ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

M

MESTRADO
ENGENHARIA INFORMÁTICA

Visualização Avançada de Logs e Análise de Padrões

Luís Carlos Lopes Teixeira

10/2020

Visualização Avançada de Logs e Análise de Padrões
Luís Carlos Lopes Teixeira

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

MESTRADO
ENGENHARIA INFORMÁTICA

Luís Carlos Lopes Teixeira. Visualização Avançada de Logs e Análise de Padrões

10/2020

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

M

MESTRADO
ENGENHARIA INFORMÁTICA

Visualização Avançada de Logs e Análise de Padrões

Luís Carlos Lopes Teixeira

João Paulo Magalhães

Agradecimentos

A realização deste trabalho foi possível através do contributo e apoio de várias pessoas, às quais quero expressar o meu sincero agradecimento.

À Escola Superior de Tecnologia de Gestão por possibilitar todas as condições necessárias à realização desta dissertação.

À minha diretora de curso, a professora Dorabela Gamboa, pela sua disponibilidade, apoio e incentivo.

Ao meu orientador, o professor João Paulo Magalhães, pelo acompanhamento do trabalho desenvolvido e pelas correções e sugestões relevantes que foram feitas durante a orientação.

À minha família, pelo seu contínuo apoio e incentivo durante todo o meu percurso académico.

Por último, a todas as pessoas que de uma forma ou de outra se disponibilizaram para ajudar para que atingisse este objetivo.

Resumo

A evolução das infraestruturas de redes informáticas em termos de dimensão e complexidade torna cada vez mais difícil de detetar problemas ou intrusos na rede atempadamente. Os equipamentos e sistemas ligados em rede produzem grande volume de dados relativos ao seu funcionamento (*logs*) contendo múltiplos níveis de detalhes referentes a cada evento capturado. Apesar dos sistemas focados na deteção de eventos específicos (SIEM) serem os principais consumidores destes *logs*, é importante facilitar a visualização dos mesmos e analisar a evolução e estado da rede informática, de forma a melhorar ou manter o seu funcionamento.

Nesta dissertação apresentamos uma *framework* com a capacidade de armazenar, visualizar e processar ficheiros de *logs* de sistemas e equipamentos da rede através de uma base de dados orientada por grafos. Esta visualização avançada tem por objetivo simplificar a análise por parte do administrador da rede, dando-lhe uma visão interativa, integrada e em tempo real facultando a identificação dos ativos na rede bem como a análise de padrões capaz de detetar desvios ao comportamento normal/esperado da rede.

Palavras-chaves: Common Log Format, Logging, Process Mining, Graph Databases, Pattern Analysis

Abstract

The evolution of computer network infrastructures in terms of size and complexity makes it increasingly difficult to detect problems or intruders in the network in a timely manner. The network equipment and devices produce a large volume of data related to their operation (logs) containing multiple levels of details related to each captured event. Systems focused on detecting specific events (SIEM) are the main consumers of these logs, but it still important to facilitate their visualization and analyze the evolution and status of the computer network, in order to improve or maintain its normal operation.

In this dissertation we present a framework with the ability to store, view and process log files for systems and network equipment through a graph-oriented database. This advanced visualization aims to simplify the analysis by the network administrator, giving an interactive, integrated and real-time view providing the identification of the assets in the network as well as the analysis by means of pattern analysis to detect potential malfunction scenarios by considering deviations from normal/expected behavior

Keywords: Common Log Format, Logging, Process Mining, Graph Databases, Pattern Analysis

Conteúdo

Lista de Figuras

Abreviaturas

IoT (Internet of Things)

CAGR (Compound annual growth rate)

PCAP (Packet Capture)

IP (Internet Protocol)

SIEM (Security Information and Event Management)

SIM (Security Information Management)

SEM (Security Event Management)

EPS (Events per Second)

DMZ (Demilitarized Zone)

SNMP (Simple Network Management Protocol)

AD (Active Directory)

LDAP (Lightweight Directory Access Protocol)

SSO (Single Sign-On)

KDC (kerberos Key Distribution Center)

PCI DSS (Payment Card Industry Data Security Standard)

GPL (GNU General Public License)

SOC (Security Operation Center)

ACID (atomicity, consistency, isolation, durability)

Capítulo 1

Introdução

Neste capítulo é apresentada a contextualização e motivação para esta dissertação e os objetivos a atingir. É apresentada também a estrutura do restante documento.

1.1 Contextualização

Com a evolução da Internet, pequenas e grandes empresas, vêem o seu volume de tráfego em constante crescimento, assim como o aumento de equipamentos e dispositivos ligados em rede. O cenário continua a ser de evolução e prova disto é o aumento de tráfego gerado por dispositivos IoT (Internet of Things) e o crescimento de dispositivos sem fios que acedem as redes móveis em todo o mundo.

Com base no estudo realizado pela Cisco, Annual Internet Report [3], em 2018, existiam 8,8 mil milhões de dispositivos móveis e ligações globais, que crescerão para 13,1 mil milhões em 2023, dando uma taxa de crescimento anual composta (CAGR - Compound annual growth rate) de 8%. Estes números são ilustrados na Figura 1.

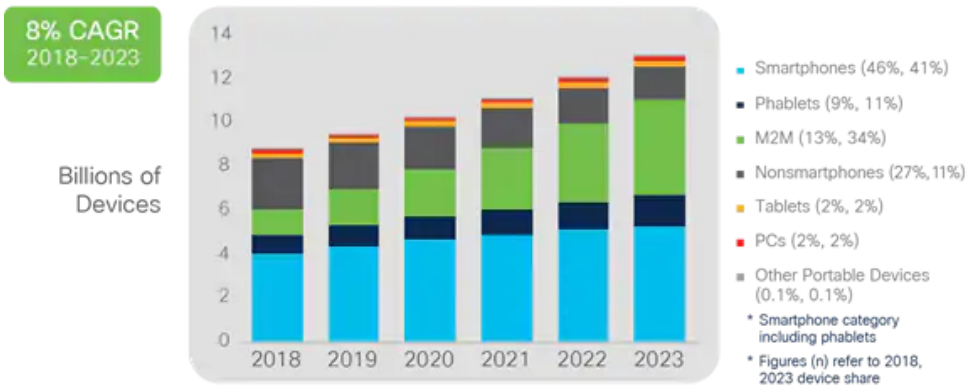


Figura 1: Cisco Annual Internet report [3]

De acordo com o apresentado em [3], até 2023, haverá 8,7 mil milhões de dispositivos pessoais, portáteis ou móveis, e de 4,4 mil milhões de ligações Machine to Machine (M2M), tal como os sistemas GPS em automóveis, os sistemas de rastreio de ativos nos sectores de transporte e produção, ou aplicações médicas que disponibilizam mais facilmente os registos dos pacientes e o estado de saúde. A nível regional, a América do Norte e a Europa Ocidental terão o crescimento mais rápido em dispositivos móveis e ligações, com 16% e 11% CAGR de 2018 a 2023, respetivamente. Como apresentado no estudo realizado pela Cisco [3] é expectável este crescimento de dispositivos, irá de algum modo afetar as redes internas das organizações, mais propriamente as redes sem fios, o que levanta desafios. Um desses desafios será a maior dificuldade em gerir e identificar cada evento de rede gerado pelos dispositivos e, entender todo o seu percurso dentro da organização, nomeadamente se o seu percurso/comportamento é normal ou poderá ser representativo de um comportamento anormal eventualmente malicioso ou prejudicial para a organização.

Considerando o desafio apresentado torna-se cada vez mais importante uma análise contínua para determinar melhorias de performance na rede e saber se existem acessos indevidos ou com finalidade maliciosa. É cada vez mais importante e necessário tirar partido dos *logs* provenientes de cada equipamento ou dispositivo na rede. Porém e considerando *logs* de diferentes tipos de informação e especificidade. Apesar dos esforços e tecnologias, mantém-se difícil interpretar cada um individualmente e ainda mais complexo, agregar eventos que os atravessam. Neste âmbito, continua a ser necessário um sistema inovador capaz de analisar o tráfego em tempo real e apresentar estes *logs* de forma estruturada e organizada, para uma fácil e rápida interpretação do estado da rede e permitindo atuar em conformidade e em tempo útil.

Para conseguir perceber o fluxo da rede, ou seja, o caminho percorrido desde o endereço de origem até ao endereço de destino, assim como a porta ou protocolo utilizado e tempos de acesso/ação, é necessário que sejam efetuadas análises em tempo real. Como o processo manual de análise de *logs*, é bastante moroso e pode não ser confiável, são utilizadas ferramentas (abordadas no capítulo seguinte) capazes de remover a informação desnecessária e apenas processar os dados importantes existentes na rede combinando-os entre si.

1.2 Motivação e Objetivos

Nesta dissertação exploramos a transformação dos *logs* de rede e dos sistemas para um formato visual, representado sobre a forma de grafos. O principal objetivo é auxiliar os administradores de redes e analistas de segurança, a interpretar e interagir com os *logs* de cada equipamento e sistema registado na rede. A transformação de um log de texto para um formato rico de visualização, como por exemplo um grafo, facilita, a deteção de problemas de desempenho na rede ou até mesmo a deteção de comportamentos anómalos que possam estar relacionados com falhas ou com a segurança informática. É também possível analisar o tráfego da rede de forma interativa e integrada.

Para alcançar tal objetivo, há um conjunto itens que necessitam de ser atendidos. Entre os quais:

- Recolher, tratar e armazenar os vários *logs* de sistema e equipamentos da rede;
- Processar esses *logs*, para aplicar os filtros ou os algoritmos necessários;
- Interligar os dados processados, com uma base de dados orientada a grafos;
- Visualizar as relações das comunicações, através dos grafos;
- Derivar datasets a partir dos grafos, para análise de padrões
- Estudar a eficácia dos modelos;
- Permitir análises em tempo real.

Desta forma, propomos uma *framework* focada na visualização avançada, interativa e estruturada, após o processamento dos *logs* dos vários sistemas e equipamentos da rede. Desta

forma será criada uma visualização sobre a forma de grafos, onde visualmente se pode interagir com o registo do fluxo de tráfego, de uma forma detalhada e organizada, abstraindo o administrador da rede ou analista da leitura intensiva dos vários formatos e detalhes desses *logs*. Com uma visualização mais ampla e de fácil compreensão é possível seguir todo o histórico de um equipamento ou evento específico, com o objetivo de determinar o seu trajeto dentro da rede em análise. Tal organização de dados, ainda possibilita uma análise de padrões, na medida em que o histórico do tráfego esperado de um equipamento respeita um padrão temporal e quando esse padrão se altera é sinónimo de alterações no seu comportamento que merecem maior atenção. Neste âmbito, a *framework* permite a deteção de desvios no comportamento dos dispositivos na rede, facilitando a sua análise e consequentemente a tomada de ações corretivas/preventivas necessárias. Esta compreensão, obtida através da monitorização e visualização avançada permite identificar ativos e pontos de rede que careçam de ser incluídos ou ajustados na configuração de outras soluções de monitorização de rede. A título de exemplo, e para clarificar esta vantagem, o resultado obtido com a *framework* poderá ser utilizado para desenvolver regras de alarmística no SIEM, a fim de monitorizar mais de perto um dado equipamento que ainda não estava a ser monitorizado, ou então que alterou o seu modo de funcionamento, para que sejam ajustadas regras, a fim de o monitorizar de forma adequada.

1.3 Estrutura da dissertação

Esta dissertação encontra-se organizada em capítulos. No capítulo 2 é apresentado o estado da arte e as ferramentas relacionadas. No capítulo 3 é apresentada a *framework* desenvolvida. No capítulo 4 apresentamos cenários de rede utilizados no estudo experimental e os resultados obtidos. Por último, no capítulo 5 é feita a conclusão do trabalho.

Capítulo 2

Estado da arte e ferramentas relacionadas

Neste capítulo é apresentado o estado da arte relacionado com a monitorização avançada de rede. Aborda nomeadamente a utilização de bases de dados orientadas a grafos no contexto da monitorização de sistemas e a recolha e análise de dados de rede, por exemplo, o modo eficiência. É também apresentado um conjunto de ferramentas comummente utilizadas para auxiliar o processo de monitorização.

2.1 Introdução

A monitorização do tráfego de rede é, do ponto de vista da segurança informática, um aspeto muito importante. Para facilitar tal monitorização são utilizadas ferramentas como os SIEM's. Um SIEM permite monitorizar, estruturar e armazenar pedaços/partes principais do tráfego da rede, como endereços de IP (Internet Protocol), portas e protocolos de comunicação utilizados, informação do tráfego dos equipamentos de rede (*router, switch, Access Point*), informação dos dispositivos (servidor, computador, laptop e outros), por último o registo da data e hora desses eventos. Através destes dados é possível, criar, armazenar e apresentar as correlações entre eles, para que todo o histórico dos dispositivos associados às redes da organização seja disponibilizado a qualquer instante. Esta necessidade de monitorizar a rede, tem como principal objetivo proteger os equipamentos de ataques informáticos, vulneráveis a possíveis utilizações indevidas na rede. São exemplos disso:a visualização de conteúdos não

permitidos (redes sociais, gambling, entre outros); identificação de um dispositivo que está a realizar comunicações suspeitas; ou um dispositivo possivelmente infetado com malware que se encontra a realizar exfiltração de dados confidenciais ou sensíveis, para o exterior.

Cada organização tem as suas prioridades/necessidades consoante cada rede informática e conforme o seu tipo de mercado. O conhecimento das prioridades e necessidades é fundamental para afinar os sistemas de monitorização, de modo a relacionar todo o tráfego da rede, a monitorizar possíveis comunicações suspeitas ou desviantes do comportamento normal e a mitigar através de medidas adequadas para proteger os recursos corporativos.

2.2 Monitorização ao nível dos dispositivos da rede

A monitorização ao nível dos dispositivos da rede pressupõe a utilização de agentes ou serviços capazes de recolher informação sobre os dispositivos a monitorizar. Estes agentes ou serviços de monitorização baseiam-se na utilização de aplicações/ferramentas que estabelecem uma comunicação ativa com o servidor de monitorização. Cada agente é instalado no dispositivo a monitorizar. Os agentes procedem à recolha de dados adotando um de dois métodos distintos: *agentfull* ou *agentless*.

A monitorização com agentes do tipo *agentfull* baseia-se na instalação de uma aplicação no dispositivo alvo, de forma recolher informação localmente como o estado dos equipamentos, serviços em carga, comunicações efetuadas, entre outros. [33] A monitorização sem agentes ou *agentless*. não necessita da instalação de uma aplicação específica. Neste caso a monitorização tira partido de protocolos standard de monitorização como o SNMP (Simple Network management Protocol), obtendo-se através do mesmo informação sobre o dispositivo. O SNMP é muito utilizado nos equipamentos de rede como *routers* ou *switchs*. [5]

É possível analisar o comportamento de qualquer dispositivo que esteja ligado à rede, como computadores, portáteis ou telemóveis, sem o recurso a agentes de monitorização. Para tal é necessário obter os dados dos equipamentos de rede, que são responsáveis por interligar todos os equipamentos e dispositivos por exemplo um core *switch*. Desta forma não é possível obter informação tão detalhada quando comparado ao uso de agentes instalados nos equipamentos, mas a nível da camada de rede é possível obter e analisar as comunicações efetuadas.

Nas próximas subsecções serão abordados alguns dos métodos de monitorização existentes no mercado e como podem ser escolhidos perante as necessidades da rede/infraestrutura tecnológica da organização.

2.2.1 SNMP versus Syslog

As ferramentas de monitorização tendem a utilizar vários tipos de agentes de monitorização. O protocolo SNMP e o Syslog são dois exemplos de protocolos que assistem a muitas ferramentas de monitorização. O SNMP é um conjunto de normas de comunicação de dispositivos numa rede de Transmission Control Protocol (TCP)/IP, tais como servidores, *routers* e *switchs*. Através do SNMP é possível obter informações como, o uso da rede, a largura de banda, monitorizar o tempo de funcionamento e executar remotamente instruções. [20] O Syslog é neste aspeto diferente. System Logging Protocol (Syslog) é um protocolo padrão utilizado para enviar registos do sistema ou mensagens de eventos para um servidor específico, chamado servidor syslog. É utilizado principalmente para recolher vários dados de dispositivos diferentes e alojar esses dados num local central para monitorização e revisão. Estes eventos são tipicamente registados localmente e podem ser revistos e analisados por um administrador, no entanto, a monitorização de equipamento a equipamento torna-se mais demorada e difícil de realizar. O Syslog ajuda a resolver este problema reencaminhando esses eventos para um servidor centralizado. Ao contrário de outros protocolos de monitorização (ex. SNMP) não existe nenhum mecanismo para inquirir os dados no syslog. [9] É comum utilizar-se uma combinação do SNMP e Syslog, de forma a garantir uma auditoria completa e mais proactiva para com o sistema agregador de *logs*, como nas soluções de monitorização SIEM.

2.2.2 Ferramentas de monitorização com agentes próprios

Existem ferramentas, que dispõem de um conjunto de agentes, que uma vez colocados no sistema a monitorizar procedem à recolha da informação, com base em ações periódicas e pré-definidas. Através dos agentes de monitorização espalhados pelos vários equipamentos na rede, é possível identificar o seu estado e comportamento. O objetivo é encontrar problemas de performance e melhorar a sua disponibilidade e eficiência, relativamente aos serviços, a serem executados nos equipamentos alvo. Desta forma é necessária uma ferramenta, capaz de armazenar o estado e comportamento de cada equipamento e apresentar esses resultados

ao administrador de rede.

Para tal pode ser utilizado como por exemplo a ferramenta Nagios [2] que permite recolher periodicamente a informação do estado dos equipamentos ou dispositivos, como computadores, servidores, *routers*, *switchs*, e também a monitorização sobre serviços de rede, tais como ftp, http, ssh, informação da carga do processador e memória, memória utilizada, espaço do disco, entre outros.

Estas ferramentas centralizam numa consola, o estado da rede e sistemas. Mantendo uma monitorização constante de todos os equipamentos da organização o que permite detetar problemas mais rapidamente.

2.3 Sistemas de monitorização ao nível da rede

Ainda que através do SNMP seja possível fazer monitorização de equipamentos ou que através do Syslog seja possível centralizar informação vinda de *logs* locais a cada sistema, as redes corporativas e os seus administradores vêem-se a braços, com uma imensidão de eventos que ocorrem a uma frequência elevada, tornando o seu processo de análise complexo. Sistemas de monitorização como o Nagios são úteis para centralizar a monitorização e definir um conjunto de ações a despoletar, em função do resultado da monitorização. E contrapartida têm de ser instalados equipamento a equipamento e não providenciam uma análise integrada e correlacionada dos eventos que ocorrem.

Neste âmbito, a existência de sistemas avançados capazes de lidar com um volume elevado de tráfego, capazes de analisar, guardar e correlacionar eventos em tempo real, através de um processo de monitorização contínuo são uma mais valia, para detetar, reportar e responder a possíveis ataques ou falhas na infraestrutura. Estes sistemas são normalmente implementados num servidor, onde todo o processo visa monitorizar a infraestrutura de rede e dados que nesta circulam. A utilização destas ferramentas de monitorização ao nível da rede, permite:

- Correlacionar os dados de qualquer fonte em tempo real para detetar incidentes antes que se tornem uma violação;
- Recolher, guardar e analisar qualquer evento de qualquer fonte dados de rede a qualquer momento;

- Integrar a equipa de SOC (Security Operation Center) em todas as operações da rede e nos seus serviços: CMDB (Configuration Management Database), business intelligence, segurança de e-mail, segurança de aplicações, entre outros;
- Capacidade de integrar feeds de inteligência de ameaça de terceiros, para uma deteção de ameaças mais precisa;
- Capacidade de visualização e reporte em *dashboards* personalizadas e/ou relatórios, a pedido ou agendados, para os administradores, analistas ou auditores.

Ao nível da rede temos sistemas como os IDS ou IPS e mais recentemente os SIEM's que atuam como agregadores e gestores de eventos facultando correlação de eventos.

2.3.1 IDS e IPS

A utilização de IDS e IPS tem como objetivo analisar o tráfego que circula nos *routers* ou *switchs* e o tráfego da placa de rede do equipamento em questão, possibilitando a deteção de intrusões (Intrusion Detection System) [29] ou o bloqueio automático dessas tentativas de intrusão IPS (Intrusion Prevention System) [29]. A recolha de dados é analisada através de identificadores de comportamento. Estes padrões ou assinaturas estão centralizados numa base de dados, que permite identificar comportamentos maliciosos e consoante o modo de operação alertar ou prevenir a propagação desse tráfego na rede. A diferença entre um IDS e um IPS é ilustrada na Figura 2.

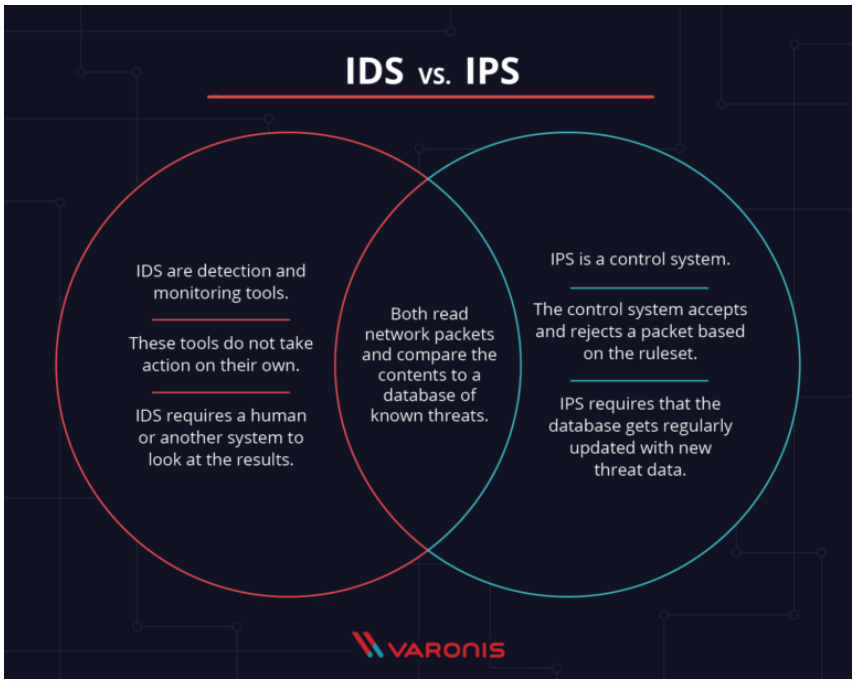


Figura 2: IDS versus IPS [29]

Uma das plataformas líder em monitorização de segurança de redes é o Zeek Network Security Monitor (também conhecido como Bro Logs). [1] Trata-se de uma ferramenta open source de monitorização que pode ser considerada como um packet sniffer, permitindo segregar os dados recebidos para o formato Bro Logs. O formato é ilustrado na Figura 3. Os dados recebidos da captura de rede são estruturados o que por sua vez, proporciona uma análise rápida e eficiente mesmo perante um grande volume de dados.

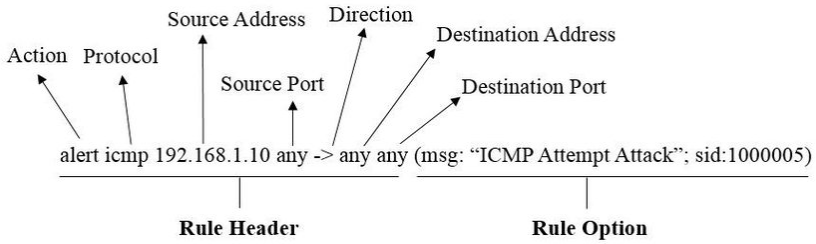


Figura 3: Formato Bro Log [26]

Com os dados da captura de tráfego uniformizados torna-se mais simples examinar o tráfego de rede contra um conjunto de regras, e de seguida alertar os administradores de sistemas sobre a atividade suspeita na rede para que possam tomar as medidas adequadas. Para este propósito é possível configurar uma ferramenta como o Snort IDS. [26] Esta ferramenta pode operar em três modos distintos:

- Sniffer mode - permite ler os pacotes de rede e exibi-los na consola/interface;
- Packet logger mode - permite armazenar os pacotes da rede em disco;
- Network intrusion detection mode – permite monitorizar o tráfego da rede e analisá-lo com base nas regras definidas pelo administrador, que desta forma, será capaz de tomar uma ação especificada, como por exemplo alertar sobre o tráfego considerado suspeito.

Um IDS como o Snort é colocado em determinados pontos da rede, para possibilitar a monitorização do tráfego dos vários dispositivos espalhados pela rede. Uma vez detetado um ataque ou um comportamento suspeito, despoleta um alerta para que o administrador do sistema tome medidas corretivas.

2.3.2 Gestores de eventos (SIEM)

Para fazer face à quantidade e diversidade de eventos, as organizações têm vindo a adotar o uso de gestores de eventos, vulgarmente designados por SIEM. O objetivo destes sistemas é agregar os dados de outros sistemas alcançando toda a rede e permitindo identificar e alertar comportamentos suspeitos, problemas nos equipamentos, falhas de comunicação entre equipamentos ou serviços, entre outros tipos de alertas possíveis de configurar. Estes sistemas têm vindo a ganhar expressão quando comparados a sistemas de monitorização autónomos.

Os SIEMs com a sua capacidade de análise em tempo real e de armazenar dados durante longos períodos, torna-se numa ferramenta poderosa, pois permite realizar o processamento de dados do tráfego da rede e a interligação desses dados já trabalhados. Um SIEM possibilita conhecer o comportamento diário da rede em monitorização, e sempre que deteta um desvio de comportamento. Por exemplo, uma máquina começou a realizar comunicações a um domínio nunca antes acedido em comparação com os outros equipamentos da rede, gera um alerta e o analista deve dar a conhecer à ferramenta se esse comportamento de facto é considerado suspeito, no caso de ser um comportamento expectável, o analista deve assignar esse evento como normal, o que permite à solução assimilar esse comportamento como fidedigno para nos próximos eventos não serem gerados alertas.

Ao longo das próximas subsecções será dado a conhecer o funcionamento de ferramenta de SIEM, como exemplo o QRadar e Darktrace. Será explicado o processo de gestão de eventos, a necessidade das organizações se tornarem compliance com algumas medidas, de forma

a assegurar no mercado níveis mínimos na área de segurança da informação, para com os seus clientes e dar a conhecer como se pode enriquecer, os dados existentes permitindo por exemplo estabelecer uma relação direta entre um endereço de IP e nome de um computador, de forma a identificar o colaborador responsável pelo mesmo.

2.3.2.1 Soluções de SIEM

As soluções SIEM são consideradas uma mais valia para a monitorização do estado dos sistemas e redes, sendo utilizada particularmente no âmbito da segurança informática para a correlação de eventos e deteção de vulnerabilidades e ciberataques.

Dois exemplos de ferramentas no âmbito da monitorização e deteção de incidentes de segurança são o QRadar da IBM e a solução da Darktrace. Ambas têm como objetivo monitorizar e gerar alarmística consoante a deteção de comportamentos maliciosos.

A ferramenta IBM QRadar [11] é uma das mais utilizadas no mercado empresarial pela sua capacidade de armazenar, correlacionar e apresentar os vários eventos já processados num formato estruturado, e que permite ao analista facilmente identificar. Tem capacidade de processar até 75 mil eventos por segundo (EPS) e recolher dados de mais de 250 dispositivos fazendo a agregação de eventos. Este sistema de monitorização coleciona os dados de *logs* da organização provenientes das suas redes informáticas, dos dispositivos de rede, agentes de monitorização, diretamente de cada sistema operativo ou aplicação, conseguindo correlacionar uma variedade de informação de forma a identificar atividades e comportamentos de cada equipamento na organização. Analisa os dados registados em tempo real, permitindo aos analistas identificar e parar rapidamente possíveis ataques ou comportamentos maliciosos. Também permite integrar feeds de inteligência de ameaça de terceiros. Na Figura 4 é ilustrado o processo de recolha de dados, em que se procede à normalização desses dados para serem processados e armazenados em disco. Em resultado do processamento podem ser despoletados alertas de comportamentos maliciosos, ou a geração de relatórios diários por exemplo o número de utilizadores ligados remotamente ao escritório.

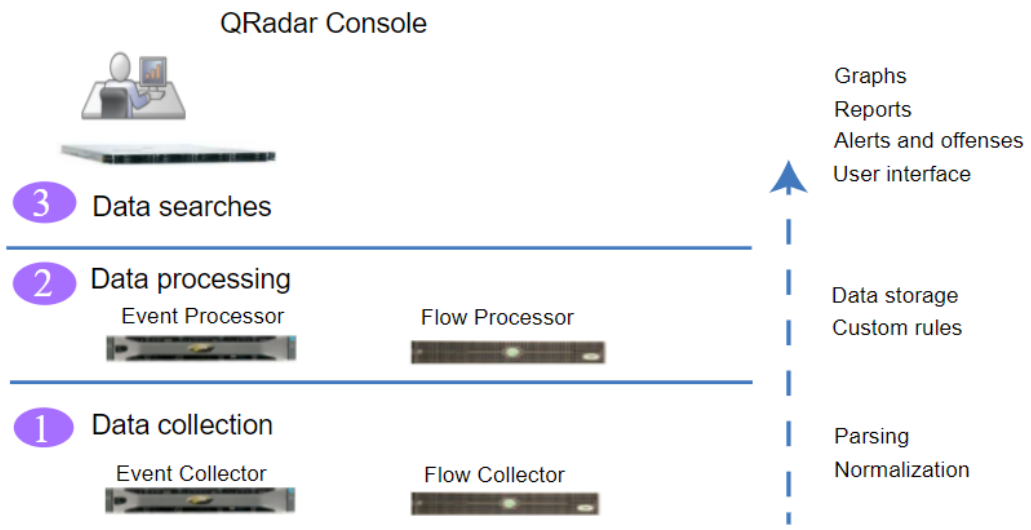


Figura 4: Funcionamento geral do SIEM IBM QRadar [11]

A solução da Darktrace [4] tira parido de aprendizagem automática sobre o estado da rede, usando inteligência artificial. A partir da aprendizagem, esta permite por exemplo detetar desvios no comportamento padrão de uma ou mais máquinas e desta forma são gerados alertas para notificar o analista. A Darktrace foi fundada recentemente por matemáticos da Universidade de Cambridge e por especialistas em inteligência cibernética dos governos americanos e britânicos. A empresa é hoje reconhecida como uma das maiores empresas de AI (Artificial Intelligence) para segurança cibernética no mundo. Altamente especializada em matemática e machine learning, além de sua vasta experiência operacional na defesa de patrimónios nacionais críticos, que permite às organizações defenderem os seus sistemas contra as mais sofisticadas ameaças cibernéticas que existem. A solução faz uso de uma tecnologia denominada Enterprise Immune System. Trata-se de tecnologia pioneira que aplica a AI à complicada questão da defesa cibernética. Os resultados revelaram que a solução é capaz de detetar ameaças cibernéticas que os sistemas mais antigos não são capazes de detetar. A aplicação da inteligência artificial para defesa cibernética marcou uma mudança fundamental na capacidade de proteger sistemas de dados críticos e infraestruturas digitais. Embora as soluções baseadas em regras e assinaturas ofereçam alguma proteção contra ameaças pré-identificadas, a realidade é que os ataques de hoje evitam essas regras e assinaturas. Uma ferramenta de aprendizagem automática não supervisionada dá resposta a estas ameaças antes de se tornarem uma crise. A plataforma Enterprise Immune System e as suas capacidades são ilustradas na Figura 5.

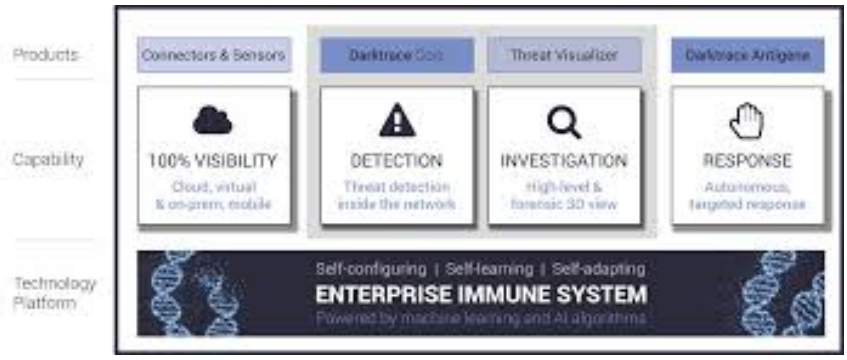


Figura 5: Enterprise Immune System da Darktrace [4]

2.3.2.2 Processo de gestão de eventos nos SIEMs

Tal como apresentado em [12], o processo de gestão de eventos pode ser descrito da seguinte forma:

1. **Recolha de dados** – Todas as fontes de informação da rede, por exemplo, servidores, sistemas operativos, *firewall*, software de antivírus, sistemas de prevenção de intrusões (IDS/IPS) são configurados para alimentar com os dados de eventos para uma ferramenta SIEM. A maioria das ferramentas modernas de SIEM usam agentes para recolher registos de eventos de sistemas empresariais, que são depois processados, filtrados, no entanto alguns SIEMs permitem a recolha de dados sem agente.

2. **Políticas** – Podem ser criados perfis pelo administrador ou analista, que definem o nível de acessos tanto em condições normais como em possíveis incidentes de segurança. Os SIEMs fornecem também regras padrão, alertas, relatórios e dashboards que podem ser parametrizados para adequar às necessidades de segurança específicas.

3. **Consolidação e correlação de dados** – Os SIEM consolidam e analisam dados recolhidos, para que esses eventos sejam categorizados com base nos filtros previamente aplicados. Com base nas regras de correlação, combinam esses eventos de dados individuais de forma a identificar possíveis questões de segurança com um contexto mais transversal. A título de exemplo podem combinar dados de autenticação numa Active Directory com os *logs* de base de dados associado a um dataleak.

4. **Notificações** – Se um evento ou conjunto de eventos desencadear uma regra com um determinado padrão de eventos, o sistema despoleta um alerta de forma a notificar a equipa responsável.

Este processo é executado continuamente pelo SIEM contribuindo para a monitorização proactiva da rede, sistemas e utilizadores.

2.3.2.3 Compliance SIEMs

É de salientar que as ferramentas SIEM podem ajudar uma organização a tornar-se compatível com determinados regulamentos ou requisitos legais.

O Payment Card Industry Data Security Standard (PCI DSS) é uma norma de segurança garante aos clientes da empresa que o seu cartão de crédito e dados de pagamento permanecerão a salvo de roubo ou uso indevido. Neste âmbito um SIEM pode satisfazer requisitos de PCI DSS, entre os quais [12]:

1. **Deteção não autorizada de ligação à rede** – As organizações em conformidade com o PCI DSS precisam de um sistema que detete todas as ligações de rede não autorizadas de/para os ativos de TI de uma organização. As soluções SIEM podem ser utilizadas como tal sistema.
2. **Procura de protocolos inseguros** – Um SIEM é capaz de documentar e justificar a utilização dos serviços, protocolos e portos permitidos de uma organização, bem como funcionalidades de segurança documental implementadas para protocolos inseguros.
3. **Inspecione os fluxos de tráfego através da DMZ** – As organizações em conformidade com o PCI precisam de implementar uma DMZ que gere as ligações entre redes não fidedignas (por exemplo, a Internet) e um servidor Web. Além disso, o tráfego de entrada na Internet para IPs dentro da DMZ deve ser limitado, enquanto o tráfego de saída que lida com os detalhes do titular do cartão deve ser avaliado.

As soluções SIEM podem satisfazer estes requisitos inspecionando o tráfego que flui através da DMZ de e para os sistemas internos, reportando sobre questões de segurança.

2.3.2.4 Enriquecimento de dados dos SIEMs

De maneira a enriquecer os dados do tráfego nas redes informáticas, os SIEM podem utilizar mecanismos de identificação, os quais permitem identificar unicamente um equipamento ou

dispositivo, de forma a detalhar todo o seu comportamento. Um dos mecanismos utilizados para enriquecer os SIEM passa por utilizar o registo de utilizadores nos sistemas de Active Directory e kerberos. Estes sistemas são responsáveis por autenticar e autorizar o acesso aos vários serviços existentes na organização. A Active Directory (AD) [15] é desenvolvido pela Microsoft para gerir a autenticação de seus sistemas operativos Windows.

O Kerberos dispõe de um serviço denominado Key Distribution Center (KDC) [28] que disponibiliza serviços de autenticação permitindo identificar cada utilizador na rede, de forma a garantir os acessos necessários para cada utilizador consoante o seu perfil de acessos na rede. É utilizado nos sistemas de monitorização de forma a associar o endereço de IP de cada dispositivo na rede a um utilizador/identificador único da organização.

De forma tornar possível numa organização, com um volume elevado de tráfego e acessos a vários equipamentos e serviços, identificar todos os equipamentos da rede com um identificador único, neste caso o identificador de cada funcionário da organização. Assim é possível, identificar quem foi o responsável por um comportamento anormal dentro da organização ou quem possivelmente terá um equipamento comprometido, com o objetivo de implementar as medidas preventivas necessárias.

Fortinet é uma solução de segurança de rede (*firewall*) que protege os utilizadores/organizações do tráfego indesejado. Baseiam-se na simples ideia de que o tráfego de ambientes menos seguros deve ser autenticado e inspecionado antes de se deslocar para um ambiente mais seguro.[8]

As *firewalls* impedem que utilizadores, dispositivos e aplicações não autorizados entrem num ambiente ou segmento de rede protegido e bloquear malware à entrada. Com base num conjunto de regras pré-programadas, podem também impedir que os utilizadores dentro da rede, acedam a determinados serviços online, podendo ser utilizados para segmentar a rede de forma a controlar e gerir o acesso a recursos internos pelos utilizadores a sistemas não autorizados. Estas remontam aos primeiros dias da Internet, e evoluíram para acompanhar o ritmo acelerado de mudança na indústria da cibersegurança. Tradicionalmente, são colocadas *firewalls* no perímetro da rede, para determinar que tipos de tráfego deixaram entrar na rede e quais devem ser bloqueados e desta forma monitorizar e proteger os recursos cooperativos, como é apresentado na Figura 6 um esquema de rede simples com uma *Firewall*.

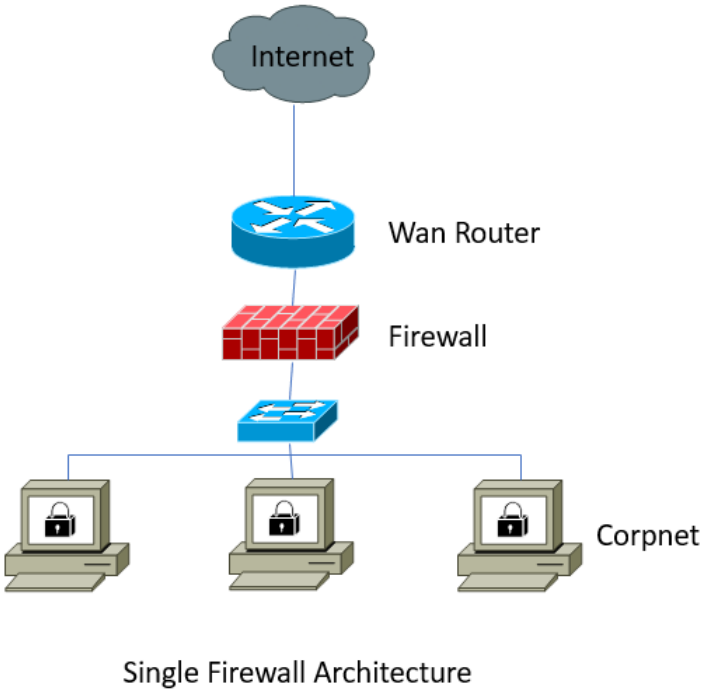


Figura 6: *Firewall* Architecture [7]

Embora uma *firewall* de última geração já não possa defender sozinha uma rede contra o complexo cenário de ameaça cibernética, estes dispositivos ainda são considerados fundamentais para criar um sistema de defesa cibernética adequado. Como parte da primeira linha de defesa contra ciberataques, as *firewalls* oferecem monitorização e filtragem essenciais a todo o tráfego, incluindo aplicações, comunicações online, e conectividade entre outros. Desta forma permite às organizações ter uma camada de proteção como de monitorização. Como as *firewalls* são responsáveis por todo o tráfego dentro da organização é possível analisar esse tráfego de *logs* com ferramentas de monitorização. Posteriormente, interligar assim estes dados com o tema deste projeto.

2.4 Base de dados orientada a grafos

Muitas das soluções de monitorização fazem uso de base de dados relacionais e ficheiros para armazenar os dados de configuração. Enquanto as primeiras obrigam à normalização dos dados para poderem ser armazenados elevando o tempo e a complexidade no processamento de *logs*, o segundo limita a capacidade de pesquisa e relacionamento entre eventos do log. Neste âmbito, as bases de dados NoSQL têm vantagens, pois permitem armazenar os dados

sem uma estrutura formal pré-definida, são eficientes tanto para o armazenamento como para a consulta de dados.

Base de dados NoSQL significa “não apenas SQL” é uma abordagem de base dados não tabular que oferece esquemas flexíveis para o armazenamento e recuperação de informação desenhados de forma diferente ao contrário das tabelas tradicionais de base de dados relacionais [6]. As bases de dados NoSQL são conhecidas pela sua facilidade de desenvolvimento, funcionalidade e desempenho em escala. Grandes empresas como Amazon, Google, Facebook, entre outras, após se depararem com problemas, como o grande volume de dados (Big Data), a escalabilidade, e a flexibilidade nos modelos de dados e alto desempenho, optaram por fazer uma mudança nos seus sistemas para conseguirem lidar facilmente com estes problemas. Isto levou a que estas e outras empresas, optassem pela utilização de base de dados NoSQL [19]. Com o passar do tempo e face às necessidades surgiram diferentes tipos de base de dados:

- Document databases: Armazena dados de forma semelhante a JSON (JavaScript Object Notation [14]). Cada documento é constituído por pares de campo e valores. Este documento normalmente se assemelha ao objeto ou modelos utilizados no desenvolvimento da aplicação em que os valores podem adotar uma variedade de tipos, como string, números, booleanos, entre outros. Uma vez da sua genérica implementação dos tipos de dados este podem ter casos de uso variados.
- Key-value databases: É uma simples forma de armazenar dados em que cada “item” é constituído por chaves e valores. Isto torna-se útil quando num conjunto de dados grandes precisamos de fazer consultas sobre dados específicos e, assim não precisamos de fazer consultas complexas para obter o resultado de um dado específico, podemos aceder a essa informação através da chave.
- Wide-column stores: Permite armazenar dados em tabelas, linhas e colunas dinamicamente, o que oferece uma grande flexibilidade em relação às bases de dados relacionais, porque não é necessário que cada linha tenha as mesmas colunas.
- Graph databases: Permite armazenar dados em nós e arestas. De certa forma, os nós armazenam informação importante como pessoas, lugares, entre outros, em quando que as arestas armazenam a informação entre os nós. Em situações em que se torna importante relacionar e cruzar os dados para descobrir padrões torna-se importante esta lógica de base de dados por grafos.

Dentro dos vários tipos de base de dados NoSQL existentes, a base de dados orientada a grafos tem um lugar de destaque, pois é possível fazer uma correspondência entre os seus elementos e os elementos que compõe uma rede informática (nós). Permite referenciar as interligações entre os nós da rede através ao adicionar relacionamentos entre os nodos na base de dados. Aceder a nós e relacionamentos numa base de dados de grafos nativos é uma operação eficiente e constante, que permite atravessar rapidamente milhões de ligações por segundo. Independentemente do tamanho total do conjunto de dados. As bases de dados de grafos primam pela gestão de dados altamente interligados e consultas complexas. Com apenas um padrão e um conjunto de pontos de partida, as bases de dados de grafos exploram os dados vizinhos em torno desses pontos iniciais de partida - recolhendo e agregando informações de milhões de nós e relacionamentos - e deixando quaisquer dados fora do perímetro de pesquisa intocados.

Tal como acontece com a maioria das tecnologias, existem poucas abordagens diferentes para o que compõe os componentes-chave de uma base de dados de grafos. Uma dessas abordagens é o modelo de propriedade de grafo, onde os dados são organizados como nós, relacionamentos e propriedades (dados armazenados nos nós ou relacionamentos), apresentado na Figura 7.

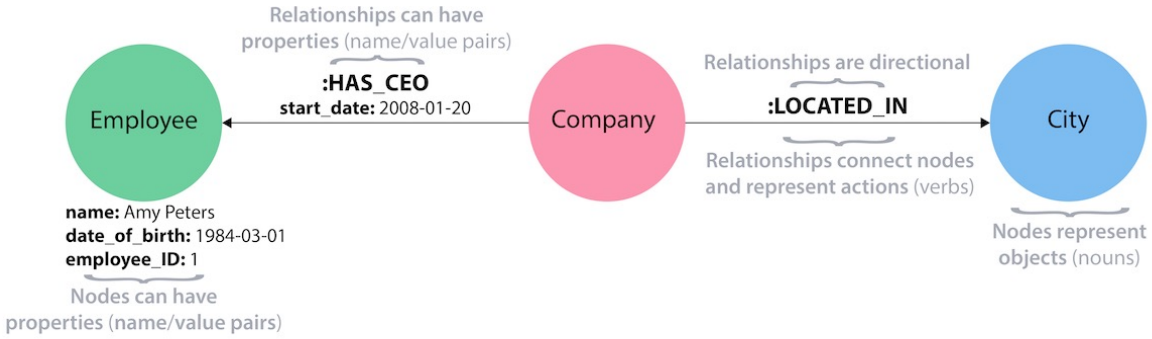


Figura 7: Grafo: Nós, relacionamentos e propriedades [25]

Os nós são as entidades no grafo, estas podem conter qualquer número de atributos (pares de valores-chave, como endereço de IP, porta de destino, utilizador, nome do equipamento, entre outros) chamados propriedades. Estes podem ser marcados com etiquetas, representando as suas diferentes funções no seu domínio, tal como podem servir para anexar metadados (como informações de índice ou restrições) a certos nós. As relações fornecem ligações direcionadas,

nomeadas, semânticas relevantes entre duas entidades de nó. Uma relação tem sempre uma direção, um tipo, um nó de partida, e um nó final. Tal como os nós, as relações também podem ter propriedades, na maioria dos casos, as relações têm propriedades quantitativas, tais como pesos, custos, distâncias, classificações ou intervalos de tempo. Devido à forma como as relações são armazenadas, dois nós podem partilhar qualquer número ou tipo de relacionamentos sem sacrificar o desempenho. Embora sejam armazenados numa direção específica, as relações podem sempre ser navegadas de forma eficiente em qualquer direção. [25]

A adoção de base de dados orientadas a grafos é transversal a um conjunto de áreas. Em relação à cibersegurança e monitorização de infraestruturas, em [13] modelaram os dados da rede numa base de dados orientada a grafos para analisar as ligações entre os clientes, servidores e equipamentos de rede ao longo do tempo para detetar padrões suspeitos de atividades maliciosas. Em [16] os autores, motivados pelo elevado número de eventos a analisar, propuseram uma base de dados orientada a grafos de dois níveis para representar objetos a partir de eventos de segurança e automatizar o processo de análise a partir da base de dados criada. A adoção de uma base de dados orientada a grafos foi também considerada em [21]. Neste trabalho, os autores, capturam de forma incremental os eventos de segurança, as vulnerabilidades e as dependências entre os sistemas existente na rede para constituir um modelo de previsão de possíveis ataques e quais os caminhos influenciados por esses ataques.

2.5 Conclusão do capítulo

As soluções de monitorização têm vindo a evoluir ao longo dos tempos, fruto da necessidade de perceber de forma mais rápida e simples os eventos e simultaneamente lidar com o elevado número de eventos gerados. Se antes os eventos se limitavam a *logs* locais, mais tarde soluções procuraram obter dados de forma remota (ex. via SNMP) centralizando a componente de monitorização para facilitar análise. Mais tarde se percebeu que os *logs* ainda que centralizados eram analisados como se fossem silos dentro de um mesmo armazém, ou seja, sem relação entre si. Os SIEMs vierem mudar o paradigma estendendo a capacidade de centralizar eventos com mecanismos de correlação de eventos facilitando uma análise transversal. Porém, as regras de deteção e alarmística continuam a ser pré-definidos e como tal incapazes de lidar com alterações dinâmicas no comportamento da rede. A adoção de inteligência artificial para detetar padrões maliciosos, e até mesmo identificar e alertar atividades suspeitas,

como novos comportamentos nas redes, surge como alternativa e as soluções começam a revelar bons resultados. A comunidade científica e a indústria procuram ainda novas formas de tratar os eventos, sendo a adoção de novas formas de representação e de análise exploradas neste âmbito.

Capítulo 3

Framework para visualização avançadas de logs – Implementação

Neste capítulo é abordada a implementação da framework, englobando desde o uso de ferramentas de monitorização até à visualização baseada em grafos, através da análise do tráfego de rede, proveniente da placa de rede de um ou mais equipamentos.

O projeto contempla o uso de várias ferramentas, que são interligadas por módulos. A divisão em módulos permite dividir o processo, em várias etapas facilitando a sua manutenção e configuração, nos equipamentos responsáveis por processar os resultados da framework. Para agilizar o processo de desenvolvimento de entrega do sistema pronto a funcionar usou-se uma abordagem de CI/CD (Continous Integrity / Continuous Delivery). De forma especifica, através da plataforma Gitlab, o software é guardado num repositório. A partir do repositório é criada uma imagem docker sendo desta forma instanciado um ambiente com o software pronto a executar. Com base nestas funcionalidades, torna possível, automatizar várias etapas, durante o desenvolvimento e implementação da framework. Como por exemplo, verificar a compatibilidade das ferramentas, interligadas entre si ou atualizar e instalar as várias dependências de cada módulo.

Ao longo do capítulo será abordado cada módulo, com todos os componentes desenvolvidos.

3.1 Introdução

Na Figura 8 é ilustrado o diagrama com os módulos que compõe a *framework*. Como se verifica pela figura, a *framework* integra um fluxo de etapas, desde a compilação do projeto, a monitorização da rede ou, na leitura de ficheiros do formato PCAP, armazenamento de dados na base de dados orientada a grafos e, apresentação desses dados através da interface web.

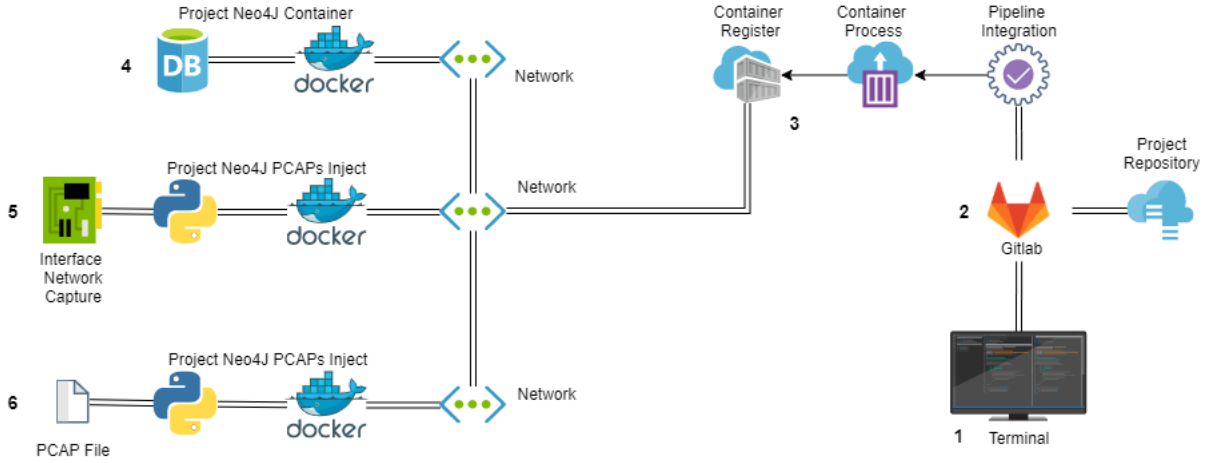


Figura 8: Diagrama Geral

Apresentado as várias etapas do projeto:

1. Terminal – tem como objetivo analisar os dados obtidos da base de dados orientada a grafos num formato visual e interativo e, compilar/afinar a *framework*, consoante as necessidades da monitorização;

2. Gitlab – responsável por armazenar o projeto desenvolvido e, executar a pipeline a pedido na etapa 1;

3. Docker – através da pipeline executa instruções, de forma a disponibilizar os ambientes Docker *containers* necessários na rede a monitorizar;

4. Base de dados Neo4J – através de ambientes Docker, disponibiliza a base de dados orientada a grafos, que permite armazenar o tráfego monitorizado nas etapas 5 e 6, processando os dados necessários para a visualização interativa da etapa 1;

5. Monitorização da placa de rede – monitoriza em tempo real o tráfego da rede em análise e, submete esses dados para a base de dados Neo4J, da etapa 4;

6. Monitorização de ficheiros PCAP – processa os dados de rede dos ficheiros PCAP, provenientes de uma captura de rede realizada e, submete esses dados para a base de dados Neo4J, da etapa 4.

Toda a arquitetura poderá ser executada diretamente em cada equipamento no seu sistema operativo ou, no caso apresentado, recorrendo a *containers* Docker, de forma a facilitar a integração e manutenção da *framework*. Desta forma, são apresentados os métodos e ferramentas utilizadas, que permitem monitorizar, processar e apresentar estes dados de rede num formato visual e interativo.

3.2 DevOps

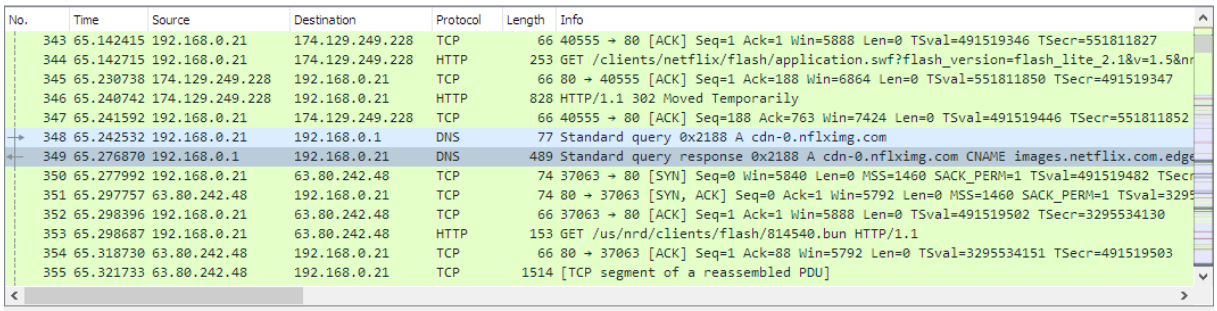
Nesta etapa, serão abordados os métodos utilizados para o desenvolvimento da *framework*, abordando como ficou planeado e elaborado. Desde o decorrer dos primeiros testes, na procura de soluções e ferramentas existentes capazes de monitorizar o tráfego de rede, diretamente através da placa de rede ou pela leitura de ficheiros de formato PCAP, até a visualização num ambiente orientado a grafos. Com base nos vários testes realizados e com o aumento da complexidade no uso de várias ferramentas e manutenção da mesma, foram abordados mecanismos para facilitar a manutenção e implementação. Assim, torna mais apetecível aplicar esta *framework* num futuro mercado empresarial, com interesse em adicionar uma camada visual com base em grafos, em vez de tabelas e gráficos muito comuns, como uma folha de calculo. Tendo como base, o projeto desenvolvido na plataforma Gitlab [10], de forma a automatizar várias etapas necessárias para a compilação da *framework*. O uso desta plataforma, torna fácil documentar cada etapa do projeto com instruções, como a manipulação/configuração da *framework* e, permite a utilização de pipelines de forma a automatizar etapas rotineiras e demoradas, como testes do código desenvolvido na linguagem Python, testes pré-produção de dependências das várias ferramentas, compilação das imagens utilizadas nos ambientes Docker *containers*, entre outros.

3.2.1 Ferramentas utilizadas

Ao longo do desenvolvimento da dissertação foram abordados vários métodos e ferramentas, capazes de monitorizar o tráfego de rede, processar esses dados e apresentá-los num formato

estruturado. Foram abordadas ferramentas como o Wireshark, que permitem analisar o tráfego de rede e estabelecer comunicação com a *framework*. Esta comunicação com a *framework*, realiza o processamento dos dados de rede, culminando com vários serviços, capazes de enriquecer esses dados recebidos, pois contém diversa informação, como as comunicações efetuadas, de forma a perceber a quem pertence o tráfego da rede em questão. Após o processamento desses dados, são criadas relações entre os vários nós da rede, juntamente com a informação adicional de enriquecimento. Após esse processo, cada relacionamento criado é submetido, para a base de dados Neo4J, através da qual, esses dados serão apresentados num formato visual, orientado a grafos. Esta visualização permite ajustar-se, através de filtros, consoante os dados fornecidos em cada nó, de forma a selecionar os grafos, pelo tipo de protocolos de comunicação, endereços IP, portos de origem ou destino, entre outros. Assim serão abordadas de forma breve, algumas das tecnologias e ferramentas.

A ferramenta Wireshark permite a análise de ficheiros no formato PCAP (Capture Packet) e, a leitura em tempo real do tráfego da placa de rede do equipamento em análise. O que torna possível a leitura de cenários de rede disponibilizados na comunidade no formato PCAP, utilizados para armazenar capturas de rede, de forma a serem analisadas e monitorizadas nos cenários de rede montados. Apresentado na Figura 9, um exemplo de tráfego de rede através da ferramenta Wireshark, que demonstra a informação recolhida, da qual são utilizados os campos mais comuns como Time, Source, Destination e Protocol.



No.	Time	Source	Destination	Protocol	Length	Info
343	65.142415	192.168.0.21	174.129.249.228	TCP	66	40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
344	65.142715	192.168.0.21	174.129.249.228	HTTP	253	GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
345	65.230738	174.129.249.228	192.168.0.21	TCP	66	80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
346	65.240742	174.129.249.228	192.168.0.21	HTTP	828	HTTP/1.1 302 Moved Temporarily
347	65.241592	192.168.0.21	174.129.249.228	TCP	66	40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
348	65.242532	192.168.0.21	192.168.0.1	DNS	77	Standard query 0x2188 A cdn-0.nflximg.com
349	65.276870	192.168.0.1	192.168.0.21	DNS	489	Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
350	65.277992	192.168.0.21	63.80.242.48	TCP	74	37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
351	65.297757	63.80.242.48	192.168.0.21	TCP	74	80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
352	65.298396	192.168.0.21	63.80.242.48	TCP	66	37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
353	65.298687	192.168.0.21	63.80.242.48	HTTP	153	GET /us/nrd/clients/flash/814540.bun HTTP/1.1
354	65.318730	63.80.242.48	192.168.0.21	TCP	66	80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
355	65.321733	63.80.242.48	192.168.0.21	TCP	1514	[TCP segment of a reassembled PDU]

Figura 9: Ferramenta Wireshark exemplo tráfego rede

O desenvolvimento da *framework* consiste na linguagem de programação Python, orientada a objetos e, de alto nível. Utilizada no desenvolvimento web, realização protótipos de software, pela sua sintaxe simples e fácil de usar. Também, pela sua fácil aplicabilidade em casos relacionados como data science e machine learning e, com várias bibliotecas disponibilizadas pela comunidade, que permitem a comunicação entre Wireshark e Neo4J. Através de bibliotecas como tshark e py2neo, permite facilmente estabelecer uma comunicação direta, mas

fulcral para o desenvolvimento da *framework*. Juntamente com outras ferramentas existentes, que possibilitam a análise do tráfego de rede e, posteriormente com possíveis algoritmos de análise de padrões.

Foram abordados serviços capazes de adicionar informação relevante para as monitorizações de rede, no qual um dos serviços que se mais destaca, corresponde à localização geográfica dos endereços de rede. Assim permite facilmente identificar o país de destino, que cada comunicação efetuada pelo *router* à Internet. Outro dado relevante como o domínio, de forma a identificar tráfego de rede externo a serviço comuns confiáveis, como por exemplos website de notícias ou bancos que não representam ameaça ao funcionamento espectável de uma rede. No qual abordado o serviço da maxmind o GeoLite2, através de uma pequena base de dados local disponibilizada gratuitamente, que contem vários endereços de IP com informação relevante como país, cidade e domínio. Esta comunicação com a base de dados é realizada através da biblioteca geoip2.database, compatível com a ferramenta Python, que permite pesquisar cada endereço de IP e adicionar o máximo de informação possível antes de criar os relacionamentos da comunicação do grafo.

Neo4j é uma base de dados orientada a grafos, nativa, de código aberto (open source), NoSQL, que fornece um backend transacional compatível com ACID (atomicity, consistency, isolation, durability). Nativo, pois implementa eficientemente a propriedade do modelo de grafo, até ao nível de armazenamento. Isto significa que os dados são armazenados exatamente como abordado, e a base de dados, utiliza apontadores para navegar e atravessar o grafo. Em contraste com o processamento de grafos ou “in memory libraries”, o Neo4j também fornece características completas de base de dados, incluindo suporte de cluster e “runtime failover”. Tornando-o adequado para usar grafos para dados em cenários de produção e monitorizações intensivas.

Apresentado na Figura 10, um cenário de rede simplista, que permite observar a comunicação interna da rede filtradas por rotas. Neste exemplo, facilmente se identifica, a rota de comunicação das diferentes interfaces de rede, e os endereçamentos IP distintos.

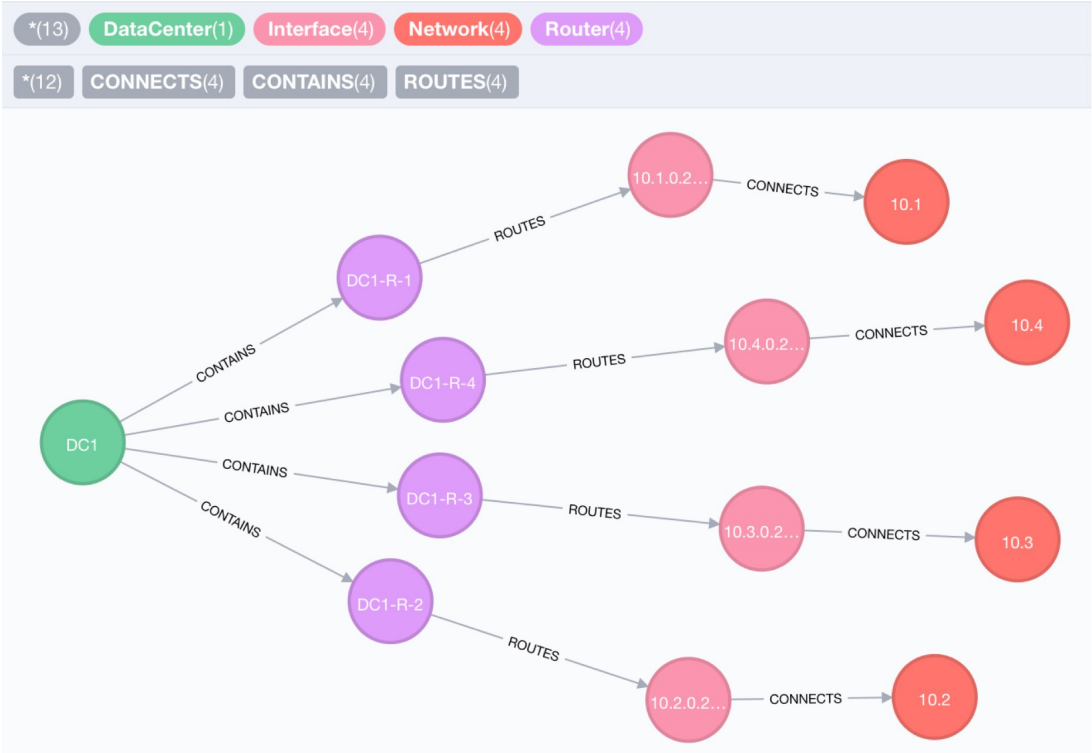


Figura 10: Cenário exemplo de rede simplista por comunicação entre dispositivos [24]

3.2.2 Docker Containers

A utilização de Docker *containers* numa fase inicial não aparenta ser necessária para a monitorização de tráfego de rede. A própria *framework* pode ser executada diretamente no equipamento em análise sem dependência de *containers* Docker, no entanto esta funcionalidade, que não fora considerada como um objetivo deste trabalho, tende a ser aplicável em ambientes de produção. Desta forma a implementação e manutenção de várias bases de dados orientadas a grafos é mais simples. Possibilita a segmentação de rede ou motorização dos dados num período especifico de tempo. A utilização de *containers* permite ainda processar e relacionar os dados de rede em bruto que são provenientes de possíveis agentes de monitorização espalhados pela rede em análise.

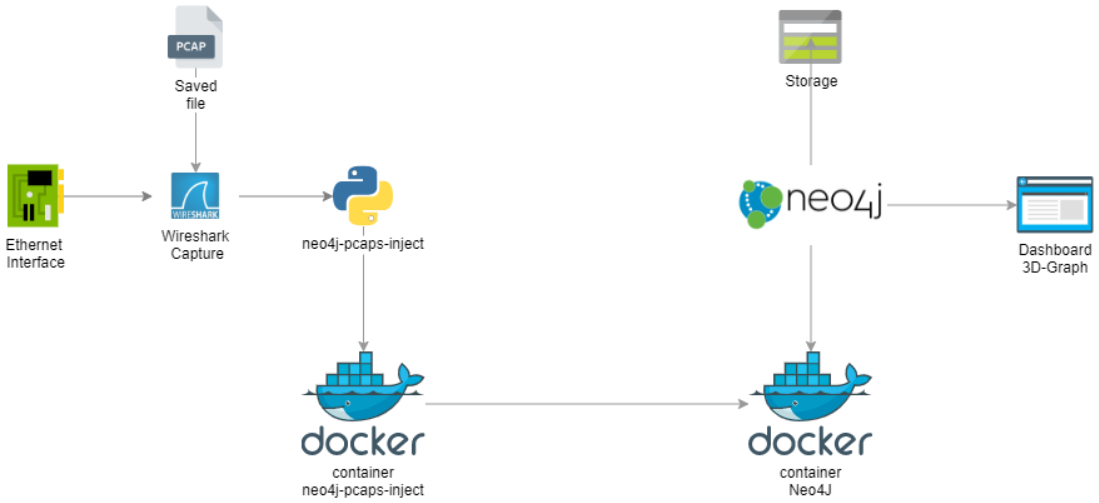


Figura 11: Diagrama comunicação da *framework* em ambiente docker

Na Figura 11 podemos observar uma estrutura simples de uma para um, ou seja, a funcionalidade de interpretar esses dados de rede e processá-los no container neo4j-pcaps-inject, e outro container Neo4J contem a base de dados orientada a grafos e dashboard.

Apesar do uso de *containers* poder ser considerado como limitado em alguns pontos deste projeto (ex. problemas de performance), o objetivo deste uso passou principalmente por aplicar na rede um ou mais *containers*, consoante a necessidade, capazes de ler o tráfego da placa de rede ou a leitura direta de ficheiros PCAPs, e emitir assim, esses dados diretamente para outro grupo de *containers* com o objetivo de processar a informação do tráfego da rede em análise.

3.2.3 Framework

A *framework* desenvolvida tem como core a linguagem Python, que permite desenvolver pequenos projetos como este, através da incorporação de várias bibliotecas, disponibilizadas pela comunidade. Permitiu a integração de funcionalidades, como por exemplo, comunicar com a API da ferramenta Wireshark, através da biblioteca pyshark. E com a biblioteca py2neo, para criar relações e comunicar com a base de dados orientada a grafos Neo4J. De grande importância, referir o projeto “Packet Communication Investigator” [18] disponibilizado na plataforma GitHub, em que disponibiliza uma demonstração simples desenvolvido na linguagem Python, que estabelece a comunicação entre a ferramenta Wireshark e a base de dados Neo4J. Durante o processo de captura de rede, é necessário processar os dados recebidos e,

estruturá-los de forma a criar as relações, que serão adicionadas à base de dados orientada a grafos. Desta forma é utilizada a ferramenta Wireshark, que permite parsear/estruturar os campos provenientes do tráfego de rede ou ficheiros PCAP, de forma a facilitar a manipulação desses dados.

Na Figura 12 é apresentado um excerto do código num estado simplista, em que podemos observar a variável *pkt* (packet) que contem a informação de um pacote em análise, de uma comunicação realizada na rede. Representando a relação a ser criada na base de dados, com o conteúdo do endereço IP de início, endereço IP de destino e protocolo de comunicação utilizado.

```
def do_the_job(pkt):
    src = pkt.source
    dst = pkt.destination
    typ = pkt.protocol
    # info = pkt.info
    # length = pkt.length
    # etc...

    a = get_node(arg=src)
    b = get_node(arg=dst)

    # PACKET_TO = Relationship.type(typ)
    # _graph_db.merge(PACKET_TO(a, b))
    _graph_db.merge(Relationship(a, typ, b))
```

Figura 12: Exemplo simples da função do_the_job

Neste tipo de relação simples é demonstrado apenas o sumário do tráfego de rede. Comummente são adicionados mais campos, para enriquecer os dados e proporcionar uma observação do tráfego, num formato visual baseado em grafos.

Isto verifica-se na função *get_node* demonstrado na Figura 13, que permite observar uma simulação em tempo real com os campos adicionais como, domínio, país e cidade do endereço IP, data de criação e última data em caso de reincidência da mesma comunicação. Para analisar o histórico das comunicações é importante o registo da data e hora de cada comunicação efetuada.

```python
if GEOIP_CITY_STATUS:
    try:
        response = _reader.city(arg)
        if response:
            country_name = response.country.iso_code
            sub_name = response.subdivisions.most_specific.iso_code
            city_name = response.city.name
    except Exception:
        pass

existing_node = Node("machine", name=arg, domain=domain_name, country=country_name,
                     sub=sub_name, city=city_name, count=1,
                     creation_date=datetime.now(timezone.utc).isoformat(),
                     last_update=datetime.now(timezone.utc).isoformat())
create_node(arg, existing_node)
```

Figura 13: Excerto da função get_node com create_node

Ao desenvolver a *framework* são necessários comandos de configuração, que permitem manipular o seu comportamento. Como por exemplo, o tipo de captura de rede a ser realizado ou a localização da base de dados a comunicar. Assim serão abordados os vários atributos necessários para executar a *framework*, sendo os mesmos atributos aplicáveis nos ambientes em Docker *containers* ou diretamente na máquina a analisar.

```
$ python pci.py -h
usage: pci.py [-h] [-f FILEPATH] [-i INTERFACE] [--only_summaries] [-r] [--NEO4J_HOST NEO4J_HOST]
              [--NEO4J_PORT NEO4J_PORT] [--NEO4J_USERNAME NEO4J_USERNAME] [--NEO4J_PASSWORD NEO4J_PASSWORD]
              [--GeoLite] [--GeoLite_File GEOIP_FILE] [--docker]

Neo4J PCAPs Injector

optional arguments:
  -h, --help            show this help message and exit
  -f FILEPATH, --file FILEPATH
                        choose PCAP file to analyze, default is disabled
  -i INTERFACE, --interface INTERFACE
                        choose interface, default eth0
  --only_summaries      only produce packet summaries, True or False, default: True
  -r, --ring            activate ring buffer
  --NEO4J_HOST NEO4J_HOST
                        choose HOST IP for Neo4J server, default is localhost
  --NEO4J_PORT NEO4J_PORT
                        choose HOST PORT for Neo4J server, default is 7474
  --NEO4J_USERNAME NEO4J_USERNAME
                        choose username for Neo4J server, default is neo4j
  --NEO4J_PASSWORD NEO4J_PASSWORD
                        choose Password for Neo4J server, or use default pw
  --GeoLite             GeoLite2 additional info to add
  --GeoLite_File GEOIP_FILE
                        GeoLite File DB, default: ./geoip/GeoLite2-City.mmdb
  --docker              running on Docker container, default vars used

LuisCarlos@LAPTOP-MSI-LCLT MINGW64 ~/GitLab_Projects/neo4j-pcaps-inject (master)
```

Figura 14: Python *framework* Menu

Na Figura 14 é apresentado o menu com os possíveis campos a inserir de forma a moldar o

comportamento da *framework*, dos quais:

- –Filepath: permite especificar a localização do ficheiro PCAP a analisar, por predefinição esta opção está desativada.
- –Interface: permite identificar a interface de rede onde o Wireshark irá executar a análise de rede, por predefinição utiliza a interface eth0.
- –only_summaries: extrai informação simples dos pacotes de rede, como Endereço de início, destino e protocolo utilizado. Este método é definido por predefinição pois permite utilizar menos recursos computacionais, com o contrapeso de extrair menos informação dos eventos de rede.
- –ring buffer: permite definir o espaço por ficheiro a ser utilizado pela captura de trafego de rede, em vez de criar um ficheiro temporário para proceder com a análise (do momento ou no futuro), desta forma segmenta por vários ficheiros de tamanho definido, em que o torna mais fácil e rápido de abrir para parsear. Logo que um ficheiro atinge o tamanho definido, será criado um novo ficheiro. Após atingir o limite de ficheiros, começa o processo de sobreposição começando pelo ficheiro mais antigo. Nota: para definir o tamanho e número de ficheiros a disponibilizar, no entanto apenas são atribuídos no momento de compilação da *framework* para evitar erros nos *containers*, por exemplo: “filesize: 10000” e “files:10”, que permite a distribuição por 10 ficheiros no formato PCAP com 10MB de capacidade cada.
- –GeoLite: permite enriquecer os dados dos endereços de IP externos a pesquisar, de forma a adicionar informação como o país, cidade e domínio de cada endereço de IP. Neste caso foi configurado o uso da base de dados da entidade Maxmind [17]
- –GeoLite_File: identifica a localização da base de dados para proceder com a pesquisa dos endereços IP.
- –docker: permite utilizar as variáveis predefinidas nas configurações internas na *framework* para comunicar com a base de dados Neo4J em docker container atribuído ao mesmo.
- –Neo4J: permite especificar as variáveis necessárias para estabelecer comunicação com a base de dados, dos quais host, port, username e password. No caso de não especificar a comunicação é estabelecida por defeito, definida no momento de compilação.

3.3 Módulos que compõem a Framework

Nesta secção é abordado modulo a modulo as ferramentas e serviços utilizados, que unem a *framework* desenvolvida. Desde os métodos utilizados na leitura do tráfego de rede, através da ferramenta Wireshark até à monitorização e visualização desses dados, na base de dados orientada a grafos através do Neo4J. É necessário enriquecer a informação a colocar em cada grafo, assim serão abordadas as propriedades necessárias a adicionar, de forma a melhorar a monitorização e análise. Outro dos métodos explorados, numa fase inicial, referente ao uso de data mining e data science com o objetivo de detetar desvio padrão na rede, através de algoritmos capazes de detetar comportamentos suspeitos na rede.

3.3.1 Métodos de leitura do tráfego rede

Numa primeira fase deste projeto, é necessário analisar o tráfego da rede, existindo a possibilidade de a realizar através de dois métodos distintos. Através da leitura de pacotes de rede, provenientes da placa de rede do equipamento ou, através da leitura de ficheiros no formato PCAP, com a finalidade de armazenar a captura de rede, previamente recolhida por uma placa de rede de um equipamento. O primeiro método a analisar, será a leitura de pacotes da placa de rede do equipamento a monitorizar, sendo possível através do uso da ferramenta Wireshark, que permite analisar este tráfego em tempo real, através de um intervalo de tempo ou tamanho de dados previamente definido ou, através de ficheiros de captura PCAP.

Permite assim, interpretar dados provenientes de capturas de rede, no formato PCAP, realizados pela entidade Screpo [27] que tem como objetivo disponibilizar para a comunidade amostras de capturas de rede, através de capturas de vários eventos de cyber segurança. Com o objetivo de identificar assim, possíveis comportamentos suspeitos, o que possibilita ajustar cenários de rede à medida e permite a inserção de atividades maliciosas no âmbito de testes, de uma forma mais simplista por ficheiros PCAP, em comparação a realizar fisicamente uma atividade maliciosa. Desta forma é possível armazenar um cenário de rede em ficheiros PCAP e futuramente analisar essas amostragens retiradas do acontecimento de uma captura de rede.

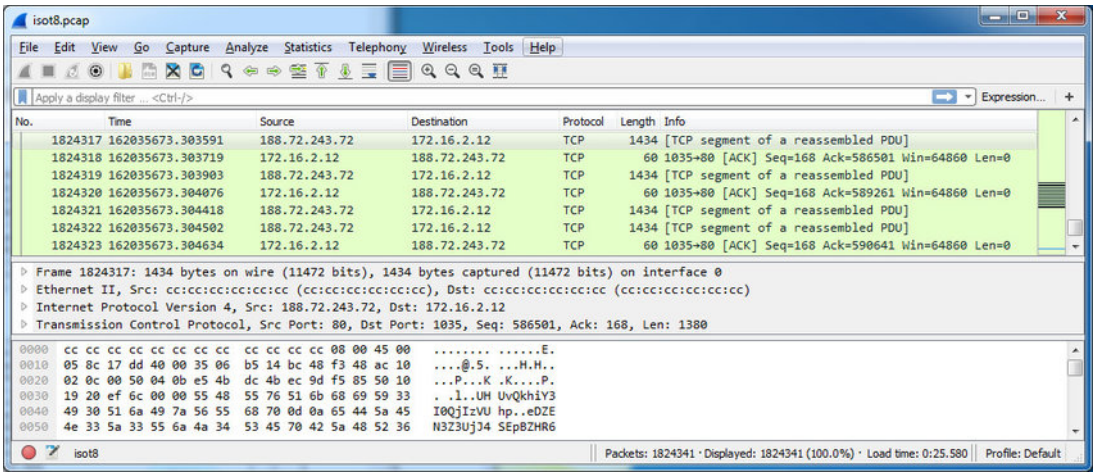


Figura 15: Wireshark exemplo *logs* de captura de rede

Apresentado na Figura 15 a ferramenta Wireshark com uma captura exemplo, com o objetivo de ilustrar a estrutura proveniente de uma captura de rede e a dificuldade para o analista em interpretar e analisar esses dados.

3.3.2 Interligação do Python com Wireshark

A *framework* desenvolvida na linguagem Python, tem a necessidade de comunicar ativamente com a ferramenta Wireshark, que permite estabelecer o elo entre os dados de rede a monitorizar, com o programa Python. Desta forma interpretar os dados recebidos, estruturá-los e posteriormente criar as relações necessárias na base de dados. Numa primeira abordagem, foi implementada a leitura de ficheiros no formato PCAPs, no entanto para tornar possível a leitura em tempo real do tráfego de rede é necessário que haja esta ligação entre a *framework* e uma placa de rede a monitorizar.

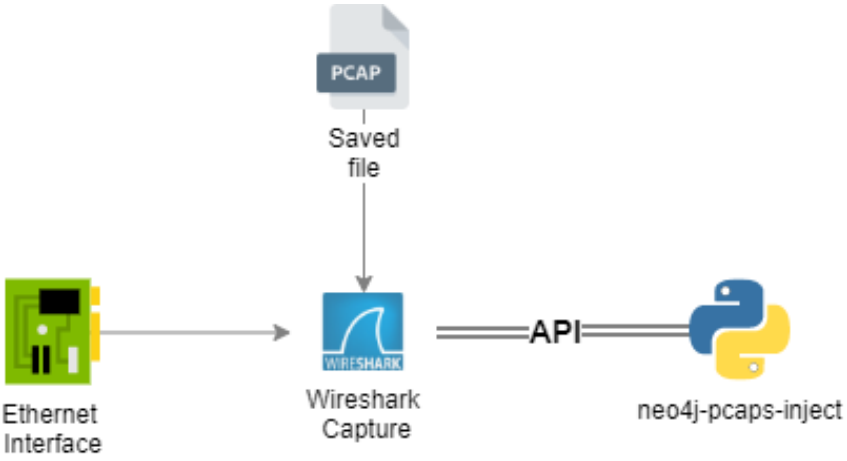


Figura 16: Diagrama comunicação Wireshark API com Python

Como é representado na Figura 16, a comunicação entre o Wireshark e Python, através da biblioteca tshark, um projeto desenvolvido pela comunidade com objetivo de comunicar diretamente com a API da ferramenta Wireshark. Na qual necessita de instruções para inicializar a análise de rede. Por exemplo, identificar a placa de rede ou a localização do ficheiro PCAP a analisar. Desta forma permite executar os comandos que seriam introduzidos no ambiente gráfico. O que possibilita aplicar uma primeira camada de filtragem desses dados de rede, através do comando "summary_only" permite filtrar apenas por alguns campos existentes, dos quais endereços de IP, portas utilizadas, data e hora dos eventos e protocolo utilizado. Desta forma é possível melhorar alguns dos pontos de performance. Num futuro cenário uns dos objetivos seria a possibilidade de trabalhar todos esses dados, que são por enquanto removidos.

3.3.3 Interligação com base de dados orientada a grafos

De forma a possibilitar a visualização dos dados de captura de rede num formato orientado a grafos, é necessário interligar o programa Python com a base de dados Neo4J, através da biblioteca py2neo. Desenvolvida para estabelecer comunicação com o Neo4J, atravéss do programa Python. Após estruturar os dados de rede obtidos, cria as relações de cada grafo e submete cada comunicação da rede, para a base de dados Neo4J. Assim, os dados são armazenados e permite representar como cada nó se conecta ou está relacionado com os outros nós.

De forma criar as relações na base de dados é utilizado a linguagem Cypher query (linguagem utilizada no Neo4J), que permite criar, alterar e pesquisar um relacionamento de forma simples. Após identificar o nó de início e o nó de fim, juntamente com o tipo de relação, na maioria dos casos abordados, utilizando o tipo de protocolo de rede, como por exemplo HTTP, DNS, DHCP, entre outros. Simulando um caso prático, imaginemos a comunicação de um computador portátil, a comunicar via browser ao site de notícias, no qual na rede é necessário identificar o endereço de IP de destino (computador portátil) e o domínio desse site de notícias. A representação deste pedido de acesso ao site de notícias, será representado como um relacionamento entre o endereço de início, o tipo de comunicação neste caso HTTP e o endereço de destino do domínio do website.

Apresentado na Figura 17 um esquema oficial do Neo4J, que representa as particularidades e características únicas.

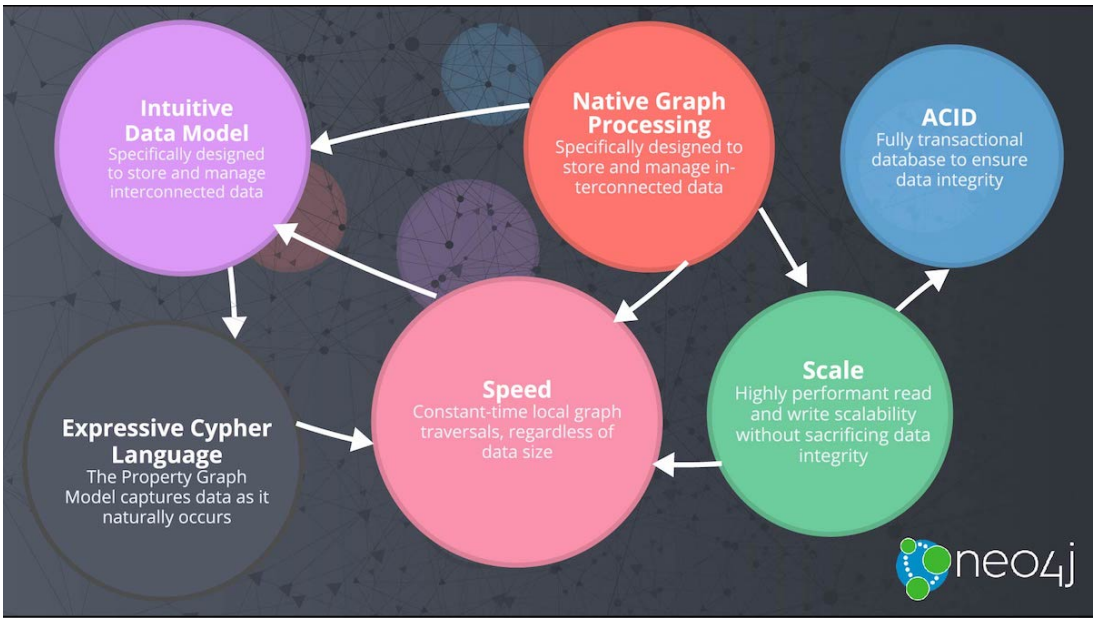


Figura 17: Particularidades do Neo4J [25]

Algumas das seguintes características particulares do Neo4J [25]:

- Cypher: uma linguagem de consulta declarativa semelhante ao SQL, otimizada para grafos.
- Constant time traversals: (Travessias de tempo constantes) em grandes grafos, tanto para profundidade como para a amplitude, devido à representação eficiente de nós e relacionamentos. Permite a escalar de milhares de milhões de nós em hardware moderado.
- Flexible: com propriedade de grafos flexíveis, que pode adaptar-se ao longo do tempo, tornando possível materializar e adicionar novas relações mais tarde.
- Drivers: para linguagens de programação populares, incluindo Python, que permitiu a integração com a *framework*.

3.3.4 Propriedades utilizadas

As propriedades abordadas nos grafos têm como principal objetivo, o enriquecimento dos dados apresentados, para permitir uma análise eficiente. Desta forma foram explorados alguns dos vários campos a definir nas propriedades do grafo, dos quais o protocolo de comunicação para estabelecer os relacionamentos, e como informação adicional em cada nó a informação

detalhada de cada endereço de IP externo ou interno e o identificador único referente ao nome utilizador de cada equipamento da organização. Para tal segue uma breve descrição, de cada propriedade de enriquecimento dos dados de rede.

Estabelecer relacionamentos nos grafos através do protocolo de rede, que possibilita filtrar por todos os tipos de protocolo utilizados na rede em análise. São um conjunto de normas, que permitem conectar todos os equipamentos com a Internet, através de uma linguagem universal, segmentada por camadas, das quais aplicação, transporte, rede (Internet) e estrutura física [30].

Apresentada uma lista dos protocolos de rede mais utilizados, que serão apresentados nos grafos como propriedades nos relacionamentos, que ajudará ao analista de rede, identificar e filtrar o tipo de tráfego.

Tipos de protocolos de Rede:

1. TCP/IP
2. HTTP
3. HTTPS
4. DHCP
5. FTP
6. SFTP
7. SSH
8. POP3
9. SMTP
10. IMAP

No enriquecimento de dados, é comum apresentar a informação dos endereços de IP externos, provenientes das comunicações com a Internet, com informação como o país, cidade, domínio, entidade responsável, entre outros. Apresentado na Figura 18, a estrutura utilizada que permite à *framework* realizar pesquisas de endereços IP a uma base de dados local ou remota. Como por exemplo, foi utilizado o serviço GeoLite2 da entidade Maxmind [17], que

permite pesquisar por endereços IP e adicionar essa informação aos campos estruturados de cada nó do grafo. O que torna possível identificar facilmente, quando equipamento ou serviço que comunica com o exterior e, permite identificar a origem ou destino dessas comunicações, agrupando ou não, por região geográfica.

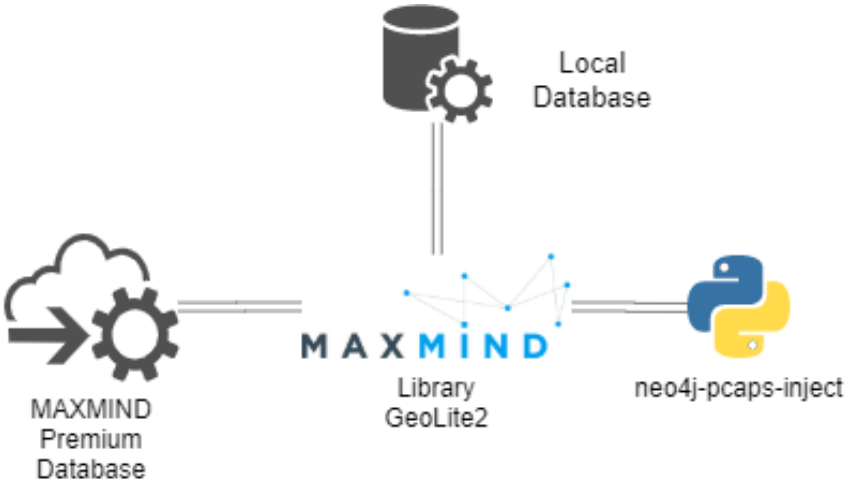


Figura 18: Esquema GeoLite2 - MAXMIND

De igual forma, é essencial manter atualizadas as informações dos endereços de IP da rede interna, através de plataformas de gestão de endereços IP, como por exemplo o CMDB da onecmdb [23], de forma a facilitar a localização física no edifício e, tipo de equipamento. Para ajudar na monitorização, a identificar o tráfego espectável desse equipamento. Na Figura 19 é apresentado o exemplo de um serviço de CMDB, em que permite observar uma lista contendo o nome, modelo, sistema operativo, endereço de IP e nome do domínio.

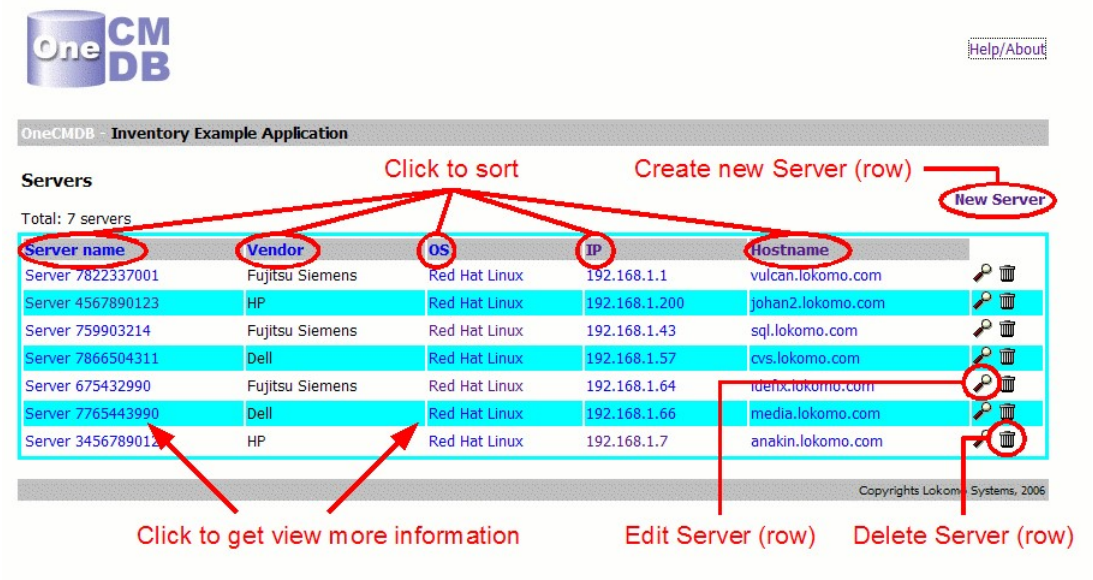


Figura 19: Gestão de equipamentos – OneCMDB [23]

Outro método de enriquecimento de dados utilizado, através do identificador único, como já abordado no capítulo anterior, a utilização de mecanismos como AD ou Kerberos, será fulcral numa organização para rapidamente identificar o proprietário de uma máquina suspeita. Assim permite associar uma máquina ou um endereço IP interno, com um identificador único que pode ser um número ou nome de utilizador, em que caracteriza unicamente um operador dessa organização. De forma a identificar se o tráfego em análise é espectável, com base nas funcionalidades e permissões do operador.

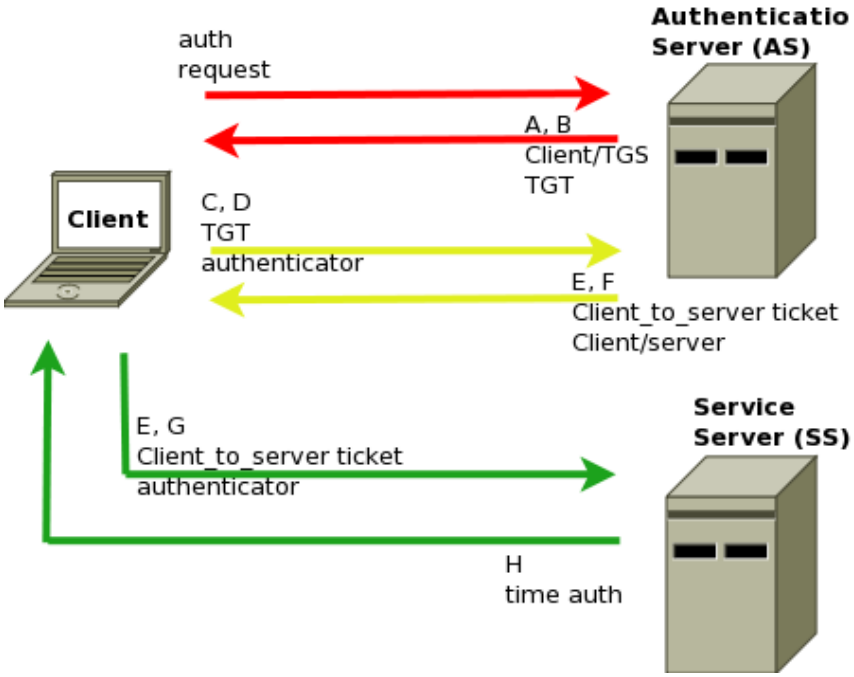


Figura 20: Esquema simplista Active Directory [31]

Como é apresentado na Figura 20, um exemplo do serviço de autenticação, em que o computador do operador realiza os pedidos de autenticação a este serviço. Utilizado nas organizações com um sistema central de pedidos de acessos aos vários serviços internos e externos da rede, sendo este responsável, por registar e estabelecer os pedidos dos vários dispositivos da rede. O que permite assim, gerar *logs* de informação, contendo os endereços de IP, nome do utilizador e, com a finalidade de associar um endereço IP unicamente a um utilizador na rede.

3.3.5 Data Mining

O processo de data mining está cada vez mais interligado nos sistemas de monitorização, serviços e funcionalidades internas das empresas. Embora exista uma grande variedade de sistemas, capazes de apoiar a execução de tais processos, no caso deste projeto, através

da monitorização de rede que permitirá auxiliar na visualização desses dados dos grafos. As práticas atuais de monitorização e análise de dados da rede, são na realidade complexas e confusas de interpretar, tal se verifica cada vez mais, com o aumento do volume de tráfego de rede, a ser monitorizado. Assim, com o uso de processos de data mining permitirá preencher essa lacuna, fornecendo meios revolucionários para a análise e monitorização desses dados da rede.

Apresentado na Figura 21 um gráfico exemplo do tráfego de rede onde se verifica o número comunicações dos payloads enviados agrupados pelos endereços de IP de inicio, juntamente com a Figura 22 onde se verifica o inverso, o número de comunicações dos payloads recebidos agrupados pelos endereços de IP de destino. Nestes exemplos podemos verificar os endereços mais utilizados na rede e, facilmente identificar os endereços com maior volume de tráfego, comummente utilizado para as métricas mensais de relatórios sobre a rede das organizações.

Figura 21: Gráfico Pacotes Enviados por Source IP [32]

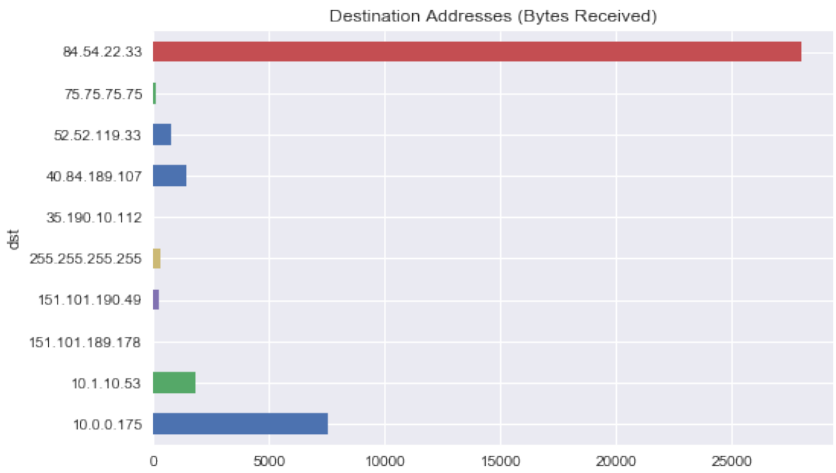


Figura 22: Gráfico Pacotes Recebidos por Destination IP [32]

Ao contrário da análise do volume de tráfego por endereços de IP, também é de igual modo importante, analisar os portos utilizados de forma a identificar os vários tipos de comunicação com os serviços internos ou externos no caso da internet. Apresentado na Figura 23 o gráfico exemplo onde se verifica os portos de início que estabelecem as ligações, agrupados pelo número de comunicações dos payloads pelo volume de tráfego. Em contraste com as ligações estabelecidas aos portos de destino apresentados na Figura 24.

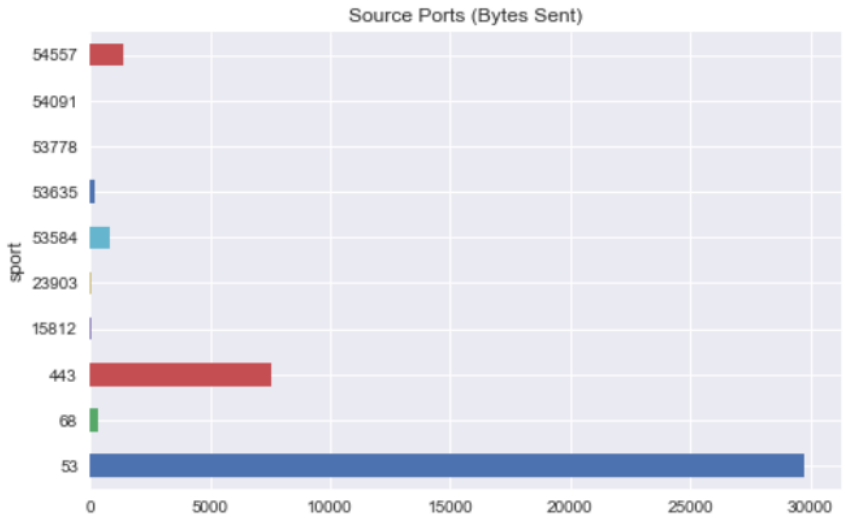


Figura 23: Gráfico Portos utilizados por Source Ports [32]

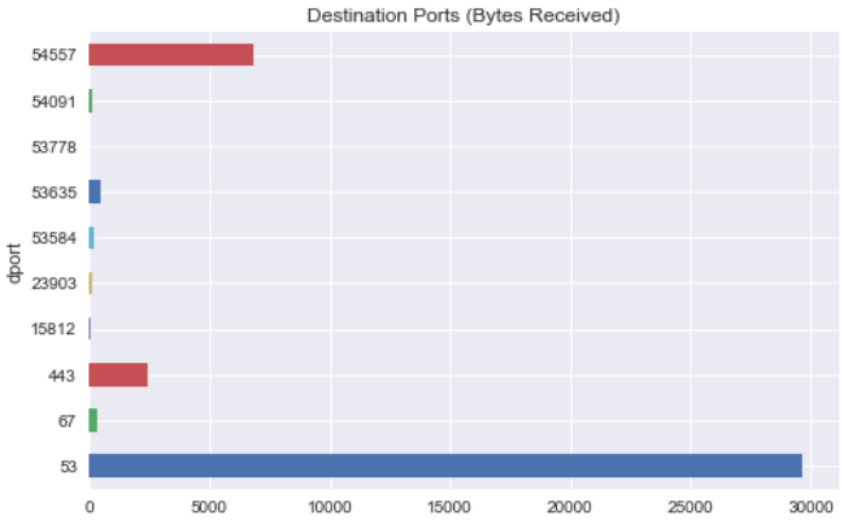


Figura 24: Grafico Portos utilizados por Destination Ports [32]

Ao lidar com grandes quantidades de dados, criar visualizações sob a forma de gráficos a que chamamos a este processo de data science, pode ser bastante útil para entender tendências e padrões. Deste modo, procurar por artefactos maliciosos em gráficos, que podem destacar picos anómalos de dados e mostrar que os dados estão a ser enviados em intervalos regulares (ou intervalos de pseudorandom). Desta forma, seria interessante analisar em gráficos, por volume de tráfego de forma agrupada, o Top 10 dos endereços de início, endereços de destino, portos de início e destino. O que permitirá identificar o endereço com maior volume de tráfego, e a porta de comunicação mais utilizada.

Assim nasce a necessidade de utilizar processos de data mining, que estão focados na extração de conhecimentos sobre um procedimento, a partir dos registos de execução e resultados. Process Mining procura obter informações sobre várias perspetivas, tais como a perspetiva do processo, o desempenho e os dados. Em que podemos entender estes registos, como o comportamento dos equipamentos na rede, que diariamente geram dados, como abordado anteriormente através da representação de gráficos, a partir de cálculos estatísticos. O processo de data mining terá de avaliar constantemente o tráfego da rede, e juntamente com machine learning permitirá detetar o comportamento padrão de todos os equipamentos da rede ao longo de vários dias e semanas. E assim, identificar artefactos suspeitos na rede de forma a alertar o analista da rede de comportamentos fora de padrão. Comportamento este, já realizado pela ferramenta Darktrace, que consiste em detetar padrões suspeitos e fora do âmbito de rede, de forma a alertar o analista e, afinar a rede com base nos seus alertas.

Assim este método de análise através de data mining, criar um dataset a partir da base de

dados orientada a grafos, de forma a treinar e afinar esse dataset da machine learning. O que possibilita classificar as comunicações de rede, na procura de padrões suspeitos na rede, com campos categorizados por tipo de alerta, grau de maliciosidade, atribuir comunicações como fora do padrão da rede. Com base nos campos atribuídos pelos algoritmos, acrescentar dados de enriquecimento nos grafos e relacionamentos considerados suspeitos na base de dados orientada a grafos, o que torna possível identificar, no formato visual detetar essas comunicações, e facilmente monitorizar a sua propagação pela rede, filtrando por departamentos ou equipamentos possivelmente infetados.

Capítulo 4

Estudo experimental

Neste capítulo são apresentados os resultados do estudo experimental realizado para avaliar o funcionamento da framework apresentada no capítulo anterior. Para efeitos de estudo foram desenhados cenários que contemplam a recolha e armazenamento do tráfego de rede, sendo feita a sua representação gráfica com base na base de dados orientada a grafos.

O computador utilizado para a realização dos cenários, fora o modelo GP62MVR 7RFX da MSI, de CPU Intel i7-7700HQ @ 2.80GHz, com 16 GB de memória RAM (DDR4 2400MHz), com o armazenamento SSD M.2 SATA de 240 GB (Kingston rbu-sns8152s3256gg5) com leituras até 550 MB/s e escrita até 330 MB/s.

O primeiro cenário representa a situação mais comum de monitorização da rede. Limita-se a relacionamentos entre endereços de IP através do protocolo de rede utilizado. Um segundo cenário de rede, permite visualizar a estrutura da rede, ou seja, demonstrar as relações dos vários equipamentos, com as rotas utilizadas desde o ponto inicial, como um computador, percorrendo todo o caminho pelo equipamento de rede até ao destino, como por exemplo o *router* de internet. O terceiro cenário simula em rede empresarial, juntamente com os possíveis mecanismos e ferramentas comumente utilizadas na recolha, processamento e monitorização dos dados de rede, juntamente com a *framework* desenvolvida, na qual existe um dispositivo infetado com malware.

Por último, é apresentada a metodologia utilizada na monitorização em tempo real e, a descrição do hardware utilizado. À medida que o estudo experimental foi feito, houve necessidade de afinar a *framework*, de forma a criar e melhorar a sua capacidade de representação gráfica dos

logs relativos ao tráfego de rede.

4.1 Representação PCAPs

O primeiro cenário realizado, teve como objetivo demonstrar a capacidade de representação visual da rede. Para o efeito foram utilizadas capturas de tráfego (PCAPs) disponibilizada pela entidade Secrepo [27]. Foram utilizados vários ficheiros no formato PCAP, sendo estes processados pela ferramenta de Wireshark e Bro Logs. O resultado do processamento foi armazenado na base de dados orientada a grafos. Na Figura 25, é ilustrado o resultado obtido através *framework* desenvolvida, sendo estes apresentados na plataforma de visualização Neo4J. Os dados de monitorização usados equivalem a aproximadamente 4 GB de dados de rede. O tempo de processamento dos PCAPs foi superior a três horas, tendo sido os ficheiros PCAPs processados sequencialmente por ordem de exportação da Secrepo. Cada PCAP tem um tamanho médio de 300 MB. Para o processamento pela *framework*, apenas foram retirados os campos mínimos necessários para gerar os relacionamentos, entre os vários nós no grafo. O tamanho final na base de dados Neo4J foi de 115 MB de dados.

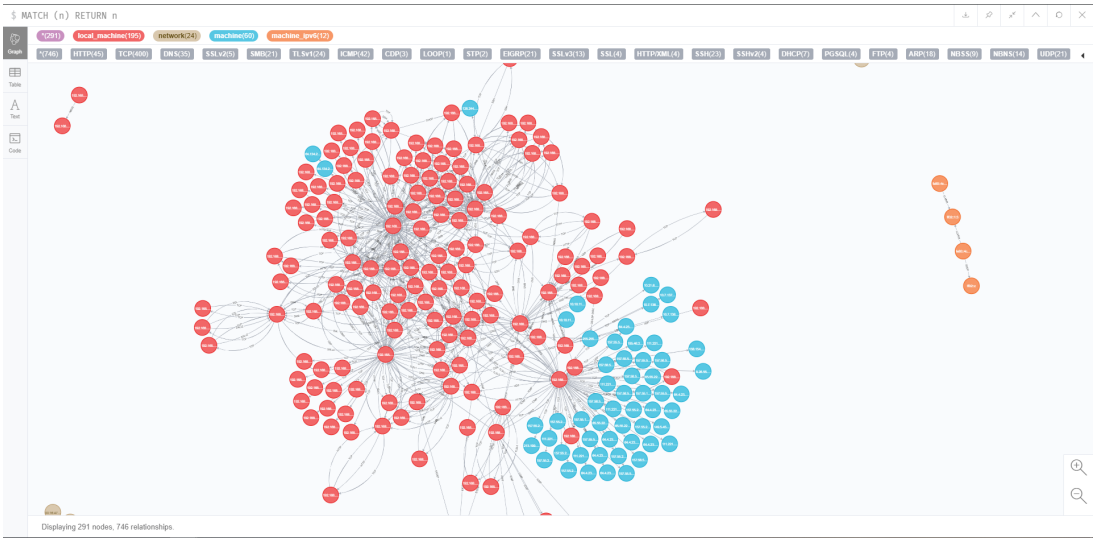


Figura 25: Neo4J – Visualização do cenário 1

Tal como ilustrado na Figura 25, os dados são apresentados diretamente na plataforma do Neo4J. O que permite manipular e filtrar de forma direta, os dados a obter. No caso demostrado, são apresentados todos os relacionamentos, sem o uso de qualquer filtro de dados, ou seja, a cypher graph query utilizada corresponde a “MATCH (n) RETURN n;”. Na figura pode-se verificar, através de um esquema de cores, o tipo de endereço de rede. A cor roxa identifica

o total de nós do grafo de 291 nós. O vermelho corresponde a “local_machine” com o total de 195, que identifica todos os equipamentos ou serviços com endereços de IP internos. O castanho identifica a rede com o total de 24 equipamentos de rede, ligados entre si identificados por mac. A cor azul corresponde aos dispositivos destino com o total de 60, ou seja, os vários endereços de IP de destino na Internet. A cor laranja corresponde a “machine_ipv6” com o total de 12, que identifica os vários equipamentos ou serviços de endereços de rede IPv6.

Outro dado fácil de observar através de uma visualização interativa, ao contrário de numa imagem estática, é o número de comunicações por protocolo de rede. Na Figura 26 verificam-se os diferentes tipos de comunicação estabelecidos na rede, no qual vários endereços se cruzam várias vezes, mas neste caso por protocolos de rede distintos.

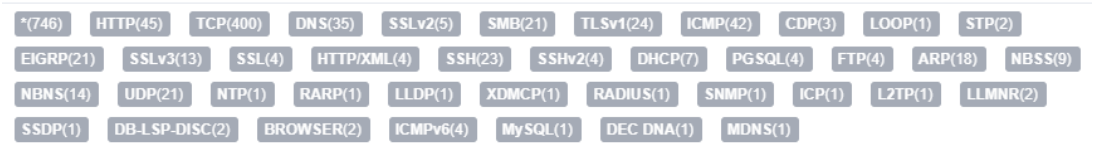


Figura 26: Cenário 1 – Representação por protocolo de rede

A visualização dos dados pode ser manipulada diretamente na plataforma Neo4J, de forma simplista, através de cliques em cada nó ou linha de relação, o que permite detalhar os vários campos de cada nó. Através desses campos, é possível alterar o aspeto de cada nó e linha de relação. Por exemplo ao invés do endereço IP, é possível analisar através dos campos comuns a todos os nós dessa categoria (ex. “local_machine” ou “machine”), como a data de criação (“timestamp”), ultima atualização de cada nó (“last_update”), o contador de relações de cada nó (“count”) que contabiliza o número de acessos que cada nó recebeu. No caso da categoria “machine” foram adicionados pela *framework* os campos extra de enriquecimento desse nó, contendo a localização geográfica, ou nome desse domínio. O resultado dessa representação é apresentado na Figura 27.

Figura 27: Cenário 1: Representação com filtro de dados

4.2 Visualização 3D

O segundo cenário realizado, com base do resultado obtido do cenário anterior, tem como objetivo visualizar os dados de rede, através da visualização no formato 3D (tridimensional). Para tal, foram utilizados componentes Web, como o “neo4j-driver” que permite estabelecer a comunicação direta com a base de dados orientada a grafos do Neo4J e, o componente web “3d-force-graph” [22] que permite renderizar a estrutura de grafos provenientes da base de dados, num espaço tridimensional e interativo. Através de uma *cypher query* predefinida, que realiza o pedido na base de dados Neo4J e, obtém-se os dados no formato JSON. Os dados no formato JSON são por sua vez renderizados num espaço tridimensional.

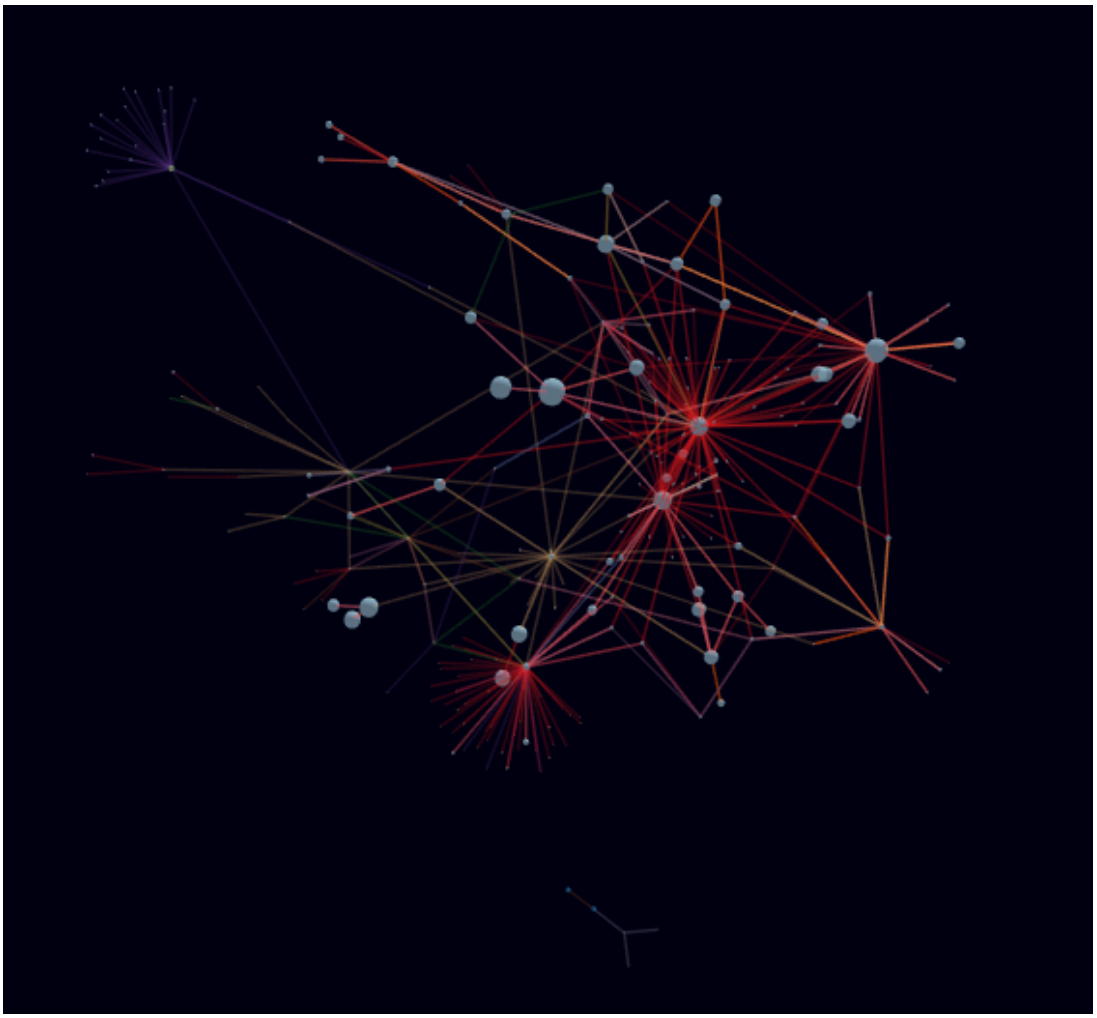


Figura 28: Cenário 2: Neo4J – Visualização HTML com 3d-force-graph

Na Figura 28, é ilustrado o resultado da visualização do cenário no espaço tridimensional. Tal representação permite facilmente identificar e comparar o diâmetro de cada nó, por toda a rede, sendo o tamanho relativo ao volume de tráfego e permite também estabelecer contacto

visual, dos relacionamentos entre si. De forma, a identificar o protocolo de comunicação, são utilizadas cores distintas por categorias. De forma a renderizar os dados da base de dados orientada a grafos, são utilizadas *cypher graph query* que efetuam o pedido diretamente à base de dados e gerem esse resultado num espaço tridimensional. Assim será abordado, a *query* utilizada para representar os dados de rede, por tipo de conexões e, principalmente representar o volume de tráfego de cada nó da rede.

Cypher graph query:

```
MATCH (n) WITH SUM(n.count) as somme
MATCH (n)-[r]->(m)
RETURN {
    id: id(n),
    label:head(labels(n)),
    caption:n.name,
    size:100.0*toFloat(n.count)/toFloat(somme),
    count:n.count,
    country:n.country,
    creation_date:n.creation_date,
    last_update:n.last_update,
    domain:n.domain
} as source,
{
    id: id(m),
    label:head(labels(m)),
    caption:m.name,
    size:100.0*toFloat(m.count)/toFloat(somme),
    count:m.count,
    country:m.country,
    creation_date:m.creation_date,
    last_update:m.last_update,
    domain:m.domain
} as target,
{
    weight:count(r),
```

```
    type:type(r)
} as rel
```

Podemos observar a estrutura destes dados a visualizar, através da variável “somme” que contem o número total de nós na base de dados. Cada relacionamento contém a informação do início e destino e, o tipo de comunicação estabelecida, através das variáveis “source”, “target” e “rel” (relacionamento). Os dados tem de ser estruturados, consoante os campos atribuídos pela *framework*, neste cenário apenas foram abordados os campos mínimos a representar uma rede. Com o acréscimo de novos campos, de enriquecimento de dados, como o nome do dispositivo e categoria (ex. equipamento de rede, servidor, computador), o nome de utilizador, a localização física ou lógica na rede interna e, os portos de início e destino utilizados. O que permitem facilitar a análise de padrões, de forma a identificar unicamente um ou vários equipamentos na rede.

4.3 Use-cases a explorar

O terceiro cenário abordado, tem como objetivo representar alguma das soluções atuais, utilizada nas empresas, para realizar as análises e monitorizações à rede, juntamente com a *framework* desenvolvida. Planeado desta forma, a construção de um cenário de rede completo, através do uso de várias máquinas virtuais e, o uso de uma *firewall* virtualizada pelo simulador de rede, que permite categorizar, bloquear e monitorizar o tráfego de rede, proveniente da Internet, o que contem informação interessante a adicionar à base de dados Neo4J.

Tendo por base, o uso da ferramenta NS3, que permite simular o comportamento dos equipamentos de rede (ex. *switchs*, *routers*, *firewall*) e, gerar ruído por exemplo, latência, perdas de ligação e principalmente, simular o modo *trunk* dos equipamentos (ex. *switch*), que permitem ás máquinas de monitorização, observar de forma completa todo o tráfego da rede. Em vez de ficar limitado ao tráfego, que chega á placa de rede dessa máquina em questão. Desta forma, possibilita uma análise de *performance e bottleneck* dos vários equipamentos na rede e, monitoria todo o tráfego das máquinas alvo correspondente aos colaboradores da empresa e os serviços compratívos. O uso deste cenário, permitirá gerar tráfego na rede de forma controlada e, posteriormente adicionar n máquinas à rede, com possíveis comportamentos maliciosos, de forma a detetar esses padrões suspeitos, o que facilitará a afinação da *framework* e estudar

cenários de rede específicos de diferentes níveis de complexidade.

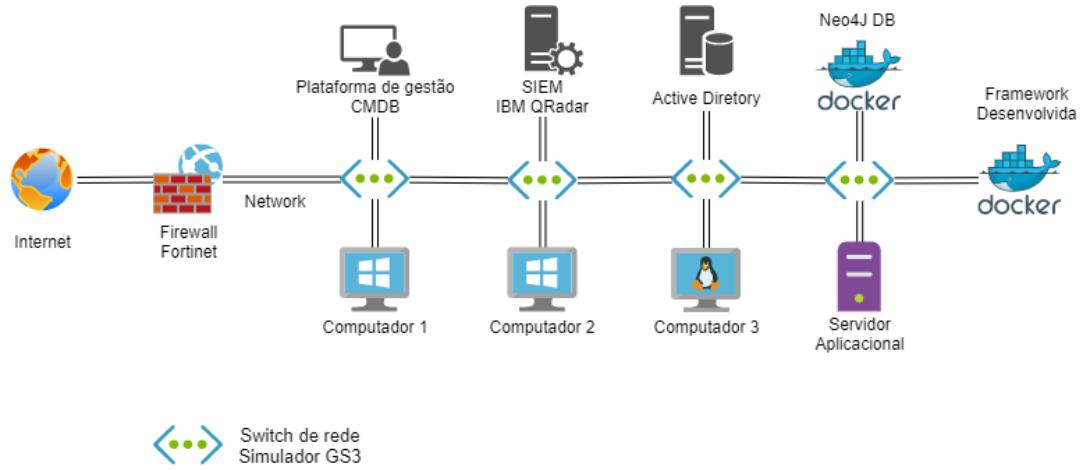


Figura 29: Cenário de rede simulado em NS3

Apresentado na Figura 29, o cenário de rede a simular, em que podemos observar três servidores, dos quais a plataforma de gestão CMDB (ex. juntamente com o Nagios), o SIEM QRadar (ex. juntamente com o Snort, entre outros agentes de monitorização) e, um servidor de AD, responsável pelos acessos às aplicações e serviços internos, o que permite também identificar os endereços IP das máquinas origem juntamente com os seus utilizadores associados. Simultaneamente com quatro máquinas, responsáveis por gerar o tráfego de rede controlado e expectável (ex. Computador 1 e servidor aplicacional). De forma a monitorizar comunicações com a Internet, através de uma *firewall*, que permite identificar e categorizar esses acessos. Por último, através de máquinas em ambiente docker, disponibilizar a base de dados Neo4J e, a *framework* desenvolvida conhecida como “Neo4J-PCAP-Inject”.

4.4 Análise em tempo real - Desafios e oportunidades

Análise em tempo real baseia-se em analisar os dados assim que estes estejam disponíveis na base de dados num formato visual, ou seja, permite tirar conclusões rapidamente após a entrada dos dados no sistema. O que permite tirar conclusões sobre os dados e, reagir em tempo útil. Assim sendo, tirar partido desta abordagem de forma a detetar possíveis problemas na rede, antes que estes cheguem a um estado critico e, bloquear ou identificar possíveis comportamentos maliciosos na rede.

A análise de dados já processados e guardados (dados históricos na base de dados) ser-

vem apenas para analisar esse conjunto de dados, a análise em tempo real pelo contrário permite visualizar, analisar e armazenar esses dados após a sua chegada à base de dados. Isto permite fazer uma análise recorrente e rápida, no entanto não devem ser ativados diversos mecanismos de enriquecimento de dados, para disponibilizar esses dados o mais rápido possível, de forma a melhorar a performance.

No caso do computador utilizado, verificaram-se latências e cargas elevadas no uso dos recursos disponíveis, limitando a monitorização da rede por longos períodos. Num pequeno cenário de testes realizado, através da leitura em tempo real da placa de rede do computador, é possível verificar uma ligeira diferença de tempo na apresentação desses dados na base de dados Neo4J, consoante o número de mecanismos de enriquecimento desses dados utilizados. Ou seja, a visualização dos dados num espaço temporal tridimensional, fora afinado para se atualizar a cada cinco segundos, o que permite ao analista analisar os dados e se manter a par da evolução constante do tráfego da rede.

Capítulo 5

Conclusão

Neste capítulo é realizada uma síntese de todo o trabalho desenvolvido ao longo desta dissertação. São descritas algumas conclusões e é elaborada uma análise crítica sobre o trabalho desenvolvido e de que forma este poderá contribuir para a implementação de novas soluções. Assim como são apresentadas ideias para trabalhos futuros.

5.1 Conclusão

A evolução das infraestruturas de redes informáticas em termos de dimensão e complexidade torna cada vez mais difícil de detetar problemas ou intrusos na rede atempadamente. Os equipamentos e sistemas ligados em rede produzem grande volume de dados relativos ao seu funcionamento (*logs*). Apesar dos sistemas focados na deteção de eventos específicos (SIEM) serem os principais consumidores destes *logs*, é importante facilitar a visualização dos mesmos e analisar a evolução e estado da rede informática, de forma a melhorar ou manter o seu funcionamento.

Nesta dissertação, desenvolvida nesse âmbito, é apresentada uma *framework* com a capacidade de armazenar, visualizar e processar ficheiros de *logs* de sistemas e equipamentos da rede através de uma base de dados orientada por grafos. Esta visualização avançada tem por objetivo simplificar a análise por parte do administrador da rede, dando-lhe uma visão interativa, integrada e em tempo real facultando a identificação dos ativos na rede bem como a análise de padrões capaz de detetar desvios ao comportamento normal/esperado da rede.

Antes do processo de desenvolvimento tecnológico foi necessário existir uma fase de estudo prévio e de análise comparativa entre diferentes tecnologias existentes no mercado. Recorreram-se a estudos realizados por outras pessoas, no que diz respeito às diferentes tecnologias. A *framework* desenvolvida possibilita analisar o tráfego de rede, e posteriormente apresentar os resultados dessa análise num formato visual, orientado a grafos. Através da utilização do renderizador 3D graph é simulado um espaço tridimensional para essa análise. Durante a visualização é possível interagir com os dados, por exemplo ao selecionar um relacionamento ou grafo obtêm-se os dados dessa comunicação.

Os dados da base de dados devem ser ajustados consoante a necessidade da pesquisa na rede e de acordo com os casos específicos de cada organização. O relacionamento mais comummente utilizado será o relacionamento por protocolos de comunicação entre os vários dispositivos na rede em análise. Ainda referente à análise, em cada grafo são apresentados alguns campos base como o endereço IP, mas existe a possibilidade de observar ao pormenor outros campos dos grafos, para tal são aplicados métodos para enriquecimentos desses dados, adicionando a informação correspondente aos endereços de IP eternos, sendo uma delas a localização geográfica.

Com isto, pode concluir-se que a utilização da *framework* possibilita auxiliar os administradores de redes e analistas de segurança, interpretar e interagir com os *logs* de cada equipamento numa rede, pois a transformação de um log de texto para um formato rico de visualização, como é o caso dos grafos, facilita a deteção de problemas de desempenho na rede ou até mesmo a deteção de comportamentos anómalos que possam estar relacionados com falhas ou com a segurança informática. Pode dizer-se que os objetivos expectados foram concluídos com sucesso, pois todas as tarefas objetivadas foram realizadas: a recolha, tratamento e armazenamentos dos registos de *logs* de redes; o processamento desses *logs*; a interligação dos dados já processados com uma base de dados orientada a grafos; a visualização das comunicações e a relação que existe entre elas, através dos grafos; a análise de padrões e a eficácia dos modelos; o que permitiu obter uma análise do tráfego da rede de forma interativa e integrada como fora proposto.

5.2 Trabalhos futuros

Fora abordado o tema de data science e data mining, com o objetivo de apresentar um método adicional de interpretação dos dados, com o principal objetivo de identificar padrões suspeitos na rede, referente a volumes de trafego, comunicação com endereços de IP e portos utilizados fora do padrão da rede, o que leva a necessidade de grandes volumes de dados de trafego de rede e constante monitorização por parte dos algoritmos de data mining, de forma a começarem a detetar o trafego padrão considerado normal, e estabelecer relações com o trafego fora de padrão possivelmente considerados suspeitos. Atravessando uma fase de diversos falsos positivos, que devem ser analisado pelo analista de forma a ajudar os algoritmos a catalogar os possíveis tráfegos maliciosos. Posteriormente com esses campos atualizados na base de dados orientada agrafos permite filtrar ou visualizar diretamente essas comunicações consideradas suspeitas e proceder com as devidas análises e afinações.

Bibliografia

[1] SANS Institute: Reading Room - Intrusion Detection. Accessed on: Sep. 10, 2020. Available: https://www.sans.org/reading-room/whitepapers/detection/paper/352.

[2] Wolfgang Barth. *Nagios, 2nd Edition: System and Network Monitoring*. October 2008.

[3] Cisco. Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper. Accessed on: Sep. 10, 2020. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[4] Darktrace. Cyber AI Platform. Accessed on: Sep. 10, 2020. Available: https://www.darktrace.com/pt/cyber-ai-platform.

[5] P. Drake. Using SNMP to manage networks. In *IEE Colloquium on Designing Resilient Architectures*, November 1991.

[6] IBM Cloud Education. nosql-databases. Accessed on: Sep. 10, 2020. Available: https://www.ibm.com/cloud/learn/nosql-databases.

[7] Fortinet. Single firewall architecture with a single IP subnet - Practical Network Scanning [Book]. Accessed on: Sep. 10, 2020. Available: https://www.oreilly.com/library/view/practical-network-scanning/9781788839235/3805fdbe-01bd-4cea-8ca1-e7bff1fc5ebf.xhtml.

[8] Fortinet. What is a network next-generation firewall? Accessed on: Sep. 10, 2020. Available: https://www.fortinet.com/fr/resources/cyberglossary/firewall.

[9] R. Gerhards. RFC 5424: The Syslog Protocol. Accessed on: Sep. 10, 2020. Available: https://www.hjp.at/doc/rfc/rfc5424.html.

[10] GitLab. Gitlab - the first single application for the entire DevOps lifecycle. Accessed on: Sep. 10, 2020. Available: https://about.gitlab.com/.

[11] IBM. QRadar architecture overview. Accessed on: Sep. 10, 2020. Available: https://www.ibm.com/support/knowledgecenter/SS42VS_7.4/com.ibm.qradar.doc/c_qradar_deployment_guide_arch.html.

[12] Imperva. What is SIEM | Security Information and Event Management Tools. Accessed on: Sep. 10, 2020. Available: https://www.imperva.com/learn/application-security/siem/.

[13] Cliff Joslyn, Sutanay Choudhury, David Haglin, Bill Howe, Bill Nickless, and Bryan Olsen. Massive scale cyber traffic analysis: a driver for graph database research. In *First International Workshop on Graph Data Management Experiences and Systems*, pages 1–6. Association for Computing Machinery, June 2013.

[14] JSON. Introducing JSON. Accessed on: Sep. 10, 2020. Available: https://www.json.org/json-en.html.

[15] Ravi Kumar. Understanding Active Directory. Accessed on: Sep. 10, 2020. Available: https://medium.com/@yoursproductly/understanding-active-directory-4e7508372b80.

[16] Laetitia Leichtnam, Eric Totel, Nicolas Prigent, and Ludovic Mé. Forensic Analysis of Network Attacks: Restructuring Security Events as Graphs and Identifying Strongly Connected Sub-graphs. September 2020.

[17] MaxMind. GeoLite2 Free Downloadable Databases. Accessed on: Sep. 10, 2020. Available: https://dev.maxmind.com/geoip/geoip2/geolite2/.

[18] MiChOo. Packet communication investigator. Accessed on: Sep. 10, 2020. Available: https://github.com/michoo/pci.

[19] MongoDB. Our Customers. Accessed on: Sep. 10, 2020. Available: https://www.mongodb.com/who-uses-mongodb.

[20] Auvik Networks. Network Basics: What Is SNMP and How Does It Work? Accessed on: Sep. 10, 2020. Available: https://www.auvik.com/franklyit/blog/network-basics-what-is-snmp/.

[21] S. Noel, E. Harley, K. H. Tam, M. Limiero, and M. Share. Chapter 4 - CyGraph: Graph-Based Analytics and Visualization for Cybersecurity. In *Handbook of Statistics*, pages 117–167. Elsevier, January 2016.

[22] npm. 3d-force-graph 3d force-directed graph. Accessed on: Sep. 10, 2020. Available: https://www.npmjs.com/package/3d-force-graph.

[23] OneCMDB. Open Source CMDB, Configuration Management Database. Accessed on: Sep. 10, 2020. Available: http://onecmdb.org/wiki/index.php?title=Main_Page.

[24] Neo4j Graph Database Platform. Neo4j network management. Accessed on: Sep. 10, 2020. Available: https://guides.neo4j.com/gcloud-testdrive/network-management.html.

[25] Neo4j Graph Database Platform. What is a Graph Database? Accessed on: Sep. 10, 2020. Available: https://neo4j.com/developer/graph-database/.

[26] Swayam Prakasha. The Growing Popularity of the Snort Network IDS. Accessed on: Sep. 10, 2020. Available: https://www.opensourceforu.com/2016/09/growing-popularity-snort-network-ids/.

[27] SecRepo. SecRepo - Security Data Samples Repository. Accessed on: Sep. 10, 2020. Available: https://www.secrepo.com/.

[28] Jennifer Steiner and Jeffrey I. Schiller. An Authentication Service for Open Network Systems. In *in Usenix Conference Proceedings*, pages 191–202, 1988.

[29] Jeff Petters Updated: 3/29/2020. IDS vs. IPS: What is the Difference? Accessed on: Sep. 10, 2020. Available: https://www.varonis.com/blog/ids-vs-ips/.

[30] Weblink. Protocolos de rede na internet. Accessed on: Sep. 10, 2020. Available: https://www.weblink.com.br/blog/tecnologia/conheca-os-principais-protocolos-de-internet/.

[31] Wikipedia. Authentication protocol. Accessed on: Sep. 10, 2020. Available: https://en.wikipedia.org/w/index.php?title=Authentication_protocol&oldid=985845491.

[32] Wikipedia. Learning Packet Analysis with Data Science | by Ronald Eddings | Hacker Valley Studio | Medium. Accessed on: Sep. 10, 2020. Available: https://medium.com/hackervalleystudio/learning-packet-analysis-with-data-science-5356a3340d4e.

[33] Marc Wilson. Agent vs Agentless Monitoring - What are the Differences & the Best Tools & Software. Accessed on: Sep. 10, 2020. Available: https://www.pcwdld.com/agent-vs-agentless-monitoring.