agentes

A comunicação entre agentes constitui um elemento central na conceção de MAS, sendo esta responsável pela coordenação, troca de informação e tomada de decisão cooperativa entre os diferentes componentes do sistema. No contexto do IDS desenvolvido, a comunicação entre agentes foi desenhada para assegurar uma deteção eficiente, colaborativa e escalável de intrusões em ambiente de rede.

A lógica de comunicação adotada é estritamente cooperativa, sendo que os agentes não competem por recursos nem executam ações autónomas de forma isolada. Cada agente assume um papel bem definido no sistema, e a interação entre estes ocorre de modo a garantir uma partilha eficaz de dados e alertas, permitindo uma resposta coordenada a potenciais ameaças.

Idealmente, procurou-se seguir os princípios estabelecidos pela especificação FIPA (Foundation for Intelligent Physical Agents), nomeadamente no que diz respeito ao uso de protocolos de comunicação normalizados e à estrutura das mensagens trocadas. No entanto, devido ao desenho da arquitetura proposta, nem todos os aspetos da norma FIPA puderam ser implementados integralmente.

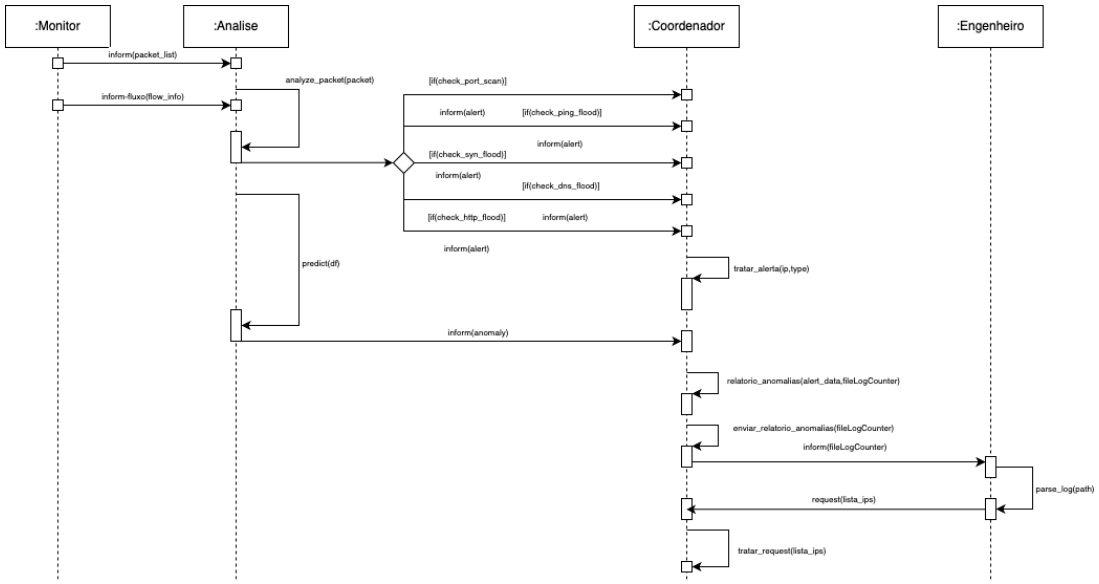

Figura 3: Diagrama de Sequência

Deste modo, conseguimos aglomerar todas as interações do sistema numa sequência de acontecimentos, traduzida na Figura 3, que pode ser traduzida pelos seguintes pontos:

1. Do "Monitor" para o agente de "Análise".
   1.1. O "Monitor" envia um conjunto de pacotes observados até ao momento do envio, para o agente de "Análise", com a *performative inform*.
   1.2. O "Monitor" envia um objeto do tipo *flow_info*, contendo estatísticas sobre os fluxos observados até ao momento do envio, para o agente de "Análise", com a *performative inform-fluxo*.
2. Do "Análise" para o agente "Coordenador".

2.1. Na eventualidade de detetar um ataque presente no conjunto das assinaturas, o agente de "Análise", dependo do ataque, informa com um alerta o agente "Coordenador", com a *performative inform*.
2.2. Na eventualidade de detetar um ataque através da variância do estado normal da rede, isto é, uma anomalia, o agente de "Análise", informa com um alerta o agente "Coordenador", com a *performative inform-fluxo*.

3. No agente "Coordenador".
3.1. Ao receber o tipo de alerta, o agente "coordenador" está equipado para reagir à ameaça encontrada, atuando exclusivamente sobre a *firewall* do sistema.
4. Do agente "Coordenador" para o agente "Engenheiro"
4.1. Ao receber uma anomalia, o agente "Coordenador" regista a informação, em formato textual, num ficheiro de logs. A cada 10 registos, é enviado um alerta ao agente "Engenheiro", contendo uma referência para o ficheiro de logs concluído.
5. Do agente "Engenheiro" para o agente "Coordenador"
5.1. Após detetar uma anomalia através da análise dos logs, o agente "Engenheiro" envia ao agente "Coordenador" uma lista com os endereços IP responsáveis pelas ocorrências anómalas.

As interações podem ser ainda representadas seguindo um diagrama de atividades, que representa as atividades executadas no sistema, no estilo de um fluxograma.

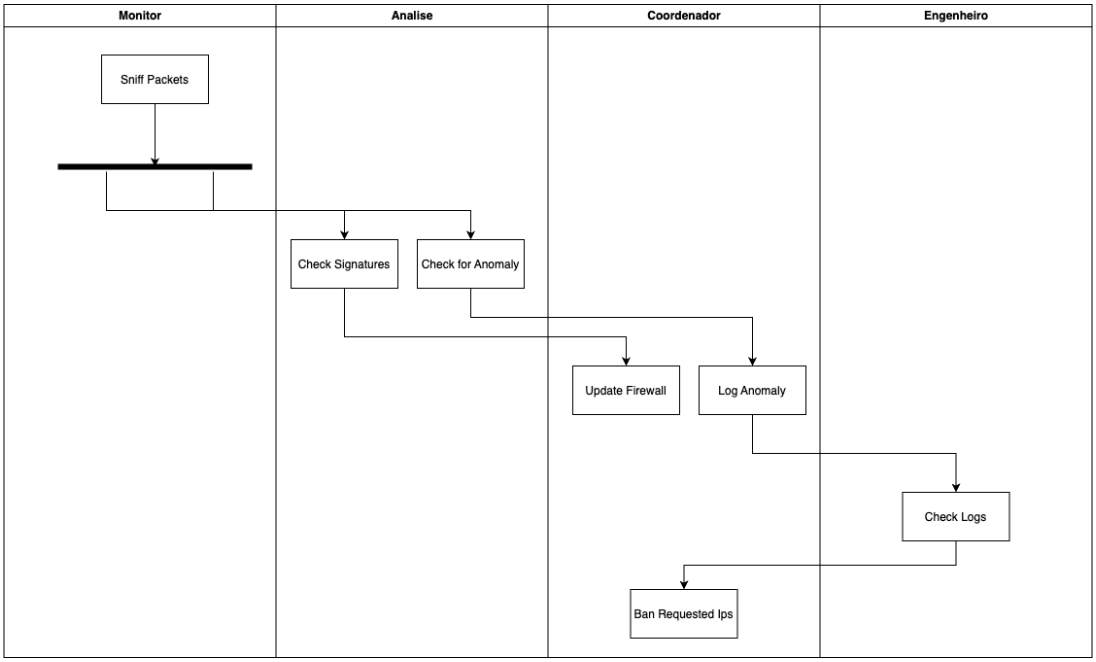

Figura 4: Diagrama de Atividade

3.3. Performatives

Relativamente às *performatives* utilizadas na comunicação entre os agentes, para os agentes de assinaturas utiliza-se a performative *"inform"*, pertencente à *FIPA ACL*. Esta *performative* é usada tanto em mensagens trocadas entre o monitor e o agente de análise, bem como entre o agente de análise e o coordenador. No caso das *performatives* destinadas aos agentes de anomalias, recorre-se à *performative "inform-fluxo"*, criada especificamente para esse propósito. Adicionalmente, o agente coordenador comunica com o agente engenheiro utilizando a

performative "inform" e, em caso de resposta, o agente engenheiro comunica com o agente coordenador utilizando a *performative "request"*.

Esta escolha deve-se à possibilidade de executar o IDS em ambos os modos no mesmo nó, de forma simultânea, exigindo o envio de pacotes para análise por assinaturas e de pacotes em fluxo para a componente de anomalias, utilizando os mesmos agentes. Dada esta sobreposição funcional, torna-se essencial uma distinção entre os diferentes tipos de mensagens. Assim, optou-se por manter esta funcionalidade no sistema, mesmo que isso implique um desvio da especificação *FIPA ACL*, com o objetivo de aumentar a flexibilidade e a adaptabilidade do sistema.

De modo a firmar o modelo de comunicação concebido, definimos o seguinte diagrama de colaboração:

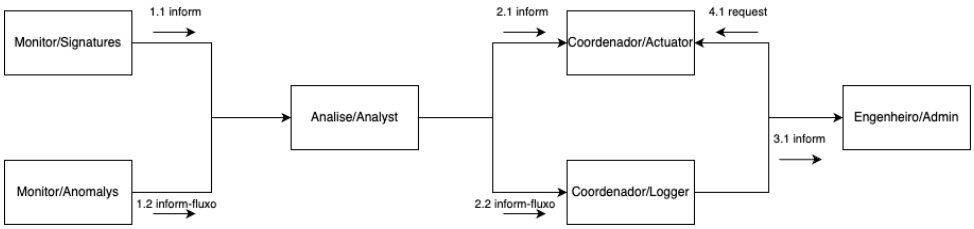

Figura 5: Diagrama de Colaboração

Da figura Figura 5, fica claro que a comunicação é feita maioritariamente num sentido, desde a função de monitorização até a respetiva atuação, e que as mesmas *performatives*, tal como referido, são utilizadas em momentos distintos e por agentes diferentes.

3.4. Inteligência dos Agentes

A inteligência dos agentes de deteção de intrusões no sistema proposto assenta numa abordagem híbrida, combinando técnicas determinísticas baseadas em regras com métodos mais flexíveis e adaptativos sustentados por *machine learning*. Esta combinação visa maximizar a capacidade de deteção de ameaças, equilibrando precisão, robustez e capacidade de adaptação a novos padrões de ataque.

Cada uma destas abordagens desempenha um papel específico e complementam-se no reforço da segurança da rede, contribuindo para um sistema de monitorização mais robusto e resiliente face à constante evolução das ameaças.

3.4.1. Assinaturas

Embora não recorra a *machine learning*, o sistema de deteção baseado em assinaturas incorpora inteligência de forma igualmente eficaz. Assenta em regras criteriosamente definidas que permitem identificar e reagir a padrões de tráfego suspeito ou malicioso.

A inteligência deste mecanismo reside na sua capacidade de: Analisar pacotes de rede em tempo real; Reconhecer comportamentos que correspondem a ataques conhecidos; Atuar de forma automática e precisa, com base nas regras definidas previamente.

Por exemplo, ao detetar um padrão de múltiplas tentativas de ligação, em curto espaço de tempo e em diferentes portas, o sistema interpreta essa atividade como um ataque de *port scanning*, desencadeando a ação correspondente, neste caso, o bloqueio do IP de origem.

As regras que definimos têm como objetivo prevenir diversos tipos de ataques, nomeadamente:

- **Port Scan:** detetado quando um mesmo IP tenta estabelecer ligações a múltiplas portas num curto intervalo de tempo. O sistema bloqueia automaticamente o IP atacante, impedindo futuras comunicações.
- **Ping Flood:** identificado quando se verifica um excesso de pacotes ICMP num curto período. Para mitigar este ataque, é imposta uma limitação na taxa de envio de pacotes ICMP por IP.
- **SYN Flood:** caracteriza-se por um volume elevado de pedidos de ligação TCP (SYN) não concluídos. A defesa consiste na limitação do número de pacotes SYN permitidos por segundo.
- **DNS Flood:** detetado através de um número anormal de pedidos UDP dirigidos à porta 53. O sistema limita o tráfego DNS por IP, rejeitando pedidos que ultrapassem o limite definido.
- **HTTP Flood:** ocorre quando um único IP tenta estabelecer um número excessivo de ligações HTTP. A resposta consiste na restrição do número de conexões simultâneas permitidas por IP.

Este sistema representa uma abordagem de inteligência determinística, assente em regras explícitas, que se tem revelado altamente eficaz na deteção e mitigação de ameaças bem definidas.

3.4.2. Anomalias

A deteção de anomalias neste sistema foi concebida como uma camada complementar à deteção por assinaturas, visando aumentar a cobertura do IDS e permitir a identificação de ataques não previamente catalogados. Para tal, adotou-se uma abordagem baseada em técnicas de *machine learning*, centrada na construção de modelos capazes de identificar desvios estatísticos no comportamento da rede.

A primeira abordagem explorada consistiu na utilização de modelos pré-treinados, baseados em conjuntos de dados públicos e genéricos. Contudo, esta solução revelou-se ineficaz. Os modelos pré-treinados, ao não refletirem com fidelidade os padrões de tráfego específicos da topologia simulada, apresentavam uma taxa elevada de falsos negativos. Esta limitação levou ao abandono desta estratégia em favor do desenvolvimento de modelos treinados com dados recolhidos diretamente no ambiente emulado. Para esse efeito, foi construído um *dataset* próprio, composto por tráfego legítimo da rede e por instâncias simuladas de ataques como *ping flood*, *port scan* e *SYN flood*.

Inicialmente, explorou-se a construção de modelos de classificação supervisionada, para identificar tipos específicos de ataque. Utilizou-se, por exemplo, o algoritmo *Random Forest* para classificar amostras em diferentes categorias de ataque. No entanto, rapidamente se constatou que esta abordagem introduzia redundância relativamente ao sistema de assinaturas já existente que, por definição, é altamente eficaz na deteção de ataques conhecidos. Assim, a abordagem supervisionada foi descartada, dando lugar à utilização de modelos de deteção de anomalias, mais alinhados com o propósito desta camada.

Entre os algoritmos considerados para deteção de anomalias, destacam-se métodos clássicos como o *One-Class SVM* e o *Isolation Forest*, ambos bastante utilizados para problemas de deteção de *outliers*. Estes modelos foram treinados unicamente com tráfego considerado "normal", sendo posteriormente utilizados para identificar desvios significativos, que poderiam corresponder a atividades maliciosas.

Paralelamente, foi também explorada uma abordagem baseada em *autoencoders*, um tipo de rede neural capaz de aprender representações comprimidas dos dados. A ideia subjacente era que, ao tentar reconstruir o tráfego normal, o *autoencoder* apresentasse erros de reconstrução significativamente maiores quando confrontado com dados anómalos. No entanto, embora conceptualmente promissora, esta abordagem não apresentou resultados superiores aos obtidos com os modelos tradicionais. Parte dessa limitação poderá ser atribuída à ausência de um processo rigoroso de otimização da arquitetura da rede, bem como à dimensão limitada do *dataset* utilizado para o treino.

Em termos práticos, nenhum dos modelos se revelou totalmente fiável quando transpostos para o ambiente simulado em tempo real. Apesar de apresentarem métricas satisfatórias em testes *offline*, a sua aplicação na deteção dinâmica resultou em diversos falsos positivos e dificuldades em manter a consistência na identificação de padrões anómalos. Estas limitações não invalidam o contributo da camada de anomalias, mas reforçam a necessidade de validação humana nos casos detetados por esta via, razão pela qual, no sistema proposto, os alertas gerados são registados em ficheiros de *log* e submetidos à análise do agente engenheiro.

Como já referido, o agente engenheiro foi concebido com o intuito de simular e, em certa medida, substituir a atuação de um administrador de rede humano. A motivação para a sua criação surgiu da observação de que, ao analisar os registos de logs, era possível distinguir com relativa facilidade os ataques reais dos falsos positivos. Assim, o agente engenheiro contribui para aumentar a fiabilidade da deteção de anomalias do sistema, sendo suficientemente confiável para tomar decisões de forma autónoma. Quando identifica uma anomalia legítima, tem a capacidade de enviar um pedido ao agente coordenador, permitindo que este atue no sentido de mitigar o ataque ou a ameaça em curso.

Apesar de o modelo de deteção por anomalias cumprir o seu objetivo principal, identificar ataques desconhecidos não cobertos pelas assinaturas, apresenta uma taxa elevada de falsos positivos. No entanto, quando complementado com o agente Engenheiro, o sistema IDS torna-se substancialmente mais robusto. Esta integração, aliada à camada de deteção por assinaturas, proporciona uma defesa sólida e abrangente contra diversos tipos de ataques.

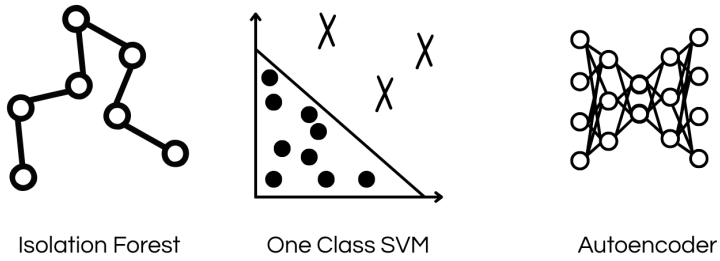

Figura 6: Modelos utilizados

3.5. Outros Aspetos

Complementarmente aos agentes, existem outros desafios do projeto, importantes para o desenho do sistema, que não sejam necessariamente um tipo ou comportamento de agente. Nesta subsecção, vamos referir aspetos, relevantes, que pertencem ao projeto, mas secundariamente.

3.5.1. Ficheiros Complementares

Um ponto importante no desenho do sistema foi a expansibilidade. IDS baseados em assinaturas são, naturalmente, dependentes da quantidade e qualidade das respetivas assinaturas. Deste modo, com o desenho de novos ataques e respetivas defesas, conceber um sistema que seja fácilmente expansível é um requisito indíspensável. Assim, optamos por separar o conjunto de assinaturas, num ficheiro designado, que pode ser posteriormente importado. Optamos por utilizar Python para este efeito, dado que todo o nosso sistema foi desenvolvido nesta linguagem, mas outra opção forte seria o uso de JSON.

```
ATTACK_SIGNATURES = {
    "port_scan": {
        "conditions": {
            "time_window": 5,  # segundos
            "min_attempts": 3
        }
    }
}

DEFENSE_SIGNATURES = {
    "port_scan": {
        "action": "block_ip",
        "description": "Bloqueia IPs que dão scan em multiplas portas num curto
intervalo de tempo.",
        "command": lambda ip: f"sudo iptables -A FORWARD -s {ip} -j DROP\n"
    }
}
```

Listagem 1: Exemplo de Assinaturas

3.5.2. Topologia

Para a simulação da rede, foi utilizado o CORE (Common Open Research Emulator), um emulador de redes que permite construir e testar ambientes de rede virtuais de forma realista. A topologia definida procurou ser relativamente complexa (dentro das limitações impostas pelo CORE) de modo a refletir um cenário mais próximo de uma rede real. Nela, foi introduzido um nó atacante, cuja comunicação com os nós protegidos só é possível passando, obrigatoriamente, por pelo menos um dos nós onde o IDS está implementado.

O nó 1 foi escolhido como o principal ponto a proteger, uma vez que, para que qualquer pacote o atinja, este tem de atravessar ambas as camadas do IDS definidas na arquitetura. Cada uma destas camadas corre num router distinto, funcionando de forma coordenada mas autónoma, reforçando a capacidade de deteção e resposta a diferentes tipos de ameaças.

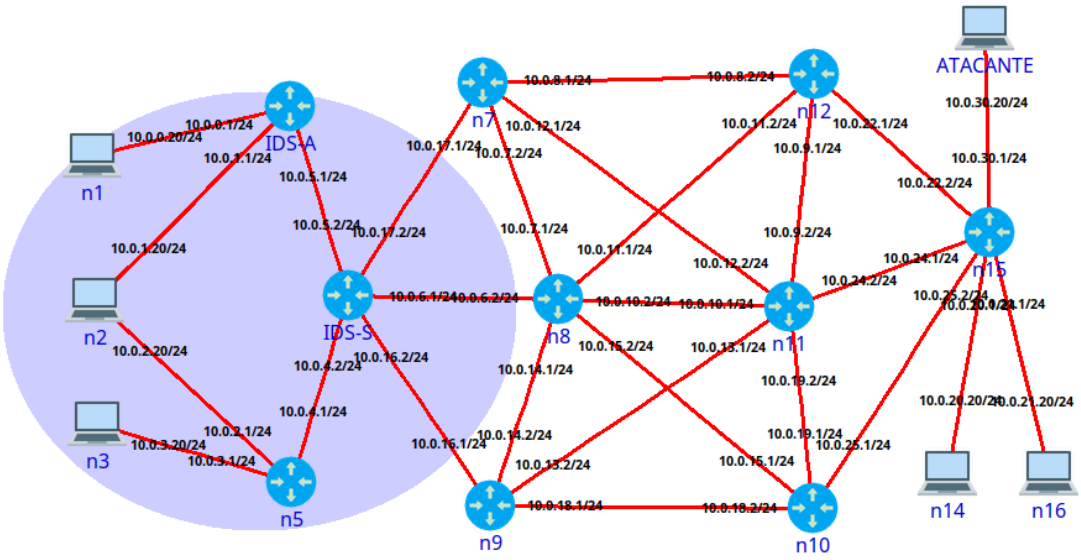

Figura 7: Topologia de rede montada

3.5.3. Ficheiros de Log

Um fator determinante para a comunicação entre o agente de coordenação e o agente engenheiro são os ficheiros de *log*. Para este trabalho, optamos por seguir uma formatação baseada em entradas, etiquetadas com a data e hora da recessão.

```
=== Anomalia Recebida - 2025-05-05 12:56:19 ===
Source IP: 10.0.5.2
Destination Port: 0    0 | 1    0 | Name: Destination Port, dtype: int64
Flow Duration: 0    0 | 1    2 | Name: Flow Duration, dtype: int64
Total Fwd Packets: 0    1 | 1    2 | Name: Total Fwd Packets, dtype: int64
Flow Bytes/s: 0     0.000000 | 1    81.969425 | Name: Flow Bytes/s, dtype: float64
SYN Flag Count: 0    0 | 1    0 | Name: SYN Flag Count, dtype: int64
RST Flag Count: 0    0 | 1    0 | Name: RST Flag Count, dtype: int64
FIN Flag Count: 0    0 | 1    0 | Name: FIN Flag Count, dtype: int64
ACK Flag Count: 0    0 | 1    0 | Name: ACK Flag Count, dtype: int64
Average Packet Size: 0    82.0 | 1    82.0 | Name: Average Packet Size, dtype:
float64
```

Listagem 2: Exemplo de entrada no ficheiro

4. Resultados

Nesta secção, apresentam-se os resultados obtidos durante os testes. Embora tenham sido realizados testes mais extensos do que os aqui documentados, será apresentado apenas um ataque efetuado com o *Nmap* (*port scanning*) à máquina alvo.

Este ataque com recurso ao *Nmap* deveria, como esperado, acionar as assinaturas previamente definidas para *port scanning* e *SYN flood* no modelo baseado em assinaturas, bem como gerar alertas nos modelos baseados em anomalias.

```
eduardo@ATACANTE:~/Secretária/IDS-ASM$ nmap 10.0.0.20
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-07 10:39 WEST
mass_dns: warning: Unable to open /etc/resolv.conf. Try using --system-dns or sp
ecify valid servers with --dns-servers: No such file or directory (2)
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
 Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 10.0.0.20
Host is up (0.00032s latency).
All 1000 scanned ports on 10.0.0.20 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds
```

Figura 8: Ataque Nmap (1)

4.1. Monitorização

A fase de monitorização é responsável pela observação contínua do tráfego de rede, analisando os pacotes em tempo real.

4.1.1. Assinaturas

No caso das assinaturas, é facilmente observável a captura e identificação dos pacotes que estão a ser detetados.

```
[Monitor] Pacote capturado do Router: [{'timestamp': '2025-05-07 10:39:53.741',
'src_ip': '10.0.6.2', 'dst_ip': '224.0.0.5', 'protocol': 89, 'size_bytes': 82, '
src_port': None, 'dst_port': None}, {'timestamp': '2025-05-07 10:39:53.741', 'sr
c_ip': '10.0.16.1', 'dst_ip': '224.0.0.5', 'protocol': 89, 'size_bytes': 82, 'sr
c_port': None, 'dst_port': None}, {'timestamp': '2025-05-07 10:39:53.741', 'src_
ip': '10.0.30.20', 'dst_ip': '10.0.0.20', 'protocol': 6, 'size_bytes': 74, 'src_
port': 50414, 'dst_port': 80}, {'timestamp': '2025-05-07 10:39:53.741', 'src_ip'
: '10.0.0.20', 'dst_ip': '10.0.30.20', 'protocol': 6, 'size_bytes': 54, 'src_por
t': 80, 'dst_port': 50414}, {'timestamp': '2025-05-07 10:39:53.741', 'src_ip': '
10.0.30.20', 'dst_ip': '10.0.0.20', 'protocol': 6, 'size_bytes': 74, 'src_port':
 52966, 'dst_port': 443}, {'timestamp': '2025-05-07 10:39:53.741', 'src_ip': '10
.0.0.20', 'dst_ip': '10.0.30.20', 'protocol': 6, 'size_bytes': 54, 'src_port': 4
```

Figura 9: Monitorização no agente de Assinaturas

4.1.2. Anomalias

Na deteção baseada em anomalias, o agente de monitorização apresenta um resultado mais compacto, pois analisa *flows* de pacotes com uma duração de 5 segundos, sobre os quais calcula estatísticas. Como o ecrã enchia-se rapidamente e tornava-se confuso, optámos por prints mais informativos do que ilustrativos.

```
[Monitor] Captura de fluxos em execução
[Monitor] Mensagem enviada!
```

Figura 10: Monitorização no agente de Anomalias

4.2. Análise

A análise consiste na interpretação dos pacotes capturados, com o objetivo de identificar potenciais ameaças ou comportamentos suspeitos.

4.2.1. Assinaturas

Nesta secção, é facilmente observável o número de pacotes recebidos, bem como os alertas acionados e os que estão a ser enviados para o agente de coordenação.

```
[Analise] 410 lista de pacotes recebidos, com 410 pacotes.
[Analise] ALERTA: {'type': 'port_scan', 'src_ip': '10.0.30.20', 'timestamp': 690
.895, 'details': 'Detetadas 3 tentativas de ligação em portas diferentes proveni
ente do ip: 10.0.30.20'}
[Analise] ALERTA: {'type': 'port_scan', 'src_ip': '10.0.0.20', 'timestamp': 690.
895, 'details': 'Detetadas 3 tentativas de ligação em portas diferentes provenie
nte do ip: 10.0.0.20'}
[Analise] ALERTA: {'type': 'syn_flood', 'src_ip': '10.0.0.20', 'timestamp': 690.
895, 'details': 'Detetados 20 pacotes TCP em 3 segundo'}
[Analise] ALERTA: {'type': 'syn_flood', 'src_ip': '10.0.30.20', 'timestamp': 690
.895, 'details': 'Detetados 21 pacotes TCP em 3 segundo'}
[Analise] A enviar alerta {'type': 'port_scan', 'src_ip': '10.0.30.20', 'timesta
mp': 690.895, 'details': 'Detetadas 3 tentativas de ligação em portas diferentes
 proveniente do ip: 10.0.30.20'}
[Analise] A enviar alerta {'type': 'port_scan', 'src_ip': '10.0.0.20', 'timestam
p': 690.895, 'details': 'Detetadas 3 tentativas de ligação em portas diferentes
proveniente do ip: 10.0.0.20'}
```

Figura 11: Análise e Alerta no agente de Anomalias

Curiosamente, aqui identifica-se um pequeno problema que encontrámos durante a elaboração do relatório: é também acionado um alerta gerado pela própria máquina a proteger (10.0.0.20). Isto acontece porque o *Nmap* realiza requisições *SYN*. Como o IDS funciona em modo *sniffer*, deixa o tráfego seguir e analisa apenas os pacotes que passam. O pacote *SYN* chega assim à máquina alvo, que responde com um *SYN ACK*.

Como existem várias requisições às quais a máquina responde, acaba por gerar também um alerta. Este comportamento é perfeitamente normal e faz sentido que o alerta seja ativado. No entanto, o agente de coordenação ignora este tipo de alerta, uma vez que verifica se o IP identificado como "atacante" corresponde ao IP da máquina protegida. Quando isso acontece, o alerta é descartado.

4.2.2. Anomalias

Aqui, é claramente visível o dataframe recebido pelo agente de análise, bem como uma amostra dos dados. Em caso de deteção de uma anomalia, o campo de previsão (*"prediction"*) assinala de forma clara a indicação de *"ANOMALY"*.

```
[Analise] Anomalia detetada no fluxo!
     Destination Port  Flow Duration  ...  Idle Min  prediction
0                  80              0  ...         0     ANOMALY
1                 443              0  ...         0     ANOMALY
2                 256              0  ...         0     ANOMALY
3                8080              0  ...         0     ANOMALY
4                3389              0  ...         0     ANOMALY
..                ...            ...  ...       ...         ...
215              5666              0  ...         0     ANOMALY
216              2401              0  ...         0     ANOMALY
217              2718              0  ...         0     ANOMALY
218               749              0  ...         0     ANOMALY
219             51493              0  ...         0     ANOMALY
```

Figura 12: Análise e Alerta no agente de Anomalias

Durante o desenvolvimento deste modelo, deparámo-nos com vários problemas relacionados com os falsos positivos gerados. Após várias tentativas e ajustes, concluímos que os dados de tráfego gerados pelo próprio CORE eram insuficientes para que o modelo conseguisse aprender um "estado normal" da rede. Por isso, decidimos criar um gerador de tráfego.

Através de três nós a gerar tráfego, construímos um dataset com cerca de 120 mil entradas. No entanto, este volume de dados revelou-se ainda insuficiente para treinar um modelo capaz de reduzir os falsos positivos de forma eficaz.

Deste problema surgiu a ideia do agente Engenheiro, uma entidade que, através da análise dos *logs*, consegue facilmente eliminar a maioria dos falsos positivos. Segue-se um exemplo de um falso positivo detetado pelo agente de Análise.

```
[Analise] DataFrame recebido com shape: (3, 77)
[Analise] Sample data:
   Destination Port  Flow Duration  ...  Idle Max  Idle Min
0                 0              0  ...         0         0
1                 0              2  ...         0         0
2                 0              4  ...         0         0

[3 rows x 77 columns]

[Analise] Anomalia detetada no fluxo!
   Destination Port  Flow Duration  ...  Idle Min  prediction
1                 0              2  ...         0     ANOMALY
2                 0              4  ...         0     ANOMALY
```

Figura 13: Análise e Falso Alerta no agente de Anomalias

Importa realçar que, apesar da deteção de ataques também identificados pelo IDS baseado em anomalias e da elevada ocorrência de falsos positivos, este modelo mantém a capacidade de detetar ataques que não estão registados nas assinaturas. Foi, por exemplo, testado com um ataque de UDP flood, tendo conseguido identificá-lo com sucesso.

Este é, aliás, o objetivo principal do modelo: detetar ataques que escapem ao sistema de assinaturas. É verdade que, nesta fase da implementação, a lista de assinaturas ainda não é muito extensa. No entanto, num contexto real, a principal função deste modelo seria precisamente

complementar essa lista, detetando comportamentos maliciosos que não sejam reconhecidos pelas regras previamente definidas.

4.3. Coordenação

Após a deteção de uma ameaça ou anomalia, o sistema age automaticamente, implementando medidas de mitigação ou registando os eventos.

4.3.1. Assinaturas

Aqui é facilmente visualizado o alerta recebido, juntamente com o comando aplicado à *firewall* e uma breve descrição do que ele faz antes de ser executado.

```
[Cordenador] Alerta Novo Recebido: {'type': 'port_scan', 'src_ip': '10.0.30.20',
 'timestamp': 690.895, 'details': 'Detetadas 3 tentativas de ligação em portas d
iferentes proveniente do ip: 10.0.30.20'}
[Cordenador] A resolver alerta 10.0.30.20 - port_scan
[Defesa] Bloqueia IPs que dão scan em multiplas portas num curto intervalo de te
mpo. em execução para o IP 10.0.30.20.
[Cordenador -> Firewall] A executar: sudo iptables -A FORWARD -s 10.0.30.20 -j D
ROP

[Cordenador] Alerta Novo Recebido: {'type': 'syn_flood', 'src_ip': '10.0.30.20',
 'timestamp': 690.895, 'details': 'Detetados 21 pacotes TCP em 3 segundo'}
[Cordenador] A resolver alerta 10.0.30.20 - syn_flood
[Defesa] Ativa proteção contra SYN flood usando regras que limitam conexoes SYN
por segundo. em execução para o IP 10.0.30.20.
[Cordenador -> Firewall] A executar: sudo iptables -A FORWARD -p tcp --syn -s 10
.0.30.20 -m limit --limit 5/second --limit-burst 10 -j ACCEPT
sudo iptables -A FORWARD -p tcp --syn -s 10.0.30.20 -j DROP
```

Figura 14: Coordenação de Alerta no agente de Assinaturas

Ao executarmos o comando "iptables -L" para listar as regras aplicadas à *firewall*, é facilmente observável que as regras foram corretamente aplicadas.

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  10.0.30.20           anywhere
ACCEPT     tcp  --  10.0.30.20           anywhere             tcp flags:FIN,SYN,
RST,ACK/SYN limit: avg 5/sec burst 10
DROP       tcp  --  10.0.30.20           anywhere             tcp flags:FIN,SYN,
RST,ACK/SYN

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figura 15: Regras de *firewall* aplicadas corretamente

Neste caso, como foram detetados dois ataques, *SYN flood* e *port scanning*, foram aplicadas duas regras sobrepostas na *firewall*. Embora esta sobreposição represente uma certa redundância, ela não acarreta problemas nem para o funcionamento da *firewall* nem ao nível da segurança. Além disso, tendo em conta que implementámos mecanismos de registo da origem dos ataques,

esta redundância não se propaga de forma significativa, não constituindo assim um problema relevante de eficiência, muito menos de segurança.

Se tentarmos realizar novamente um *Nmap*, o próprio comando responde com *"hosts seems down"*, pois não consegue aceder ao alvo.

```
eduardo@ATACANTE:~/Secretária/IDS-ASM$ nmap 10.0.0.20
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-07 10:59 WEST
mass_dns: warning: Unable to open /etc/resolv.conf. Try using --system-dns or sp
ecify valid servers with --dns-servers: No such file or directory (2)
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
 Try using --system-dns or specify valid servers with --dns-servers
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.06 seconds
```

Figura 16: Tentativa de Nmap após mitigação

4.3.2. Anomalias e Agente Engenheiro

Novamente, nas anomalias, optámos por *prints* mais informativos do que explícitos, pois o alerta, juntamente com as estatísticas geradas, é bastante extenso para ser impresso diretamente no terminal. Dessa forma, apenas é indicado que o alerta foi recebido e o IP de origem que acionou o ataque, sendo também informado que o evento foi acrescentado ao relatório de anomalias.

```
[Cordenador] Anomalia Recebida proveniente de: 10.0.30.20
[Cordenador] Relatório atualizado!
```

Figura 17: Coordenação de Alerta no agente de Anomalias

Segue então um excerto do relatório de anomalias, contendo as anomalias registadas.

```
=== Anomalia Recebida - 2025-05-05 20:28:29 ===
Source IP: 10.0.30.20
Destination Port: 0       80 | 1       443 | 2       113 | 3       993 | 4      3389 |       ...   | 213    2008 | 214    2020 | 215   16000 | 216    1022 | 217    9929 | Name: Destination Port, Length: 218, dtype: int64
Flow Duration: 0      0 | 1      0 | 2      0 | 3      0 | 4      0 |       .. | 213    0 | 214    0 | 215    0 | 216    0 | 217    0 | Name: Flow Duration, Length: 218, dtype: int64
Total Fwd Packets: 0      1 | 1      1 | 2      1 | 3      1 | 4      1 |       .. | 213    1 | 214    1 | 215    1 | 216    1 | 217    1 | Name: Total Fwd Packets, Length: 218, dtype: int64
Flow Bytes/s: 0      0 | 1      0 | 2      0 | 3      0 | 4      0 |       .. | 213    0 | 214    0 | 215    0 | 216    0 | 217    0 | Name: Flow Bytes/s, Length: 218, dtype: int64
SYN Flag Count: 0      1 | 1      1 | 2      1 | 3      1 | 4      1 |       .. | 213    1 | 214    1 | 215    1 | 216    1 | 217    1 | Name: SYN Flag Count, Length: 218, dtype: int64
RST Flag Count: 0      0 | 1      0 | 2      0 | 3      0 | 4      0 |       .. | 213    0 | 214    0 | 215    0 | 216    0 | 217    0 | Name: RST Flag Count, Length: 218, dtype: int64
FIN Flag Count: 0      0 | 1      0 | 2      0 | 3      0 | 4      0 |       .. | 213    0 | 214    0 | 215    0 | 216    0 | 217    0 | Name: FIN Flag Count, Length: 218, dtype: int64
ACK Flag Count: 0      0 | 1      0 | 2      0 | 3      0 | 4      0 |       .. | 213    0 | 214    0 | 215    0 | 216    0 | 217    0 | Name: ACK Flag Count, Length: 218, dtype: int64
Average Packet Size: 0      74.0 | 1      74.0 | 2      74.0 | 3      74.0 | 4      74.0 |        ...   | 213    74.0 | 214    74.0 | 215    74.0 | 216    74.0 | 217    74.0 | Name: Average Packet Size, Length: 218, dtype: float64
```

Figura 18: Exemplo de relatório de anomalias

Como mencionado anteriormente, quando o agente Engenheiro deteta uma anomalia através da análise dos ficheiros de logs, é emitido um aviso com o seguinte formato:

```
[Engenheiro] A analisar log...
[Engenheiro] Anomalias encontradas nestes IPs: ['10.0.30.20']
```

Figura 19: Exemplo de anomalia na *log* em agente Engenheiro

Posteriormente, são efetivamente aplicadas regras na *firewall* pelo agente Coordenador, com o objetivo de mitigar o ataque detetado:

```
[Cordenador] Request Recebido: ['10.0.30.20']
[Cordenador] A bloquear o IP: 10.0.30.20
[Cordenador -> Firewall] A executar: sudo iptables -A FORWARD -s 10.0.30.20 -j D
ROP
```

Figura 20: Mitigação de ataque no Coordenador de anomalias

Mais uma vez, ao analisarmos as regras da firewall, é possível verificar que estas foram efetivamente aplicadas, conseguindo travar com sucesso futuros ataques.

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  10.0.30.20          anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figura 21: Regras de *firewall* aplicadas corretamente (2)

4.4. Observação Adicional

Uma limitação observada ocorreu no caso de ataques de *flood*, como o *ping flood*. Embora esses ataques continuem a ser capturados pelos agentes de monitorização, tal fenómeno não indica falha na proteção. Isto acontece porque a biblioteca Scapy, utilizada para a recolha dos pacotes, opera ao nível do kernel, ou seja, antes mesmo de os pacotes serem bloqueados pela *firewall*. Em termos práticos, o ataque não chega a ultrapassar a *firewall*, o que é facilmente verificável através da utilização do *tcpdump* no nó seguinte, onde se confirma que todos os pacotes do ataque estão a ser devidamente bloqueados. No entanto, os agentes de monitorização continuam a recolher esses pacotes e a enviá-los para o agente de análise. Ainda assim, como o sistema mantém um registo de ameaças previamente detetadas, não são gerados novos alertas redundantes, evitando sobrecarga desnecessária no sistema.

5. Proteção de Dados

A proteção de dados é um aspeto essencial no desenho de qualquer IDS, sobretudo num contexto em que se monitoriza tráfego de rede. A sensibilidade da informação observada e o potencial para violações de privacidade exigem uma abordagem consciente e ética, garantindo que os mecanismos de segurança não se tornam, eles próprios, fontes de exposição indevida de dados pessoais.

No desenvolvimento do nosso IDS, adotámos uma postura de minimização de dados. O sistema opera exclusivamente com base nos cabeçalhos dos pacotes, nunca acedendo aos respetivos *payloads*. Esta decisão garante que o conteúdo das comunicações não é processado ou armazenado, reduzindo o risco de exposição de dados sensíveis. As informações analisadas limitam-se a métricas agregadas e estatísticas, como a média de tamanho dos pacotes num determinado fluxo, que não permitem identificação de utilizadores ou interpretação do conteúdo.

Outra medida importante de proteção adotada foi a exclusão deliberada dos endereços IP nos conjuntos de dados utilizados para treinar os modelos de deteção por anomalias. Embora o IP seja uma peça de informação frequentemente usada em contextos de segurança, considerámos que a sua inclusão poderia introduzir critérios discriminatórios, especialmente em situações onde o modelo pudesse aprender a associar certos IPs a comportamentos maliciosos com base

em padrões regionais. Tal preocupação é corroborada por estudos recentes, que apontam a possibilidade de discriminação geográfica associada à exposição forçada de IPs, levantando questões éticas relevantes sobre a privacidade e liberdade de expressão dos utilizadores online. [1]

Por fim, e talvez mais crucial, nenhum pacote é persistido pelo sistema. Os pacotes capturados são processados em tempo real, e imediatamente descartados após a análise, tirando partido do sistema de *garbage collection* nativo do Python. Esta abordagem garante que não existem dados históricos armazenados, eliminando riscos associados à retenção de tráfego e simplificando as exigências legais de compliance com normas como o RGPD.

Com este conjunto de medidas, procuramos garantir um equilíbrio responsável entre eficácia na deteção de ameaças e respeito pela privacidade dos utilizadores, reforçando a ideia de que segurança e ética não são mutuamente exclusivas, mas sim complementares.

Figura 22: Medidas de Proteção de Dados

6. Trabalhos Futuros

Apesar dos resultados promissores obtidos com o sistema atual, existem várias direções que poderão ser exploradas para evoluir e robustecer a solução desenvolvida:

O mecanismo de deteção por assinaturas revelou-se eficaz, mas naturalmente limitado à quantidade e abrangência das regras definidas. Assim, um dos principais vetores de desenvolvimento futuro passa pela expansão significativa do conjunto de assinaturas, incluindo novos padrões de ataque, variantes conhecidas e cenários mais complexos. Além disso, seria benéfico adotar uma abordagem modular para o carregamento dinâmico de assinaturas, possibilitando a atualização contínua do sistema sem necessidade de interrupções.

A camada de deteção por anomalias, embora funcional em ambiente controlado, apresenta desafios em tempo real, nomeadamente no que toca a falsos positivos e estabilidade da deteção. Um passo natural será trabalhar no sentido de a tornar efetivamente fiável, refinando os modelos utilizados, otimizando os parâmetros e explorando abordagens como a aprendizagem contínua (*partial learning*). Estas estratégias poderão permitir uma adaptação progressiva do sistema ao tráfego real, sem comprometer a integridade do modelo.

A arquitetura atual assenta numa hierarquia ligeiramente limitada, com um *host* centralizado na primeira camada do IDS. Uma evolução desejável (e que constituiu a ideia inicial de implementação) seria a transição para uma hierarquia verdadeiramente multinível, onde diferentes IDS, para além de autónomos, seriam individualmente hospedados. Assim, além de assumi-

rem a responsabilidade por domínios específicos, como por exemplo um por *router*, seria possível a sua total individualização. Esta abordagem permitiria uma maior escalabilidade, uma especialização regional mais eficaz nas decisõe e uma distribuição mais equilibrada da carga computacional no sistema. No entanto, esta funcionalidade não foi explorada devido a limitações do CORE, que não permite hospedar múltiplas instâncias simultâneas do Openfire, mesmo em nós distintos. Como trabalho futuro, poderia considerar-se o desenvolvimento em emuladores de redes mais avançados, como o GNS3, que, graças à maior capacidade de virtualização em cada nó, poderá permitir essa separação de hospedagem. Alternativamente, esta abordagem poderá ser testada num ambiente real, onde esta limitação deixaria de se colocar. Relativamente a esta alteração, importa salientar que a mudança em questão foi cuidadosamente ponderada e implicaria uma transformação significativa na arquitetura do MAS. Esta alteração poderia dar origem a cenários de negociação, como, por exemplo, a possibilidade de um coordenador reivindicar soberania sobre determinada política, com o intuito de negociar o local mais apropriado para a aplicação de uma regra, evitando assim redundâncias. O ponto que se levanta neste contexto é que, embora a redundância possa não ser a solução mais elegante, ela contribui para a segurança do sistema. Trata-se, portanto, de uma redundância justificada, não sendo meramente desnecessária. Esta questão mantém-se relevante para a articulação com a matéria, ainda que, neste caso específico, a sua utilidade possa ser limitada.

Surge ainda como trabalho futuro, uma ideia que surgiu durante o desenvolvimento, mas que não pôde ser implementada devido às limitações do CORE. A proposta passava por substituir o mecanismo atual de análise de logs, assente em *parsing*, por uma abordagem baseada em *agentic AI*. Embora o *parsing* se tenha revelado eficaz no contexto específico do estudo realizado no CORE, poderá não ser a solução mais adequada em cenários mais complexos e realistas. A utilização de *agentic AI* permitiria ao agente engenheiro tomar decisões mais autónomas e flexíveis, determinando de forma mais inteligente e contextual se uma determinada ocorrência corresponde ou não a uma anomalia. Esta abordagem traria uma capacidade de adaptação muito superior à atual, aproximando o comportamento do agente ao de um verdadeiro especialista humano, com a vantagem de escalar e operar de forma contínua. Contudo, esta ideia não foi concretizada devido às dificuldades associadas à comunicação com o exterior do ambiente CORE, como já referido anteriormente, e também porque, tendo em conta que o método de *parsing* se revelou totalmente adequado para o tráfego gerado no contexto do CORE, optou-se por manter essa abordagem nesta fase.

7. Conclusão

O desenvolvimento deste Sistema de Deteção de Intrusões, baseado em Agentes e Sistemas Multi-Agente, demonstrou a viabilidade de uma abordagem distribuída e colaborativa para a monitorização e mitigação de ameaças em redes informáticas. Ao integrar técnicas de deteção por assinaturas e por anomalias, o sistema proposto revelou-se eficaz na identificação e resposta a uma variedade de ataques conhecidos mantendo simultaneamente a capacidade de identificar comportamentos invulgares que possam indicar ameaças emergentes.

A arquitetura hierárquica e cooperativa adotada permitiu uma distribuição eficiente das tarefas entre agentes especializados, promovendo a escalabilidade e a resiliência do sistema, mesmo perante as limitações técnicas impostas pelo ambiente de emulação. Apesar dos

desafios encontrados, como a necessidade de centralizar a comunicação devido a restrições do Openfire e do CORE, a solução final demonstrou-se funcional e promissora. Destaca-se a eficácia da camada de deteção por assinaturas, associada a uma baixa taxa de falsos positivos, enquanto a deteção por anomalias, embora com potencial, requer ainda aperfeiçoamento para melhorar a sua fiabilidade em cenários dinâmicos.

Do ponto de vista ético, o sistema foi concebido com especial atenção à privacidade dos dados, limitando-se à análise de metadados de tráfego e evitando o armazenamento persistente de informações sensíveis. Esta abordagem está em conformidade com as boas práticas de segurança e os requisitos do RGPD.

Propõe-se, como trabalho futuro, a expansão do conjunto de assinaturas, a otimização dos modelos de *machine learning* para redução de falsos positivos e a exploração de uma arquitetura verdadeiramente distribuída, eventualmente em ambientes de emulação mais avançados ou em redes físicas reais. A incorporação de mecanismos de aprendizagem contínua e a adoção de políticas de coordenação mais dinâmicas entre agentes poderão ainda potenciar a adaptabilidade do sistema perante ameaças cada vez mais sofisticadas.

Pensamos ter cumprido todos os objetivos inicialmente propostos, apesar dos constrangimentos técnicos e operacionais enfrentados ao longo do desenvolvimento. Este trabalho foi importante para o nosso crescimento académico e técnico, permitindo-nos aplicar conhecimentos teóricos a um problema real e desafiante na área da cibersegurança. Em suma, este projeto não só validou a utilidade dos MAS na proteção de redes informáticas, como também lançou as bases para investigações futuras, sublinhando o potencial dos agentes inteligentes na construção de sistemas de defesa cibernética mais robustos, autónomos e flexíveis.

Bibliografia

[1] C. Zhang, «Ethical review of compulsory display of user IP addresses on multiple network platforms», *Journal of Social Science Humanities and Literature*, vol. 7, n.º 1, pp. 113–117, 2024, doi: 10.53469°.

Indice de Figuras