O Regulamento Geral de Proteção de Dados e a Pseudonimização de *Logs*

Mestrado em Cibersegurança e Informática Forense

Artur Eduardo Lago Torres Varanda

Leiria, setembro de 2019

O Regulamento Geral de Proteção de Dados e a Pseudonimização de *Logs*

Mestrado em Cibersegurança e Informática Forense

Artur Eduardo Lago Torres Varanda

Trabalho de Projeto realizado sob a orientação do Professor Doutor Carlos Manuel da Silva Rabadão, Professor Coordenador do Departamento de Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria

Leiria, setembro de 2019

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Curso de Mestrado em Cibersegurança e Informática Forense, no ano letivo 2018/2019, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Assinado por : **ARTUR EDUARDO LAGO TORRES VARANDA**
Num. de Identificação: BI098192663
Data: 2019.09.28 23:36:07 Hora de Verão de GMT

Dedicatória

À Paula, Gonçalo e Tiago

Agradecimentos

Passados dois anos em que iniciei o curso de Mestrado em Cibersegurança e Informática Forense, gostaria de agradecer às pessoas que de algum modo contribuíram para a concretização do presente trabalho, em particular:

Ao Professor Doutor Carlos Rabadão, por ter assumido a orientação deste trabalho, pela confiança que depositou em mim, pela sua boa disposição, compreensão, conselhos e total disponibilidade que revelou em todas as fases deste projeto.

Ao Professor Miguel Frade, pelas sugestões e conselhos prestados na apresentação intermédia.

Ao Engenheiro Adaíl Oliveira, pelas orientações, revisões finais e valiosas críticas e sugestões.

Aos meus amigos Paula Joaquim e Bruno Severino, colegas e companheiros de curso, pelo incentivo a participar nesta aventura e pelo apoio e motivação constantes, fundamentais para a conclusão deste trabalho.

Ao meu amigo Sotero Freitas, pelo apoio e colaboração que me deu na fase de revisão.

Aos meus colegas de Mestrado, pela motivação e boa disposição que me transmitiram ao longo do primeiro ano do curso.

Por último, à minha mulher Paula e filhos, Gonçalo e Tiago, pelo apoio que me deram ao longo destes dois últimos anos e pelos sacrifícios que fizeram, ao serem privados do tempo comum em família, especialmente durante as férias e fins-de-semana.

After a while you learn that privacy is something you can sell, but you can't buy it back.

Bob Dylan

Resumo

O novo Regulamento Geral de Proteção de Dados (RGPD) impõe que sejam utilizados vários níveis de proteção para garantir que os dados pessoais cumpram os requisitos de segurança da informação. Uma das técnicas recomendadas pelo RGPD para proteger dados pessoais é a Pseudonimização, que consiste em substituir, num documento, nomes reais por nomes fictícios (pseudónimos). Apesar de grande parte dos dados pessoais estarem contidos em documentos e bases de dados, ao nível da administração de sistemas é possível encontrar informações pessoais (endereços IP, *e-mails* e nomes de utilizador) nos ficheiros de registo recolhidos (*logs*), entre outros fins, para demonstrar a conformidade com o princípio da responsabilidade, imposto pelo regulamento. Numa primeira parte, este trabalho consiste no estudo dos processos de registo dos sistemas mais comuns e na caracterização da informação que é necessário pseudonimizar, de modo a que estes registos fiquem em conformidade com as diretivas impostas pelo RGPD. Em seguida, são apresentadas e analisadas algumas estratégias de pseudonimização possíveis, sendo avaliadas as respetivas vantagens e desvantagens de cada uma. Finalmente, são implementadas e testadas algumas soluções concretas de pseudonimização da informação contida nos ficheiros de registo dos sistemas e apresentadas as respetivas conclusões.

Palavras-chave: Dados Pessoais, Logs, Pseudonimização, Privacidade, RGPD

Abstract

The new General Data Protection Regulation (GDPR) demands various levels of protection to be used to ensure that personal data meets information security requirements. One of the techniques recommended by the GDPR to protect personal data is Pseudonymization, which consists of replacing, in a document, real names with fictitious names (pseudonyms). Although much of the personal data is contained in documents and databases, at the system administration level we can find personal information (IP addresses, *e-mails* and usernames) in the log files collected, among others to demonstrate compliance with the principle of accountability imposed by the Regulation. In the first part, this work studies the generation of logs of the most common systems and the characterization of the information that is necessary to pseudonymize, so that these logs comply with the requirements of the GDPR. Possible pseudonymization strategies are presented and analysed; and the respective advantages and disadvantages of each are evaluated. Finally, concrete solutions to the pseudonymization of the information contained in the system's log files are implemented and tested; and the respective conclusions are presented.

Keywords: Personal Data, Logs, Pseudonymization, Privacy, GDPR

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CA	Certification Authority
CPU	Central Processing Unit
CSV	Comma-separated values
DNS	Domain Name Service
DPIA	Data Protection Impact Assessment
ELK	Elasticsearch, Logstash & Kibana
FTP	File Transfer Protocol
GDPR	General Data Protection Regulation
GELF	Graylog Extended Log Format
HMAC	Hash-based message authentication code
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IIS	Internet Information Services
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISP	Internet Service Provider
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control
MD5	Message-Digest algorithm 5
NIST	Nation Institute of Standards and Technology
NTP	Network Time Protocol

RAM	Random-access memory
RCM	Resolução do Conselho de Ministros
REST	Representational State Transfer
RFC	Request for Comments
RGPD	Regulamento Geral de Proteção de Dados
SDK	Software Development Kits
SHA	Secure Hash Algorithm
SIEM	Security Information & Event Management
SPL	Search Processing Language
SSD	Solid-state drive
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TJUE	Tribunal de Justiça da União Europeia
TLS	Transport Layer Security
UDP	User Datagram Protocol
UE	União Europeia
UFW	Uncomplicated Firewall
URL	Uniform Resource Locator
UUCP	Unix to Unix Copy Protocol
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
W3C	World Wide Web Consortium
XML	Extensible Markup Language
WWW	World Wide Web

1. Introdução

O Regulamento Geral de Proteção de Dados (RGPD) contém um conjunto de disposições e requisitos relativos ao tratamento de dados pessoais dos indivíduos e impõe que os controladores de dados pessoais implementem medidas técnicas e organizacionais apropriadas para cumprir com os princípios de proteção de dados.

1.1. Motivação e Objetivos

O regulamento exige que sejam implementadas medidas para garantir o princípio da responsabilidade, o qual exige que o responsável pelo tratamento dos dados se responsabilize pelo que faz com os dados pessoais. Para isso, entre outras medidas, deve fazer registos apropriados para demonstrar a conformidade. Além destes, os registos de eventos dos sistemas operativos e aplicações também poderão conter informação classificada como "dados pessoais".

O primeiro objetivo deste trabalho é identificar a informação contida nos *logs* gerados por sistemas e aplicações que necessita ser anonimizada ou pseudonimizada, de forma a garantir o cumprimento das obrigações impostas pelo RGPD.

Para que se já possível proceder à transformação da informação que necessita ser pseudonimizada, é necessário compreender como é que os sistemas e aplicações manipulam os respetivos registos de eventos.

O segundo objetivo do trabalho é estudar e analisar os processos de registo dos sistemas operativos e aplicações mais comuns de forma a compreender como é que são gerados, estruturados e armazenados.

Uma das medidas técnicas recomendada explicitamente pelo regulamento é a pseudonimização de dados pessoais como uma das várias maneiras de reduzir os riscos do ponto de vista de segurança da informação, de forma a aumentar a privacidade e facilitar o processamento de dados pessoais para além dos propósitos originais de recolha. Esta técnica é usada para reduzir consideravelmente a probabilidade de se identificar um indivíduo a partir dos registos, substituindo os identificadores originais por identificadores fictícios.

O terceiro objetivo deste trabalho é estudar e avaliar diversas estratégias para a anonimização e pseudonimização da informação contida nos *logs* de sistemas e aplicações.

A fim de avaliar as respetivas estratégias, é de todo conveniente implementar algumas soluções concretas. O desenvolvimento destas soluções é um contributo para projetos futuros e permite dar mais visibilidade aos estudos realizados.

O quarto objetivo deste trabalho é desenvolver, testar e comparar várias soluções concretas de pseudonimização da informação contida nos logs de sistemas e aplicações.

1.2. Resumo das Principais Contribuições

Todos os estudos e pesquisas efetuadas, ao longo deste trabalho, a artigos científicos, livros da especialidade, pareceres de grupos de trabalho, regulamentos e guias, estão aqui agregados de forma coerente, de forma a cumprir os primeiros dois objetivos. Esta contribuição, apesar de elementar, apresenta uma visão geral sobre quais os dados pessoais contidos nos *logs* que têm de ser processados, onde é que eles podem ser encontrados, como é que podem ser transformados e por que razão devem ser pseudonimizados.

A metodologias apresentadas para a pseudonimização registos nas diferentes das fases do processo de aglomeração de registos dentro de uma infraestrutura de gestão de *logs*, derivam de uma reflexão sobre possíveis abordagens ao problema. A classificação proposta, em função das fases referidas, apesar de simples, lógica e intuitiva, é uma contribuição útil e original.

Finalmente, de forma a cumprir o quarto objetivo, são apresentados três protótipos para a implementação de soluções concretas de pseudonimização.

1.3. Estrutura do Trabalho

Este trabalho está dividido em seis capítulos que correspondem às etapas percorridas para atingir os objetivos anteriormente determinados.

Neste primeiro capítulo é apresentada a motivação que originou o presente trabalho e definidos os principais objetivos que se pretendem atingir. Em seguida é apresentado um resumo das principais contribuições deste trabalho para definição de estratégias e desenvolvimento de soluções concretas para pseudonimização de dados pessoais contidos em

de *logs* de sistemas e aplicações. Por fim, é a presentada a estrutura deste documento, com o intuito de facultar uma visão geral e abrangente dos respetivos conteúdos.

No capítulo 2, são apresentadas diversas considerações extraídas do RGPD que estão na origem das motivações que levam a realização deste trabalho. Após a definição de alguns conceitos chave, é apresentado o conjunto de direitos básicos atribuídos aos titulares dos dados pessoais que lhes permitem ter um maior controlo sobre esta informação. Em seguida, são apresentadas algumas medidas técnicas e organizativas eficazes, propostas pelo regulamento, para assegurar um nível de segurança adequado. Por fim, são identificados os tipos de dados que de acordo com o regulamento são considerados dados pessoais.

No capítulo 3, são apresentadas algumas recomendações para a recolha de dados pessoais em registos de sistemas e aplicações. Em seguida, são identificados os diversos tipos de fontes geradoras de *logs* e respetivos formatos. São abordados, em particular e de forma mais detalha, os registos de sistema do Windows e do Linux. Ao nível das aplicações, é dado especial relevo à estrutura dos registos de acesso de servidores *Web*, *firewalls* e servidores SSH. Por fim, é abordado o tema da gestão centralizada de registos, identificando as suas principais vantagens e apresentando a arquitetura típica deste tipo de sistemas.

No capítulo 4, são abordadas as principais técnicas usadas para a anonimização e pseudonimização de dados. São apresentados alguns argumentos que justificam a utilização da função HMAC como das mais adequadas para a pseudonimização de informação. Por fim, são discutidos quatro métodos alternativos para a pseudonimização de dados pessoais contidos em registos de sistemas e é feita a respetiva análise comparativa.

No capítulo 5, são apresentadas três soluções concretas para a pseudonimização dos dados pessoais contidos nos registos de sistemas e aplicações. Cada solução utiliza uma infraestrutura de gestão de *logs* diferente pelo que são apresentadas as respetivas particularidades para a implementação de cada uma: configuração de entradas, extração de campos das mensagens, pesquisa e visualização de registos. Nas secções relativas à pseudonimização da informação, são explicados os respetivos processos de transformação e a forma como está desenvolvido o respetivo código. Por fim, é feita uma análise comparativa às três soluções e são apresentadas algumas conclusões.

A conclusão é feita no capítulo 6, onde é apresentado um resumo do trabalho realizado e são feitas algumas considerações sobre os objetivos alcançados. São também referidas as

principais contribuições deste trabalho para comunidade e explicada a respetiva utilidade para projetos futuros. Finalmente, são avançados alguns tópicos para o eventual desenvolvimento do trabalho realizado.

2. Regulamento Geral de Proteção de Dados (RGPD)

O novo Regulamento Geral de Proteção de Dados (RGPD) substitui a Diretiva de Proteção de Dados 95/46/EC e entrou em vigor a partir de 25 de maio de 2018. Este Regulamento é diretamente aplicável em cada Estado Membro e garante um maior grau de harmonização da proteção de dados entre os países da UE [1].

As alterações principais no RGPD começam, desde logo, pelo seu âmbito e pelos sujeitos a quem se destina. Importa, por isso, começar por distinguir, de acordo com o artigo 4.º do RGPD, as figuras do responsável pelo tratamento de dados e do subcontratante.

O responsável pelo tratamento dos dados é a pessoa singular ou coletiva, ente público ou privado, agência, instituição ou qualquer outro organismo que decide como e porque é que os dados são processados. Assim, é a pessoa física ou jurídica que, isoladamente ou em conjunto com outros, determina os fins e meios de processamento de dados pessoais. De acordo com o artigo 5.º do RGPD, o responsável pelo tratamento de dados tem o dever de provar o cumprimento dos princípios relativos ao tratamento de dados pessoais a que está vinculado.

O subcontratante é a pessoa singular ou coletiva, ente público ou privado, agência, instituição ou qualquer outro organismo que trata os dados pessoais por conta do responsável pelo tratamento destes, ou seja, é aquele que processa dados pessoais em nome do responsável pelo tratamento.

De acordo com o artigo 28.º do RGPD, o tratamento pode ser efetuado em nome de um responsável pelo tratamento, mas este deve subcontratar apenas entidades que forneçam garantias suficientes de cumprimento do RGPD, isto é, subcontratantes que demonstrem evidências da implementação das medidas técnicas e organizativas adequadas para que o processamento de dados satisfaça os requisitos do Regulamento. Assim, tanto o responsável pelo tratamento como o subcontratante têm de implementar os controlos necessários para garantir a conformidade com o RGPD, se os dados a serem processados forem de cidadãos da UE.

2.1. Direitos Básicos

O Regulamento fornece oito direitos básicos que proporcionam aos titulares dos dados um melhor controlo sobre os seus dados pessoais:

- O direito de ser informado obriga o responsável pelo tratamento dos dados a fornecer informações sobre como e quais os dados que são processados. Isto normalmente é feito através de um aviso de privacidade muito claro e fácil de entender;
- O direito de acesso confere aos titulares dos dados o direito de obter confirmação sobre o processamento dos dados recolhidos, o acesso aos seus dados pessoais e informações suplementares;
- O direito de retificação permite que os titulares exijam a retificação de dados, caso estes estejam incompletos ou imprecisos. Se o responsável pelo tratamento dos dados tiver divulgado as informações a um subcontratante, é obrigado a informá-lo sobre a necessidade de retificação;
- O direito ao esquecimento permite que os indivíduos solicitem a remoção de dados pessoais dos sistemas de informação;
- O direito de restringir o processamento permite que os titulares de dados exijam aos responsáveis pelo tratamento que estes deixem de processar os dados recolhidos;
- O direito à portabilidade dos dados concede aos titulares o direito de obter os seus dados e transferi-los para outro responsável. Os dados deverão poder ser transferidos de um sistema para outro sem que isso comprometa a segurança ou afete a respetiva usabilidade;
- O direito de contestar consiste em dar aos titulares dos dados o direito de se oporem ao processamento para fins de pesquisa científica e/ou histórica, *marketing* ou processamento em geral. Os responsáveis pelo tratamento dos dados devem interromper o processamento, a não ser que possam apresentar uma razão legítima e lícita para o fazer.

A notificação por violação de dados passa a ser obrigatória em todos os Estados Membros, sempre que puder "resultar num risco para os direitos e liberdades dos indivíduos". A notificação deve ser feita dentro das 72 horas após o conhecimento da ocorrência. Os subcontratantes têm de notificar, atempadamente, os responsáveis pelo tratamento.

2.2.Medidas de Segurança

O RGPD obriga as entidades que tratam dados pessoais a realizar uma grande reforma em termos operacionais. Assim, os responsáveis pelo tratamento têm de aplicar medidas técnicas e organizativas eficazes para assegurar um nível de segurança adequado ao risco. Entre estas medidas, incluem-se:

- pseudonimização e/ou cifragem dos dados pessoais;
- capacidade de assegurar a confidencialidade, integridade, disponibilidade e resiliência permanentes dos sistemas e dos serviços de tratamento;
- capacidade de restabelecer a disponibilidade e o acesso aos dados pessoais de forma atempada no caso de um incidente físico ou técnico;
- processos que permitam testar, apreciar e avaliar regularmente a eficácia das medidas técnicas e organizativas por forma a garantir a segurança do tratamento da informação.

O princípio da responsabilidade exige não só o dever de cumprir todas as obrigações, mas também o de demonstrar que as mesmas são cumpridas. Para poder demonstrar conformidade com o RGPD, o responsável pelo tratamento dos dados deve implementar medidas que atendam aos princípios de proteção de dados desde a conceção e por omissão (artigo 25.º). Este princípio exige que as medidas de proteção de dados sejam projetadas no desenvolvimento de projetos de produtos e serviços. Estas medidas incluem a pseudonimização de dados pessoais (considerando 78).

De acordo com o considerando 82 que serviu de base para a redação do artigo 30.º, "a fim de comprovar a observância do presente Regulamento, o responsável pelo tratamento ou o subcontratante deverá conservar registos de atividades de tratamento sob a sua responsabilidade. Os responsáveis pelo tratamento e subcontratantes deverão ser obrigados a cooperar com a autoridade de controlo e a facultar-lhe esses registos, a pedido, para fiscalização dessas operações de tratamento".

Isto significa que o responsável pelo tratamento tem a obrigação de documentar a forma como a transferência de dados ocorre (através de uma avaliação de impacto) e registar todas as atividades envolvidas através de *logs* de auditoria.

A Resolução do Conselho de Ministros n.º 41/2018, considera fundamental definir orientações técnicas para a Administração Pública, recomendando-as ao setor empresarial do Estado, em matéria de arquitetura de segurança das redes e sistemas de informação e procedimentos a adotar de modo a cumprir as normas do RGPD [2].

Um dos requisitos técnicos gerais exigidos pela RCM n.º 41/2018 prende-se precisamente com a Capacidade de monitorização, registo e análise de toda a atividade de acessos de modo a procurar ameaças prováveis. As medidas obrigatórias necessárias para cumprir com este requisito são as seguintes:

- Deve ser guardado registo de atividade (log) de todas as ações que um utilizador efetue sobre dados pessoais, independentemente do seu perfil e função;
- Todos os registos de atividade (log) devem ser armazenados apenas em modo de leitura, devendo, com uma periodicidade máxima de 1 mês, ser englobados num único bloco de registos e assinado digitalmente (garantia de integridade);
- Deve ser guardado registo de atividade (log) de todos os acessos e tentativas falhadas de acesso, obedecendo aos requisitos anteriores;
- Garantir que os registos de atividade provenientes dos diversos subsistemas (Sistemas Operativos, aplicações, browsers, Sistema de Gestão de Base de Dados — SGBD, etc.) são inequivocamente associados à sua origem;
- Os registos de atividade (log) devem conter, no mínimo, o endereço de acesso (IP e Porto), Host, HASH da conta do utilizador que efetuou a ação, ação efetuada (CRUD), Tipo de Dado Pessoal onde a ação foi efetuada, data/hora/ minuto/segundo (TimeStamp) da ação, alteração efetuada sobre o dado pessoal.

Os sistemas estão normalmente configurados para recolher e armazenar *logs* de acesso, de erro e de auditoria de segurança. Todos estes *logs* contêm dados pessoais. Por exemplo, os endereços IP são especificamente definidos como dados pessoais de acordo com o Regulamento.

Apesar de não poderem ser recolhidos nem armazenados dados pessoais sem o consentimento dos titulares, podem ser recolhidos *logs* com o objetivo legítimo de detetar e impedir fraudes e acessos não autorizados e para garantir a segurança dos sistemas (considerando 49).

2.3.Dados Pessoais no contexto do RGPD

De acordo com o RGPD, "dados pessoais" são quaisquer informações relacionadas com um indivíduo vivo identificado ou identificável. Também constituem dados pessoais as diferentes informações, recolhidas em conjunto, que podem levar à identificação de uma determinada pessoa.

Os dados pessoais que forem pseudonimizados ou cifrados, mas que possam ser usados para reidentificar uma pessoa, permanecem como dados pessoais.

Por outro lado, os dados pessoais que forem anonimizados, de forma a tornar o titular não identificável, não são considerados dados pessoais. Para que os dados sejam verdadeiramente anonimizados, o processo tem de ser irreversível.

O RGPD, no considerando 30, clarifica que os endereços IP devem ser considerados dados pessoais, pois o texto inclui "identificadores por via eletrónica" na definição de "dados pessoais".

Assim são exemplos de dados pessoais:

- o nome e apelido;
- o endereço de uma residência;
- o *e-mail*, por exemplo, nome.apelido@empresa.com;
- o número de um cartão de identificação;
- dados de localização geográfica (por exemplo, num telemóvel);
- um endereço IP;
- informação de navegação (*cookie*);
- o identificador de publicidade no telefone;
- os dados recolhidos por um hospital ou médico, que permitam identificar uma pessoa de forma inequívoca.

Por outro lado, não são considerados dados pessoais:

- o número de registo de uma empresa;
- o *e-mail* da empresa, por exemplo, info@empresa.com;
- dados anonimizados.

Tem havido um grande consenso em considerar os endereços IP estáticos como dados pessoais. O mesmo não acontece em relação aos endereços IP dinâmicos. Embora a maioria das autoridades concorde que um endereço dinâmico na posse de um Fornecedor de Acesso à Internet (Internet Service Provider - ISP) pode ser usado para identificar um indivíduo, sendo por isso considerado um dado pessoal, há pouco consenso sobre se os endereços IP dinâmicos devem ser considerados dados pessoais.

Em 19 de outubro de 2016, o Tribunal de Justiça da União Europeia (TJUE) publicou o acórdão do Processo 582/14 - Patrick Breyer vs. Alemanha, no qual considerou que os endereços IP dinâmicos são dados pessoais em determinadas circunstâncias [3]. O TJUE decidiu que um endereço IP dinâmico será um dado pessoal nas mãos de um operador se:

- existe uma forma de se poder vincular o endereço IP dinâmico à identidade de um indivíduo (como acontece se o operador for um ISP);
- o responsável pelo tratamento de dados tem um meio legal de obter acesso à informação mantida pelo ISP para identificar o indivíduo.

Se uma empresa recolher e processar endereços IP, mas não tiver meios legais de vincular esses endereços IP às identidades dos utilizadores, é improvável que esses endereços IP sejam dados pessoais. No entanto, as empresas devem observar que se tiverem informações suficientes para vincular um endereço IP a um indivíduo específico (por exemplo, por meio de detalhes de login, cookies ou qualquer outra informação ou tecnologia), esse endereço IP será um dado pessoal e estará sujeito à legislação da UE em matéria de proteção de dados. Para muitas empresas, é provável que isto exija uma revisão de como os endereços IP são tratados no contexto e propósito das respetivas atividades.

2.4. Síntese

Neste capítulo foram apresentadas várias determinações introduzidas pelo RGPD. Foram definidos alguns conceitos chave e apresentado o conjunto de direitos básicos atribuídos aos titulares dos dados pessoais. Foram também abordadas algumas medidas técnicas e organizativas eficazes, propostas pelo regulamento, para assegurar um nível de segurança adequado. Finalmente, foram identificados os tipos de dados que de acordo com o regulamento são considerados dados pessoais.

No capítulo seguinte vai ser caracterizada a informação contida nos registos de eventos de sistemas e vão ser estudados e analisados os processos de registo dos sistemas mais comuns.

3. Registos de Sistemas e Aplicações (*Logs*)

Ao nível da administração de sistemas é possível encontrar informações pessoais nos ficheiros de registo recolhidos (*logs*), nomeadamente, para demonstrar a conformidade com o princípio da responsabilidade, imposto pelo regulamento. Neste capítulo é caracterizada a informação contida nesses registos que é necessário pseudonimizar, de modo a que fiquem em conformidade com as diretivas impostas pelo RGPD. Numa segunda parte, são estudados e analisados os processos de registo dos sistemas mais comuns.

De acordo com o RGPD, os *logs* de acesso, os *logs* de erro e os *logs* de auditoria de segurança são agora passíveis de conter informações pessoais. As empresas devem, por isso, tratar os dados de IP e os *cookies* como se fossem identificadores pessoais. No entanto, os dados pessoais poderão ser recolhidos e armazenados nos ficheiros de registo de eventos de servidores e de aplicações, para ajudar a detetar e evitar fraudes ou acessos não autorizados.

Além de serem úteis para a organização, ao fornecer pistas sobre atividades hostis que afetam a rede e fornecer informações úteis para identificar e solucionar problemas de equipamentos, os ficheiros de registo podem também fornecer informações a *hackers* e cibercriminosos que as podem utilizar para comprometer o sistema e a segurança dos dados.

3.1. Recomendações para a recolha de dados pessoais em *Logs*

Um relatório preliminar do Grupo de Trabalho da IETF IntArea revela que as mudanças nos regulamentos de proteção de dados fizeram com que as melhores práticas estabelecidas se tornassem inadequadas. O relatório oferece uma lista com um conjunto de atualizações para o RFC6302 (Recomendações para servidores voltados para a Internet), para permitir a conformidade com o RGPD e sugere que os administradores de sistemas adotem uma abordagem de minimização de dados para configurar os *logs* do servidor [4]:

- os endereços IP completos só devem ser armazenados pelo tempo necessário para fornecer um serviço;
- os *logs* devem incluir apenas os dois primeiros octetos de endereços IPv4 ou os três primeiros octetos de endereços IPv6;
- os registos de endereços IP de entrada não devem durar mais do que três dias (suficiente para cobrir um fim de semana);

- não devem ser registados identificadores desnecessários (portos de origem e destino, registos de data e hora, números de protocolos de transporte);
- os *logs* devem ser protegidos contra acesso não autorizado.

Não estão cobertos pela proposta os registos obrigatórios para cumprir a legislação de retenção de dados de telecomunicações.

3.2. Fontes Geradoras de *Logs*

Os sistemas e respetivas aplicações geram *logs* de vários tipos, incluindo os originados por servidores de *e-mail*, servidores *Web* e servidores de bases de dados. Estes *logs* podem fornecer informações úteis sobre indivíduos que realizam ataques aos componentes do sistema. As principais fontes geradoras de *logs* são:

- *Software* de Segurança (Anti-malware, IDS, *Firewalls*, *Routers*)
- Sistema Operativo (Eventos do Sistema, Registos de Auditoria)
- Aplicações (*E-mail*, Servidores *Web*, Bases de Dados)

O documento do NIST, *SP 800-92 - Guide to Computer Security Log Management*, define um conjunto de fontes geradoras de *logs* para cada um dos três tipos apresentados acima [5].

3.2.1. *Software* de Segurança

O *software* de segurança instalado na rede ou nos *host*s é utilizado para detetar atividade maliciosa, proteger sistemas e informação e para auxiliar o trabalho de resposta a incidentes. Assim, este *software* constitui a maior fonte de registos de eventos de segurança. Entre os diversos tipos de *software* de segurança, encontram-se [6]:

- *Software anti-malware* – O *software anti-malware* mais comum é o *software* antivírus, que regista os eventos de deteção de *malware*, tentativas de desinfeção e ficheiros colocados em quarentena. Este *software* também pode registar quando são efetuadas análises de varrimento e quando são feitas atualizações do *software* e das assinaturas do *malware*;
- Sistemas de Deteção e Prevenção de Intrusão – Os sistemas de deteção e de prevenção de intrusão registam informações detalhadas de atividades suspeitas e de ataques detetados, bem como as respostas efetuadas pelo *software* de segurança para anular a atividade maliciosa. Este *software* também efetua, periodicamente,

rotinas de verificação da integridade de ficheiros, gerando os registos correspondentes;

- *Software* de Acesso Remoto – O acesso remoto é normalmente assegurado através de redes privadas virtuais (VPN). Os sistemas VPN costumam registar as tentativas de autenticação, bem como as datas e as horas de abertura e encerramento das sessões de cada utilizador, bem como o volume de dados enviados e recebidos em cada sessão;
- *Web Proxies* – Os *Web Proxies* são máquinas intermediárias através das quais os servidores *Web* são acedidos. Podem fazer pedidos de páginas *Web* em nome dos utilizadores e armazenam cópias das respostas numa *cache* para tornar as futuras pesquisas mais eficientes. Também podem ser usados para restringir o acesso a páginas e para adicionar uma camada de proteção entre os clientes e os servidores. Normalmente, os *Web Proxies* registam todos os URLs acedidos pelos clientes;
- *Software* de Gestão de Vulnerabilidades – Este *software* regista o histórico de instalação de atualizações de *software* (*patches*) e o estado de vulnerabilidade de cada *host*. Isto inclui as vulnerabilidades conhecidas e o *software* de atualização em falta;
- Servidores de Autenticação – Estes servidores que incluem, entre outros, os servidores de diretório (ex: Active Directory ou OpenLDAP), normalmente registam as tentativas de autenticação, incluindo a origem, o nome de utilizador, a data e a hora;
- *Software* de Configuração de *Routers* – O *software* de configuração dos *routers* da rede serve para permitir ou bloquear certos tipos de tráfego de rede, de acordo com uma política pré-definida. Normalmente, estes *routers* registam certas características da atividade bloqueada;
- *Firewalls* – Tal como os *routers*, as *firewalls* podem permitir ou bloquear a atividade de acordo com o uma política pré-definida. Estas, no entanto, utilizam métodos muito mais sofisticados de análise do tráfego de rede. Também podem rastrear o tráfego e analisar o conteúdo. Normalmente, têm políticas de gestão de tráfego muito mais complexas e geram registos muito mais detalhados do que os *routers*;
- *Software* de Controlo de Acesso à Rede - Muitas vezes, antes de ser autorizado o acesso à rede de um determinado *host*, são avaliadas, remotamente, as respetivas configurações de segurança. Isto é feito, normalmente, através de um servidor de

controlo de acesso à rede e de agentes instalados nos *host*s. Os *host*s que não respondem aos pedidos de verificação do servidor, ou que falham nos testes, são colocados em quarentena num segmento separado de uma rede virtual local (VLAN). Estes servidores de controlo de acesso registam toda a informação relacionada com os *host*s colocados em quarentena.

3.2.2. Sistema Operativo

Os sistemas operativos de servidores, *host*s e dispositivos de rede (*routers* e comutadores) registam, normalmente, uma grande variedade de informação relacionada com a segurança. Os tipos mais comuns de dados de segurança relativos a sistemas operativos são:

- Eventos do Sistema – Os eventos do sistema são funções realizadas por componentes do sistema operativo, por exemplo, o encerramento da sessão ou a inicialização de um serviço. Normalmente, são registados os eventos falhados e alguns eventos de relevo bem-sucedidos. No entanto, os sistemas operativos permitem que os administradores especifiquem quais os eventos a serem registados. Os detalhes registados também podem variar muito. Normalmente é registada a data/hora (*timestamp*) e outra informação de suporte: evento, estado, código do erro, nome do serviço e nome do utilizador ou da conta relacionada com o evento;
- Registos de Auditoria – Os registos de auditoria contêm informação relativa a eventos de segurança, por exemplo, tentativas de autenticação, acesso a ficheiros, alterações da política de segurança, alterações de conta (criação, eliminação ou mudança de privilégios de uma conta) e utilização de privilégios. Os sistemas operativos permitem que os administradores especifiquem quais os tipos de eventos que devem ser auditados e quais as atividades que devem ser registadas.

Os registos do sistema operativo também contêm informação do *software* de segurança e de outras aplicações instaladas no sistema. Estes registos são dos mais importantes para identificar ou investigar atividades suspeitas que envolvem um *host* particular. Assim que uma atividade suspeita é identificada pelo *software* de segurança, os sistemas operativos são consultados para se obter mais informação sobre essa atividade. Por exemplo, um dispositivo de rede pode detetar um ataque a um determinado *host*. Os registos do sistema operativo desse *host* podem indicar se algum utilizador estava a usar esse *host* no momento do ataque e se o ataque foi bem-sucedido.

Como é visto mais adiante, alguns registos do sistema operativo são gerados no formato *Syslog*. Outros, como os dos sistemas baseados em Windows, são armazenados em formatos proprietários.

3.2.3. Aplicações

Os sistemas operativos e o *software* de segurança garantem a instalação e a proteção de aplicações que são usadas para armazenar, aceder e operar os dados usados nos processos de negócio de uma organização. Algumas aplicações geram os seus próprios ficheiros de registo, enquanto outras utilizam as capacidades de registo do sistema operativo onde são instaladas. O tipo de informação que as aplicações registam varia consideravelmente. Entre os tipos de registo de aplicações mais comuns, encontram-se:

- Pedidos de Cliente e Respostas de Servidor – Estes registos podem ser muitos úteis na reconstrução da sequência de eventos e na determinação do respetivo resultado. Se a aplicação registar as autenticações de utilizador bem-sucedidas, é possível determinar qual o utilizador que fez um determinado pedido. Algumas aplicações efetuam registos bem detalhados, por exemplo, os servidores de *e-mail* registam o emissor, os recetores, o assunto e os nomes dos anexos de cada *e-mail*; os servidores *Web* registam os pedidos de URL e o tipo de resposta dada pelo servidor; e as aplicações financeiras registam a informação acedida por cada utilizador. Esta informação pode ser usada para identificar ou investigar incidentes e para monitorizar a utilização da aplicação para efeitos de auditoria e conformidade com os regulamentos;
- Informação da Conta – Estes registos incluem as tentativas de autenticação, alterações de conta (criação, eliminação ou mudança de privilégios de uma conta) e utilização de privilégios. Além de identificar eventos de segurança como ataques de força bruta para adivinhar palavras passe e escalamento de privilégios, podem ser usados para identificar quem usou a aplicação e quando é que a utilizou;
- Informação de Utilização – Estes registos incluem o tráfego realizado por unidade de tempo e o volume total do tráfego. Isto pode ser útil para certos tipos de monitorização de segurança, por exemplo, um grande aumento de atividade de *e-mail* pode indiciar uma nova ameaça de *malware* baseada em *e-mail* ou uma disponibilização indevida de informação;

- Atividades de operação significativas – Os registos de inicialização ou o encerramento de uma aplicação, falhas da aplicação ou alterações significativas na configuração da aplicação podem ser usadas para identificar falhas de segurança. Muita desta informação, em particular para aplicações que apenas são usadas através de ligações de rede seguras, apenas pode ser registada pelas aplicações, o que torna estes registos particularmente valiosos para o tratamento de incidentes de segurança, auditoria e garantia de conformidade, relacionados com as respetivas aplicações. O problema é que estes registos são feitos em formatos proprietários, o que dificulta a sua utilização. Além disso, os dados dependem muito do contexto, o que faz com que sejam necessários recursos adicionais para analisar o respetivo conteúdo.

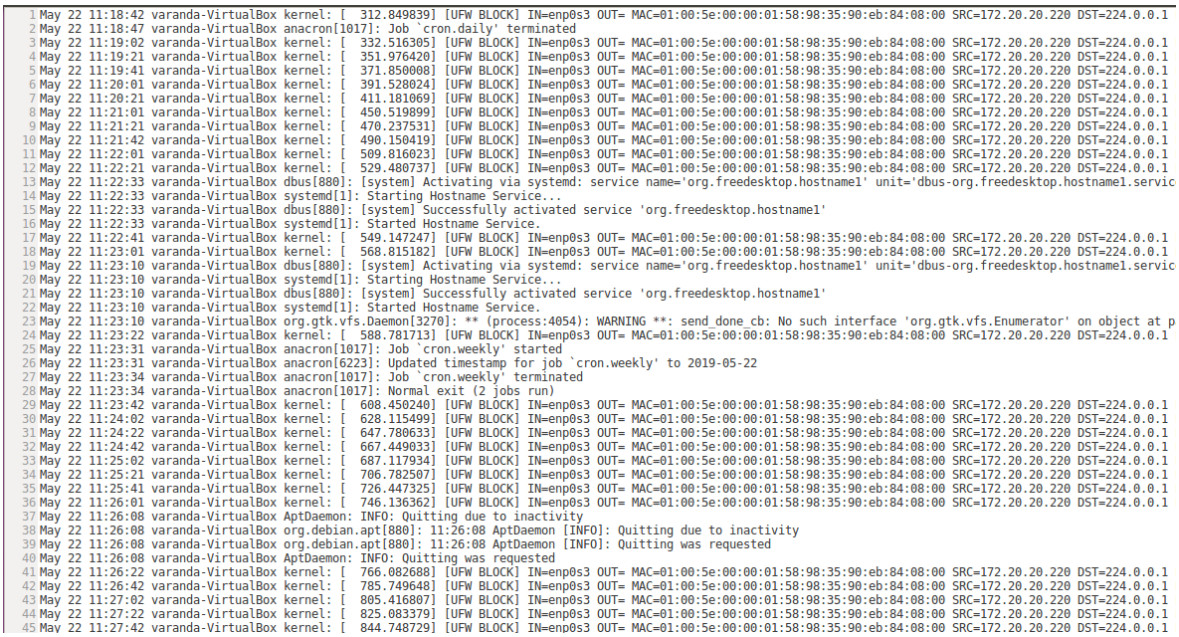
3.3.Formatos dos Registos

O formato e a sintaxe dos registos definem a forma como estas mensagens são formadas, transportadas, armazenadas, monitorizadas e analisadas. No caso mais simples, os eventos registados podem ser tratados como sequências de caracteres de texto e a análise dos registos pode ser feita através de pesquisa de texto simples. Em processos automáticos de análise de eventos é necessário que os geradores e os utilizadores dos registos se entendam e que concordem com a sintaxe utilizada.

Cada campo de um registo de eventos, devidamente formatado, contém a representação de uma determinada informação. Infelizmente, existem muitos registos que não seguem qualquer formato específico ou predeterminado, ou o seguem apenas em parte, como por exemplo no campo data/hora (*timestamp*), sendo por isso considerados texto livre.

Existem vários aspetos a considerar nos formatos dos registos. Em primeiro lugar, o ficheiro pode ser binário ou pode ser um ficheiro de texto ASCII. Depois, o ficheiro pode ser lido com um editor de texto simples ou podem ser necessárias ferramentas de conversão para se poder ter acesso à informação. Um exemplo bem conhecido de um registo legível em ASCII é o formato *Syslog* do UNIX, como se pode ver na Figura 3.1.

Os servidores *Web*, as *Firewalls* e muitas outras aplicações, também geram registos em ficheiros de texto, qualquer que seja a plataforma onde estão instalados, podendo ser lidos e analisados facilmente.



```
May 22 11:18:42 varanda-VirtualBox kernel: [  312.849839] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:18:47 varanda-VirtualBox anacron[1017]: Job `cron.daily' terminated
May 22 11:19:02 varanda-VirtualBox kernel: [  332.516305] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:19:21 varanda-VirtualBox kernel: [  351.976420] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:19:41 varanda-VirtualBox kernel: [  371.850008] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:20:01 varanda-VirtualBox kernel: [  391.528024] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:20:21 varanda-VirtualBox kernel: [  411.181069] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:21:01 varanda-VirtualBox kernel: [  450.519899] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:21:21 varanda-VirtualBox kernel: [  470.237531] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:21:42 varanda-VirtualBox kernel: [  490.150419] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:22:01 varanda-VirtualBox kernel: [  509.816023] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:22:21 varanda-VirtualBox kernel: [  529.480737] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:22:33 varanda-VirtualBox dbus[880]: [system] Activating via systemd: service name='org.freedesktop.hostname1' unit='dbus-org.freedesktop.hostname1.servic
May 22 11:22:33 varanda-VirtualBox systemd[1]: Starting Hostname Service...
May 22 11:22:33 varanda-VirtualBox dbus[880]: [system] Successfully activated service 'org.freedesktop.hostname1'
May 22 11:22:33 varanda-VirtualBox systemd[1]: Started Hostname Service.
May 22 11:22:41 varanda-VirtualBox kernel: [  549.147247] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:23:01 varanda-VirtualBox kernel: [  568.815182] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:23:10 varanda-VirtualBox dbus[880]: [system] Activating via systemd: service name='org.freedesktop.hostname1' unit='dbus-org.freedesktop.hostname1.servic
May 22 11:23:10 varanda-VirtualBox systemd[1]: Starting Hostname Service...
May 22 11:23:10 varanda-VirtualBox dbus[880]: [system] Successfully activated service 'org.freedesktop.hostname1'
May 22 11:23:10 varanda-VirtualBox systemd[1]: Started Hostname Service.
May 22 11:23:10 varanda-VirtualBox org.gtk.vfs.Daemon[3270]: ** (process:4054): WARNING **: send_done_cb: No such interface 'org.gtk.vfs.Enumerator' on object at p
May 22 11:23:22 varanda-VirtualBox kernel: [  588.781713] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:23:31 varanda-VirtualBox anacron[1017]: Job `cron.weekly' started
May 22 11:23:31 varanda-VirtualBox anacron[6223]: Updated timestamp for job `cron.weekly' to 2019-05-22
May 22 11:23:34 varanda-VirtualBox anacron[1017]: Job `cron.weekly' terminated
May 22 11:23:34 varanda-VirtualBox anacron[1017]: Normal exit (2 jobs run)
May 22 11:23:42 varanda-VirtualBox kernel: [  608.450240] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:24:02 varanda-VirtualBox kernel: [  628.115499] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:24:22 varanda-VirtualBox kernel: [  647.780633] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:24:42 varanda-VirtualBox kernel: [  667.449033] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:25:02 varanda-VirtualBox kernel: [  687.117934] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:25:21 varanda-VirtualBox kernel: [  706.782507] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:25:41 varanda-VirtualBox kernel: [  726.447325] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:26:01 varanda-VirtualBox kernel: [  746.136362] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:26:08 varanda-VirtualBox AptDaemon: INFO: Quitting due to inactivity
May 22 11:26:08 varanda-VirtualBox org.debian.apt[880]: 11:26:08 AptDaemon [INFO]: Quitting due to inactivity
May 22 11:26:08 varanda-VirtualBox org.debian.apt[880]: 11:26:08 AptDaemon [INFO]: Quitting was requested
May 22 11:26:08 varanda-VirtualBox AptDaemon: INFO: Quitting was requested
May 22 11:26:22 varanda-VirtualBox kernel: [  766.082688] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:26:42 varanda-VirtualBox kernel: [  785.749648] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:27:02 varanda-VirtualBox kernel: [  805.416807] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:27:22 varanda-VirtualBox kernel: [  825.083379] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
May 22 11:27:42 varanda-VirtualBox kernel: [  844.748729] [UFW BLOCK] IN=enp0s3 OUT= MAC=01:00:5e:00:00:01:58:98:35:90:eb:84:08:00 SRC=172.20.20.220 DST=224.0.0.1
```

Figura 3.1 - Ficheiro de registo no formato *Syslog*.[1]

É preciso ter em atenção, no entanto, que texto legível e texto formatado não significa necessariamente o mesmo. Por exemplo, não é qualquer indivíduo que consegue ler um ficheiro de texto de um registo formatado em XML, com linhas muito longas e diversos identificadores.

O exemplo mais comum de um ficheiro de registo binário é um ficheiro de registo de eventos do Windows (*Windows Event Log*). A leitura destes registos necessita de utilizar a ferramenta de visualização de eventos do Windows, o *Event Viewer*. Esta ferramenta converte o ficheiro de registo binário num evento legível.

Os registos feitos em formato binário são normalmente mais pequenos do que os correspondentes em formato ASCII, ocupam um menor espaço no disco e utilizam menor tráfego de rede ao serem transportados. Além disso, requerem um menor processamento para serem interpretados e contêm estruturas de dados claramente definidas, o que torna a sua análise mais eficiente. Um interpretador de registos no formato ASCII tem de processar mais dados e normalmente tem de usar um identificador de padrões para extrair os dados dos diversos campos da mensagem.

[1] Fonte: Captura de ecrã feita pelo autor.

3.4. Geração de *Logs* no Microsoft Windows

A Microsoft decidiu há muito tempo desenvolver o seu próprio sistema de geração e recolha de registos de eventos, denominado *Event Log*. O sistema evoluiu ao longo dos anos e é quase tão antigo como o próprio Windows. Este sistema é usado para recolher e visualizar dois grandes tipos de registos: registos do Windows, e registos de Serviços e Aplicações.

Os registos do Windows englobam, entre outros, os registos de Aplicação, de Segurança e de Sistema. O tipo de registos de Segurança é muito importante, pois é aqui que se podem encontrar os registos relativos à abertura e ao encerramento da sessão do utilizador e aos acessos a recursos do sistema operativo. O tipo de registos de Aplicação contém informação relativa ao estado, erros e outras informações relevantes, relativas às aplicações [7].

Como é dito acima, os registos do Windows são arquivados em ficheiros binários com a extensão *evtx* e podem ser visualizados através do *Event Viewer*, apresentado na Figura 3.2.

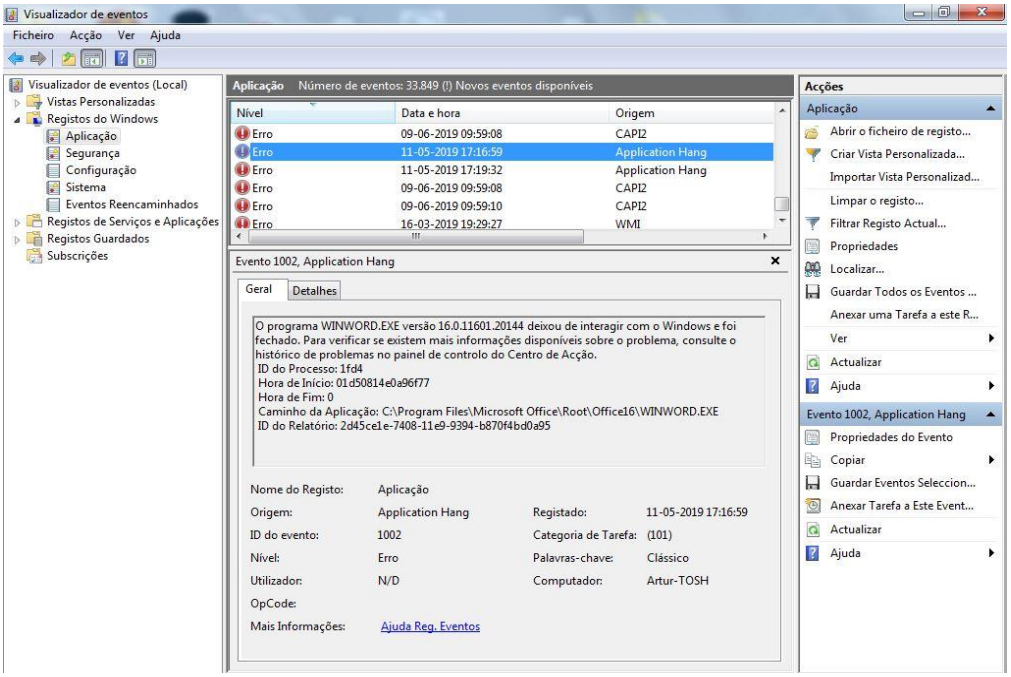


Figura 3.2 – Visualizador de eventos do Windows.[2]

Do lado esquerdo do visualizador, podem ser vistos os vários tipos de registo. Na parte central, em cima, podem ser vistas as mensagens de registo para o tipo selecionado (no caso da Figura 3.2, são registos do tipo Aplicação). Por baixo, ainda na parte central, podem ser visualizados os detalhes da mensagem selecionada na parte de cima.

[2] Fonte: Captura de ecrã feita pelo autor.

3.5. Geração de *Logs* no Linux

Nos sistemas Unix, os registos podem ser encontrados em ficheiros localizados na pasta */var/log*. Normalmente, é usado o protocolo *Syslog* para fazer o registo de mensagens do *kernel* e de várias aplicações. Originalmente, o *Syslog* foi desenvolvido para recolher informação de depuração. Como resultado, apresenta algumas limitações que não são ótimas para efeitos de análise de registos de segurança. Ainda assim, o *Syslog* tornou-se o método mais comum para registar eventos de sistemas baseados em Unix [8].

O *Syslog* consiste num *daemon* (*syslogd*), isto é, um processo que corre em plano de fundo (*background*), normalmente iniciado no arranque e finalizado no encerramento do sistema. As aplicações comunicam com o *syslogd* através de chamadas da biblioteca do *Syslog*. O *syslogd* recebe as mensagens de registo das aplicações e do *kernel* através da *socket* de domínio U que basicamente é uma interface para troca de dados entre processos do sistema operativo.

O *syslogd* também pode, opcionalmente, receber dados de *hosts* remotos, através de mensagens UDP, normalmente no porto 514. Versões mais modernas e substitutos do *Syslog*, como o *rsyslog* e o *syslog-ng*, também permitem utilizar o protocolo TCP. A Figura 3.3 mostra o ficheiro de configuração do *rsyslog* do Ubuntu, onde é possível habilitar a utilização dos protocolos UDP e TCP para receber mensagens de *hosts* remotos.

```
9  #################
10 #### MODULES ####
11 #################
12
13 module(load="imuxsock") # provides support for local system logging
14 module(load="imklog")   # provides kernel logging support
15 #module(load="immark")  # provides --MARK-- message capability
16
17 # provides UDP syslog reception
18 #module(load="imudp")
19 #input(type="imudp" port="514")
20
21 # provides TCP syslog reception
22 #module(load="imtcp")
23 #input(type="imtcp" port="514")
24
25 # Enable non-kernel facility klog messages
26 $KLogPermitNonKernelFacility on
```

Figura 3.3 – Ficheiro de configuração do *rsyslog*.

A configuração do *syslogd* pode ser feita através da edição do ficheiro *syslog.conf*, normalmente localizado na pasta */etc*. No *rsyslog*, a configuração é feita no ficheiro */etc/rsyslog.conf*.

O *daemon* lê as mensagens de registo através da *socket* de domínio Unix (o nome pode variar de acordo com a versão do Unix/Linux) e escreve em um ou em mais ficheiros de saída, ou reencaminha a mensagem via UDP para um *host* coletor.

No ficheiro de configuração, apresentado na Figura 3.4 é possível ver as origens e os destinos das mensagens de registo. Por exemplo, as mensagens originadas pelos recursos *auth* e *authpriv* são registadas no ficheiro *auth.log*. As mensagens do *kernel* e do *mail* são registadas nos ficheiros *kernel.log* e *mail.log*, respetivamente.

A linha 17 indica que as mensagens de todos os recursos são enviadas por UDP para o porto 514 do *host* local (se fosse por TCP teriam de ser usados dois @@ em vez de um único @).

Qualquer alteração no ficheiro de configuração requer a reinicialização do *daemon*, para que seja executada a nova configuração.

```
1  #  Default rules for rsyslog.
2  #
3  #  For more information see rsyslog.conf(5) and /etc/rsyslog.conf
4
5  #
6  # First some standard log files.  Log by facility.
7  #
8  auth,authpriv.*
9  *.*;auth,authpriv.none
10 #cron.*
11 #daemon.*
12 kern.*
13 #lpr.*
14 mail.*
15 #user.*
16
17 *.* @127.0.0.1:514
18
19 #
20 # Logging for the mail system.  Split it up so that
21 # it is easy to write scripts to parse these files.
22 #
23 #mail.info
24 #mail.warn
25 mail.err
```

Figura 3.4 – Ficheiro de configuração do *rsyslog*.

3.5.1. Classificação das Mensagens no *Syslog*

As mensagens do *Syslog* contêm dois atributos usados pelo *syslogd*, para este saber como as encaminhar: *Recurso* e *Prioridade*.

O *Recurso* serve para fornecer uma classificação geral sobre a origem da mensagem e apenas pode ser um dos que constam numa lista pré-definida, não sendo possível usar qualquer outro. O conjunto de recursos que podem ser usados é o apresentado na Tabela 3.1:

Tabela 3.1 - Tabela de recursos do *Syslog*

Código do Recurso	Nome do Recurso	Descrição
0	kern	Mensagens do *kernel*
1	user	Mensagens ao nível do utilizador
2	mail	Sistema de mail
3	daemon	*Daemons* do sistema
4	auth	Mensagens de segurança e autenticação
5	syslog	Mensagens geradas internamente pelo *syslogd*
6	lpr	Subsistema de impressora de linha
7	news	Subsistema da rede de notícias
8	uucp	Subsistema UUCP (*Unix to Unix Copy Protocol*)
9	cron	*Daemon* do relógio
10	authpriv	Mensagens de segurança e autenticação
11	ftp	*Daemon* FTP (*File Transfer Protocol*)
12	ntp	Subsistema NTP (*Network Time Protocol*)
13	security	Auditoria de registos
14	console	Alerta de registos
15	solaris-cron	*Daemon* de agendamento
16-23	local 0 – local 7	Recursos usados localmente

Os nomes destes recursos variam ligeiramente entre as distribuições Unix/Linux e alguns sistemas operativos têm algumas categorias que outros não têm.

O programador que desenvolve a aplicação escolhe qual o recurso a utilizar, sem qualquer tipo de restrição. Pode escolher usar os recursos *mail* ou *kern*, mesmo que não seja usada qualquer aplicação de *mail* ou de *kernel*.

3.5.2. Prioridade de mensagens no *Syslog*

A *Prioridade* serve para indicar a importância da mensagem. O conjunto de prioridades disponíveis é indicado na tabela seguinte, por ordem decrescente de gravidade:

Tabela 3.2 – Prioridade das mensagens do *Syslog* por ordem decrescente de gravidade

Valor	Gravidade	Nome	Descrição
0	Emergency	emerg	O sistema está inutilizável
1	Alert	alert	Deve ser tomada uma medida imediatamente
2	Critical	crit	Condições críticas
3	Error	err	Condições de erro
4	Warning	warning	Condições de aviso
5	Notice	notice	Condições normais mas significaticas
6	Informational	info	Mensagens informativas
7	Debug	debug	Mensagems de depuração

A prioridade de um determinado evento é determinada pelo programador da aplicação, pelo que nada garante que a gravidade do evento seja de facto significativa.

3.5.3. Conteúdo das Mensagens *Syslog*

Uma entrada do *Syslog* é constituída por três campos: a data/hora (*timestamp*), o nome do *host* que gerou a entrada e respetiva mensagem:

Aug 6 18:27:36	varanda-VirtualBox	kernel: [0.004000] console [tty0] enabled
timestamp	*host*	*mensagem (MSG)*

A mensagem (MSG), por sua vez, é composta por dois campos: TAG que é o nome do programa ou do processo que gerou a mensagem (*kernel*, no exemplo acima) e CONTENT que contém os detalhes da mensagem.

3.6. Geração de Registos por Servidores *Web*

Os servidores *Web* geram e mantêm, automaticamente, os respetivos ficheiros de registo. Todos os pedidos ao servidor, incluindo visualizações de documentos HTML, imagens ou outros objetos, são registados pelo servidor. O formato do registo é, essencialmente, uma linha de texto por cada solicitação feita. Estes registos contêm informação sobre quem visitou as páginas, qual a sua origem e toda a atividade realizada no servidor.

O *World Wide Web Consortium* (W3C) criou um formato padrão para os registos de servidores *Web*, embora existam outros formatos proprietários.

3.6.1. Registos do Microsoft IIS

Os servidores *Web* da Microsoft, IIS nas versões 4.0, 5.0, 6.0 e 7.0, utilizam o formato de registo W3C estendido [9].

A Figura 3.5 mostra um exemplo de um ficheiro de registo feito no formato W3C estendido, gerado por um servidor IIS da Microsoft. Em cima, é mostrada a localização dos ficheiros e em baixo, é apresentado um exemplo de um ficheiro de registo.

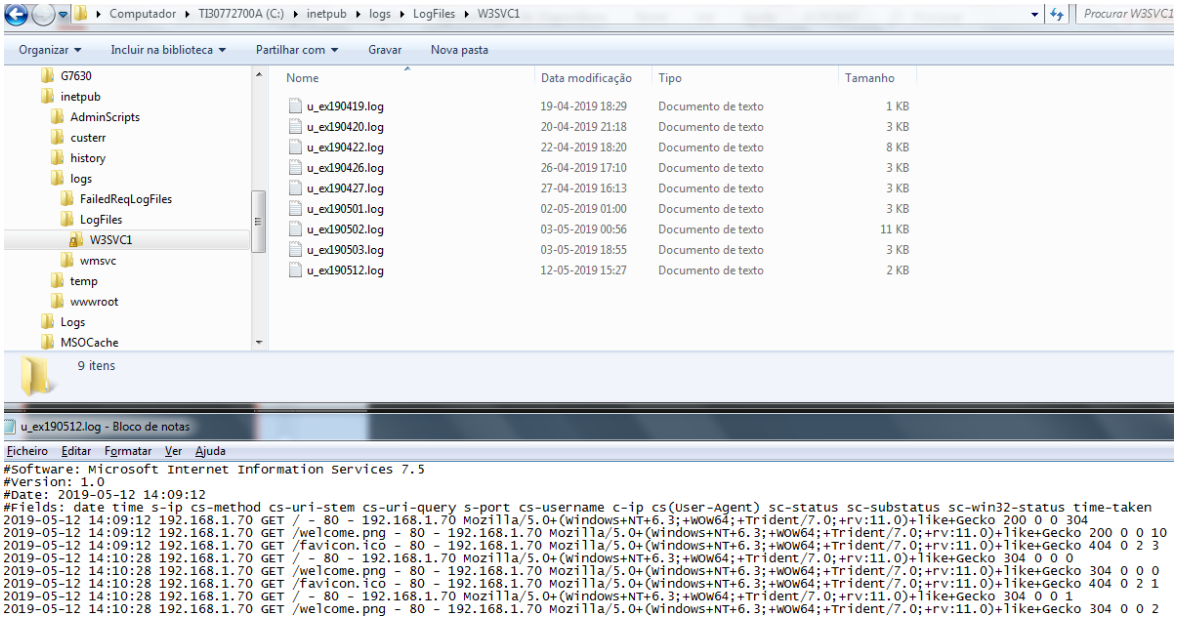


Figura 3.5 – Ficheiros de registo do IIS da Microsoft.[3]

[3] Fonte: Captura de ecrã feita pelo autor.

Estes ficheiros contêm sequências de caracteres ASCII. Cada linha pode conter uma *diretiva* ou uma *entrada*. No exemplo acima, as primeiras quatro linhas são *diretivas* e a restantes linhas são *entradas*.

As *entradas* são sequências de campos separados por espaços em branco, que dão informação sobre comunicações feitas através do protocolo HTTP. Os campos em que a informação está omissa são substituídos por travessões "-".

As *diretivas* são linhas que fornecem informações sobre o próprio processo de registo e começam com um cardinal "#".

O formato W3C estendido define as seguintes *diretivas*:

Tabela 3.3 – Diretivas definidas pelo formato W3C Estendido

Nome	**Estrutura de Dados**	**Descrição**
Software	String	Identifica o *software* que originou o registo.
Version	<integer>,<integer>	Versão do formato de registo estendido usada.
Date	<date> <time>	Data e hora em que a entrada foi registada.
Fields	[<specifier> …]	Lista a sequência de campos usados nas entradas.

As diretivas *Version* e *Fields* devem estar sempre presentes e devem preceder todas as entradas do registo. A diretiva *Fields* contém uma sequência de identificadores de campos que especificam os dados registados nos campos de cada entrada. Os indentificadores dos campos podem conter prefixos definidos da seguinte forma:

Tabela 3.4 – Prefixos usados pelo formato W3C Estendido

Prefixo	**Descrição**
s-	Ações do servidor
c-	Ações do cliente
cs-	Ações do cliente para o servidor
sc-	Ações do servidor para o cliente

Os seguintes identificadores não necessitam de prefixo:

Tabela 3.5 – Identificadores do W3C Estendido que não requerem prefixo

Nome	Descrição
date	Data do evento
time	Hora do evento
time-taken	Duração do evento em milissegundos

Os identificadores que requerem um prefixo são os seguintes:

Tabela 3.6 – Identificadores do W3C Estendido que requerem prefixo

Campo	Descrição
c-ip	Endereço IP do cliente que acedeu ao servidor
cs-username	Nome de utilizador autenticado que acedeu ao servidor
s-ip	Endereço IP do servidor que gerou o registo
s-port	Número do porto ao qual o cliente estava ligado
cs-method	A ação que o cliente tentou realizar (por exemplo, método GET).
cs-usi-stem	Recurso ao qual o cliente tentou aceder (por exemplo, pagina.html)
cs-uri-query	Pesquisa efetuada pelo cliente ao servidor
sc-status	Estado da ação de acordo com os protocolos HTTP ou FTP
sc-win32-status	Estado da ação de acordo com Microsoft Windows
cs(User-Agent)	Navegador da *Web* (*Browser*) usado pelo cliente

3.6.2. Registos do Servidor Apache

O servidor *Web* da Apache regista os pedidos efetuados pelos clientes, as respostas dadas e as respetivas operações internas. Existem dois grandes tipos de registo que são gerados: *acesso* e *erro*.

Os registos de *acesso* contêm informação sobre pedidos feitos ao servidor, incluindo as páginas visitadas pelos clientes, os estados dos pedidos e os tempos de resposta. Um exemplo de uma entrada de registo de acesso é:

192.168.1.102 - - [29/Apr/2019:23:19:21 +0100] "GET /teste?nome:artur HTTP/1.1" 404 203

Os registos de *erro* contêm informação relativa aos erros encontrados pelo servidor ao processar os pedidos dos clientes, como ficheiros não existentes ou informação de diagnóstico sobre o próprio servidor. Um exemplo de um registo de *erro* é:

[Fri May 03 01:44:19.619446 2019] [mpm_winnt:notice] [pid 9168:tid 308] AH00430: Parent: Child process 8856 exited successfully.

A Figura 3.6 mostra a localização dos registos do servidor Apache instalado no Windows 7, onde se podem ver os ficheiros de registo de três tipos: acesso, erro e instalação. Na parte de baixo, é apresentado um exemplo com o conteúdo do ficheiro com os registos de acesso.

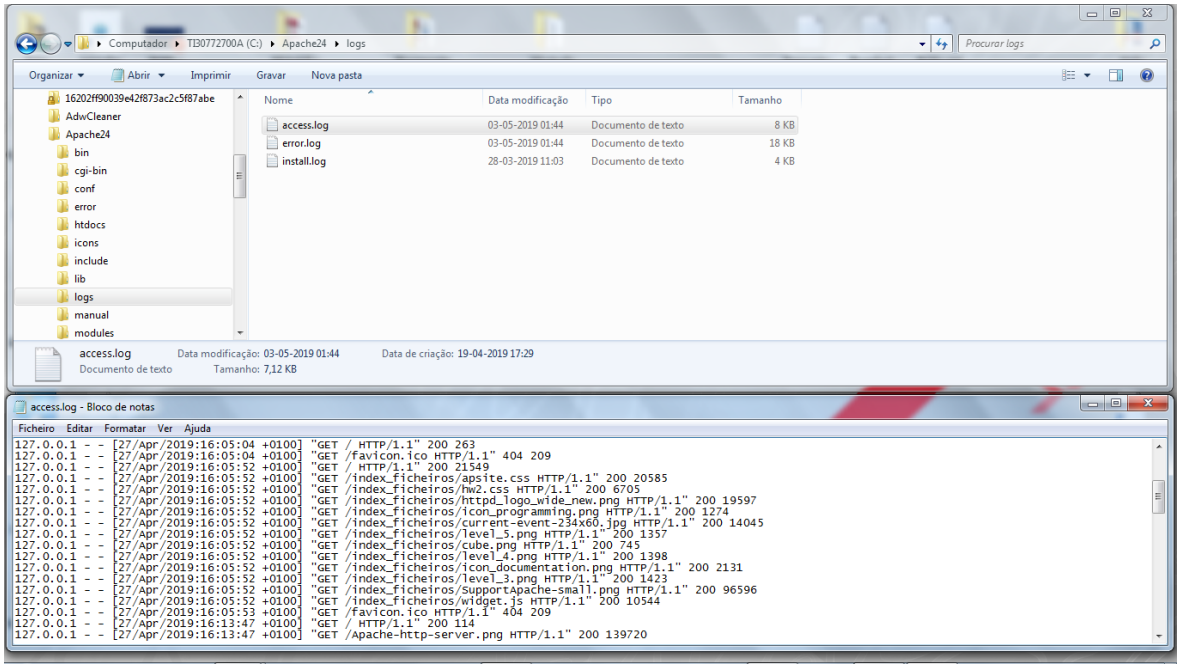


Figura 3.6 – Registos de um servidor *Web* da Apache.[4]

3.7. Geração de registos por servidores SSH

O serviço SSH é normalmente usado para estabelecer uma ligação segura entre uma estação de trabalho e um servidor remoto. Normalmente, quando é feita uma ligação a uma máquina remota via SSH, é pedida uma palavra-passe. No entanto, se é necessário fazer a ligação com alguma frequência, perde-se bastante tempo, principalmente se a palavra-passe for segura e extensa. O servidor SSH permite-nos, em alternativa, usar autenticação baseada em chaves. Para isso, é necessário criar chaves públicas e privadas e colocar uma cópia da chave pública no servidor remoto ao qual se pretende ligar. Quando se tenta ligar, o servidor remoto verifica que a nossa chave privada corresponde à chave publica lá armazenada e autoriza a autenticação.

[4] Fonte: Captura de ecrã feita pelo autor.

Por norma, no Linux, o servidor SSH envia a informação de registo para o *Syslog* com o nível de prioridade *info* e classifica-a como pertencente ao recurso *auth*. A Figura 3.7 mostra as entradas de registo de duas sessões no servidor SSH, adicionadas ao ficheiro */var/log/auth.log*:

```
16:38:58 varanda-VirtualBox sshd[874]: Server listening on 0.0.0.0 port 22.
16:38:58 varanda-VirtualBox sshd[874]: Server listening on :: port 22.
16:38:59 varanda-VirtualBox sshd[874]: Received SIGHUP; restarting.
12:11:57 varanda-VirtualBox sshd[954]: Server listening on 0.0.0.0 port 22.
12:11:57 varanda-VirtualBox sshd[954]: Server listening on :: port 22.
12:12:45 varanda-VirtualBox sshd[17291]: Accepted password for varanda from 192.168.43.72 port 1256 ssh2
12:12:45 varanda-VirtualBox sshd[17291]: pam_unix(sshd:session): session opened for user varanda by (uid=0)
12:13:29 varanda-VirtualBox sshd[17354]: Received disconnect from 192.168.43.72 port 1256:11: disconnected by user
12:13:29 varanda-VirtualBox sshd[17354]: Disconnected from 192.168.43.72 port 1256
12:13:29 varanda-VirtualBox sshd[17291]: pam_unix(sshd:session): session closed for user varanda
15:10:38 varanda-VirtualBox sshd[999]: Server listening on 0.0.0.0 port 22.
15:10:38 varanda-VirtualBox sshd[999]: Server listening on :: port 22.
15:20:33 varanda-VirtualBox sshd[999]: Received SIGHUP; restarting.
15:20:43 varanda-VirtualBox sshd[999]: Server listening on 0.0.0.0 port 22.
15:20:43 varanda-VirtualBox sshd[999]: Server listening on :: port 22.
15:24:22 varanda-VirtualBox sshd[3446]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.43.72  user=varanda
15:24:24 varanda-VirtualBox sshd[3446]: Failed password for varanda from 192.168.43.72 port 1476 ssh2
15:24:49 varanda-VirtualBox sshd[3497]: Received disconnect from 192.168.43.72 port 1476:11: disconnected by user
15:24:49 varanda-VirtualBox sshd[3497]: Disconnected from 192.168.43.72 port 1476
15:24:49 varanda-VirtualBox sshd[3446]: pam_unix(sshd:session): session closed for user varanda
```

Figura 3.7 – Registos do servidor SSH no Linux.

No Windows, os registos do servidor SSH são armazenados no ficheiro *OpenSSH_Admin.evtx* e podem ser visualizados através do visualizador de eventos do Windows (Figura 3.8):

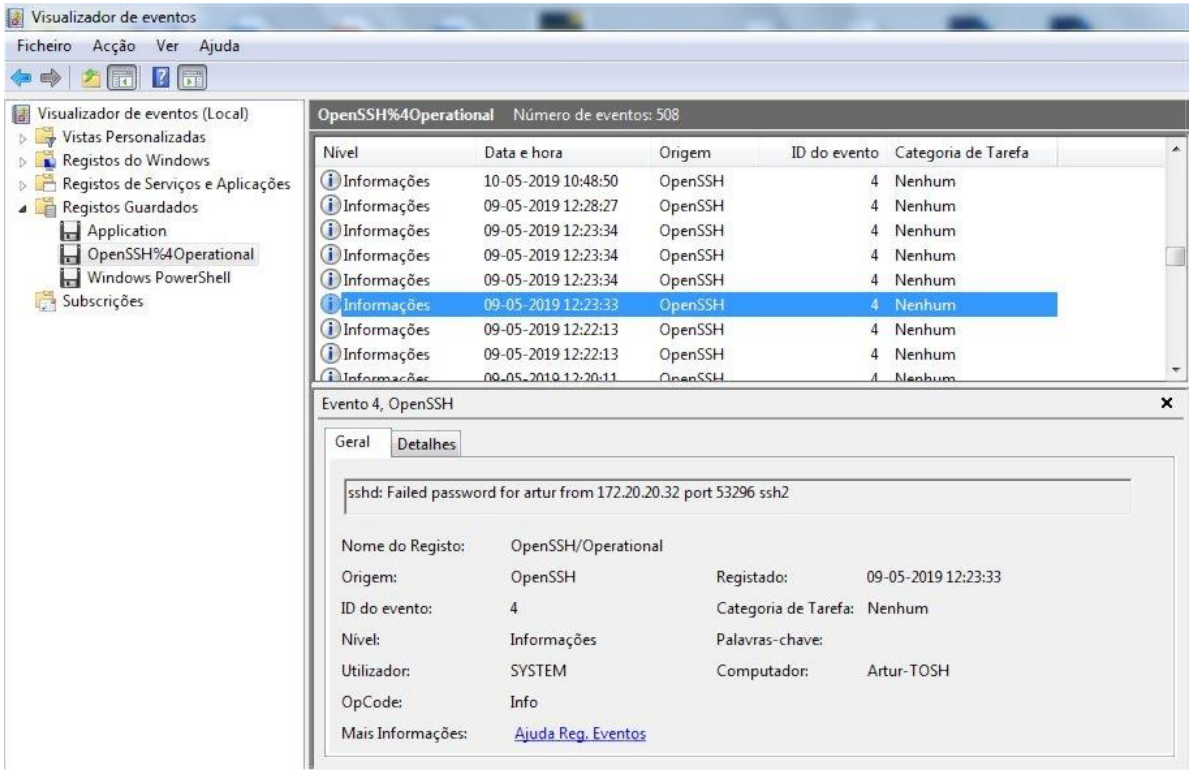


Figura 3.8 – Registos do servidor SSH no *Windows Event Log*.[5]

As entradas típicas de inicialização do servidor são do tipo:

sshd: Server listening on 0.0.0.0 port 22

[5] Fonte: Captura de ecrã feita pelo autor.

As tentativas falhadas de autenticação são registadas como:

sshd: Failed password for artur from 172.20.20.32 port 53296 ssh2

E as tentativas bem-sucedidas são apresentadas da forma seguinte:

sshd: Accepted password for artur from 192.168.1.102 port 41842 ssh2

3.8. Geração de registos por *Firewalls*

3.8.1. *Firewall* do Windows

Normalmente, o ficheiro com o registo da *firewall* do Windows está localizado na pasta *C:\Windows\system32\logfiles\firewall\pfirewall.log* e pode ser visualizado com um editor de texto. O principal objetivo de registar a atividade da *firewall* é o de permitir verificar se as regras acrescentadas estão a funcionar convenientemente, ou para as corrigir, caso não estejam a funcionar como é esperado.

Ao analisar o comportamento da *firewall* do Windows, é importante verificar os seguintes atributos:

- Endereço IP local do sistema;
- Endereço IP público do sistema, quando está ligado a uma VPN;
- Protocolos utilizados, por exemplo, UDP ou TCP;
- Estado de um dado pacote, por exemplo, ALLOW ou DROP.

O ficheiro de registo da *firewall* do Windows tem a seguinte apresentação:

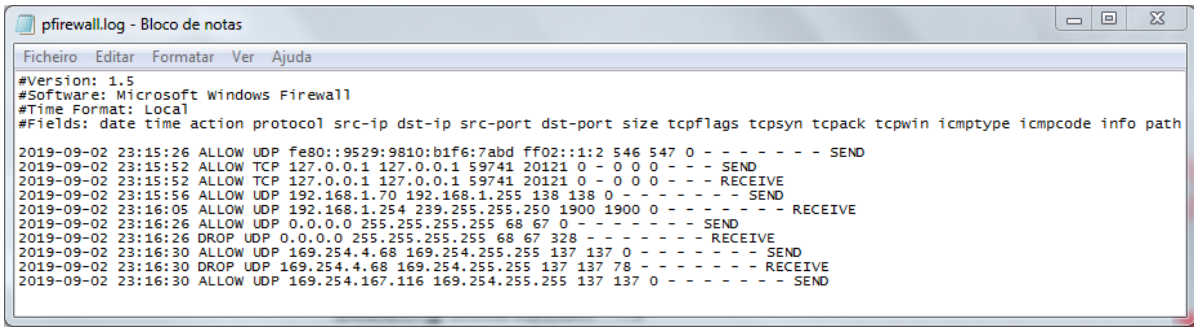


Figura 3.9 – Ficheiro de registo da Firewall do Windows.[6]

[6] Fonte: Captura de ecrã feita pelo autor.

Para cada entrada, podem ser vistos a data/hora, o tráfego TCP ou UDP que é autorizado ou rejeitado, os endereços IP de origem e de destino e os respetivos portos.

De acordo com a documentação técnica da Microsoft, o cabeçalho do ficheiro de registo da *firewall* do Windows contém as seguintes diretivas:

Tabela 3.7 – Diretivas do cabeçalho do ficheiro de registo da Firewall do Windows

Nome	**Descrição**
Version	Versão da Firewall do Windows
Software	*Software* que gerou o ficheiro de registo
Time	Data/hora do registo na hora local
Fields	Lista de campos disponíveis para as entradas de registo

As entradas de registo contêm os seguintes campos:

Tabela 3.8 – Campos das entradas de registo da Firewall do Windows

Nome	**Descrição**
date	Data no formato YYYY-MM-DD
time	Hora no formato HH:MM:SS
action	Ações da firewall (ALLOW – ligação permitida; DROP – ligação rejeitada)
protocol	Protocolo usado (TCP, UDP ou ICMP)
src-ip	Endereço IP da máquina que está a tentar estabelecer a ligação
dst-ip	Endereço IP da máquina à qual se esta a tentar estabelecer a ligação
src-port	Porto usado pela máquina que está a tentar estabelecer a ligação
dst-port	Porto da máquina à qual se está a tentar estabelecer a ligação
size	Volume da trama, em bytes
tcpflags	Informação das *flags* de controlo do cabeçalho da trama TCP
tcpsyn	Número de sequência da trama TCP
tcpakn	Código de aceitação da trama TCP
tcpwin	Volume da janela da trama TCP, em bytes
icmptype	Informação sobre mensagens ICMP
icmpcode	Informação sobre mensagens ICMP
info	Valor que depende do tipo de ação que foi tomada.
path	Direção da comunicação (SEND – envio; RECEIVE – receção)

3.8.2. *Firewall* do Linux

Por norma, a ferramenta de configuração da *firewall* do Linux é a UFW (*Uncomplicated Firewall*). Esta ferramenta permite configurar, de uma forma muito simples, uma *firewall* IPv4 ou IPv6 instalada localmente no *host*. Os registos da *firewall* são essenciais para reconhecer ataques, para resolver de problemas que possam surgir com as regras da *firewall* e para detetar atividade suspeita na rede. Para que sejam geradas entradas de registo, é necessário que sejam previamente criadas regras através da ferramenta de configuração.

Podem ser vistos, abaixo, dois exemplos de registos gerados pela UFW. O primeiro corresponde a um bloqueio de uma ligação de entrada UDP/IPv6 e o segundo corresponde a uma autorização de uma ligação de entrada TCP/IPv4:

Aug 6 18:28:03 varanda-VirtualBox kernel: [33.192625] [UFW BLOCK] IN=enp0s3 OUT= MAC= SRC=fe80:0000:0000:0000:e33f:fc18:b68e:280a DST=ff02:0000:0000:0000:0000:0000:0000:0001 LEN=64 TC=0 HOPLIMIT=1 FLOWLBL=611903 PROTO=UDP SPT=8612 DPT=8610 LEN=24

Aug 7 12:2:05 varanda-VirtualBox kernel: [4304885.870000] NEW_HTTP_CONN: IN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1 DST=127.0.0.1 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=58288 DF PROTO=TCP SPT=53981 DPT=80 WINDOW=32767 RES=0x00 SYN URGP=0

Os campos das entradas de registo estão descritos na tabela seguinte:

Tabela 3.9 – Campos das entradas de registo da UFW

Nome	**Descrição**
Timestamp	Data e hora do registo
Hostname	Nome do *host*
Uptime	Tempo decorrido desde a inicialização do sistema, em segundos
Logged Event	Pequena descrição do evento, por exemplo, UFW BLOCK
IN	Caso contenha informação, indica que o evento foi uma receção
OUT	Caso contenha informação, indica que o evento foi um envio
MAC	Cabeçalho MAC da trama de Ethernet Type II
SRC	Endereço IP da máquina que está a tentar estabelecer a ligação
DST	Endereço IP da máquina à qual se esta a tentar estabelecer a ligação
LEN	Tamanho da trama, em bytes
TOS	*Type of Service* – Tipo de serviço do IPv4

TC	*Traffic Class* - Equivalente ao TOS do IPv4, mas usado pelo Ipv6
PREC	*Precedence* – Medida de importância da trama IPv4
TTL	*Time to Live* - Número de saltos entre máquinas que os pacotes IPv4 podem dar na rede antes de serem descartados
HOPLIMIT	Equivalente ao TTL do IPv4, mas usado pelo IPv6
ID	*Identification* - Um valor de identificação atribuído pelo remetente para ajudar na montagem dos fragmentos de um pacote IPv4.
FLOWLBL	*Flow Label* - Equivalente ao ID do IPv4, mas usado pelo IPv6
PROTO	Protocolo usado (TCP ou UDP)
SPT	Porto usado pela máquina que está a tentar estabelecer a ligação
DPT	Porto da máquina à qual se está a tentar estabelecer a ligação
WINDOW	Volume da janela da trama TCP, em bytes
RES	Campo reservado para futura utilização
SYN URGP	SYN indica que essa ligação requer um *handshake* triplo, típico em TCP. URGP indica que o valor é relevante. Se o valor for 0, significa que não é urgente.

3.9. Gestão Centralizada de Registos

Normalmente, os *logs* são armazenados no *host* onde são gerados. Caso haja um comprometimento do sistema local, estes ficheiros ficam expostos e poderão ser corrompidos. A solução para este problema consiste em recolher e agregar as entradas de registo dos *logs* de uma forma centralizada, o que apresenta os seguintes benefícios :

- Agregação – Cada serviço envia as suas entradas de registo para o servidor, onde são agregadas e disponibilizadas para análise;
- Pesquisa – Através da indexação, é possível pesquisar com eficiência dados estruturados inerentes às entradas;
- Acesso – Facilidade em separar os dados pessoais dos dados técnicos;
- Segurança – Os registos não ficam armazenados nos *hosts* que os geraram. No caso destes terem um comprometimento ou falha de *hardware*, os registos estão protegidos no sistema central;
- Monitorização e Alerta – Visualização das entradas de registo num painel e possibilidade de criar regras para gerar alarmes;
- Análise – Os registos agregados num sistema central fornecem um contexto muito mais rico para a análise do que os que são armazenados individualmente.

Até alguns anos atrás, a agregação de registos centralizada consistia simplesmente num *host* Unix com grande capacidade de armazenamento, a correr o serviço *syslogd* e a recolher os *logs* do ambiente para a sua própria pasta */var/log/* [10].

Atualmente, existem soluções SIEM (*Security Information & Event Management*) que desempenham este papel, para além de disponibilizarem outras funcionalidades. Estas soluções permitem também correlacionar *logs* com outras informações de segurança para beneficiar a análise. A *ELK Stack*, o *AlienVault*, o *OSSIM* e o *Splunk*, são exemplos deste tipo de plataformas [11].

A agregação de *logs* é útil para outros fins para além da gestão centralizada. Os *logs* que abrangem vários *hosts* fornecem um contexto muito mais rico que permite detetar determinados eventos de segurança [12].

3.10. Arquitetura de uma Infraestrutura de Gestão de *Logs*

Uma infraestrutura de gestão de registos consiste em *hardware*, *software*, redes e dispositivos usados para gerar, transmitir, armazenar, analisar e descartar dados. Normalmente, a infraestrutura está dividida em três níveis:

- Geração - O primeiro nível contém os *hosts* que geram os *logs*. Alguns *hosts* registam os *logs* de aplicações ou serviços clientes e disponibilizam-nos, através da rede, a servidores de *logs* na segunda camada (*push*). Outros *hosts* permitem que os servidores se autentiquem e recolham cópias dos registos (*pull*);
- Análise e Armazenamento – O segundo nível consiste num ou mais servidores de *logs* que recebem ou copiam os registos dos *hosts* do primeiro nível. Os dados tanto podem ser transferidos entre os servidores em tempo real, como podem ser programados para serem transferidos periodicamente, ou sempre que atingirem um determinado volume. Os servidores que recebem *logs* de vários geradores são normalmente denominados de *coletores* ou *agregadores*. Os *logs* podem ser armazenados nos servidores ou arquivados num sistema à parte;
- Apresentação – O terceiro nível consiste em consolas que são usadas para visualizar ou analisar os registos. Estas consolas também podem ser usadas para gerar relatórios, ou para gerir os servidores e os clientes.

A figura Figura 3.10 mostra um exemplo de uma configuração típica de uma infraestrutura de gestão de *logs*. Do lado esquerdo, são apresentados dois clientes que geram ficheiros de registo provenientes de várias fontes que são recolhidos pelos respetivos agentes. Os agentes encaminham as mensagens via TCP ou UDP para um servidor. Este servidor contém uma plataforma de gestão de *logs* que recolhe os registos dos *host*s clientes e do seu próprio sistema operativo. A plataforma disponibiliza uma consola de visualização que normalmente é uma interface *Web*.

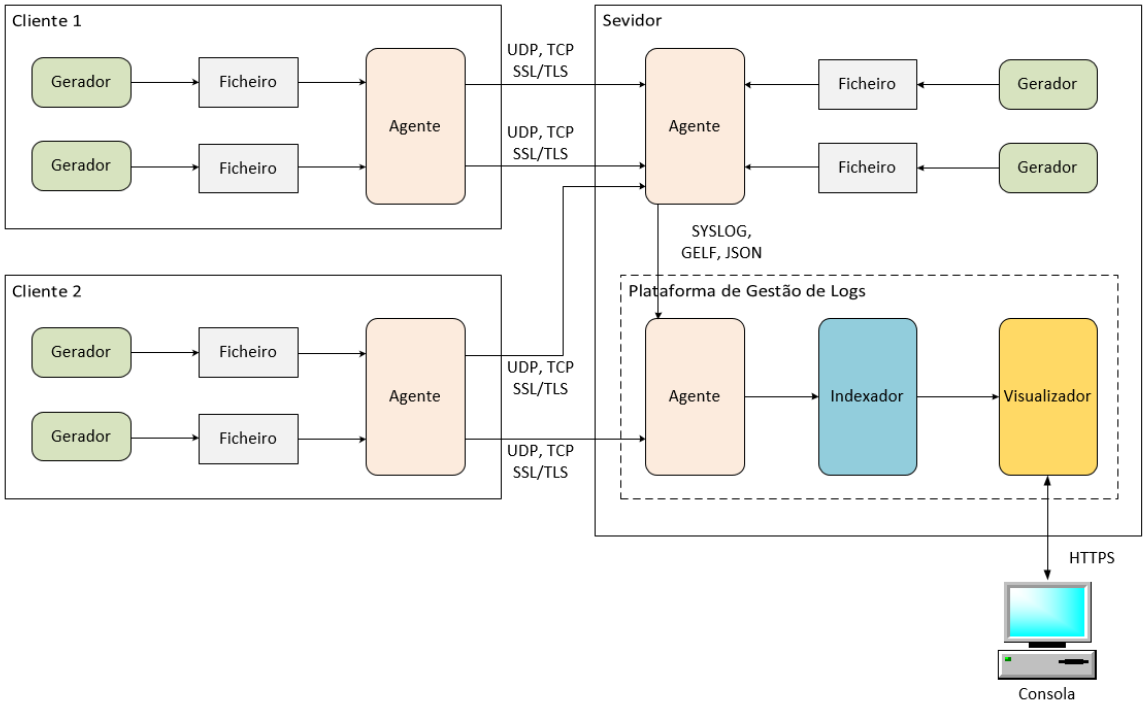


Figura 3.10 – Configuração típica de uma infraestrutura de gestão de *logs*.

3.11. Síntese

Neste capítulo foram apresentadas algumas recomendações para a recolha de dados pessoais em ficheiros de registo.

Foram também apresentadas a principais fontes geradoras de logs: software de segurança, sistema operativo e aplicações. Em particular, foram estudados e analisados os registos de eventos provenientes dos sistemas operativos Microsoft Windows e Linux, registos de acesso de servidores *Web* , registos de *firewalls* de ambos os sistemas operativos e registos de servidores SSH.

Finalmente, foram discutidas as vantagens da implementação de uma gestão centralizada de registos e apresentada a arquitetura típica de uma infraestrutura deste tipo.

No próximo capítulo vão ser abordadas algumas técnicas e metodologias para a pseudonimização da informação contida nos registos dos sistemas e aplicações.

4. Anonimização e Pseudonimização de Dados

O novo Regulamento Geral de Proteção de Dados (RGPD) impõe que sejam utilizados vários níveis de proteção para garantir que os dados pessoais cumpram os requisitos de segurança da informação. Uma das técnicas recomendadas pelo regulamento para proteger dados pessoais é a pseudonimização, que consiste em substituir, num documento, nomes reais por nomes fictícios (pseudónimos).

Neste capítulo são apresentadas algumas técnicas para a pseudonimização da informação contida nos registos dos sistemas e aplicações. São também discutidas quatro metodologias diferentes para efetuar o processo de pseudonimização: na fase de geração de registos, na fase de ingestão, através de duplicação de índices e na fase de apresentação. No final, é feito um estudo comparativo entre as diferentes metodologias.

Os registos de auditoria são uma parte importante de qualquer sistema e precisam ser cuidadosamente preparados para fornecer uma representação fiel das atividades ocorridas no sistema.

Embora seja importante que os auditores possam inspecionar *logs* de auditoria, o conteúdo de um registo pode conter informações confidenciais ou dados pessoais e, portanto, deve ser protegido contra acessos não autorizados. A proteção do conteúdo dos *logs* de auditoria através de criptografia, como forma eficiente de pesquisa por auditores autorizados, representa um problema complexo [13].

Os *logs* são um dos meios para a deteção de eventos mais poderosos do sistema. No entanto, requerem uma quantidade substancial de tempo e de recursos para poderem ser usados eficientemente. A dimensão dos sistemas distribuídos aumenta constantemente e o volume de *logs* gerados pelos sistemas é proporcional a esse aumento. Assim, a recolha e o armazenamento destes *logs* representa, a longo prazo, um grande desafio [14].

A análise dos *logs* do sistema requer técnicas avançadas de processamento de texto. Têm sido utilizadas várias técnicas de anonimização para proteger a privacidade dos seus utilizadores e dos seus dados. A utilidade dos *logs* com os dados anonimizados pode diminuir de forma significativa, dificultando os processos de análise pretendidos.

Devido à presença de um grande número de identificadores pessoais nos *logs* do sistema, a privacidade dos dados é um pré-requisito para cumprir com a regulamentação. Algumas abordagens estabelecem um equilíbrio entre privacidade e a utilidade dos dados de forma a cumprir o objetivo da análise. Estas abordagens consistem em métodos de codificação irreversível, através da utilização de funções de resumo resistentes à colisão (funções de *hash*), de modo a garantir a privacidade dos dados [15].

4.1.Pseudonimização e Anonimização

Entre as várias técnicas de segurança da informação disponíveis, a pseudonimização e a anonimização são recomendadas pelo RGPD. Estas técnicas reduzem o risco e auxiliam os responsáveis pelo tratamento de dados no cumprimento do Regulamento. Por isso, a sua utilização deve ser generalizada e recorrente.

A principal diferença entre os dados pseudonimizados e os dados anonimizados reside na possibilidade dos dados pseudonimizados poderem ser reidentificados com um esforço razoável em termos computacionais, enquanto que os dados anonimizados não podem ser reidentificados [16].

A anonimização de dados consiste em processá-los de forma irreversível, impossibilitando a sua reidentificação (Figura 4.1):

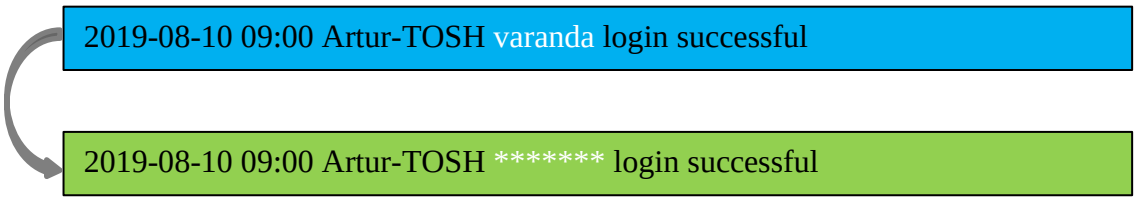


Figura 4.1 - Processo de Anonimização.

A pseudonimização de dados consiste em substituí-los por valores fictícios (pseudónimos), impossibilitando a reidentificação de forma direta (Figura 4.2):

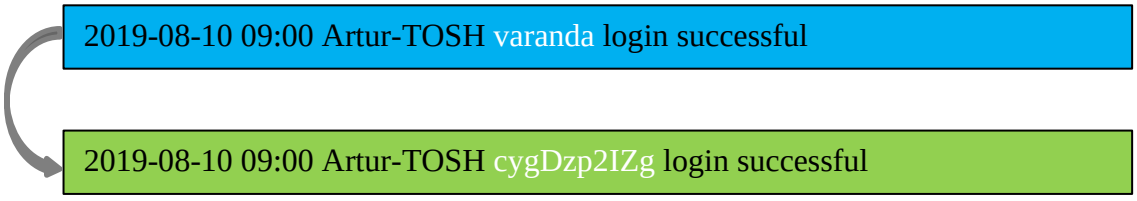


Figura 4.2 - Processo de Pseudonimização.

A pseudonimização pode facilitar o processamento de dados pessoais para além dos propósitos de recolha originais. O Regulamento permite que os responsáveis pelo tratamento pseudonimizem os dados pessoais a fim de os poderem processar para uma finalidade diferente daquela para a qual foram recolhidos (alínea e) do número 4 do artigo 6.º.

4.1.1. Cifragem

Os responsáveis pelo tratamento de dados são obrigados a implementar medidas baseadas no risco para proteger a segurança dos mesmos. Uma dessas medidas é a pseudonimização com cifragem de dados pessoais (artigo 32.º).

No contexto da pseudonimização, o processo de cifragem consiste em usar uma chave para transformar um nome (texto original) num pseudónimo (texto cifrado). A decifragem é o processo inverso: consiste em transformar o texto cifrado no texto original e corresponde ao processo de reidentificação.

Os processos de cifragem simétrica utilizam a mesma chave para cifragem e para a decifragem, enquanto que os processos de cifragem assimétrica usam um par de chaves, uma pública e outra privada. A chave pública é usada para a cifragem, enquanto a chave privada é usada apenas para a decifragem.

4.1.2. Funções de Resumo (funções de Hash)

Nos casos em que não está prevista a necessidade de reidentificação, o mascaramento dos valores identificativos pode ser feito através de uma função de resumo. Este método permite que seja possível verificar se dois pseudónimos correspondem a nomes idênticos sem efetuar qualquer reidentificação. Caso se verifique a correspondência, diz-se que os pseudónimos estão relacionados em termos de igualdade.

Uma função de resumo (função de *hash*) é uma função unidirecional, isto é, uma função matemática simples que é difícil de inverter. Isto significa que é simples calcular o valor de saída, mas a inversão é extremamente difícil, ou seja, é difícil obter o valor de entrada a partir do valor de saída. Os conceitos "simples" e "difícil" devem ser entendidos no sentido da teoria da complexidade computacional. Neste contexto, “difícil” significa "praticamente impossível dentro de um período razoável" [17].

A função de *hash* é uma função unidirecional resistente a colisões que atribui um valor de resumo de comprimento fixo (*hash*) a uma entrada de qualquer tamanho. A "resistência à colisão" é a propriedade que torna praticamente impossível que diferentes valores de entrada originem o mesmo *hash* de saída.

Algumas funções *hash* bem conhecidas são as funções MD5, SHA-1 e SHA-256, embora existam muitas mais. A segurança dos algoritmos MD5 e SHA-1 foi seriamente comprometida, tendo as respetivas falhas sido exploradas [18, p. 353-356.].

4.1.3. Cifragem determinística e cifragem probabilística

Os processos de cifragem são normalmente determinísticos, ou seja, o mesmo texto original é transformado no mesmo texto cifrado, se for usada a mesma chave. Por outro lado, os processos de criptografia que geram textos cifrados diferentes quando é usada a mesma chave, são chamados processos de cifragem probabilísticos.

As funções *hash* são desenvolvidas de modo a prevenir a possibilidade de se converter o resumo por elas criadas, no valor original. No entanto, apesar de ser praticamente impossível reverter o processo, a salvaguarda de dados não está completamente garantida.

Os *hackers* podem usar tabelas de consulta de *hashes* pré-calculados (*rainbow tables*) para encontrar o valor original a partir do resumo. Estas tabelas são dicionários com listas de milhares, milhões, ou até milhares de milhões de resumos, juntamente com os valores originais correspondentes.

Apesar deste processo não ser propriamente uma reversão do algoritmo de resumo é, no entanto, um processo muito simples de realizar. Na verdade, como não existem tabelas de consulta que contenham todas as correspondências entre valores originais e respetivos resumos, apenas são úteis para valores simples, como palavras-passe fracas.

A tabela seguinte mostra uma versão simplificada de uma tabela de consulta com as correspondências entre os valores originais e os respetivos resumos, calculados através da função de *hash* SHA-1:

Tabela 4.1 – Exemplo de tabela de consulta com *hashes* pré-calculados

Valor original	***Hash* (SHA-1)**
123456	7c4a8d09ca3762af61e59520943dc26494f8941b
password	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
127.0.0.1	4b84b15bff6ee5796152495a230e45e3d7e947d9
admin	d033e22ae348aeb5660fc2140aec35850c4da997
root	dc76e9f0c0006e8f919e0c515c66dbba3982f785
1234	7110eda4d09e062aa5e4a390b0a572ac0d2c0220
0000	39dfa55283318d31afe5a3ff4a0e3253e2045e43
artur_varanda	e0d80f16cf7c4be2a4ccd0bc9d033677c8477a1a

O *hacker* tem, como é óbvio, de saber qual é a função de *hash* usada para gerar os resumos, de modo a descobrir os valores originais.

Alguns dados identificativos existentes nos registos de eventos, como endereços IPv4 ou nomes de utilizador, que tenham sido pseudonimizados com funções de resumo, podem ser facilmente reidentificados através de tabelas de consulta.

Para mitigar os estragos provocados pelas tabelas de consulta e pelos ataques de dicionário, deve ser anexado um valor aleatório, denominado "sal", ao texto original, antes de ser calculada a função de resumo [17]. O "sal" é um valor aleatório fixo, robusto em termos criptográficos, acrescentado ao texto original e processado pela função *hash*, de modo a gerar resumos distintos, apesar dos textos originais serem idênticos. Assim, um processo de cifragem determinístico é transformado num processo de cifragem probabilístico, através da introdução do "sal" na equação. Isto quer dizer que dois valores originais idênticos, como palavras-passe duplicadas, por exemplo, terão valores de resumo distintos.

4.1.4. Função HMAC

Algumas funções, em particular as funções de resumo aplicadas a palavras-passe, necessitam da transformação adicional proporcionada pelo "sal" sem, no entanto, necessitar da confidencialidade dos "sais" utilizados, o que facilita o armazenamento dos mesmos.

Na prática, o "sal" usado para calcular o resumo das palavras-passe é armazenado no mesmo local onde é armazenado o respetivo resumo, por vezes numa coluna adjacente da mesma

tabela, para que possa ser recalculado e confrontado com o resumo do valor introduzido pelo utilizador durante o processo de autenticação.

A função HMAC incorpora um segredo adicional sob a forma de uma chave. Esta chave é combinada com o valor original de uma forma muito mais profunda do que o "sal", podendo ser usada, no entanto, juntamente com este último. O resumo resultante não é tão vulnerável como o de uma função de resumo simples com "sal". Além disso, o HMAC é um método muito mais robusto para produzir um resumo autenticado.

A verificação de palavras-passe pode ser feita de um modo mais sofisticado, usando uma função HMAC. Num sistema que verifique palavras-passe processadas com "sal" e uma função de resumo, esta última pode ser substituída por uma função HMAC, usando uma chave secreta K. Este método é seguro contra "ataques de dicionário", desde que a chave K seja secreta. Manter segura uma chave K de 128 bits não é uma tarefa fácil, mas sempre é mais fácil do que manter segura uma base de dados com "sais" [18, p. 393-394].

A função HMAC tem ainda outra vantagem em relação às funções de resumo simples com utilização de um "sal": é imune a "ataques de extensão de comprimento". Nestes ataques, são acrescentados dados a mensagens falsas de modo a terem o mesmo resumo que a mensagem original (existência de colisão). As funções MD5, SHA-1 e SHA-2, baseadas no algoritmo *Merkle–Damgård* são suscetíveis a este tipo de ataque [19].

A definição da função HMAC, de acordo com a RFC 2104, é a seguinte [20]:

$$\text{HMAC}(K, m) = \text{H}\left((K' \oplus opad) \parallel \text{H}\left((K' \oplus ipad) \parallel m\right)\right)$$

$$K' = \begin{cases} \text{H}(K) & \text{se o tamanho de } K \text{ for maior que o tamanho do bloco} \\ K & \text{no caso contrário} \end{cases}$$

onde:

H é a função de *hash*

m é a mensagem a ser autenticada

K é a chave secreta

K' é uma chave de tamanho fixo derivada da chave *K*; o tamanho fixo do bloco é garantido com preenchimento com zeros à direita se a chave for mais curta, ou resumindo a chave para um tamanho inferior e acrescentando zeros à direita

$\parallel$ denota concatenação

⊕ denota disjunção exclusiva (XOR)

opad é o bloco de preenchimento exterior com bytes sucessivos de valor 0x5C

ipad é o bloco de preenchimento interior com bytes sucessivos de valor 0x36

A Figura 4.3 ilustra a geração do resumo usando a função HMAC juntamente com a função de *hash* SHA-1.

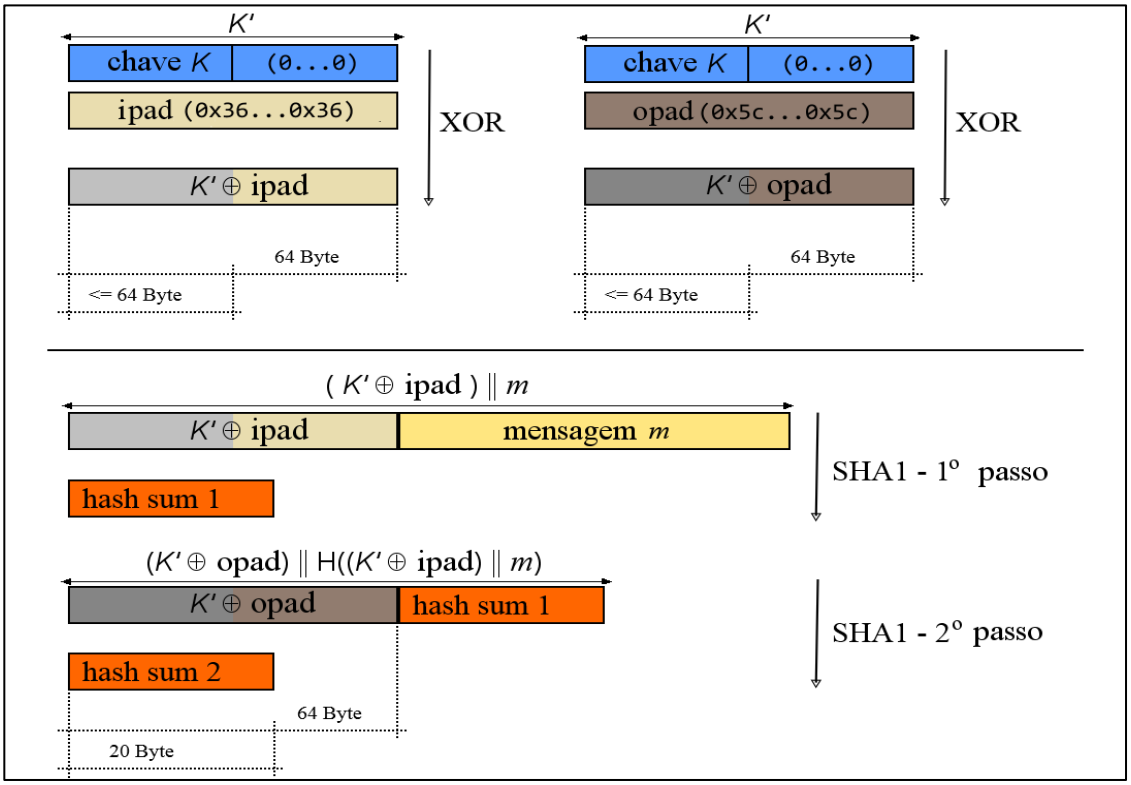


Figura 4.3 – Geração de resumo de mensagem usando HMAC SHA-1.

Depois da primeira iteração com a função de resumo SHA-1 obtém-se o valor *hash sum 1*. Este é o resultado da operação H ((*K'* ⊕ *ipad*) || *m*).

Na segunda iteração com a função de resumo SHA-1, obtém-se o valor *hash sum 2* que é o resultado da operação H ((*K'* ⊕ *opad*) || *hash sum 1*). Este é o resumo resultante da operação HMAC SHA-1 aplicada à mensagem *m*, usando a chave *K*.

4.2. Implementação da Pseudonimização de Registos

Existem várias abordagens possíveis para implementar soluções de pseudonimização de dados existentes em registos de eventos. À partida, existem dois cenários básicos que nos vêm à ideia. No primeiro, a pseudonimização é feita pelo sistema operativo ou pelas aplicações, na fase de geração de *logs*. No segundo, a pseudonimização é feita na fase de agregação.

A pseudonimização dos *logs*, no entanto, pode ser feita em qualquer uma das fases da infraestrutura da gestão de *logs*: geração, análise, armazenamento e apresentação.

A fase de análise incluí vários processos sequenciais:

- *Parsing* – Tradução do formato das mensagens;
- Agregação – Consolidação de entradas semelhantes numa única entrada que contém um campo com número de ocorrências de um dado evento;
- Transformação – Transformação de dados através da utilização de filtros;
- Indexação – Criação de índices com as palavras contidas nos registos para facilitar o processo de pesquisa.

4.2.1. Pseudonimização na fase de Geração dos Registos

A primeira possibilidade para implementação de um sistema de pseudonimização consiste em pseudonimizar os dados pessoais na fase de geração os *logs*. Assim, as entradas que são acrescentadas aos ficheiros de registo de eventos já contêm, à partida, os dados pessoais pseudonimizados. Esse processo pode ser feito diretamente pelo sistema operativo ou pelas aplicações, ou através de *plugins* que são instalados para o efeito (Figura 4.4).

Figura 4.4 – Pseudonimização na fase de geração dos registos.

Toda a infraestrutura de gestão de *logs* trabalha com os registos que foram pseudonimizados no momento em que foram gerados, pelo que não é necessário tomar medidas adicionais ao nível da pseudonimização para garantir a conformidade com o regulamento.

Ao serem tratados numa fase inicial do processo, os dados pessoais ficam desde logo salvaguardados e não ficam expostos nas fases de transporte, análise, armazenamento e apresentação.

Como a pseudonimização é feita pela aplicação ou pelo sistema operativo, a responsabilidade pela pseudonimização dos dados pessoais é do proprietário da informação, libertando o gestor da infraestrutura de gestão dessa responsabilidade. O problema de garantir a privacidade dos dados pessoais armazenados também está desde logo resolvido.

A grande desvantagem deste método é que para cada aplicação, fonte ou método de geração de *logs*, é necessário desenvolver uma solução específica. Se a aplicação ou o serviço não integrarem uma componente de pseudonimização de dados pessoais, os registos gerados não poderão ser ingeridos na infraestrutura de gestão, ou terão de ser tratados à parte através de um método alternativo, como qualquer um dos métodos seguintes.

4.2.2. Pseudonimização na fase de Ingestão dos Registos

Após a geração dos *logs* e respetivo armazenamento em ficheiros de registo, é possível pseudonimizar os dados neles contidos antes que estes sejam disponibilizados ao sistema de gestão (Figura 4.5).

Figura 4.5 – Pseudonimização na fase de ingestão dos registos.

A pseudonimização pode ser feita através de um módulo integrado num dos agentes responsáveis pela transmissão da informação entre o *host* e o servidor de gestão de *logs*, à saída dos clientes ou à entrada do servidor.

Se o processo for feito no lado do servidor, apenas terá de ser desenvolvido um módulo no agente do servidor.

Se o processo for feito à saída dos clientes, terão de ser desenvolvidos módulos para cada agente cliente. No entanto, isto resulta numa distribuição do processamento pelas várias máquinas, tornando o processo mais eficiente.

A pseudonimização também pode ser feita no lado do servidor, através de um módulo de pseudonimização entre o agente e o serviço de gestão de *logs*. Como o módulo de pseudonimização é independente das fontes que geram os *logs* e do sistema de gestão, existe uma grande flexibilidade nos requisitos inerentes ao seu desenvolvimento, podendo ser utilizados quaisquer métodos de cifragem ou resumo.

A desvantagem deste método reside na necessidade de se desenvolverem e configurarem módulos capazes de processar os diversos tipos de entradas. Cada módulo terá de ser configurado para interpretar os dados dos diversos tipos de documentos de entrada, identificar os campos com dados pessoais e proceder à respetiva pseudonimização.

4.2.3. Pseudonimização de Registos usando Duplicação de Índices

Um método simples de abordar o problema da pseudonimização consiste em duplicar os dados na fase de indexação (Figura 4.6).

Um dos índices contém os valores identificativos originais, os quais são cifrados e armazenados de forma segura num repositório apropriado, para permitir uma eventual consulta por pessoal devidamente credenciado e autorizado.

O segundo índice é idêntico ao primeiro, com a exceção dos dados identificativos que são pseudonimizados através de funções de resumo.

Este método envolve as fases ingestão, indexação e armazenamento, uma vez que a duplicação é feita na ingestão e a indexação é feita sobre os documentos originais e sobre os que contêm os dados pseudonimizados. Os índices são, por sua vez, armazenados em repositórios distintos.

Figura 4.6 – Pseudonimização de registos usando duplicação de índices.

Para garantir a privacidade e a confidencialidade dos dados, o repositório com os valores identificativos deve ser sujeito a medidas de segurança adicionais, através da implementação de mecanismos de cifragem e controlo de acessos.

A duplicação de índices é um método fácil de implementar, manter e utilizar. No entanto, como a informação é duplicada, este método consome recursos adicionais de armazenamento e licenciamento.

4.2.4. Pseudonimização na fase de Apresentação dos Registos

A pseudonimização dos valores identificativos pode ser feita apenas na fase de apresentação dos dados, na interface de monitorização. Os dados identificativos não poderão ser pesquisáveis e a página de resposta do motor de pesquisa é transformada num documento equivalente, mas com os valores identificativos pseudonimizados (Figura 4.7).

Figura 4.7 – Pseudonimização na fase de apresentação dos registos.

A pseudonimização dos dados, feita imediatamente antes da sua apresentação na interface de monitorização, causa necessariamente um atraso considerável no processo de pesquisa.

Como a pseudonimização é feita apenas na fase de apresentação, todos os dados identificativos estão armazenados com os valores originais, pelo que o utilizador comum não poderá ter acesso direto à base de dados. O acesso apenas poderá ser feito através de uma interface pré-definida (*dashboard*), ou utilizando um perfil com permissões de pesquisa e visualização limitadas.

4.3. Análise comparativa

Qualquer solução de pseudonimização de registos de eventos tem de intervir numa ou em mais fases do fluxo de dados, dentro da infraestrutura de gestão de *logs*: geração, ingestão e apresentação.

O processamento na fase de geração de registos, efetuado pelas aplicações ou pelos sistemas, implica desenvolver soluções independentes para cada aplicação. A principal vantagem é que não é necessário tomar medidas adicionais ao nível da pseudonimização para garantir a conformidade com o regulamento, o que liberta o gestor da infraestrutura de qualquer responsabilidade a este nível.

O processamento dos dados na fase de ingestão, antes de serem indexados e armazenados é normalmente feito por um módulo de transformação interno ao sistema de gestão de registos. O módulo de transformação pode ser desenvolvido livremente de acordo com um conjunto de requisitos pré-definidos, permitindo liberdade total no *design* do processo e nas funções utilizadas. A principal desvantagem deste método reside na necessidade de desenvolvimento de código específico para processar diferentes tipos de entradas, principalmente através da definição de padrões de extração de campos.

O método de pseudonimização através de duplicação de índices, implica sempre uma duplicação de mensagens arquivadas, sendo que um índice vai conter os valores originais e o outro vai conter os valores pseudonimizados. Se a replicação das mensagens for feita nos *hosts* que as geram, vai haver também uma duplicação do tráfego de rede, no transporte entre os clientes e o servidor de gestão de *logs*. A grande vantagem deste método é que, tanto uma pesquisa feita no índice com os dados originais, como uma pesquisa feita no índice com dados pseudonimizados, é igualmente rápida e eficiente, uma vez que não há necessidade de efetuar processamento adicional.

O processamento na fase de apresentação, aplicando filtros apropriados à página devolvida pelo motor de pesquisa, é outro método possível. Neste caso, as mensagens não necessitam de ser pseudonimizadas antes de serem arquivadas, o que torna o processo de ingestão muito mais rápido e eficiente. Além disso, para um utilizador com privilégios de acesso aos dados identificativos, a pesquisa também é muito mais eficiente porque não existe necessidade de efetuar qualquer reidentificação. A desvantagem deste método reside pesquisa e visualização de registos efetuada por utilizadores com privilégios limitados e sem acesso aos dados

identificativos. Como a informação tem de ser pseudonimizada antes de ser apresentada, a disponibilização dos resultados vai ser muito mais demorada.

A tabela Tabela 4.2 resume a principais vantagens e desvantagens das metodologias de pseudonimização apresentadas acima.

Tabela 4.2 - Principais vantagens e desvantagens das diferentes metodologias de pseudonimização

	Vantagens	**Desvantagens**
Pseudonimização na fase de Geração dos Registos	• As entradas que são acrescentadas aos ficheiros de registo de eventos já contêm, à partida, os dados pessoais pseudonimizados; • Não é necessário tomar medidas adicionais ao nível da pseudonimização para garantir a conformidade com o regulamento; • Liberta o gestor da infraestrutura de gestão dessa responsabilidade pelo processo de pseudonimização.	• Para cada aplicação, fonte ou método de geração de logs, é necessário desenvolver uma solução específica. • Se a aplicação ou sistema não pseudonimizarem os respetivos logs, estes terão de ser pseudonimizados por um dos métodos alternativos.
Pseudonimização na fase de Ingestão dos Registos	• Grande flexibilidade nos requisitos inerentes ao seu desenvolvimento.	• Necessidade de desenvolver código específico para processar diferentes tipos de entradas.
Pseudonimização de Registos usando Duplicação de Índices	• Método fácil de implementar, manter e utilizar; • Pesquisas feita no índice com os dados originais e no índice com dados pseudonimizados são igualmente rápidas e eficientes.	• Como a informação é duplicada, este método consome recursos adicionais.
Pseudonimização na fase de apresentação dos registos	• As mensagens não necessitam de ser pseudonimizadas antes de serem arquivadas, o que torna o processo de ingestão muito mais rápido. • Para um utilizador com privilégios de acesso aos dados identificativos, a pesquisa é muito mais eficiente.	• Como a pseudonimização dos dados é feita na interface de monitorização, o processo de pesquisa para um utilizador sem acesso a dados identificativos é menos eficiente. • O acesso para um utilizador comum apenas poderá ser feito através de uma interface pré-definida ou utilizando um perfil com permissões de pesquisa e visualização limitadas.

4.4. Síntese

Neste capítulo foram abordados os conceitos de anonimização e pseudonimização. Foram também apresentados argumentos para a utilização de uma função de resumo probabilística no processo de pseudonimização, nomeadamente a função HMAC.

Foram também discutidas quatro metodologias diferentes para efetuar o processo de pseudonimização, nomeadamente na fase de geração de registos, na fase de ingestão, através de duplicação de índices e na fase de apresentação. No final, foi feito um estudo comparativo entre as diferentes metodologias.

No próximo capítulo são apresentados três cenários de testes, para os quais foram desenvolvidas soluções concretas de pseudonimização da informação contida nos registos de eventos de sistemas e aplicações.

5. Cenários de Testes

Neste capítulo vão ser apresentadas três soluções concretas para a pseudonimização de dados contidos em registos de sistemas e aplicações. Cada solução utiliza uma infraestrutura de gestão de *logs* própria, baseada em soluções de gestão centraliza de registos: *Graylog*, *Splunk* e *ELK Stack*. As apresentações incluem os processos de configuração de entradas, extração de campos, transformação de mensagens, pesquisa e visualização de registos e criação de perfis. No final é feito um estudo comparativo entre as três soluções.

Os testes efetuados ao longo deste trabalho são todos realizados numa máquina física local a correr o sistema operativo Windows 7 Home Premium, juntamente com máquinas virtuais instaladas com o *software* de virtualização *Oracle VM VirtualBox,* versão 6.0.6.

Todos os servidores de gestão de registos testados, nomeadamente o *Graylog*, o *Splunk* e a *ELK Stack* estão instalados nas respetivas máquinas virtuais, todas elas a correr o sistema operativo Linux Ubuntu 16.04 de 64 bits. As máquinas estão configuradas com 4 CPU's, 4 GB de RAM, 40 GB de disco alocados dinamicamente e placa de rede em modo *Bridged*, para poder comunicar com outras máquinas virtuais e com a máquina física.

Como fontes geradoras de registos da máquina física local são usados os registos do *Event Log*, os registos de acesso dos servidores *Web* (Microsoft IIS e Apache HTTP Server) e os registos da *firewall*. Os registos de eventos do servidor SSH instalado na máquina física também são usados, embora estejam integrados no *Event Log*.

Nas máquinas virtuais são usados os registos provenientes do *Syslog*, os quais já integram os registos do servidor *OpenSSH* e da UFW, a *firewall* do Linux. São também usados os registos de acesso do servidor *Web* Apache instalado na máquina virtual.

Cada um dos três cenários envolve apenas a máquina física local e uma máquina virtual a correr o respetivo sistema de gestão de registos.

5.1.*Graylog*

O servidor *Graylog* é uma solução *Open Source* que apenas está disponível para sistemas operativos Linux e é suportado para as versões CentOS, Debian e Ubuntu. Este sistema surgiu por iniciativa de Lennart Koopman no ano de 2009, devido ao facto de na altura só existirem soluções de gestão de registos muito dispendiosas.

Apesar do processo de instalação deste gestor de registos (Anexo A – Instalação do *Graylog*) ser extenso, é uma solução bastante simples de configurar e utilizar.

A instalação do *Graylog* requer a pré-instalação do Java, do *ElasticSearch* e do *MongoDB*. É recomendada a utilização de bastante memória RAM e de um disco SSD para aumentar a eficiência do indexador *ElasticSearch*. A Figura 5.1 apresenta a arquitetura mínima de um servidor *Graylog* [21]:

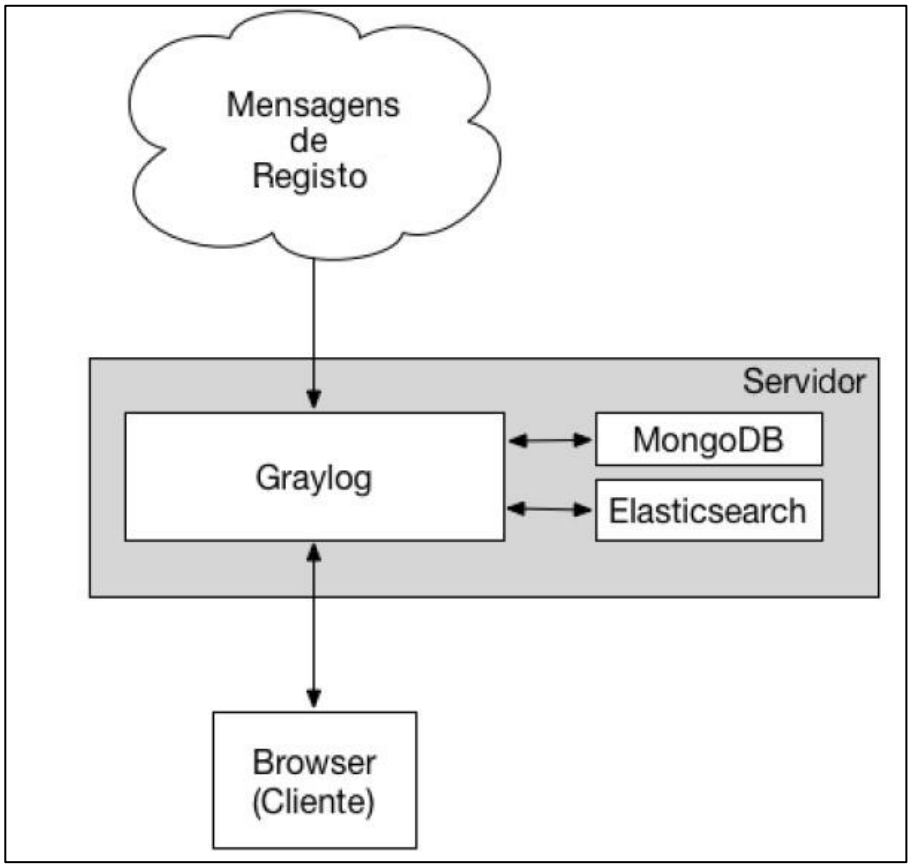


Figura 5.1 – Arquitetura mínima de um servidor *Graylog*[7]

[7] Fonte: Architecture — Graylog 2.4.6 documentation

O *MongoDB* é a tecnologia de base de dados que é usada para guardar as configurações do servidor. O conteúdo da base de dados *MongoDB* pode ser facilmente visualizado através de *software* próprio. A Figura 5.2 mostra a interface *MongoDB Compass*, onde podem ser visualizados os diversos componentes de configuração do servidor *Graylog*. Neste exemplo, são mostrados os detalhes da configuração de algumas entradas de registos.

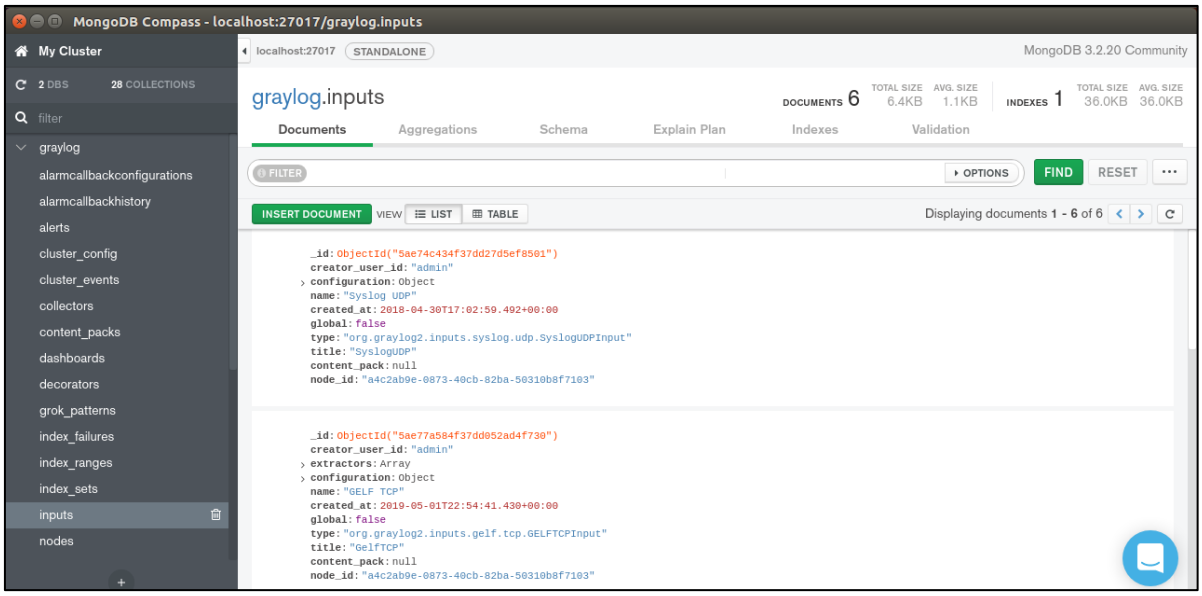


Figura 5.2 – Interface de visualização dos dados de configuração do servidor *Graylog*.[8]

O *ElasticSearch* é o *software* de indexação, pesquisa e armazenamento das mensagens, pelo que o servidor onde está instalado deverá possuir bons recursos de *hardware*. Este poderá estar instalado no mesmo *host* onde estão os restantes elementos que compõem o *Graylog*. De facto, esta é a configuração mais comum e a mais recomendada para projetos mais simples.

No caso em que os dados a analisar sejam críticos e tenham volumes consideráveis, é recomendável que a carga seja distribuída por vários servidores a executar o *ElasticSearch* de forma redundante e partilhada. A Figura 5.3 mostra a configuração típica de uma arquitetura distribuída.

[8] Fonte: Captura de ecrã feita pelo autor.

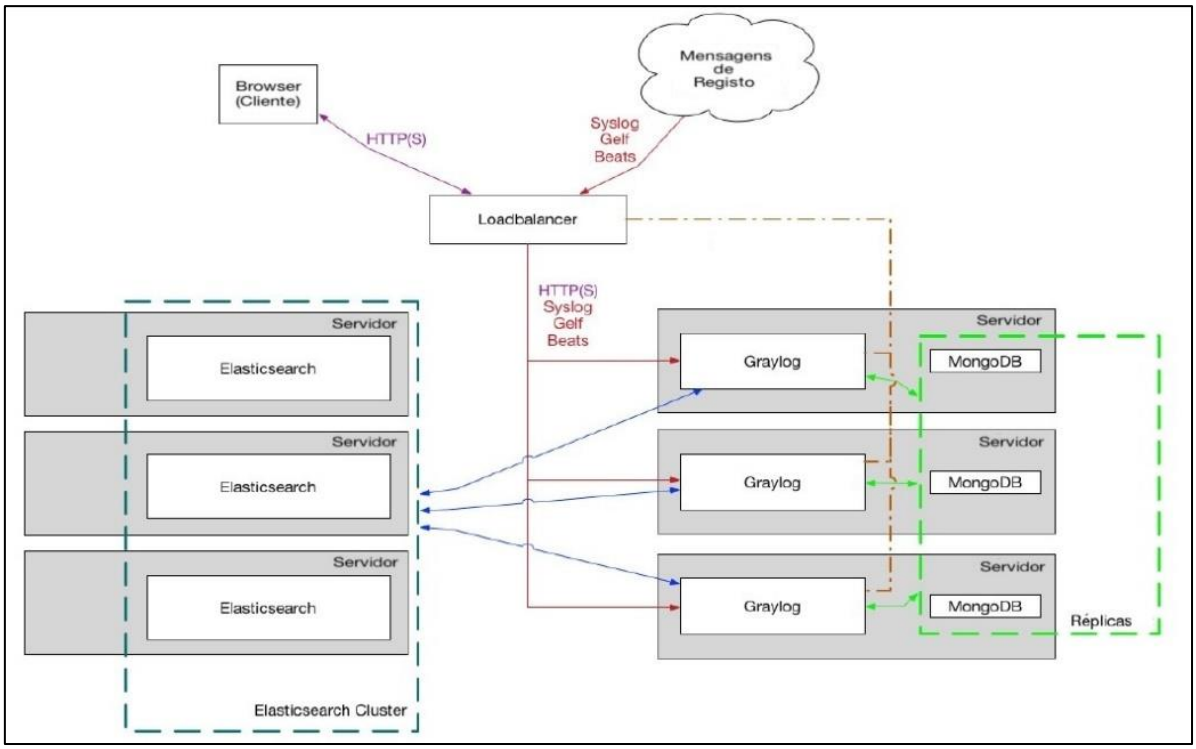


Figura 5.3 – Arquitetura de uma configuração distribuída de servidores *Graylog*.[9]

No cenário testado neste trabalho, apresentado na Figura 5.4, é utilizada a configuração mais simples, onde todos os componentes do sistema de gestão de registos estão instalados na máquina virtual.

As mensagens provenientes das aplicações e do sistema operativo da máquina física são reencaminhadas através de um agente de recolha e transmissão, o *NXLog*, para o servidor de gestão de registos.

O *NXLog* recolhe as entradas dos ficheiros de registo do Microsoft IIS, da *firewall* do Windows e do Event Log. Em seguida, reencaminha-as via UDP ou TCP, para o gestor de entradas do *Graylog (Graylog Input Manager)*.

[9] Fonte: Architecture — Graylog 2.4.6 documentation

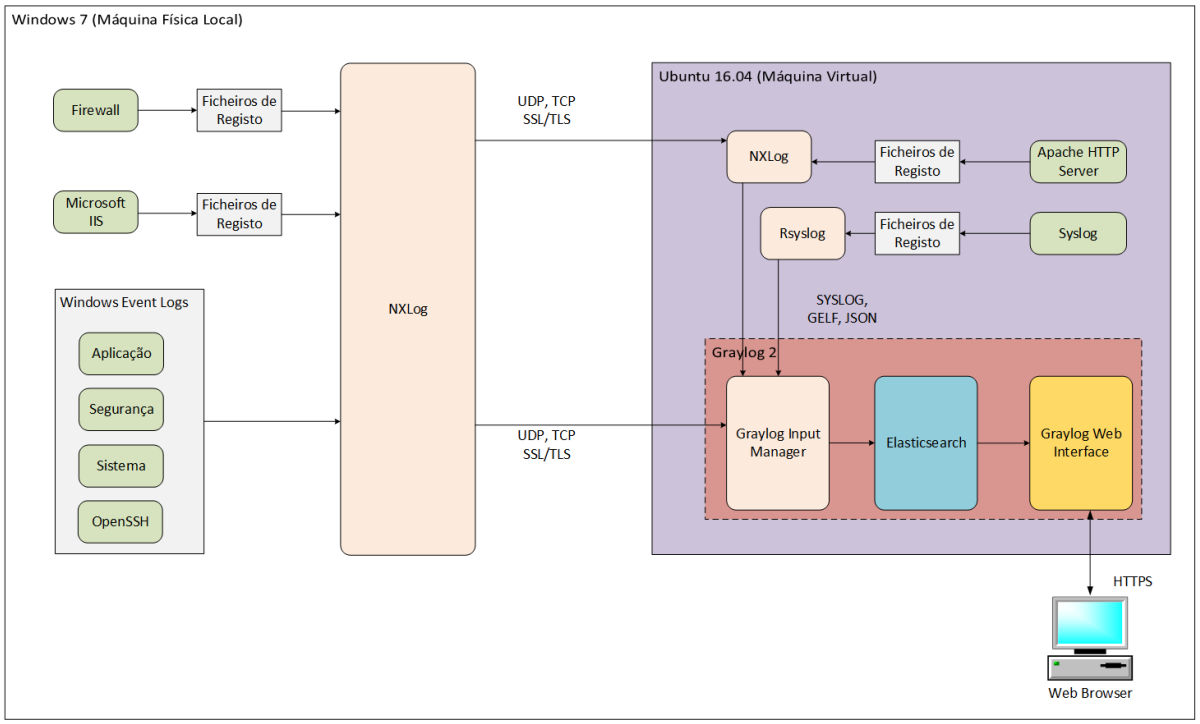


Figura 5.4 – Arquitetura do cenário de testes efetuados com o *Graylog*.

As mensagens de registo do servidor *Web Apache* instalado na máquina virtual são recolhidas pelo agente *NXLog* local e reencaminhas para o *Graylog*. As mensagens provenientes do *Syslog* são enviadas diretamente para o *Graylog*, via UDP.

Depois de darem entrada no *Graylog*, as mensagens são indexadas e arquivadas no servidor, para poderem ser pesquisadas e analisadas através de uma interface *Web*.

5.1.1. *NXLog*

A configuração do *NXLog* é feita a partir da edição do ficheiro *nxlog.conf* [22].

A Figura 5.5 mostra as linhas do ficheiro onde estão configuradas as entradas dos registos do *Event Log*, do *Microsoft IIS*, da *firewall* do Windows e do servidor *Web* da Apache, respetivamente.

As entradas do *Event Log* são recolhidas através do módulo *im_msvistalog*. Normalmente, o *NXLog* transporta versões curtas e truncadas das mensagens dos registos do *Event Log*. A linha **EXEC $ShortMessage = $Message** substitui as versões curtas pelas versões completas.

```
<Input winevent>
   Module im_msvistalog
   Exec $ShortMessage = $Message;
</Input>

<Input iis>
    Module im_file
    File C:\inetpub\logs\LogFiles\W3SVC1\u_ex*
</Input>

<Input WinFirewallLog>
   Module im_file
   File C:\Windows\system32\LogFiles\Firewall\pfirewall.log
</Input>

<Input WinApache>
    Module im_file
    File C:\Apache24\logs\access.log
</Input>
```

Figura 5.5 – Configuração das entradas no *NXLog*.

O modulo *im_file* é um componente do *NXLog* que recolhe entradas de registo dos ficheiros de *logs*. Neste caso, o módulo é usado para recolher as entradas de *logs* do IIS, da *firewall* do Windows e do servidor Apache, respetivamente.

A configuração das saídas é feita através dos módulos *om_tcp* e *om_ssl*. O primeiro é usado para encaminhar mensagens via TCP e o segundo é usado para encaminhar mensagens através de uma ligação segura SSL/TLS. A Figura 5.6 mostra a configuração das saídas do *NXLog*:

```

<Output winevent_graylog>
    Module om_tcp
    Host %GRAYLOG%
    Port 12201
    OutputType GELF_TCP
</Output>

<Output iis_graylog>
    Module          om_ssl
    Host            %GRAYLOG%
    Port            12212
    CAFile          %CERTDIR%/rootCA.pem
    CertFile        %CERTDIR%/artur-TOSH.crt
    CertKeyFile     %CERTDIR%/artur-TOSH.key
    AllowUntrusted  TRUE
</Output>

<Output WinFirewall_graylog>
   Module om_tcp
   Host %GRAYLOG%
   Port 12203
   Exec $ShortMessage = $raw_event;
   OutputType GELF_TCP
</Output>
```

```
26
27  <Output Apache_graylog>
28     Module om_tcp
29     Host %GRAYLOG%
30     Port 12204
31     Exec $ShortMessage = $raw_event;
32     OutputType GELF_TCP
33  </Output>
```

Figura 5.6 – Configuração das saídas do *NXLog*.

Neste cenário apenas é configurada uma saída SSL/TLS para efeitos de estudo. Numa solução real em produção, deverão ser usadas ligações seguras em todas as comunicações.

A saída SSL/TLS é encaminhada para o porto 12212 do servidor, o qual está a ser escutado por outro agente *NXLog*, configurado para receber entradas via SSL/TLS. Este, por sua vez, reencaminha-as para o gestor de entradas do *Graylog* via TCP para o porto 12202.

As saídas são enviadas para o servidor, cujo endereço IP é o que consta na variável %GRAYLOG%, nos portos 12201, 12212, 12203 e 12204, respetivamente.

As mensagens de saída enviadas via TCP estão configuradas para serem apresentadas no formato *Graylog Extended Log Format (GELF)*. Uma mensagem GELF é uma cadeia JSON com os campos [21]:

version | host | short_message | full_message | timestamp | level | facility | line | file | [additional field]

Estas mensagens apresentam uma estrutura idêntica à apresentada na Figura 5.7:

```
{
  "version": "1.1",
  "host": "example.org",
  "short_message": "A short message that helps you identify what is going on",
  "full_message": "Backtrace here\n\nmore stuff",
  "timestamp": 1385053862.3072,
  "level": 1,
  "_user_id": 9001,
  "_some_info": "foo",
  "_some_env_var": "bar"
}
```

Figura 5.7 – Estrutura típica de uma mensagem GELF [10].

A configuração da saída *iis_graylog*, usa uma ligação segura SSL/TLS e requer a utilização de três ficheiros existentes na pasta *nxlog/cert*. Os ficheiros, *artur-TOSH.key*, *artur-TOSH.crt* e *rootCA.pem*, são a chave privada, a chave pública e o certificado CA, respetivamente. Todos estes ficheiros são gerados através da ferramenta *OpenSSL*[23, p. 230].

[10] Fonte: GELF Payload Specification - Graylog 2.4.6 documentation

A chave privada da CA, *rootCA.key***,** é gerada com o comando:

$ openssl genrsa -out rootCA.key 2048

O certificado *rootCA.pem* é gerado através do comando:

$ openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 730 -out rootCA.pem

A chave privada do servidor artur-TOSH.key é gerada com o comando:

$ openssl genrsa -out artur-TOSH.key 2048

A chave privada é usada para efetuar um pedido de assinatura do certificado *artur-TOSH.csr*:

$ openssl req -new -key server.key -out server.csr

Finalmente, o pedido é assinado pela CA, gerando o certificado *artur-TOSH.crt*:

$ openssl x509 -req -in server.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out Artur-TOSH.crt -days 500 -sha256

O reencaminhamento das entradas para as saídas é feito através de instruções de direcionamento, como mostra a Figura 5.8:

```
1  <Route 1>
2       Path winevent => winevent_graylog
3  </Route>
4
5  <Route 2>
6       Path  iis => iis_graylog
7  </Route>
8
9  <Route 3>
10      Path  WinFirewallLog => WinFirewall_graylog
11 </Route>
12
13 <Route 4>
14      Path  WinApache => Apache_graylog
15 </Route>
```

Figura 5.8 – Configuração de encaminhamentos do *NXLog*

5.1.2. Pesquisa e Visualização de Registos

O acesso à interface *Web* do *Graylog* pode ser feito usando um navegador, através do endereço IP do servidor no porto 9000. No caso do acesso ser feito a partir de um navegador local, é usado o endereço **http://localhost:9000**.

A pesquisa e visualização de registos é feita a partir do separador *Search* da barra superior, como é mostrado na Figura 5.9.

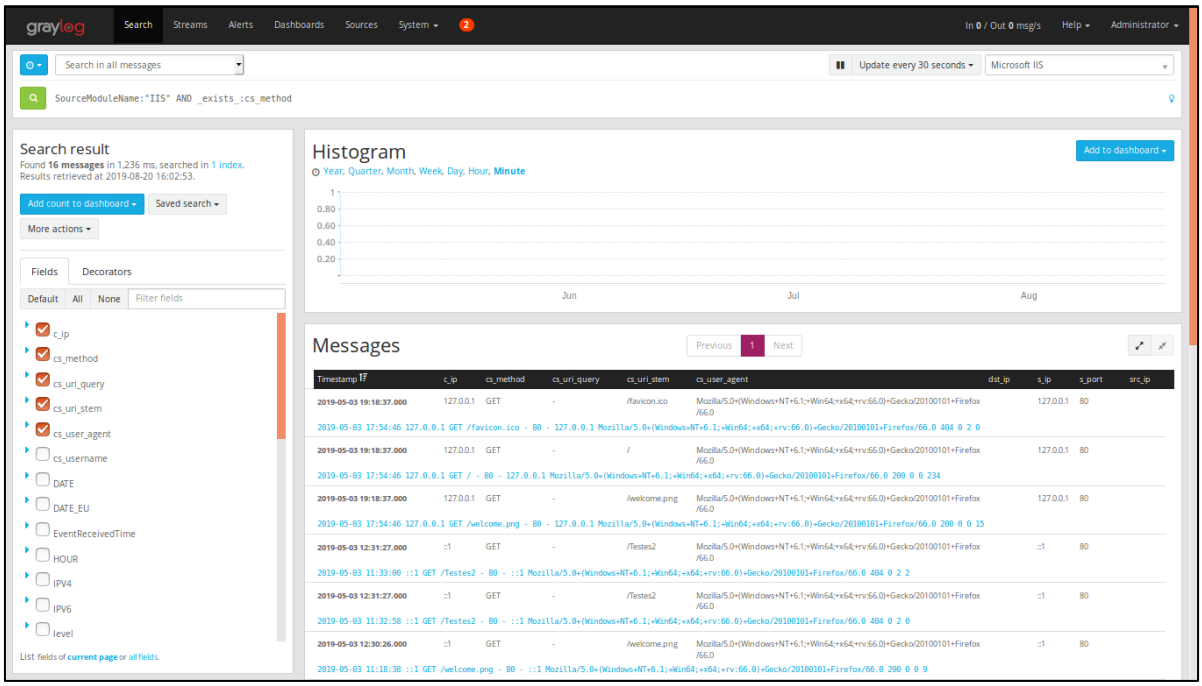


Figura 5.9 – Pesquisa e Visualização de Registos no Graylog.[11]

Na terceira linha, junto ao botão verde com uma lupa desenhada, é possível escrever uma expressão de pesquisa usando uma sintaxe própria. Esta sintaxe, desenvolvida pela Apache para o seu motor de pesquisa *Lucene*, é também a sintaxe usada pelo *Elasticsearch*, o motor de pesquisa usado pelo *Graylog*. Como é visto no terceiro cenário, o *Elasticsearch* é também o motor de pesquisa que integra a *ELK Stack*. Um exemplo de uma expressão de pesquisa é:

SourceModuleName:"IIS" AND _exists_:cs_method

Esta expressão filtra todos os registos cujo campo *SourceModuleName* contém o valor "IIS" e que contêm o campo *cs_method*. Por outras palavras, a expressão filtra todas as mensagens de acesso vindas do servidor *Web* da Microsoft que correspondem a entradas de registo, descartando as mensagens correspondentes a diretivas.

A expressão pode ser aplicada a todos os registos, ou pode ser selecionada uma janela de tempo, cujos intervalos podem ir de cinco segundos até 30 dias.

[11] Fonte: Captura de ecrã feita pelo autor.

As mensagens são apresentadas no quadro *Messages*, acima do qual é também possível visualizar um histograma com o volume de tráfego.

Do lado esquerdo, no quadro *Search result*, é possível selecionar os campos que se pretendem visualizar. A partir deste quadro é também possível visualizar a expressão de pesquisa no formato JSON, através da opção: *More actions* > *Show query*. Também é possível, a partir deste quadro, exportar o resultado da pesquisa no formato *csv*, através da opção: *More actions* > *Export as CSV*.

5.1.3. Configuração de Entradas

Para que o servidor possa receber mensagens provenientes das diversas fontes, têm de ser configuradas as respetivas entradas. Isso é feito através da interface *Web* do *Graylog*, entrando no separador *System* > *Inputs* e escolhendo a opção *Launch new input*.

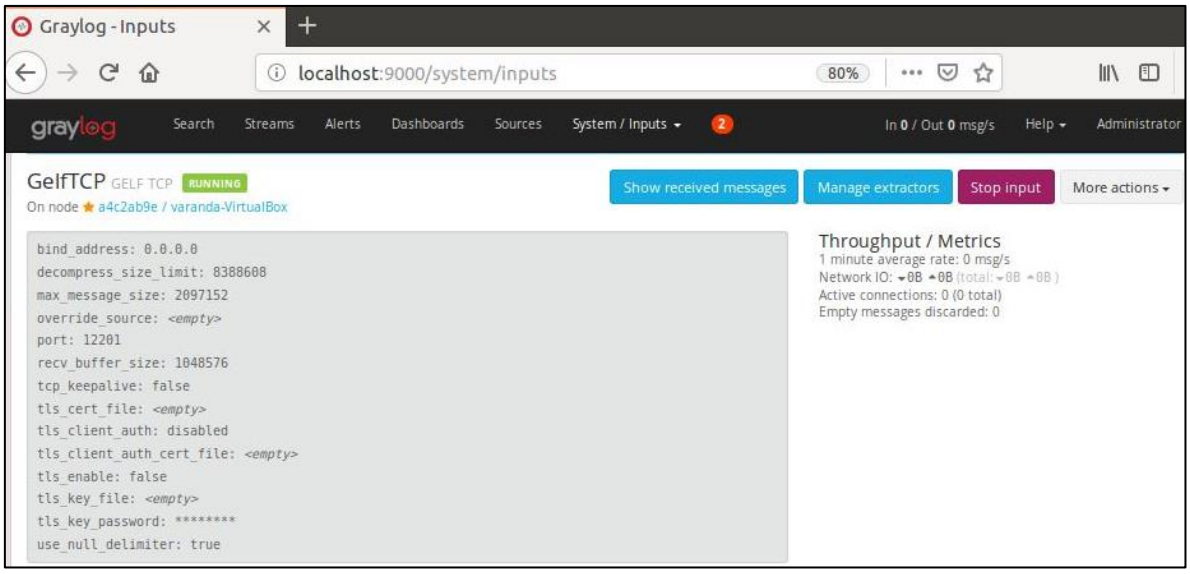


Figura 5.10 – Configuração de uma entrada no *Graylog*. [12]

As entradas das mensagens provenientes do *Syslog* são do tipo **Syslog UDP** e entram no porto 1514. As mensagens provenientes do Windows que entram nos portos TCP 12201, 12203 e 12204, são configuradas usando o tipo **GELF TCP** nos portos respetivos. Depois de configuradas, as entradas exibem o estado "RUNNING".

A Figura 5.10 mostra a configuração da entrada recebida por TCP no porto 12201, onde entram as mensagens provenientes do *Event Log* do Windows.

[12] Fonte: Captura de ecrã feita pelo autor.

5.1.4. Extração de Campos das Mensagens

Depois de passarem pelo gestor de entradas do *Graylog* e antes de entrarem no indexador, as mensagens são analisadas por extratores de campos. Estes extratores são usados para extrair dados das mensagens de texto e colocá-los em campos apropriados. Assim, depois de indexados, poderão ser filtrados e analisados mais facilmente.

Os extratores podem ser configurados a partir da página de configuração das entradas, através da opção *Menage extractors* e são aplicados a todas as mensagens recebidas na respetiva entrada onde estão configurados. A configuração é iniciada pela opção *Add extractor* > *Get started*, a qual dá acesso à opção *Load Message*, que permite ler uma mensagem típica desta entrada, para configurar a extração dos respetivos campos.

A Figura 5.11 mostra a lista de campos que surge após a escolha da opção *Load Message*. Neste caso, interessa extrair dados do campo *message*. A extração pode ser feita através de vários métodos, por exemplo, através de um padrão *Grok* ou de uma estrutura JSON.

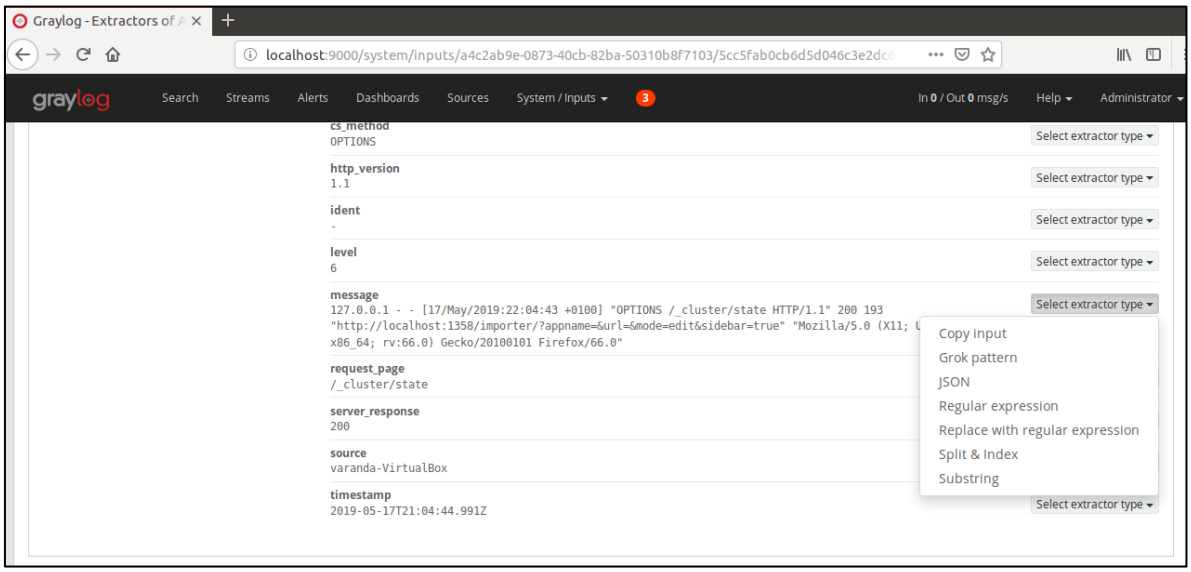


Figura 5.11 – Configuração de extratores no Graylog.[13]

Neste cenário, a extração de campos é feita através de padrões *Grok*, também usados pelo *Logstash* no terceiro cenário. A lista de padrões usados em ambos os cenários pode ser consultada no Anexo C – Padrões *Grok* de extração de campos

A configuração dos padrões é feita através de sintaxe própria, embora já existam expressões pré-configuradas para a maioria dos tipos de registos conhecidos. De qualquer forma, as expressões são relativamente simples de construir.

[13] Fonte: Captura de ecrã feita pelo autor.

A sintaxe para um padrão *Grok* é do tipo %{SYNTAX:SEMANTIC}.

SYNTAX é o nome do padrão que corresponde ao texto. Por exemplo, 2019-05-03 17:54:46 corresponde ao padrão DATESTAMP e 127.0.0.1 corresponde ao padrão IP. O padrão NOTSPACE é um padrão especial que corresponde a uma cadeia de caracteres sem espaço. Assim, se os campos da mensagem estão apenas divididos por espaços, é possível construir facilmente uma expressão *Grok* da seguinte forma:

%{NOTSPACE:campo1} %{NOTSPACE:campo2} %{NOTSPACE:campo3} …

SEMANTIC é o nome do identificador de campo dado ao texto correspondido. Por exemplo, para um endereço IP, um identificador de campo pode ser *s_ip* ou *c_ip*, que correspondem aos campos *s-ip* e *c-ip* de registos do Microsoft IIS. O identificador UNWANTED é usado para se descartar um campo extraído.

A título de exemplo, a configuração de um padrão *Grok* para extração de dados de mensagens de registo de acesso do Microsoft IIS,

2019-05-03 17:54:46 127.0.0.1 GET /favicon.ico - 80 - 127.0.0.1 Mozilla/5.0 404 0 2 0

pode ser feita usando a expressão:

```
%{DATESTAMP:UNWANTED} %{IP:s_ip} %{WORD:cs_method}
%{URIPATHPARAM:cs_uri_stem} %{NOTSPACE:cs_uri_query} %{INT:s_port}
%{NOTSPACE:cs_username} %{IP:c_ip} %{NOTSPACE:cs_user_agent}
%{INT:sc_status} %{INT:sc_substatus} %{INT:sc_win32_status}
%{INT:time_taken}
```

Note-se que os identificadores criados (*s_ip, cs_method, cs_uri_stem, cs_uri_query*, etc…) correspondem aos nomes dos campos componentes de registo de acesso do IIS.

A página de configuração do extrator permite ir testando-o, à medida que se constrói a expressão. Também é possível aceder à página de padrões arquivados na opção "*stored pattern*", a partir da qual é possível ver, criar e editar todos os padrões armazenados no *Graylog*.

Depois de devidamente configurados, os extratores podem ser exportados em formato JSON através a opção *Actions* > *Export extrators*.

5.1.5. Geração de *Streams*

Uma *stream*, no Graylog, é uma sequência de mensagens que estão agrupadas numa categoria, de acordo com um conjunto de regras pré-definidas. A *stream* principal, a *root stream*, contém todas as mensagens recebidas pelo servidor. A Figura 5.12 mostra a geração de duas *streams*, a partir da *root stream*, definidas de acordo com as respetivas regras.

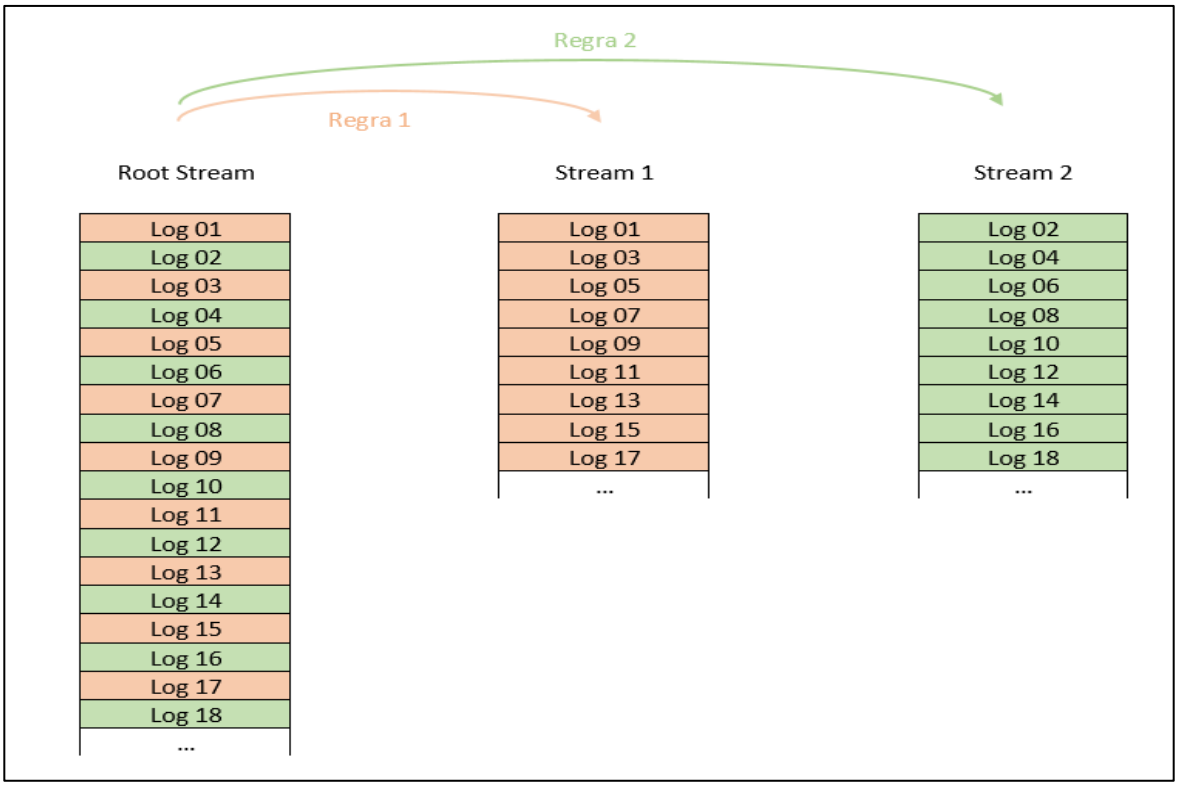


Figura 5.12 – Geração de *streams* no *Graylog* de acordo com a definição de regras.

As *streams* são geradas à medida que as mensagens vão sendo ingeridas pelo servidor, o que permite a geração de alertas em tempo real e o encaminhamento para outros sistemas.

As mensagens pertencentes a uma determinada *stream* podem ser visualizadas a partir do separador *Streams*. A expressão de pesquisa correspondente pode ser vista através da opção *More actions* > *Show query* e é do tipo: *streams:5cc9756c4e9cfa052f661554*. Este é um identificador único aplicado a todas as mensagens da *stream*.

A utilização de uma *stream* para listar uma categoria de mensagens é muito mais eficiente do que uma pesquisa na base de dados de acordo com um conjunto de regras, pois as mensagens da *stream* já estão marcadas com o identificador único.

A configuração de *streams* é feita entrando no separador *Streams* e escolhendo a opção *Create Stream*. Depois de atribuir um nome e uma descrição guarda-se a *stream* carregando na opção

Save. As regras são definidas através da opção *Manage Rules*, escolhendo a entrada à qual vão ser aplicadas e carregando na opção *Add stream rule*. Depois de definida a regra, esta é gravada carregando na opção *Save*. Depois de definidas todas as regras, carrega-se na opção *I'm done!*. Finalmente, a *stream* é ativada na opção *Start stream*.

5.1.6. Pseudonimização usando *Pipelines*

Configuração de *Pipelines*

No *Graylog*, os campos das mensagens de uma *stream* podem ser processados através de regras definidas pelo utilizador, usando um mecanismo denominado *pipelines*. As regras definidas nas *pipelines* podem usar funções pré-definidas pelo Graylog e tem o aspeto de pequenas rotinas de código. Estas rotinas usam expressões de controlo de fluxo do tipo WHEN…THEN como mostra o exemplo seguinte:

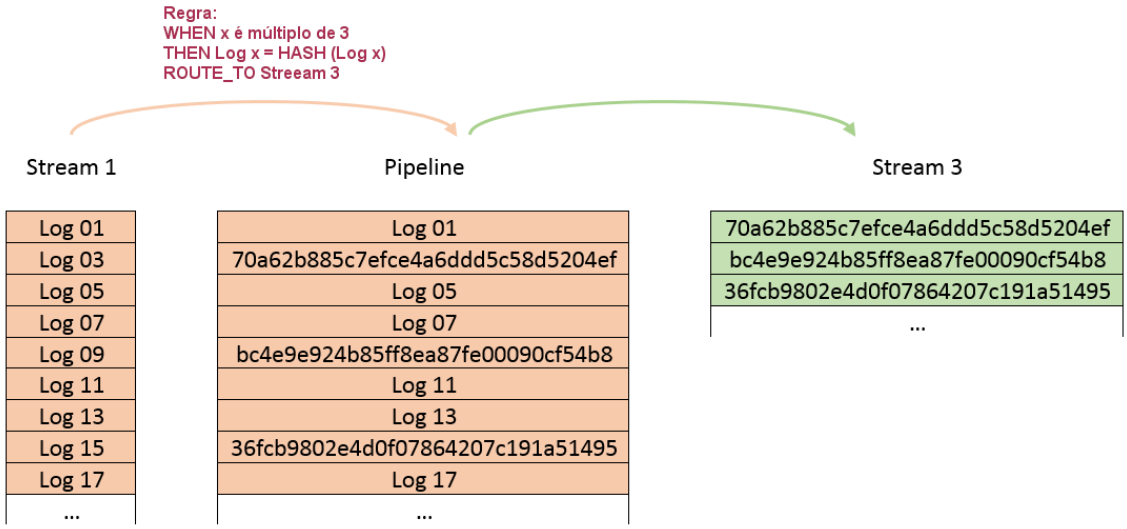


Figura 5.13 – Processamento de mensagens usando *pipelines*.

Na Figura 5.13, a *pipeline* aplicada à *stream 1* está definida com uma regra que substitui as mensagens com índice múltiplo de três pelo resultado de uma função de *hash* e reencaminha-as para a *stream 3*.

A configuração das regras das *pipelines* é feita entrando no separador *System* > *Pipelines*. A partir daqui é possível acrescentar novas *pipelines* através da opção *Add new pipeline*. Depois de atribuir um nome e uma descrição, guarda-se a *pipeline* carregando na opção *Save*.

A associação da *pipeline* à respetiva *stream* é feita através a opção *Edit connections*. As regras a aplicar são definidas *à priori* através da opção *Manage rules*. Depois de criadas, estas podem ser aplicadas a qualquer *pipeline* através da opção *Edit*.

Pseudonimização de dados usando *Pipelines*

A pseudonimização de mensagens de registo usando *pipelines* pode ser feita utilizando uma de duas estratégias possíveis: duplicação de índices e criação de mensagens de reidentificação.

Usando a primeira estratégia, apresentada na Figura 5.14, as mensagens provenientes das várias fontes são enviadas pelo *NXLog* em duplicado, para permitir a pseudonimização por duplicação de índices. Neste caso, a *root stream* é composta por conjuntos duplicados de mensagens. Um deles segue diretamente para o indexador. O outro é subdividido, por tipos, em várias *streams* que vão ser aplicadas nas entradas das respetivas *pipelines*. Em cada *pipeline*, os dados são transformados de acordo com regras próprias. Os dados identificativos são pseudonimizados usando funções de *hash* e remetidos para as respetivas *streams*, as quais são ingeridas pelo indexador.

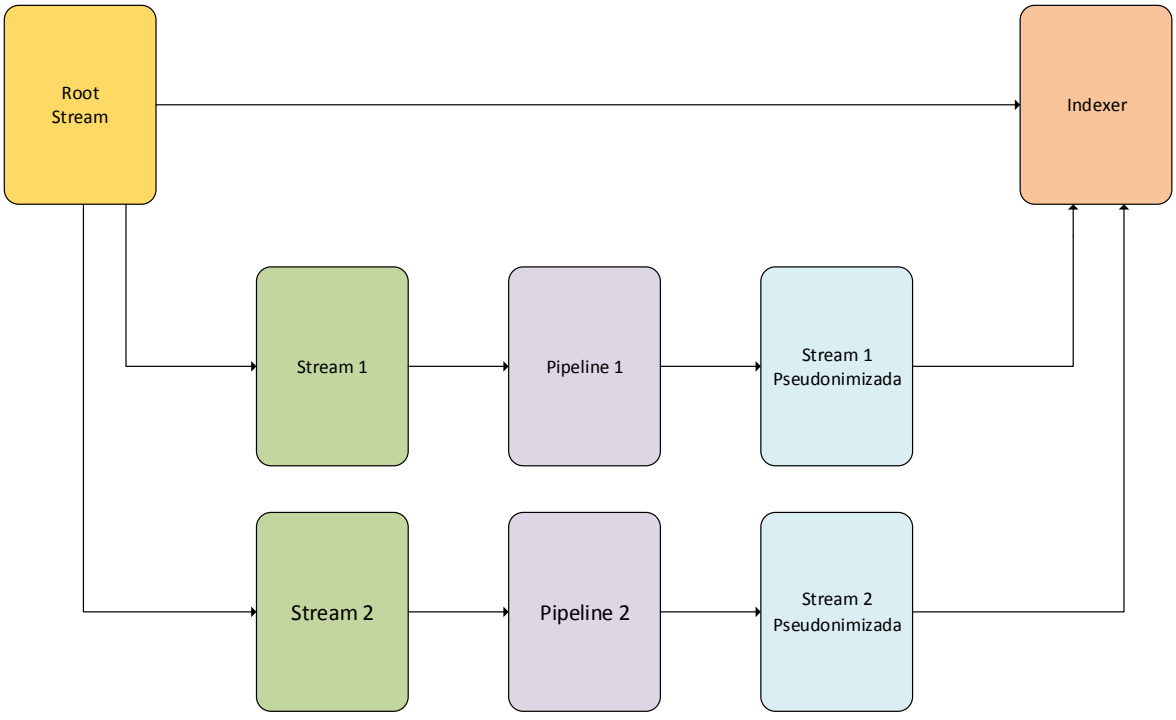


Figura 5.14 - Fluxo de dados do processo de pseudonimização usando duplicação de índices.

Usando a segunda estratégia, apresentada na Figura 5.15, são criadas mensagens de reidentificação, compostas pelos pares correspondentes aos valores identificativos e aos

respetivos resumos, as quais são reencaminhadas para *streams* de identidades. Estas *streams* funcionam como arquivos de identidades e apenas estão acessíveis aos utilizadores com um perfil privilegiado, para permitir a reidentificação nos termos estipulados pelo regulamento de proteção de dados.

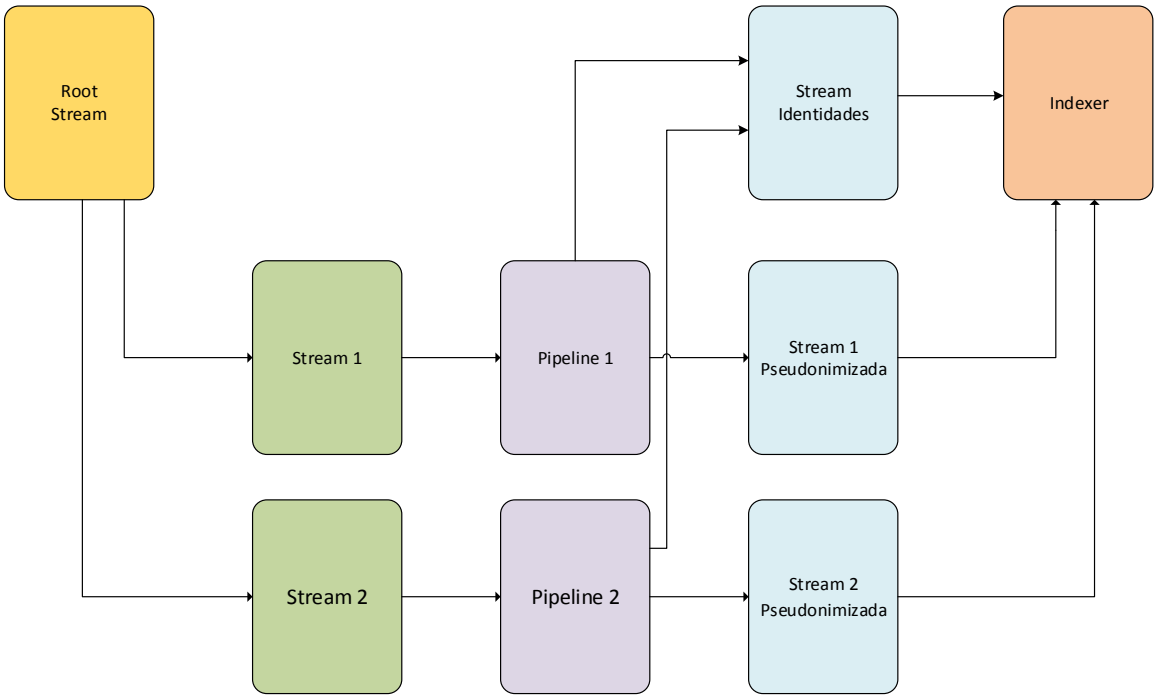


Figura 5.15 – Fluxo de dados do processo de pseudonimização usando mensagens de reidentificação.

Na primeira estratégia, a duplicação de mensagens à saída dos *hosts* duplica o volume de dados no transporte e no armazenamento, mas permite a pesquisa de identidades num índice não pseudonimizado, cujo acesso é exclusivo a um perfil privilegiado.

Na segunda estratégia, a criação de mensagens de reidentificação e respetivo encaminhamento para um arquivo de identidades envolve menos recursos, tanto a nível de tráfego como a nível de armazenamento, mas não existe a possibilidade de se efetuarem pesquisas de identidades.

As regras usadas na configuração das *pipelines* usadas neste trabalho podem ser consultadas no Anexo E – Configuração de pipelines do *Graylog*.

Por exemplo, a pseudonimização dos dados identificativos de mensagens de registo de um servidor SSH, pode ser feita através da seguinte regra:

```
1  rule "pseudonymize ssh"
2  when
3      has_field("sshd_client_ip") && has_field("sshd_user")
4  then
5      let h_ip = sha256(to_string($message.sshd_client_ip));
6      let h_user = sha256(to_string($message.sshd_user));
7      set_field("sshd_client_ip", h_ip);
8      set_field("sshd_user", h_user);
9      remove_field("payload");
10     remove_field("full_message");
11     remove_field("IPV4");
12
13     set_field("message", "sshd: Accepted password for $sshd_user from
   $sshd_client_ip port $sshd_port ssh2");
14     set_field("Pseudonymised", "Yes");
15     route_to_stream("Pseudonymised Messages");
16 end
```

A primeira linha atribui à regra o nome "pseudonymize ssh".

As linhas 2 e 3 constituem uma expressão de controlo de fluxo que aplica a regra apenas às mensagens que contêm os campos *sshd_client_ip* e *sshd_user*.

As linhas 5 e 6 definem as variáveis *h_ip* e *h_user*, nas quais são colocados os valores resultantes da função de *hash* SHA-256 aplicada aos conteúdos dos campos *sshd_client_ip* e *sshd_user*, respetivamente.

As linhas 7 e 8 substituem os conteúdos dos campos *sshd_client_ip* e *sshd_user* pelos respetivos valores de resumo que foram armazenados nas variáveis *h_ip* e *h_user*.

As linhas 9 a 11 removem da mensagem os campos redundantes *payload*, *full_message* e *IPV4* que contêm dados identificativos.

As linhas 13 e 14 criam novos campos na mensagem. O campo *messagem* vai conter a mensagem original com os dados identificativos pseudonimizados. O campo *Pseudonymised*, preenchido com o valor *"Yes"*, identifica a mensagem como sendo uma mensagem pseudonimizada.

Finalmente, a linha 15 reencaminha a mensagem para a *stream* "*Pseudonymised Messages"*.

Na Figura 5.16 é possível observar as mensagens do servidor SSH com os campos *sshd_user* e *sshd_client_ip* pseudonimizados.

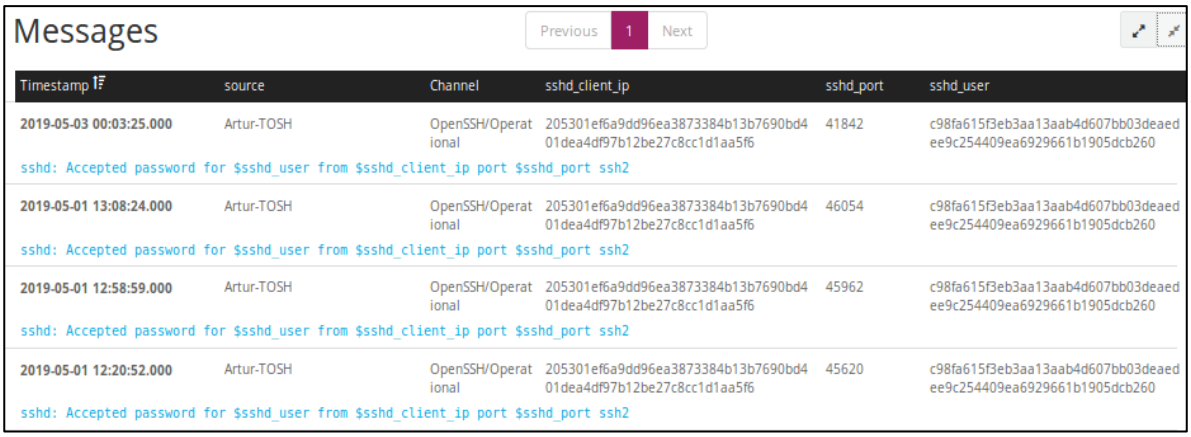


Figura 5.16 – Mensagens do servidor SSH com campos pseudonimizados.[14]

5.1.7. Criação de perfis de utilizador no *Graylog*

A partir do separador *System* > *Authentication* é possível criar perfis de utilizador, selecionando os recursos a que cada perfil tem acesso.

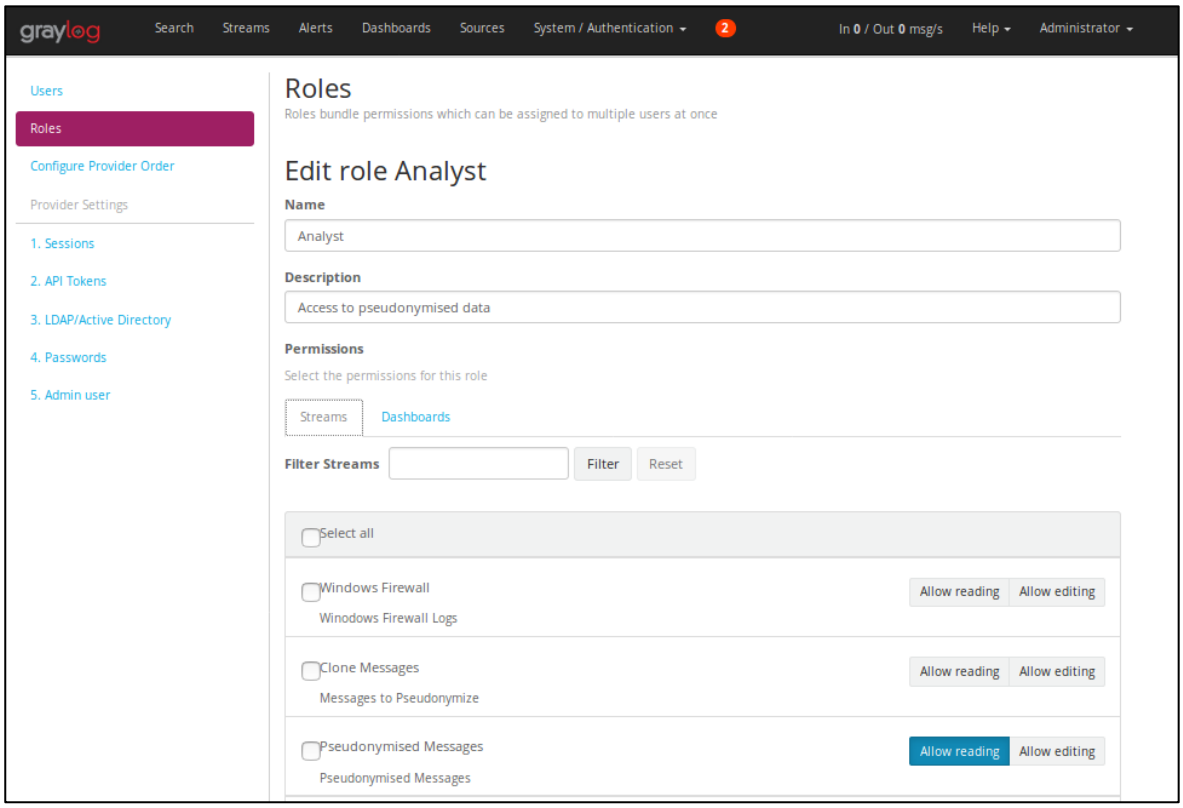


Figura 5.17 – Configuração de perfis de utilizador no Graylog.[15]

[14] Fonte: Captura de ecrã feita pelo autor.
[15] Fonte: Captura de ecrã feita pelo autor.

Por exemplo, pode ser criado um perfil de analista com acesso apenas às *streams* que contêm registos como dados identificativos pseudonimizados. Desta forma é possível garantir a conformidade com o RGPD.

Ao mesmo tempo, é possível criar um perfil de administrador, com acesso às *streams* com dados identificativos, para efeitos de auditoria e nos termos do regulamento.

A Figura 5.17 mostra as permissões do analista, apenas com acesso de leitura às mensagens pseudonimizadas.

5.2.*Splunk*

O *Splunk* está disponível em três versões, *Splunk Enterprise*, *Splunk Free* e *Splunk Cloud*, sendo suportado por sistemas operativos Windows, Linux e OSX. O processo de instalação no Ubuntu é simples e direto, bastando, para isso, extrair o ficheiro de instalação. Para o cenário apresentado é usada a versão *splunk-7.2.6-c0bf0f679ce9-linux-2.6-amd64.deb* [24].

O *Splunk Enterprise* pode ser instalado numa variedade de configurações, desde uma instalação centrada num único servidor, até uma infraestrutura distribuída. Na configuração mais simples, é usado um servidor único para gerir todas as funções do *Splunk,* incluindo gestão de entradas, interpretação, análise, indexação e pesquisa de dados. É a configuração ideal para provas de conceito, estudos, projetos pessoais ou projetos mais simples.

Num ambiente de produção real, à medida que o sistema aumenta, é necessário atribuir algumas funcionalidades a servidores especializados, como agentes encaminhadores de mensagens, indexadores e motores de pesquisa, para aumentar a capacidade de processamento. Os indexadores e os motores de pesquisa podem ser replicados e agregados em *clusters*, para garantir que os dados estejam sempre disponíveis.

Neste trabalho, a versão utilizada é a versão *Splunk Free*, a qual apenas permite a instalação numa arquitetura centrada num único servidor. Esta versão também não suporta a utilização por mais do que um utilizador, nem permite a configuração de alertas ou a definição de perfis de utilizador.

A arquitetura do cenário de testes efetuados com o *Splunk* é idêntica à usada no cenário anterior (ver Figura 5.4). A única diferença é que esta solução usa um indexador e um motor de pesquisa próprios, ao contrário da solução do cenário anterior que usa o indexador e motor de busca *Elasticsearch*.

Neste cenário, as mensagens enviadas pelo *NXLog* para o servidor não estão formatadas, ao contrário do que acontece no cenário anterior, o qual usa o formato GELF.

5.2.1. Configuração de Entradas

Normalmente, o *Splun*k fica instalado na pasta */opt/splunk/* e pode ser inicializado através do comando **$ splunk start**. Outros dois comandos bastante usados durante os testes são o

comando para encerrar o servidor **$ splunk stop** e o comando para purgar os índices **$ splunk clean eventdata**.

Depois de inicializado, o acesso à interface *Web* do *Splunk* pode ser feito usando um navegador, através do endereço IP do servidor no porto 8000.

Embora existam vários métodos para configurar as entradas no *Splunk*, apenas dois são usados durante os testes: através da interface *Web* ou através da edição do ficheiro de configuração *inputs.conf*.

A configuração através da interface *Web* pode ser feita a partir do separador *Settings* > *Data Inputs*. Podem ser configuradas entradas de vários tipos, como mostram as opções do lado esquerdo da Figura 5.18.

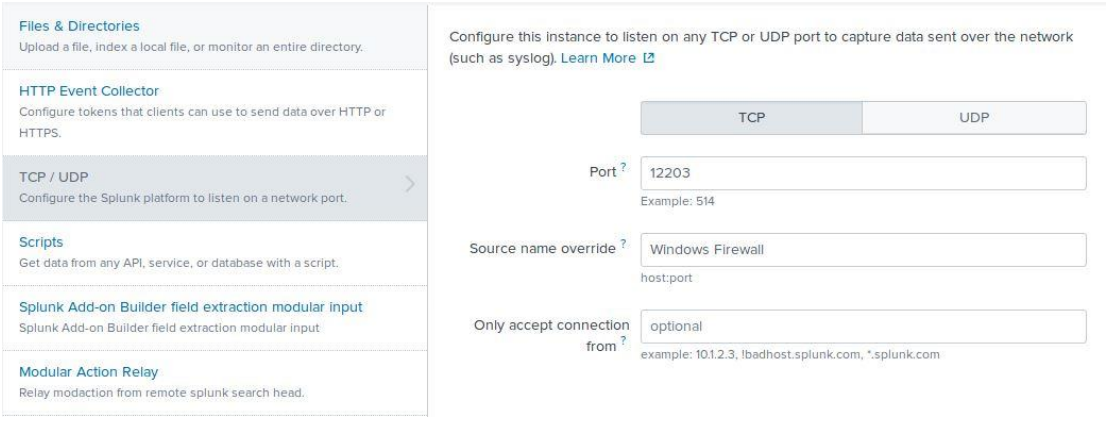


Figura 5.18 – Configuração de entradas no *Splunk*, usando a interface *Web*.[16]

Neste caso, são usadas entradas de portos TCP para as mensagens provenientes do Windows e do porto UDP 1514 para as mensagens provenientes do *Syslog* local.

A Figura 5.18 mostra a configuração da entrada TCP no porto 12203, onde entram as mensagens provenientes da *firewall* do Windows.

A configuração de entradas através da edição do ficheiro de configuração é necessária no caso de se pretender transformar as entradas na fase de ingestão, antes da indexação, por exemplo, para anonimizar ou pseudonimizar campos com dados pessoais identificativos.

A configuração da entrada do exemplo acima, através da edição do ficheiro *inputs.conf*, é feita com a introdução das seguintes linhas:

[16] Fonte: Captura de ecrã feita pelo autor.

```
[tcp://12203]
disabled = false
connection_host=dns
source = Windows_Firewall
sourcetype = generic_single_line
```

A primeira linha identifica o porto TCP que está a ser escutado, neste caso é o porto 12203. A segunda linha ativa a entrada. A terceira linha identifica o *host*. Neste caso é usado o nome de domínio do servidor que gera as mensagens. Se a linha fosse *connection_host=ip*, o *host* era identificado pelo respetivo IP, em vez de ser identificado pelo nome de domínio. A terceira linha identifica a fonte das mensagens, cujo nome pode ser qualquer um definido pelo utilizador. Como as mensagens são provenientes da *firewall* do Windows, é esse o nome escolhido. Finalmente, a última linha identifica o tipo de fonte, neste caso, é uma linha de texto, identificada por *generic_single_line*. Outros identificadores possíveis são, entre outros: *syslog*, *_json*, *csv* e *iis*.

5.2.2. Pesquisa e Visualização de Registos

Após o arranque da interface *Web* do *Splunk*, são apresentadas, do lado esquerdo, um conjunto de aplicações. No início, apenas uma está visível, nomeadamente a aplicação *"Search & Reporting"*. No entanto, podem ser instaladas mais aplicações através da opção *Explore Splunk* > *Splunk Apps*. Também podem serem desenvolvidas novas aplicações, usando ferramentas SDK para o *Splunk.*

A visualização, pesquisa e análise de registos no *Splunk* é feita através da interface disponibilizada pela aplicação *"Search & Reporting"*.

A interface apresenta seis componentes principais apresentados na Figura 5.19.

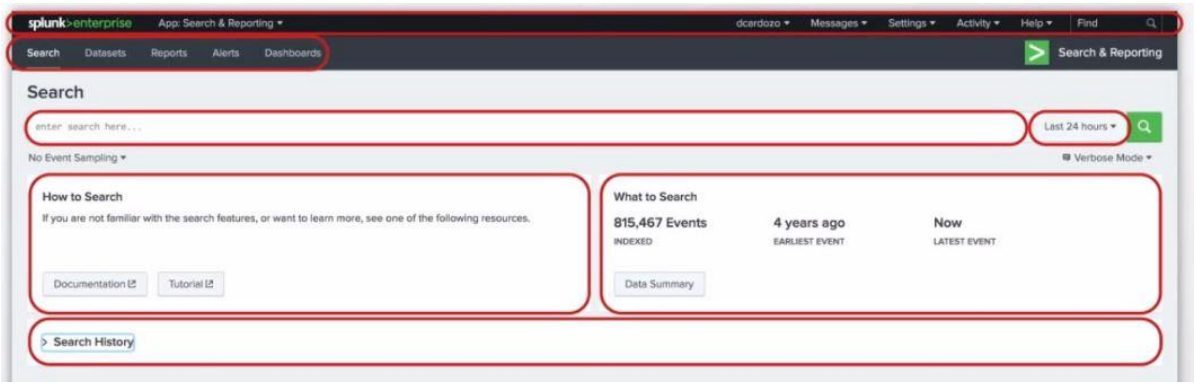


Figura 5.19 – Interface de pesquisa do *Splunk*.

A barra superior aparece em todas as páginas do *Splunk* e pode ser usada para alternar entre aplicações, configurar a conta do utilizador, ver mensagens do servidor, gerar e editar configurações, monitorizar processos de pesquisa e obter ajuda.

A barra existente na segunda linha contém um conjunto de separadores usados para navegar pela aplicação.

A barra "*Search*" permite executar comandos de pesquisa. As pesquisas são feitas através da introdução de um conjunto de parâmetros numa linguagem própria designada *Search Processing Language (SPL)*. Do lado direito, é possível escolher o intervalo de tempo a aplicar à pesquisa. Por baixo desta barra existe o painel "*How to Search*" com acesso a documentação diversa, incluindo um tutorial sobre o processo de pesquisa. O painel "*What to Search*" apresenta um sumário dos dados indexados no servidor. Finalmente, o menu *"Search History"* permite rever e executar expressões de pesquisa introduzidas anteriormente.

Para filtrar, por exemplo, todas as mensagens da *firewall* do Windows correspondentes a ligações recusadas, pode ser usada a expressão de pesquisa **host="artur-tosh" AND source="windows firewall" AND DROP**, como mostra a Figura 5.20.

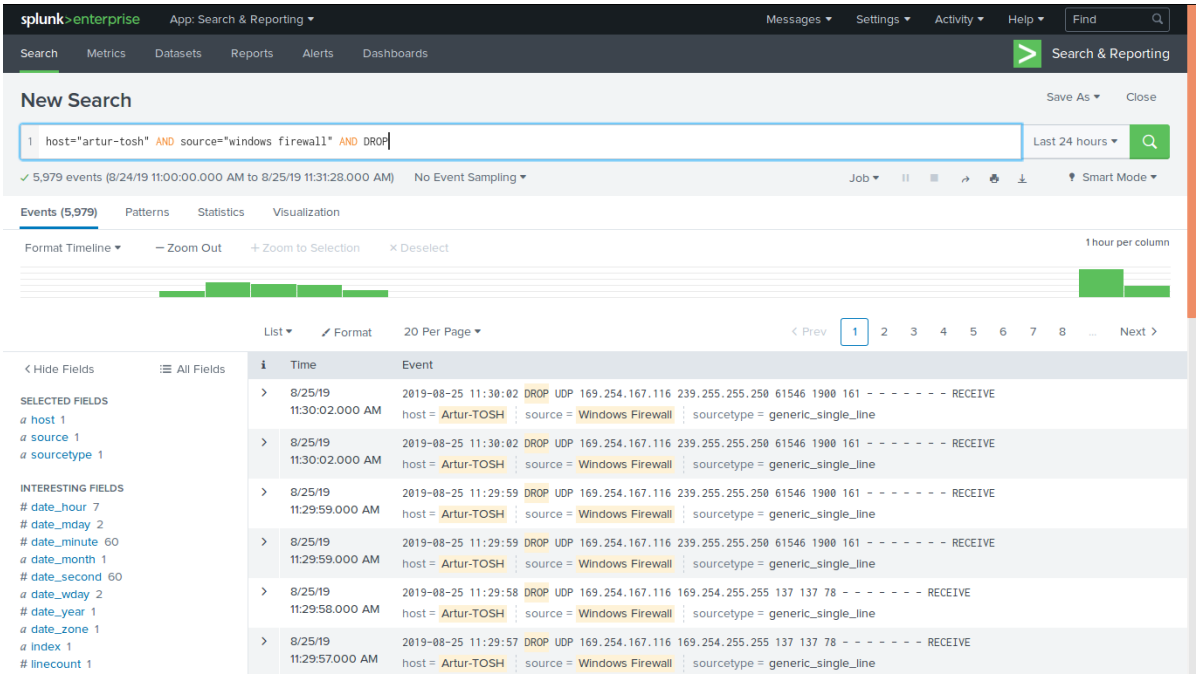


Figura 5.20 – Pesquisa de mensagens no *Splunk*.[17]

[17] Fonte: Captura de ecrã feita pelo autor.

Após a expressão de pesquisa ser executada, surge uma nova página com a apresentação dos resultados. A interface apresenta agora o menu "*Save As*", separadores com informação sobre o resultado da pesquisa, botões de ação a partir dos quais é possível imprimir o resultado ou exportá-lo para um ficheiro *csv* e um histograma com a linha de tempo.

As mensagens filtradas são apresentadas numa lista de eventos no lado direito e os campos extraídos são apresentados numa lista de campos no lado esquerdo.

5.2.3. Extração de Campos das Mensagens

A extração de campos das mensagens pode ser feita através da interface da aplicação "*Search & Reporting*" ou através da edição do ficheiro *transforms.conf*.

A configuração de extratores através da edição do ficheiro *transforms.conf* é explicada mais à frente, quando for abordado o tema da pseudonimização de dados na fase de ingestão.

Na aplicação, a extração de campos é feita a partir da página de apresentação de resultados de uma pesquisa. Para isso, abre-se uma mensagem, à qual se pretende extrair um campo, carregando no símbolo ">", existente do lado esquerdo. Aparece então uma janela com informação detalhada sobre a mensagem, como mostra a Figura 5.21. Em seguida, carrega-se na opção *Event Actions* > *Extract Fields* que dá acesso à seleção do método de extração.

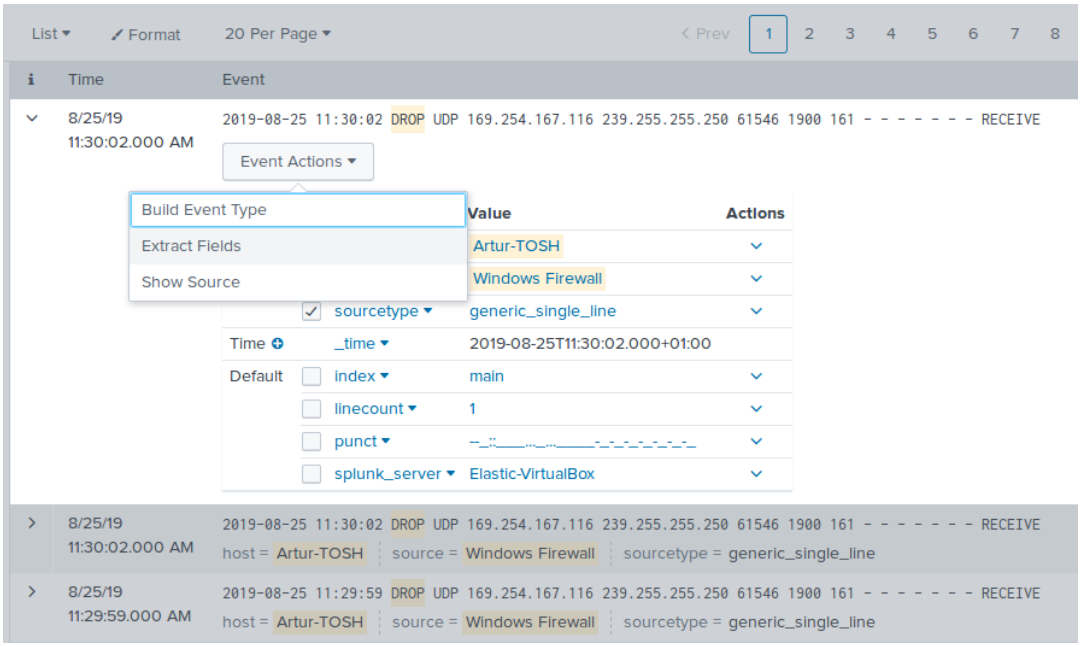


Figura 5.21 – Acesso à funcionalidade de extração de campos da aplicação *Search & Reporting*.[18]

[18] Fonte: Captura de ecrã feita pelo autor.

Existem dois métodos possíveis para se proceder à extração: através de delimitadores ou através de expressões regulares.

A extração através de delimitadores é feita escolhendo o delimitador adequado. Para a extração das mensagens da *firewall* do Windows, por exemplo, usa-se o caracter espaço " ". Outros delimitadores possíveis são a vírgula, a tabulação e a barra vertical, entre outros. A interface permite nomear os campos extraídos e pré-visualizar o resultado da extração.

A extração através de uma expressão regular é igualmente simples: seleciona-se a sequência de caracteres correspondente ao campo, atribui-se um nome e carrega-se na opção *Add Extraction*. A interface permite pré-visualizar o resultado da extração e mostrar a expressão regular que foi usada através da opção "*Show Regular Expression*". As extrações podem ser reconfiguradas através da opção *Settings > Fields > Field extractions*.

5.2.4. Pseudonimização na Fase de Apresentação

A linguagem de pesquisa do *Splunk, Search Processing Language (SPL)*, é constituída por cinco componentes: termos de pesquisa, comandos, funções, argumentos e cláusulas. Os termos de pesquisa servem para filtrar as mensagens. Os comandos servem para criar gráficos, calcular estatísticas e formatar a apresentação dos resultados das pesquisas. As funções são usadas para transformar os dados. Os argumentos são as variáveis aplicadas às funções. Finalmente, as cláusulas são usadas para definir e agrupar os resultados das funções.

Numa expressão de pesquisa, são aplicados comandos, separados por barras verticais. O primeiro comando é denominado por "comando de pesquisa" e está sempre presente. A cada um dos comandos seguintes é aplicado ao resultado do comando anterior.

Por exemplo, o código seguinte usa vários comandos de uma expressão de pesquisa para pseudonimizar os campos *src_ip* e *dst_ip* de uma mensagem proveniente da Firewall do Windows:

```
host="Artur-TOSH" AND source="windows firewall"
| eval h_src_ip = sha256(src_ip)
| rename h_src_ip as "src_ip"
| eval h_dst_ip = sha256(dst_ip)
| rename h_dst_ip as "dst_ip"
| eval _raw=date+" "+time+" "+action+" "+protocol+" src_ip"+" dst_ip
"+src_port+"_" +dst_port+" "+size+" "+tcpflags+" "+tcpsyn+" "+tcpakn+"
"+tcpwin+" "+icmptype+"_" +icmpcode+" "+info+" "+path
```

A primeira linha corresponde ao comando de pesquisa e filtra as mensagens provenientes da *firewall* do Windows. As linhas seguintes, substituem os valores IP dos campos s*rc_ip* de *dst_ip* pelos respetivos resumos resultantes da função de *hash* SHA-256. A última linha transforma a mensagem do registo por uma idêntica à original, mas com os valores dos campos pseudonimizados substituídos pelos nomes dos campos respetivos. O resultado da expressão é o apresentado na Figura 5.22:

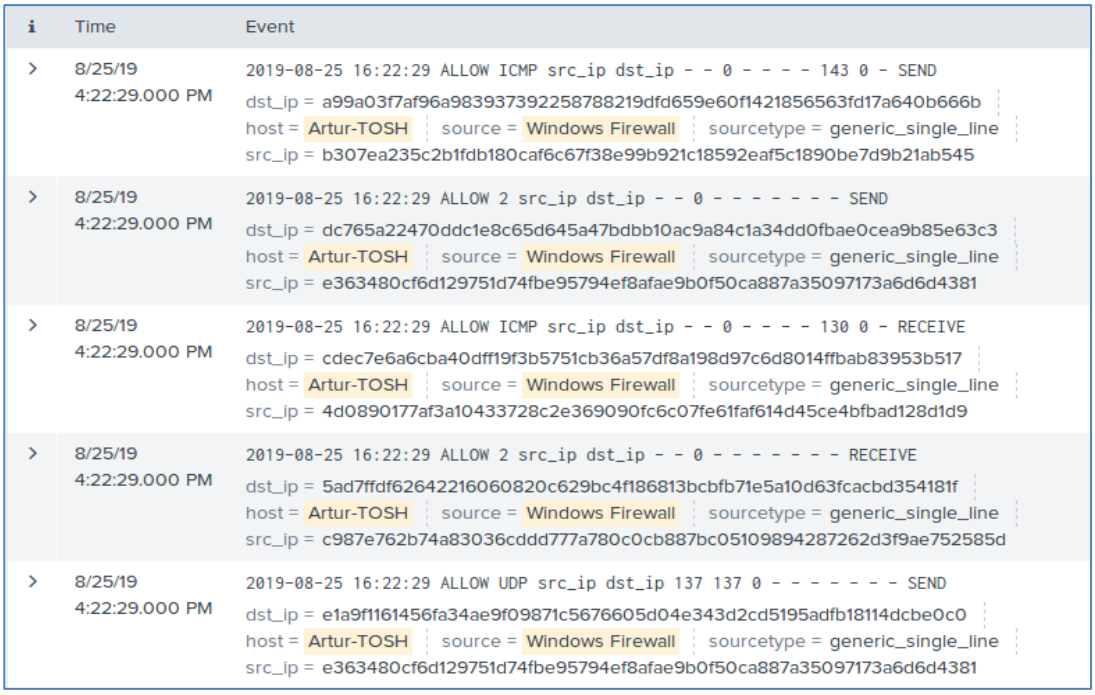


Figura 5.22 – Pseudonimização na fase de apresentação, usando uma expressão de pesquisa.[19]

Pode verificar-se que a primeira linha das mensagens esconde o valor dos IPs e que os campos *src_ip* e *dst_ip* apresentam os resumos respetivos, resultantes da função de *hash*.

A pseudonimização de dados na fase de apresentação, usando expressões de pesquisa pré-definidas, pode ser uma solução para o problema da pseudonimização de dados pessoais em mensagens de registo. Para isso, é necessário criar um perfil limitado a expressões de pesquisa pré-definidas. Em alternativa, pode ser configurado um perfil com acesso exclusivo a painéis de monitorização (*dashboards*) que apresentam, periodicamente ou em tempo real, resultados de pesquisas programadas.

Em qualquer dos casos, as soluções de pseudonimização na fase de apresentação implicam uma limitação para o utilizador que fica limitado em termos de opções de pesquisa.

[19] Fonte: Captura de ecrã feita pelo autor.

5.2.5. Pseudonimização na Fase de Ingestão

Outra possibilidade para a pseudonimização, é a transformação de registos na fase de ingestão. A grande diferença em relação ao método apresentado anteriormente é que os registos são indexados e arquivados já com os valores identificativos pseudonimizados.

Para que seja possível pesquisar e analisar dados identificativos, é necessário que se faça, em paralelo, uma duplicação dos índices, em que um deles mantém os valores originais. Em alternativa, pode ser criada uma tabela de correspondência entre as identidades e os respetivos resumos. Em qualquer dos casos, é necessário que o acesso aos dados seja restrito e que sejam cumpridos os requisitos de segurança da informação impostos pelo regulamento.

A transformação dos registos na fase de ingestão é feita através da configuração de quatro ficheiros: *inputs.conf*, *props.conf*, *transforms.conf* e *fields.conf*. Todos estes ficheiros estão localizados na pasta *splunk/etc/system/local*. A configuração destes ficheiros, usada neste trabalho, pode ser consultada no Anexo F – Configuração da *pipeline* do *Splunk*.

Em seguida, é explicado como é feita a configuração destes ficheiros para pseudonimizar os campos *src_ip* e *dst_ip* dos ficheiros de registo da *firewall* do Windows.

Como é explicado acima, a configuração da entrada dos registos da *firewall* do Windows é feita através da edição do ficheiro *inputs.conf*, adicionando o seguinte código:

```
[tcp://12203]
disabled = false
connection_host=dns
source = Windows_Firewall
sourcetype =  generic_single_line
```

No ficheiro *props.conf* são declaradas as etapas de transformação efetuadas na fase de ingestão:

props.conf

```
[source::Windows_Firewall]
TRANSFORMS-step1 = raw_backup
TRANSFORMS-step2 = src_ip_extration
TRANSFORMS-step3 = src_ip_pseudonymization
TRANSFORMS-step4 = raw_restore1
TRANSFORMS-step5 = dst_ip_extraction
TRANSFORMS-step6 = dst_ip_pseudonymization
TRANSFORMS-step7 = raw_restore2
TRANSFORMS-step8 = raw_transformation
```

No ficheiro *transforms.conf* são definidas as operações efetuadas em cada uma das etapas de transformação:

transforms.conf

```
[raw_backup]
INGEST_EVAL = _rawtmp=_raw

[src_ip_extraction]
REGEX=^(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)
FORMAT = $5
DEST_KEY = _raw

[src_ip_pseudonymization]
INGEST_EVAL = src_ip=sha256(_raw)

[raw_restore1]
INGEST_EVAL = _raw=_rawtmp

[dst_ip_extraction]
REGEX=^(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)
FORMAT = $6
DEST_KEY = _raw

[dst_ip_pseudonymization]
INGEST_EVAL = dst_ip=sha256(_raw)

[raw_restore2]
INGEST_EVAL = _raw=_rawtmp

[raw_transformation]
REGEX=^(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)
FORMAT = $1 $2 $3 $4 src-ip dst-ip $7 $8 $9 $10 $11 $12 $13 $14 $15 $16 $17
DEST_KEY = _raw
```

A transformação *INGEST_EVAL* coloca o resultado de uma função num determinado campo. As funções usadas pela transformação apenas podem ser aplicadas aos conteúdos dos campos internos: *_raw*, *_time*, *_indextime*, *_cd* e *_bkt*.

O campo usado durante as etapas de transformação é o campo *_raw* que contém a mensagem de entrada. Assim, no primeiro passo *[raw_backup]*, guarda-se o conteúdo do campo *_raw*, que é o campo que contém a mensagem, num novo campo auxiliar *_rawtmp*.

Em seguida, no passo *[src_ip_extraction]*, é feita a extração do valor correspondente campo *src_ip* e colocado no campo *_raw*, para que possa ser processado no passo seguinte.

A função *REGEX* aplica uma expressão regular ao campo *_raw*, usando o espaço como delimitador. Os valores dos campos extraídos são colocados nas variáveis *$1* a *$17*. A função

FORMAT formata a entrada com o valor da variável $5 (conteúdo de *src_ip*) e a função *DEST_KEY* coloca o resultado no campo *_raw*.

O passo *[src_ip_pseudonymization]* calcula o resultado da função de *hash sha-256* aplicada ao conteúdo do campo *_raw* e coloca o resultado no campo *src_ip*. Por outras palavras, o conteúdo original do campo *src_ip* é substituído pelo respetivo *hash*.

O passo *[raw_restore1]* reestabelece o valor original de *_raw* para que se possa repetir o processo e pseudonimizar o campo *dst_ip*, através dos passos *[dst_ip_extraction]* e *[dst_ip_pseudonymisation]*.

O passo *[raw_restore2]* reestabelece o valor original de *_raw* para que no passo *[raw_transformation]* se possa gerar a mensagem *_raw* original, com as identidades substituídas pelos nomes dos campos correspondentes.

Para que os novos campos, *src_ip* e *dst_ip* possam ser indexados, têm de ser declarados no ficheiro *fields.conf*:

```
1  [src_ip]
2  INDEXED = True
3
4  [dst_ip]
5  INDEXED = True
```

Agora, na aplicação "*Search & Reporting*", uma pesquisa com a expressão **host="Artur-TOSH" AND source="windows firewall"**, é apresentado um resultado semelhante ao da Figura 5.22, onde os valores dos campos *src_ip* e *dst_ip* estão pseudonimizados e o campo *_raw* apresenta a mensagem completa com as identidades mascaradas.

O método de pseudonimização na fase de ingestão pode tornar o processo de ingestão mais demorado, uma vez que as mensagens têm de ser processadas antes da indexação. Isto significa um maior atraso na disponibilização das mensagens. No entanto, torna-se muito mais eficiente na pesquisa, uma vez os dados já foram transformados na fase de ingestão.

O processamento na fase de ingestão pode ser acelerado usando módulos dedicados denominadas "entradas modulares" que permitem customizar o processamento antes da fase

de indexação. A customização é feita através da utilização de rotinas, normalmente em *Java* ou em *Pyton*, usadas por aplicações do *Splunk*.

Para além do processamento se tornar mais rápido, a utilização de entradas modulares permite o desenvolvimento de módulos de tratamento de dados mais complexos que integrem, por exemplo, a função HMAC ou funções de cifragem simétrica.

5.2.6. Criação de perfis de utilizador no *Splunk*

A criação de perfis de utilizador no *Splunk* apenas está disponível na versão *Enterprise*. No entanto, é possível usar a licença *Trial* que está disponível até 30 dias a partir da data de instalação.

O novo perfil pode ser configurado através da opção *Settings* > *Access controls* > *Roles* > *+Add new*, a partir da qual é possível selecionar os índices que estão acessíveis, entre um conjunto variado de atributos de configuração:

- Restrições de pesquisa;
- Herança (usar atributos definidos em outros perfis);
- Permissões;
- Índices pesquisados por definição;
- Índices acessíveis.

A Figura 5.23 mostra uma parte da página de configuração, onde é possível ver o nome atribuído ao perfil e a atribuição de índices permitidos. Neste caso, os utilizadores deste perfil apenas têm acesso ao índice *main*, selecionado no campo *Selected search indexes*.

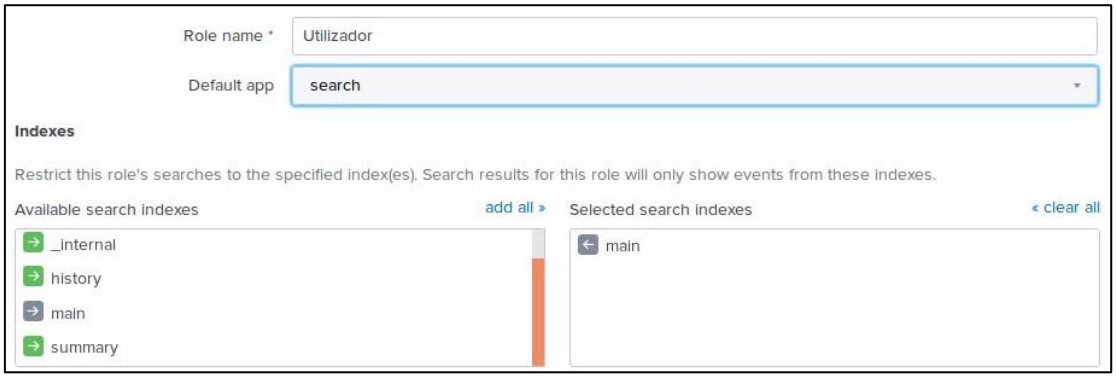


Figura 5.23 – Página de criação de um novo perfil no *Splunk*.[20]

[20] Fonte: Captura de ecrã feita pelo autor.

A criação de novos utilizadores é feita a partir da opção *Settings* > *Access controls* > *Users* > *+Add new*. Por exemplo, o utilizador *varanda* associado ao perfil *Utilizador* é adicionado como mostra a Figura 5.24:

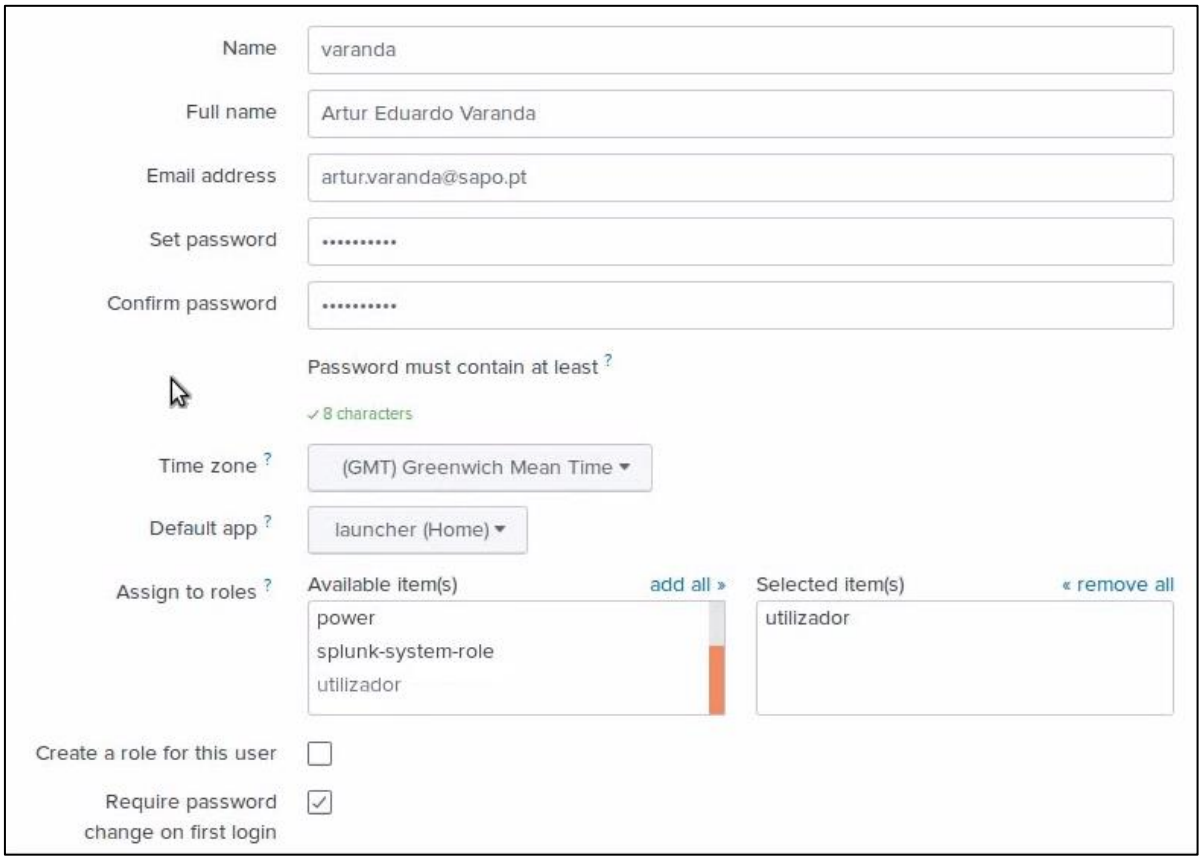


Figura 5.24 – Página de criação de um novo utilizador no *Splunk*.

Os perfis atribuídos ao utilizador *varanda* estão selecionados na secção *Assign to roles*, no campo *Selected item(s)*. Neste caso, apenas está atribuído o perfil *utilizador*.

5.3.*ELK Stack*

A *ELK Stack*, cujo processo de instalação para este cenário pode ser visto no Anexo B – Instalação da *ELK Stack*, é um conjunto de módulos *open source*, constituídos essencialmente pelo *Elasticsearch*, pelo *Logstash* e pelo *Kibana*. O prefixo "ELK" corresponde, precisamente, às iniciais dos três módulos. A interligação dos módulos pode ser vista na Figura 5.25 e é explicada em seguida.

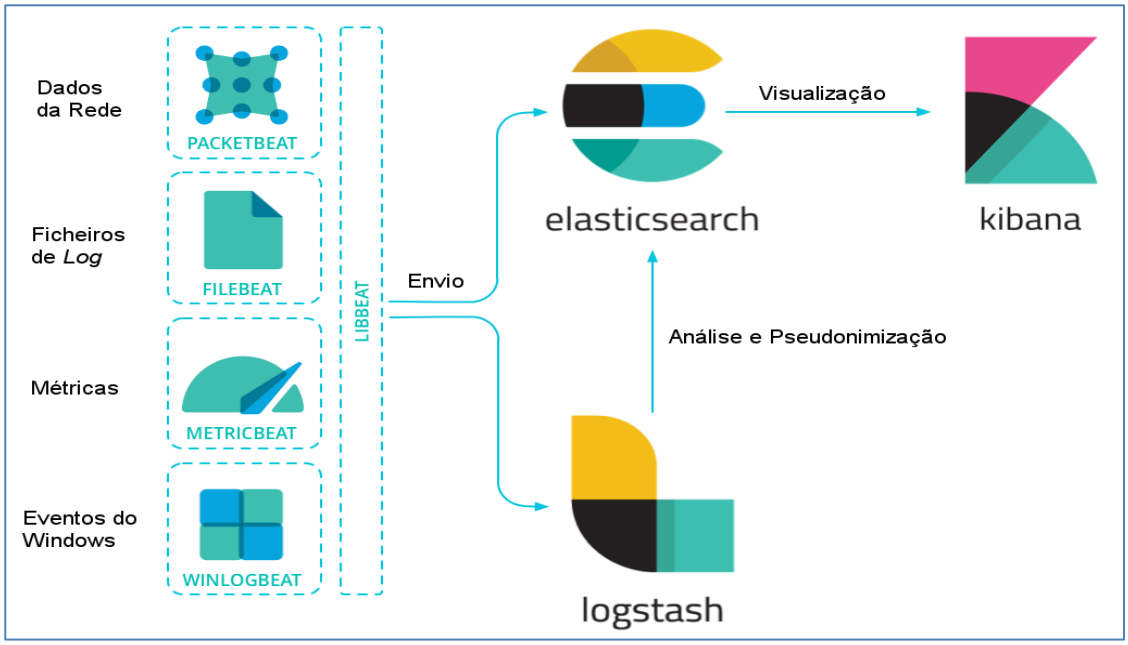


Figura 5.25 – Interligação dos módulos da *ELK Stack*.

O módulo central, o *Elasticsearch*, permite executar análises complexas em tempo real e funciona como um indexador, um motor de pesquisa e um analisador de eventos.

O *Kibana* é a interface *Web* da *ELK Stack*. Este módulo liga-se ao *Elasticsearch* e permite criar painéis de monitorização (*dashboards*) em tempo real, bem como configurar os recursos e fazer a administração da *Stack* como um todo.

O *Logstash* é o módulo responsável por receber as mensagens dos agentes de recolha e encaminhamento (como o *NXLog*, o *Syslog* ou o *Filebeats*), extrair os respetivos campos, transformá-los e encaminhá-los para o *Elasticsearch* [25].

Os *Beats* são módulos de recolha de mensagens de várias fontes, desde ficheiros (*Filebeat*), elementos de rede (*Packetbeat*), métricas (*Metricbeat*) ou eventos do Windows (*Winlogbeat*),

entre outros. Estes módulos podem reencaminhar as mensagens para o *Logstash* para serem transformadas, ou enviá-las diretamente para o *Elasticsearch*, sem serem processadas.

A arquitetura do cenário de testes efetuados com a *ELK Stack* é idêntica às arquiteturas usadas nos cenários anteriores (ver Figura 5.4). A grande diferença relativamente aos cenários anteriores é que as mensagens enviadas pelo *NXLog* para o *Logstash* usam o formato JSON.

5.3.1. Configuração de Entradas

A *pipeline* do *Logstash* tem dois blocos obrigatórios que são o bloco de *Entradas* e o bloco de *Saídas*; e um bloco opcional, que é o bloco de *Filtros*. As entradas recebem as mensagens de diversas fontes (ex: *Filebeats* ou portos TCP e UDP), os filtros transformam as mensagens de acordo com um conjunto de regras e as saídas enviam as mensagens para um determinado destino, normalmente um índice do *Elasticsearch* (Figura 5.26).

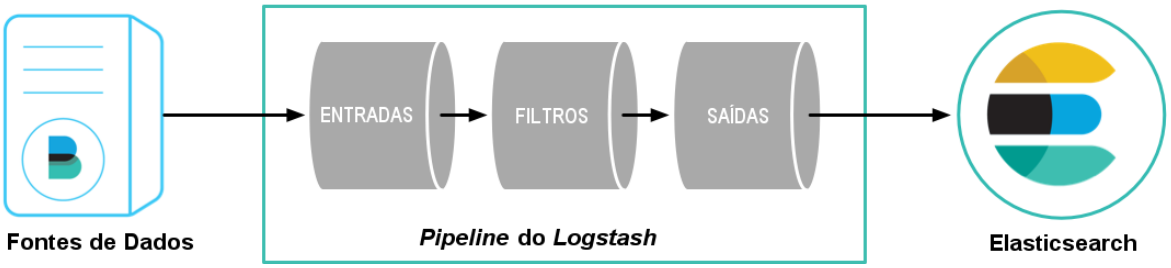


Figura 5.26 – *Pipeline* do *Logstash*.[21]

A configuração destes blocos é feita através de edição do ficheiro *logstash.conf*, localizado na pasta de instalação do *Logsatsh*, cuja estrutura é do tipo:

```
input {
  ...
}

filter {
  ...
}

output {
  ...
}
```

[21] Fonte: Stashing Your First Event - Logstash Reference [7.3]

Configuração de Entradas provenientes do Windows

Como nos cenários anteriores, as mensagens provenientes do Windows são enviadas para o servidor via *NXLog*. Neste caso, o *NXLog* pode ser configurado para enviar as mensagens do Windows diretamente para o *Elasticsearch* substituindo o *Logstash*; ou pode ser configurado para as encaminhar para o *Logstash*, funcionando como um coletor.

Caso se pretenda enviar as mensagens diretamente para o *Elasticsearch*, o *NXLog* necessita usar um módulo de saída adicional (*om_elasticsearch*) que apenas está disponível na versão comercial *NXLog Enterprise Edition*. No entanto, como no nosso cenário se pretende transformar as mensagens usando filtros de pseudonimização, convém fazê-las passar pelo *Logstash*.

A configuração das saídas do *NXLog* para o *Logstash* é feita como mostra o exemplo seguinte:

```
Output out>
   Module       om_tcp
   Host         %SERVERIP%
   Port         12204
   Exec         $ShortMessage = $raw_event;
   Exec         to_json();
</Output>
```

A função *to_json()* transforma a saída no formato JSON e requer a utilização da extensão *xm_json*, disponível na versão gratuita *NXLog Community Edition* [26].

No *Logstash*, a entrada e respetivo encaminhamento são configurados no ficheiro *logstash.conf* como mostra o exemplo seguinte:

```
input {
   tcp {
      type => "win_apache"
      codec => json_lines { charset => CP1252 }
      port => "12204"
      tags => [ "tcpjson" ]
   }

filter { }

output {
   elasticsearch {
      hosts => localhost
      index => "logstash-main"
   }
      stdout { codec => rubydebug }
}
```

No bloco *input*, é acrescentada uma entrada do tipo TCP, escutada no porto 12204, para onde são enviadas as mensagens de acesso do servidor *Web* da Apache, instalado no Windows.

O bloco *filter* ainda não contém qualquer filtro. No entanto, é necessário acrescentar vários filtros para configurar a extração de campos das mensagens e para efetuar a pseudonimização dos dados neles contidos, como é visto mais adiante.

No bloco *output*, é acrescentada uma saída para o *Elasticsearch*, especificando o *host* (neste caso é o *localhost*) e o índice de destino, *logstash-main*.

Na última linha, é configurada a saída de eventos do *Logstash* para o terminal, usando o *codec rubydebug* (*Ruby Awesome Print Library*). Esta linha é necessária para fazer o depuramento do código de configuração do *Logstash* e para acompanhar, no terminal, o encaminhamento das mensagens em formato JSON para o *Elasticsearch*.

Configuração de Entradas do *Syslog*

A configuração das entradas provenientes do *Syslog* pode ser feita de duas formas.

Uma possibilidade é escutar o porto UDP 1514, como é feito nos cenários anteriores. A entrada das mensagens provenientes do *Syslog* através do porto UDP 1514 é configurada no *Logstash*, acrescentando as seguintes linhas ao bloco *input*:

```
udp {
    type => "syslog_udp"
    codec => plain
    port => "1514"
}
```

Como as mensagens provenientes do *Syslog* são linhas de texto simples, é usado o *codec plain*.

A outra possibilidade, usada neste cenário, é a utilização do módulo coletor *Filebeat* para recolher as mensagens dos *logs* do *Linux*, normalmente localizados na pasta */var/log*.

A configuração do *Filebeat* para recolher as entradas dos *logs* é feita através da edição do ficheiro *filebeat.yml*:

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/*.log
```

O *Filebeat* suporta uma grande variedade de saídas. Se não for necessário efetuar qualquer processamento, as mensagens podem ser enviadas diretamente para o *Elasticsearch.* Caso contrário, terão de ser enviadas para o *Logstash*.

Para se enviar a saída diretamente para o *Elasticsearch*, configura-se o ficheiro *filebeat.yml* com a localização do *host* onde o *Elasticsearch* está instalado, neste caso é o *localhost*:

```
output.elasticsearch:
  hosts: ["localhost:9200"]
```

Como se pretende processar as mensagens provenientes do *Syslog*, é necessário configurar a saída do *Filebeat* para o *Logstash*. Para isso, é necessário desabilitar a saída para o *Elasticsearch*, introduzindo caracteres de comentário # no início de cada uma das linhas de configuração apresentadas acima e habilitar a saída para o *Logstash*:

```
output.logstash:
  hosts: ["localhost:5044"]
```

O porto 5044 é o que está configurado, no *Logstash*, para escutar as entradas provenientes das ligações de módulos *Beats*, entre os quais o *Filebeat*.

Configuração de Múltiplas Entradas

Apesar de ser necessário processar diversos tipos de entradas de acordo com regras muito específicas, apenas se pode usar uma *pipeline* no *Logstash*. Cada mensagem pode, então, ser identificada com um tipo único, através do campo *type*.

No bloco *filter*, cada conjunto de filtros a aplicar a cada tipo de mensagens pode ser definido usando operadores de controlo de fluxo.

De igual forma, no bloco *output*, podem ser usados operadores de controlo de fluxo para remeter cada tipo de mensagens para o respetivo destino.

O algoritmo usado para identificar, processar e encaminhar os diferentes tipos de mensagens no *Logstash* é o apresentado no exemplo seguinte:

```
input {

   tcp {
      type => "Win_Apache"
      codec => json_lines {charset => CP1252}
      port => "12204"
      tags => ["tcpjson"]
   }

   beats {
      type => "Syslog_filebeat"
      port => "5044"
   }
}

filter {

    if [type] == "Win_Apache" {
            # Processamento .......
    }

    if [type] == "Syslog_filebeat" {
            # Processamento .......
    }
}

output {

    if [type] == "Win_Apache" {
            # output to elasticsearch index_Windows
    }

    if [type] == "Syslog_filebeat" {
            # output to elasticsearch index_Syslog
    }
}
```

Neste caso, apenas são identificados, processados e encaminhados dois tipos diferentes de mensagens, provenientes do servidor *Web* da Apache do Windows e do *Syslog* da máquina virtual local, embora este método possa ser aplicado a mais tipos de entradas.

5.3.2. Pesquisa e Visualização de Registos

A visualização e pesquisa de registos, neste cenário, é feita através do *Kibana*, a interface *Web* da *ELK Stack*. Antes se abrir a interface, é necessário iniciar os serviços do *Elasticsearch* e do *Kibana*:

$ elasticsearch-6.3.2/bin/elasticsearch

$ kibana-6.3.2-linux-x86_64/bin/kibana

Para ativar a receção de mensagens é também necessário iniciar o serviço do *Logstash*:

$ logstash-6.3.2/bin/logstash -f logstash.conf

Finalmente, para ativar a coleção de registos do *Syslog*, é necessário ativar o serviço do *Filebeat*:

$ sudo service filebeat start

Uma vez iniciados os serviços, é possível abrir a interface *Web* no navegador local, através da ligação **http://localhost:5601**, cuja janela inicial é semelhante à apresentada na Figura 5.27.

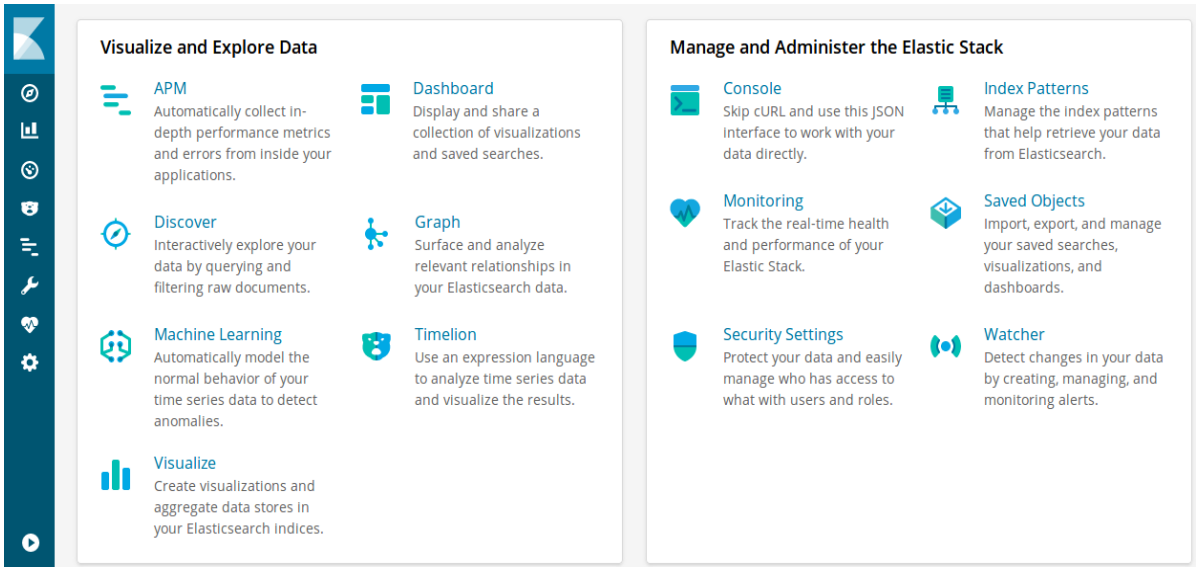


Figura 5.27 – Janela inicial do *Kibana*.[22]

Das várias opções disponibilizadas pela interface, apenas são usadas, neste trabalho, as opções *Discover*, *Dev Tools* e *Management*, correspondentes ao primeiro, antepenúltimo e último ícones da barra azul existente do lado direito.

Management

Antes de se poder pesquisar e visualizar dados, é necessário criar padrões associados aos índices. Para isso, através da opção *Management -> Index Patterns -> Create index pattern*, definem-se padrões correspondentes à estrutura das mensagens existentes em cada um. Concluído este processo, é possível efetuar pesquisas através da opção *Discover*.

[22] Fonte: Captura de ecrã feita pelo autor.

Após serem criados os padrões, também é possível ocultar determinados campos das mensagens através do separador *Source filters*, acrescentando-os à lista de filtros.

Discover

A opção *Discover* abre uma nova página, semelhante à da Figura 5.28, a partir da qual é possível pesquisar os dados existentes nos vários índices.

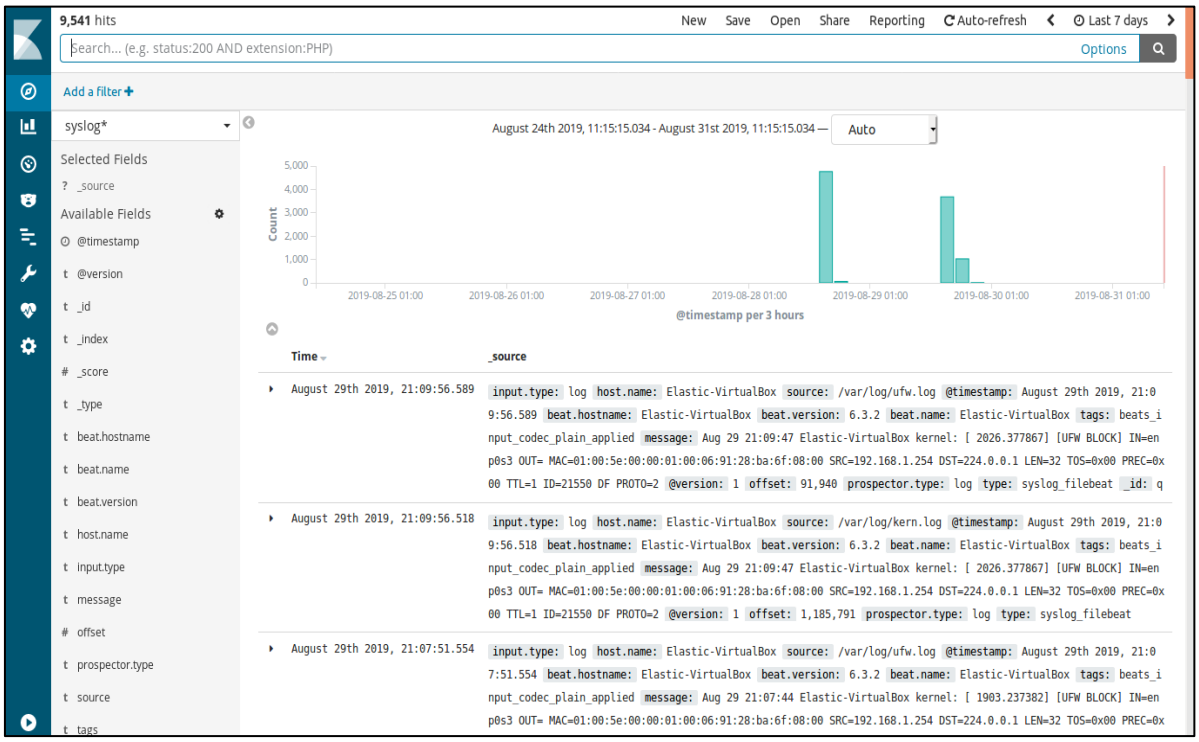


Figura 5.28 – Janela de pesquisa e visualização do *Kibana*.[23]

A interface de pesquisa e visualização do *Kibana* é semelhante às interfaces usadas no *Graylog* e no *Splunk*.

A barra superior apresenta um conjunto de opções que permitem gravar, ler, partilhar e apresentar pesquisas efetuadas. Do lado direito, é possível escolher o intervalo de tempo a aplicar à pesquisa.

Na segunda linha existe uma barra que permite introduzir expressões de pesquisa baseadas em *Apache Lucene*, idênticas às usadas para realizar as pesquisas no *Graylog*. Este facto não é de estranhar, dado que as duas soluções usam o mesmo motor de pesquisa, o *Elasticsearch*.

[23] Fonte: Captura de ecrã feita pelo autor.

Após a expressão de pesquisa ser executada, surge uma nova página com a apresentação dos resultados. As mensagens filtradas são apresentadas numa lista de eventos no lado direito e os campos respetivos são apresentados numa lista de campos no lado esquerdo.

Por cima da lista de campos, é possível selecionar os índices ao quais se pretende aplicar a pesquisa. No exemplo acima, são todos os índices definidos pelo padrão *syslog**.

Por cima da lista de eventos, é possível visualizar um histograma das mensagens, com a respetiva linha de tempo.

Dev Tools

A opção *Dev Tools* (*Development Tools*) abre uma nova página que contém ferramentas de desenvolvimento e que podem ser usadas para interagir com os dados do *Kibana*.

A consola permite interagir com a REST API do *Elasticsearch*, a partir da qual é possível, por exemplo, eliminar índices, usando o comando:

DELETE índice_a_apagar

Pode ser apresentada uma visualização detalhada dos índices existentes, usando o comando:

GET /_cat/indices?v

Após a execução do comando, é apresentada uma tabela semelhante à da Figura 5.29:

```
health status index          uuid                   pri rep docs.count docs.deleted store.size pri.store.size
green  open   .kibana        prLW7Ji0SeGXlsTR7vyAJA   1   0        157            6    185.9kb        185.9kb
yellow open   syslog         x-xlgDb1RWSj6AT0xXG6AA   5   1       9541            0      3.6mb          3.6mb
yellow open   logstash-main  Zr5X_iAaRbOyFEPusA_Ihw   5   1         25            0    289.7kb        289.7kb
yellow open   logstash-pairs VfHf10RkQOOW0IFNs0N-bQ   5   1          5            7     27.3kb         27.3kb
```

Figura 5.29 – Lista de índices do *Elasticsearch*.[24]

5.3.3. Extração de Campos

A extração de campos é feita através da utilização do filtro **grok** na *pipeline* do *Logstash*. Por exemplo, a extração de campos das mensagens provenientes do servidor *Web* da Apache é feita usando o filtro juntamente com a seguinte expressão regular:

[24] Fonte: Captura de ecrã feita pelo autor.

```
1  grok {match => ["ShortMessage", "%{IPORHOST:c_ip} %{HTTPDUSER:ident}
   %{USER:auth} \[%{HTTPDATE:timestamp}\] \"%{WORD:cs_method}
   %{NOTSPACE:request_page} HTTP/%{NUMBER:http_version}\"
   %{NUMBER:server_response} (?:%{NUMBER:bytes}|-)"]}
2  if "_grokparsefailure" in [tags] {
3     drop { }
4  }
```

Em caso de falha na extração de uma mensagem, o filtro **grok** acrescenta automaticamente uma etiqueta com o identificador *"_grokparsefailure"*. Caso se pretendam descartar estas mensagens da *pipeline* do *Logstash*, pode ser usado o filtro **drop**, conforme é definido nas últimas três linhas.

Esta purga de mensagens é especialmente útil quando se pretende pseudonimizar alguma informação. Uma mensagem mal processada pelo filtro **grok** poderá encaminhar para o indexador dados identificativos no campo *_raw*, uma vez que os filtros de pseudonimização apenas são aplicados aos campos extraídos.

5.3.4. Pseudonimização na *ELK Stack*

Neste cenário, a pseudonimização é efetuada na *pipeline* do *Logstash* na fase de ingestão, usando as funções de *hash* que integram o filtro **fingerprint**.

A função usada neste cenário é a função HMAC-SHA256, mas existem alternativas, como SHA1, SHA384, SHA512, MD5. Quando é incluída uma chave no filtro, é usada a função HMAC associada a uma das funções anteriores. As funções MD5 e SHA1 não são recomendadas, pelos motivos explicados na secção 4.1.4.

O filtro **cipher** permite usar funções de cifra, em alternativa às funções de resumo e integra todos os algoritmos de cifra fornecidas pela livraria *Ruby OpenSSL*.

As mensagens pseudonimizadas são enviadas para um índice para poderem ser pesquisadas e visualizadas.

Ao mesmo tempo, são criadas mensagens de reidentificação, compostas pelos pares de campos correspondentes às identidades e aos resumos correspondentes, as quais são enviadas para um índice separado.

Este índice funciona como um arquivo de identidades e apenas está acessível aos utilizadores com um perfil privilegiado, para permitir a reidentificação nos termos estipulados pelo regulamento de proteção de dados.

A Figura 5.30 mostra o processo de geração das mensagens pseudonimizadas e das mensagens de reidentificação, a partir da mensagem original.

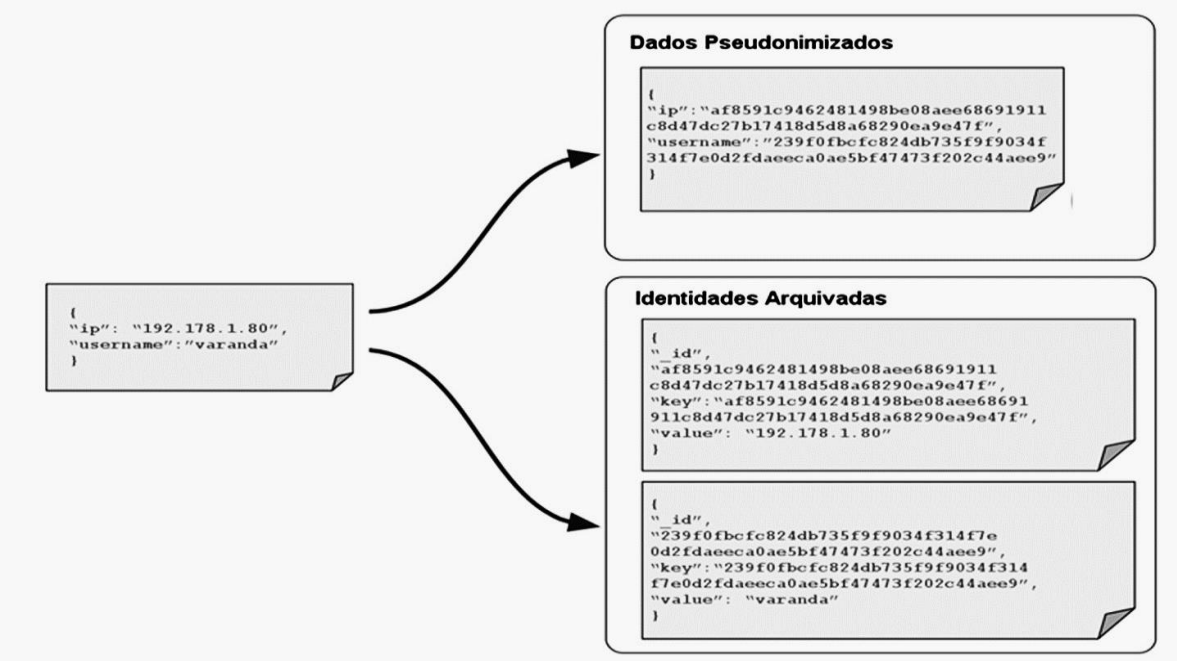


Figura 5.30 – Geração das mensagens pseudonimizadas e das mensagens de reidentificação.

A geração das mensagens pseudonimizadas é feita após a extração dos campos da mensagem original, substituindo os valores dos campos com dados identificativos pelos restivos valores de resumo. A mensagem original completa, guardada no campo *ShortMessage* também é substituída por uma mensagem idêntica, mas com os valores identificativos substituídos pelos nomes dos campos correspondentes.

A geração das mensagens de reidentificação é feita usando réplicas da mensagem pseudonimizada, através do filtro **clone**. Os campos com os dados identificativos são mantidos e são acrescentados campos com os resumos correspondentes. Todos os outros campos são descartados usando o filtro **prune**.

Para evitar duplicação de mensagens de reidentificação no índice de arquivo de identidades, estas são etiquetadas com um *_id* cujo valor corresponde ao resultado da função de *hash MurmurHash* aplicada ao valor do campo de resumo. A função *MurmurHash* é uma função

de *hash*, baseada em operações de multiplicação e rotação. O nome *Murmur* resulta do encadeamento das operações **MU**ltiplicação > **R**otação > **MU**ltiplicação > **R**otação.

Se a mensagem contiver o mesmo valor de *_id* que uma mensagem indexada anteriormente, isso significa que é composta pelo mesmo par (valor, resumo) e pode substituir, no índice, a mensagem anterior. Desta forma, é garantido que as mensagens de reidentificação não se repetem no índice.

Exemplo – Pseudonimização de registos de acesso do servidor *Web* da Apache

O código seguinte apresenta o processo usado na pseudonimização de mensagens de acesso do servidor *Web* da Apache na *pipeline* do *Logstash*, incluído no ficheiro *logstash.conf*.

As linhas relativas à configuração da entrada já foram explicadas acima, na respetiva secção.

```
1  input {
2     tcp {
3        type => "win_apache"
4        codec => json_lines {charset => CP1252}
5        port => "12204"
6        tags => ["tcpjson"]
7     }
```

Note-se que entrada configurada acima é rotulada com o tipo "win_apache".

As primeiras linhas respeitantes à configuração dos filtros criam cópias das mensagens de entrada rotuladas com o tipo "win_apache_pairs".

```
 9  filter {
10     if [type] == "win_apache" {
11        clone {
12           clones => ["win_apache_pairs"]
13        }
```

As linhas respeitantes à extração de campos também já foram explicadas acima. Neste caso, o campo que se pretende pseudonimizar é o campo *c_ip*, correspondente ao endereço IP do cliente que tentou aceder ao servidor *Web*:

```
15  grok {
       match => ["ShortMessage", "%{IPORHOST:c_ip} %{HTTPDUSER:ident}
16  %{USER:auth} \[%{HTTPDATE:timestamp}\] \"%{WORD:cs_method}
```

```
    %{NOTSPACE:request_page} HTTP/%{NUMBER:http_version}\"
    %{NUMBER:server_response} (?:%{NUMBER:bytes}|-)"]
17  }
18
19  if "_grokparsefailure" in [tags] {
20     drop { }
21  }
22
```

Agora que existem dois tipos de mensagens e os respetivos campos já estão extraídos, o processamento diverge.

As mensagens originais do tipo "win_apache" são pseudonimizadas e as mensagens clonadas do tipo "win_apache_pairs" dão origem a mensagens de reidentificação.

As linhas seguintes correspondem à pseudonimização das mensagens do tipo "win_apache".

```
23  if [type] == "win_apache" {
24   fingerprint {
25     method => "SHA256"
26     key => "HMAC_SECRET_KEY"
27     source => "c_ip"
28     target => "c_ip"
29   }
30
31   mutate {
32     replace => {"ShortMessage" => "c_ip %{ident} %{auth} [%{timestamp}]
    \"%{cs_method}     %{request_page} HTTP/%{http_version}\" %{server_response}
    %{bytes}" }
33          }
34
35  }
```

O filtro **fingerprint** substitui o conteúdo do campo *c_ip* pelo respetivo resumo, resultante da função HMAC SHA256, usando a chave "HMAC_SECRET_KEY".

O filtro **mutate** usa a opção **replace**, para substituir o conteúdo do campo *ShortMessage*, por uma *string* semelhante, mas com a identidade correspondente a campo *c_ip*, substituída pelo nome do campo. Por outras palavras, o endereço IP do cliente, na mensagem completa, é substituído pela *string* "c_ip".

As linhas seguintes geram mensagens de reidentificação a partir das mensagens clonadas correspondentes ao tipo "win_apache_pairs":

```
38  if [type] == "win_apache_pairs" {
39    fingerprint {
40    method => "SHA256"
41    key => "HMAC_SECRET_KEY"
42    source => "c_ip"
43    target => "resumo"
44  }
45
46  mutate {
47    add_field => {"valor" => "%{c_ip}"}
48  }
49
50  prune {
51    interpolate => true
52    whitelist_names => ["type", "valor", "resumo"]
53  }
54
55  fingerprint {
56    source => "resumo"
57    target => "fingerprint_id"
58    method => "MURMUR3"
59  }
60  }
```

O filtro **fingerprint** usado é idêntico ao que foi aplicado às mensagens pseudonimizadas. A única diferença é que o valor original é mantido no campo *c_ip*. Por sua vez, o resumo correspondente é armazenado no novo campo *resumo.*

Como o índice que armazena as mensagens de reidentificação contém valores provenientes de várias fontes para além do servidor *Web* da Apache, convém que o campo com os dados originais tenha um nome genérico em vez de ter o nome *c_ip*. Assim, o novo campo *valor* é criado com o filtro **mutate** associado à opção **add_field**, sendo-lhe atribuído o conteúdo do campo *c_ip*.

Como se pretende que a mensagem de reidentificação apenas contenha os campos *valor* e *resumo*, é usado o filtro **prune** para descartar os campos restantes. O campo *type* é mantido porque é utilizado na configuração da saída.

Finalmente, é usado um segundo filtro **fingerprint** para etiquetar a mensagem com um *fingerprint_id* resultante da função de resumo *MurmurHash* aplicada ao valor contido no campo *resumo*. As mensagens que têm o mesmo *fingerprint_id* correspondem a mensagens de reidentificação idênticas, com os mesmos conteúdos nos campos *valor* e *resumo*.

As saídas são configuradas da forma seguinte:

```
62
63  output {
64     if [type] == "win_apache_pairs" {
65
66        elasticsearch {
67           hosts => localhost
68           index => "logstash-pairs"
69           document_id => "%{fingerprint_id}"
70        }
71
72       stdout {codec => rubydebug}
73     }
74
75     if [type] == "win_apache" {
76        elasticsearch {
77           hosts => localhost
78           index => "logstash-main"
79        }
80
81        stdout {codec => rubydebug}
82     }
83
84  }
```

As mensagens do tipo "win_apache_pairs" são enviadas para o índice *logstash-pairs* e armazenadas na posição correspondente ao conteúdo do campo *fingerprint_id*. É desta forma que se evita a repetição de mensagens de reidentificação no índice.

O conteúdo do índice *logstash-pairs*, apresentado na Figura 5.31, apenas está acessível aos utilizadores com um perfil privilegiado, como é explicado acima.

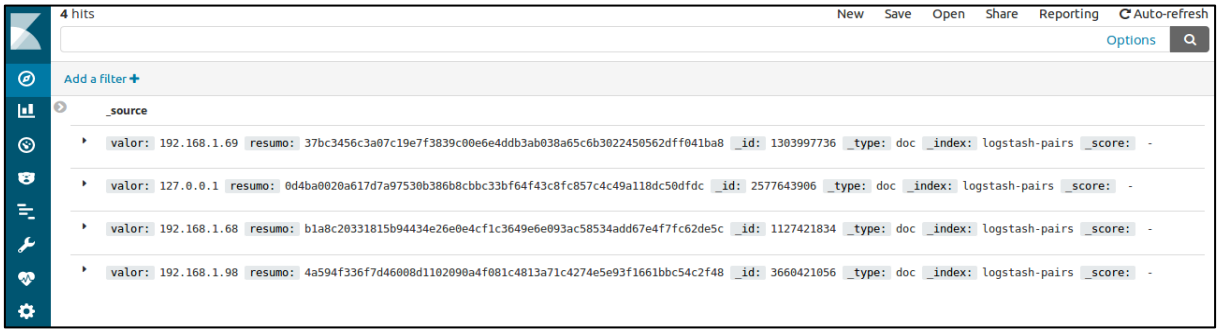


Figura 5.31 – Mensagens de reidentificação constantes no índice *logstash-pairs*,[25]

Por fim, as mensagens do tipo "win_apache" são enviadas para o índice *logstash-main*, cuja amostra do conteúdo é apresentada na Figura 5.32.

[25] Fonte: Captura de ecrã feita pelo autor.

Figura 5.32 – Mensagens de acesso a um servidor *Web* da Apache, devidamente pseudonimizadas.[26]

Note-se que o campo *c_ip* contém os resumos resultantes da função HAMC SHA256 correspondentes a pseudonimização dos endereços IP dos clientes que tentaram aceder ao servidor. Além disso, o campo *ShortMessage*, contém as mensagens completas, com o valor dos respetivo IPs substituídos pela *string* "c_ip".

5.3.5. Criação de perfis de utilizador no *Kibana*

A criação de perfis de utilizador e a gestão de utilizadores no *Kibana* apenas está disponível na versão *Platinum* que contem a extensão *X-Pack*. Apesar do *X-Pack* estar disponível na versão Basic, não é fornecida a licença para usar o modulo de Segurança. Felizmente, é possível usar uma licença *Trial* durante 30 dias, que permite testar todas as funcionalidades. A licença é ativada a partir da interface do *Kibana*, usando a opção *Management -> License Management*.

O processo de instalação e configuração do módulo de segurança, apresentado no Anexo B – Instalação da *ELK Stack*, não é fácil, uma vez que a informação existente na documentação técnica está bastante dispersa.

Uma vez terminado o processo de configuração do módulo de segurança, o arranque da interface *Web* do *Kibana* apresenta uma janela para introdução de credenciais de acesso.

[26] Fonte: Captura de ecrã feita pelo autor.

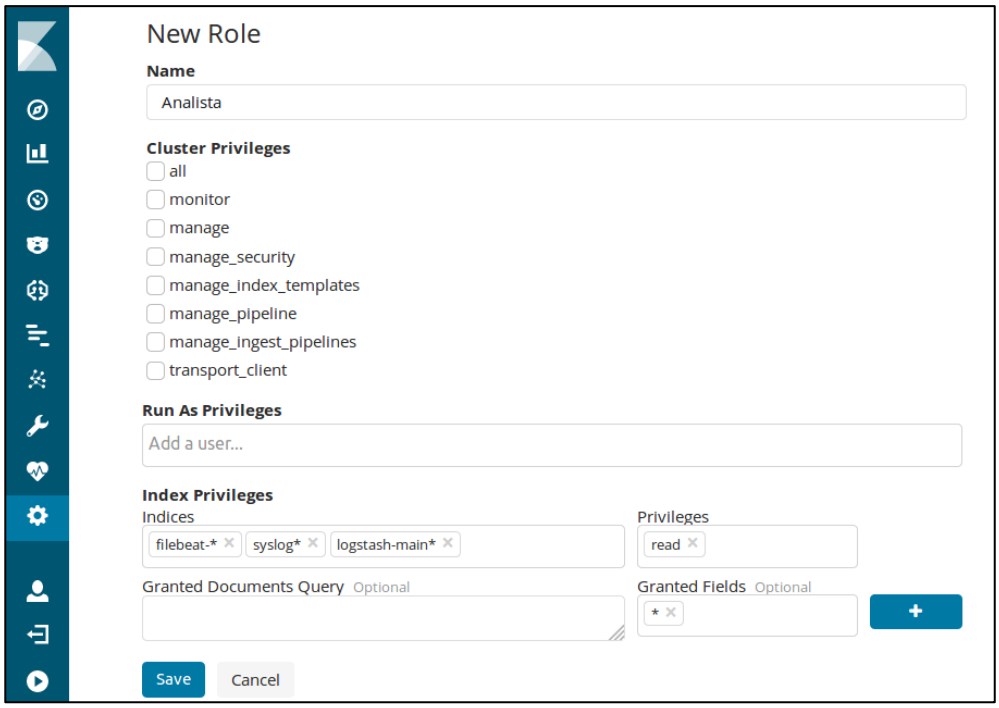


Figura 5.33 – Página de criação de um novo perfil no *Kibana*.[27]

A página *Management* apresenta agora um novo grupo *Security* com duas opções: *Roles* e *Users*. A criação de novos perfis de utilizador é feita a partir da opção *Roles -> Create role.*

Caso se pretenda criar um perfil de *Analista*, com acesso de leitura aos índices *filebeat*, *syslog* e *logstash_main* e portanto, sem acesso ao índice *logstash_pairs* que é o arquivo de identidades, a configuração é feita conforme mostra a Figura 5.33. Os índices acessíveis por este perfil estão introduzidos campo *Indices* e o privilégio de leitura está selecionado no campo *Privileges*. No campo *Granted Fields* é dado acesso de leitura a todos os campos dos índices selecionados, usando a *wildcard* *.

A criação de novos utilizadores é feita a partir da opção *Users -> Create user*. A criação, por exemplo, do utilizador *Artur* associada ao perfil *Analista* é feita como mostra a Figura 5.34. Os perfis atribuídos ao utilizador estão selecionados no campo *Role*s. Neste caso, apenas está atribuído o perfil *Analista*.

[27] Fonte: Captura de ecrã feita pelo autor.

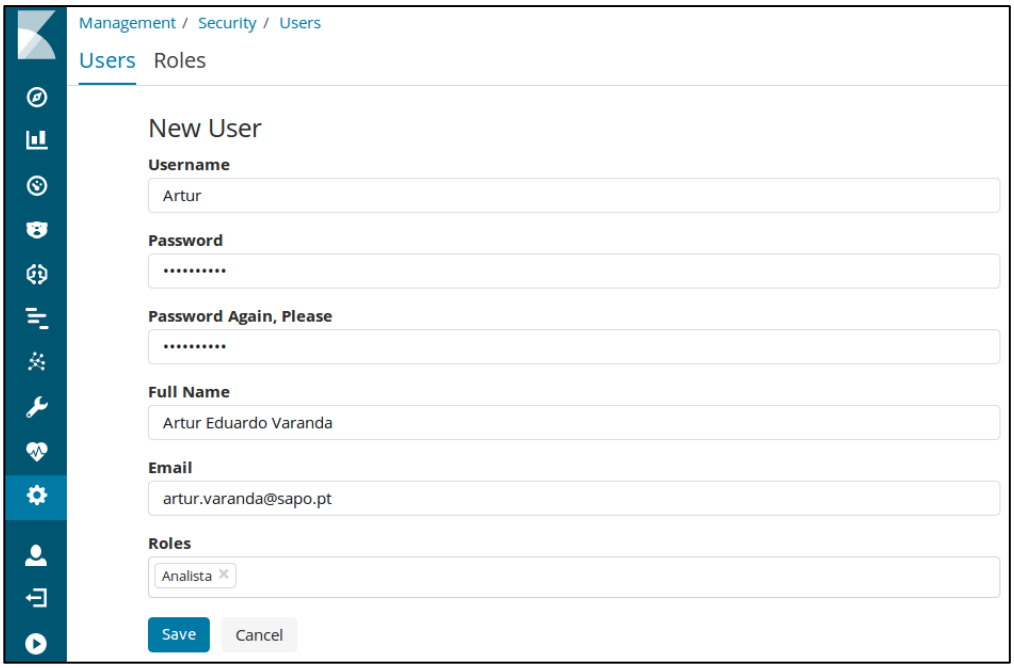


Figura 5.34 – Página de criação de um novo utilizador no *Kibana*.[28]

5.4. Análise comparativa dos diferentes cenários

Os três cenários apresentados neste trabalho usam sistemas de gestão de registos que no essencial são mito semelhantes entre si. Todos eles respeitam os princípios típicos de uma infraestrutura de gestão de registos. Permitem configurar entradas para receber dados enviados por clientes remotos ou recolher dados de ficheiros. Estão preparados para extrair os campos das mensagens recebidas e indexar os respetivos dados, para poderem ser pesquisados e visualizados. Incluem interfaces *Web* para gerir o sistema e para realizar pesquisas nos índices.

As páginas de pesquisa e apresentação dos resultados são muito semelhantes, com os diversos componentes localizados nas posições típicas: barra de pesquisa no topo; seleção da janela de tempo localizada ao lado direito da barra; os campos das mensagens apresentados em baixo, no lado esquerdo e as mensagens apresentadas por linhas, no lado direito. Entre a barra de pesquisa e a lista de mensagens costuma haver um histograma que permite visualizar o fluxo de dados, com possibilidade de filtrar as mensagens por janelas de tempo.

As soluções permitem criar perfis e novos utilizadores. No entanto, estas funcionalidades apenas costumam estar disponíveis nas versões comerciais.

[28] Fonte: Captura de ecrã feita pelo autor.

Graylog

O *Graylog*, estudado no primeiro cenário, é um sistema mais limitado que os outros dois, no que respeita ao processamento de mensagens.

As transformações de dados dos campos são possíveis através da configuração de *pipelines*, mas as funções de *hash* disponíveis são limitadas. A função HMAC não está disponível, embora seja possível implementar este algoritmo usando as funções existentes.

Uma grande vantagem do *Graylog* em relação às outras soluções, é o facto de incluir a gestão de perfis de utilizadores numa versão gratuita.

Splunk

O *Splunk*, estudado no segundo cenário, é o servidor mais fácil de instalar. No entanto, devido ao facto da documentação de suporte ser pouco detalhada, a utilização de funções complexas torna-se um pouco difícil. Outra desvantagem é o facto das funções apenas poderem ser aplicadas a campos internos do servidor, o que origina atrasos no processo de ingestão. A configuração de perfis de utilizadores apenas está disponível na versão *Enterprise*, embora esta versão possa ser testada por um período de 30 dias.

ELK Stack

A *ELK Stack*, estudada no terceiro cenário, revela-se como a solução mais versátil. Os módulos da *Stack* têm de ser instalados individualmente e interligados através de ficheiros de configuração. Cada componente corresponde a um serviço que pode correr em separado.

Esta solução dispõe de um conjunto muito interessante de módulos e extensões. O *Logstash*, por exemplo, integra um grande número de filtros de processamento. Além disso, a documentação disponibilizada é extensa e muito completa.

O protótipo desenvolvido com a *ELK Stack* é o único que permite implementar, ao mesmo tempo, duas funcionalidades requeridas à partida, nomeadamente: a pseudonimização de dados através da utilização de uma função HMAC e o arquivo de identidades num índice separado de acesso restrito.

Uma réplica deste protótipo está instalada em *cloud* para que seja possível verificar as várias funcionalidades. As instruções de acesso ao servidor estão disponíveis no Anexo I – Protótipo da *ELK Stack* na *Cloud*.

A tabela seguinte apresenta um resumo comparativo das principais funcionalidades das soluções desenvolvidas:

Tabela 5.1 - Resumo comparativo das principais funcionalidades das três soluções desenvolvidas

	Graylog	**Splunk**	**ELK Stack**
Formato da Mensagem	GELF	Plain Text	JSON
Configuração de Entradas	Interface Web	Interface Web ou Ficheiro de Config.	Interface Web ou Ficheiro de Config.
Extração de Campos	GROK	Expressões Regulares	GROK
Transformação de mensagens	Interface Web	Ficheiro de Configuração	Ficheiro de Configuração
Função de Resumo usada	SHA-256	SHA-256	HMAC SHA-256
Geração de pares identidade/resumo	Sim	Não	Sim
Descarte de pares repetidos	Não	-	Sim
Configuração de perfis de utilizador	Sim	Sim, na versão *Enterprise*	Sim, na versão Enterprise

5.5. Síntese

Neste capítulo foram apresentadas três soluções concretas para a pseudonimização da informação contida nos registos de sistemas e aplicações. Para cada solução foram apresentadas as respetivas particularidades para a respetiva implementação: configuração de entradas, extração de campos das mensagens, pesquisa e visualização de registos. No final, foi efetuada uma análise comparativa das três soluções e são apresentadas algumas conclusões.

No último capítulo são observadas as conclusões, apresentadas as principais contribuições disponibilizadas por este trabalho e sugeridos tópicos para desenvolvimento futuro.

6. Conclusões

Pretendeu-se com este trabalho, em primeiro lugar, identificar a informação contida nos *logs* gerados por sistemas e aplicações que necessita pseudonimizada, para estar em conformidade com o RGPD. Para isso, foi necessário estudar e analisar os processos de registo dos sistemas operativos e aplicações mais comuns, entre os quais, os registos dos eventos dos sistemas operativos Windows e Linux, os registos de acesso de servidores *Web*, registos de *firewalls* e registos de servidores SSH.

A fim de reduzir os riscos do ponto de vista de segurança da informação, aumentar a privacidade e facilitar o processamento de dados pessoais para além dos propósitos originais de recolha, foram estudas e avaliadas diversas estratégias para a pseudonimização da informação contida nos *logs* de sistemas e aplicações.

No final, foram desenvolvidas, testadas e comparadas três soluções concretas de pseudonimização da informação contida nos *logs* de sistemas e aplicações, usando cada uma delas um servidor de gestão centralizada de registos distinto.

6.1. Principais Contribuições

O trabalho desenvolvido na criação dos três protótipos, apesar de focado na transformação de *logs*, poderá ser muito útil para projetos de gestão centralizada de mensagens onde é necessário, de alguma forma, pseudonimizar algum tipo de informação. Assim, as principais contribuições apresentadas neste trabalho são as seguintes:

(i) *Pseudonimização de dados pessoais numa infraestrutura de gestão de logs baseada num servidor Graylog.*

A arquitetura apresentada utiliza servidor *Graylog* a funcionar como um agregador de mensagens de registo provenientes de várias fontes. A pseudonimização é feita através de funções de transformação aplicadas a *pipelines* de acordo com determinadas regras. Da mesma forma, são criados documentos compostos pelos valores originais e pelos respetivos pseudónimos, que são encaminhados para um segmento de acesso restrito, para permitir a reidentificação.

(ii) *Pseudonimização de dados pessoais numa infraestrutura de gestão de logs baseada num servidor Splunk.*

A arquitetura apresentada utiliza servidor *Splunk* a funcionar como um agregador de mensagens de registo provenientes de várias fontes. A pseudonimização é feita através de funções de transformação definidas num ficheiro de configuração próprio.

(iii) *Pseudonimização de dados pessoais numa infraestrutura de gestão de logs baseada na ELK Stack.*

A arquitetura apresentada utiliza a ELK Stack para agregar mensagens de registo provenientes de várias fontes. A pseudonimização é feita através de funções de transformação aplicadas na *pipeline* de processamento de *logs*. Ao mesmo tempo, são criados documentos compostos pelos valores originais e pelos respetivos pseudónimos, que são encaminhados para um índice de acesso restrito, para permitir a reidentificação

6.2. Tópicos para Trabalho Futuro

As fontes geradoras de registos de eventos e respetivas estruturas de dados são imensas e muito diversificadas. Para este trabalho, foi necessário fazer uma seleção das principais aplicações e sistemas geradores de mensagens de registo, a fim de obter dados para serem ingeridos pelos protótipos desenvolvidos. Para que seja possível aplicar a solução a um conjunto mais alargado de fontes geradoras de *logs*, é necessário analisar as respetivas estruturas de dados e definir os padrões para efetuar a competente extração de campos.

A definição de extratores de campos poderá revelar-se bastante complexa nos casos em que os dados estão, à partida, mal estruturados. A este nível é necessário desenvolver ferramentas de *parsing* que transformem as mensagens originais em mensagens fáceis de poderem ser processadas.

Para cada um dos protótipos desenvolvidos, poderão ser considerados diferentes opções para futuro desenvolvimento.

Ao nível ao protótipo desenvolvido no *Graylog*, tem de ser criado um mecanismo que agregue os pares compostos pelos dados identificativos e pelos respetivos pseudónimos, isto é, que descarte pares idênticos já existentes no índice.

Relativamente ao protótipo desenvolvido para o *Splunk*, tem de ser criado um mecanismo que possibilite a reidentificação dos dados pessoais, para além do método de duplicação de índices. Em alternativa às funções de transformação usadas nos ficheiros de configuração da *pipeline*, poderão ser desenvolvidas "entradas modulares" para acelerar o processamento na fase de ingestão.

Finalmente, ao nível da *ELK Stack*, o protótipo desenvolvido já cumpre todas propostas de desenvolvimento referidas para os restantes protótipos. A evolução desta solução passa por refinar os processos de *parsing* e extração de campos de outras fontes geradoras de registos, como foi referido no início desta secção.

Bibliografia

[1] U. Europeia, "REGULAMENTO (UE) 2016/679 DO PARLAMENTO EUROPEU E DO CONSELHO de 27 de abril de 2016," *J. Of. da União Eur.*, 2016.

[2] "Resolução do Conselho de Ministros n.º 41/2018," *Diário da República, 1.ª série — N.º 62 — 28 março 2018*, pp. 1424–1430, 2018.

[3] A. S. Reid, "The European Court of Justice case of Breyer," *J. Inf. Rights, Policy Pract.*, vol. 2, no. 1, 2017.

[4] A. Andersdotter, "An update to RFC6302 on Logging Recommendations for Internet-Facing Servers," *IntArea Work. Gr.*, 2018.

[5] K. Kent and M. P. Souppaya, "Guide to Computer Security Log Management," *NIST Spec. Publ. 800-92*, 2006.

[6] S. Datt, *Learning Network Forensics*. Packt Publishing Ltd, 2016.

[7] S. Davidoff and J. Ham, *Network forensics: tracking hackers through cyberspace*. Prentice hall, 2012.

[8] A. A. Chuvakin, K. J. Schmidt, C. Phillips, and P. Moulder, *Logging and log management : the authoritative guide to understanding the concepts surrounding logging and log management*. Syngress, 2013.

[9] P. M. Hallam-Baker and B. Behlendorf, "Extended log file format," *WWW J.*, vol. 3, p. W3C, 1996.

[10] J. Turnbull, P. Lieverdink, and D. Matotek, *Pro Linux System Administration*. Springer, 2009.

[11] D. R. Miller, S. Harris, A. Harper, S. VanDyke, and C. Blask, *Security Information and Event Management (SIEM) Implementation*. McGraw-Hill Education, 2010.

[12] L. Brotherston and A. Berlin, *Defensive Security Handbook*. 2017.

[13] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an Encrypted and Searchable Audit Log," in *NDSS*, 2004, vol. 4, pp. 5–6.

[14] S. Ghiasvand and F. M. Ciorba, "Anonymization of System Logs for Privacy and Storage Benefits," *Proc. Futur. Inf. Commun. Conf.*, 2018.

[15] S. Ghiasvand and F. M. Ciorba, "Assessing Data Usefulness for Failure Analysis in Anonymized System Logs," *arXiv Prepr. arXiv1805.01790*, 2018.

[16] M. Mourby *et al.*, "Are 'pseudonymised' data always personal data? Implications of the GDPR for administrative data research in the UK," *Comput. Law Secur. Rev.*, 2018.

[17] R. Schwartmann and S. Weiß, "White Paper on Pseudonymization Drafted by the Data Protection Focus Group for the Safety, Protection, and Trust Platform for Society and Businesses in Connection with the 2017 Digital Summit.," *Digit. Summit*, vol. 2017, p. 44, 2017.

[18] W. Stallings, *Cryptography and Network Security, 4/E*. Pearson Education, 2006.

[19] J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya, "Merkle-Damgård Revisited: How to Construct a Hash Function," in *Advances in Cryptology -- CRYPTO 2005*, 2005, pp. 430–448.

[20] H. Krawczyk, M. Bellare, and R. Canetti, "RFC 2104: HMAC: Keyed-hashing for message authentication," *Internet Eng. Task Force*, vol. 252, 1997.

[21] "Graylog 2.2 Documentation," *Graylog Inc.* 2017.

[22] "NXLOG Community Edition Reference Manual," *Community, Nxlog Ref. Ed.*, no. November, 2018.

[23] "NXLog User Guide," *NXLog Ltd*, 2006.

[24] "Splunk® Enterprise Documentation," *Splunk Inc.* 2019.

[25] "Logstash Reference [6.3]," *Elastic*. 2018.

[26] T. Bray, "The javascript object notation (json) data interchange format," 2014.

Anexo A – Instalação do *Graylog*

Instalação

Antes da instalação do servidor Graylog, são instalados todos os componentes necessários para o seu funcionamento: Java, MongoDB e Elasticsearch.

Em primeiro lugar é instalada a versão 1.8 o Java Runtime Environment:

 # add-apt-repository -y ppa:webupd8team/java

 # apt-get update

 # apt-get install -y oracle-java8-installer

A versão instalada deve ser verificada com o comando:

 # java –version

java version "1.8.0_171"
Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)

Em seguida é instalado o Elasticsearch:

 # wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | apt-key add -

 # echo "deb https://packages.elastic.co/elasticsearch/2.x/debian stable main" | tee -a /etc/apt/sources.list.d/elasticsearch.list

 # apt-get update && apt-get install -y elasticsearch

A configuração do *Elasticsearch* é feita através da edição do ficheiro *elasticsearch.yml*:

```
cluster.name: graylog
script.inline: false
script.indexed: false
script.file: false
```

Finalmente o serviço é ativado e iniciado:

 # systemctl enable elasticsearch.service

 # systemctl start elasticsearch.service

Deve ser feita a verificação do estado do serviço usando o comando:

 # service elasticsearch status

- elasticsearch.service - Elasticsearch

  Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled; vendo

  Active: active (running) since Sex 2018-05-04 14:46:14 WEST; 45min ago

Em seguida é instalado o MongoDB:

 # apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927

 # echo "deb http://repo.mongodb.org/apt/debian wheezy/mongodb-org/3.2 main" | tee /etc/apt/sources.list.d/mongodb-org.list

 # apt-get update

 # apt-get install -y mongodb-org

O serviço é ativado e iniciado:

 # systemctl enable mongod

 # systemctl start mongod

Deve ser verificado que o serviço está a ativo usando o comando:

 # service mongod status

- mongod.service - LSB: An object/document-oriented database

  Loaded: loaded (/etc/init.d/mongod; bad; vendor preset: enabled)

  Active: active (running) since Sex 2018-05-04 14:46:15 WEST; 52min ago

Finalmente é instalado o servidor Graylog:

 # wget https://packages.graylog2.org/repo/packages/graylog-2.2-repository_latest.deb

 # dpkg -i graylog-2.2-repository_latest.deb

 # apt-get install -y apt-transport-https

 # apt-get update

 # apt-get install -y graylog-server

Configuração

A configuração do servidor Graylog requer uma chave cifrada (CHAVE). Para isso, tem de ser instalado um gerador de chaves e gerada uma chave de 96 caracteres:

 # apt-get install -y pwgen

 # pwgen -N 1 -s 96

> 4jCDiSyz6ynlqj7FacshDR11klU2FT31NTfDNsdeDPKmautbGuKWiFhv4v7Lh82Yc
> AavNXDchzfWJq25vYp4yfFkJ4Xt8y22

A palavra passe de acesso do administrador do servidor é igualmente cifrada. Por exemplo, para a palavra-passe PASSWORD, o HASH é:

 # echo -n PASSWORD | shasum -a 256

> 0be64ae89ddd24e225434de95d501711339baeee18f009ba9b4369af27d30d60

A configuração do servidor Graylog é feita através da edição do ficheiro *server.conf*:

```
is_master = true
password_secret = CHAVE
root_password_sha2 = HASH
root_timezone = UTC
elasticsearch_discovery_zen_ping_unicast_hosts = 127.0.0.1:9300
elasticsearch_max_docs_per_index = 20000000
elasticsearch_max_number_of_indices = 20
```

```
elasticsearch_shards = 1
rest_listen_uri = http://0.0.0.0:12900/
web_listen_uri = http://0.0.0.0:9000/
```

Os valores “CHAVE” e “HASH” devem ser substituídos pelas chaves obtidas acima.

Finalmente, o serviço é ativado e iniciado:

 # systemctl daemon-reload

 # systemctl enable graylog-server.service

 # systemctl start graylog-server.service

Deve ser verificado que o serviço está a correr:

 # service graylog-server status

- graylog-server.service - Graylog server

  Loaded: loaded (/usr/lib/systemd/system/graylog-server.service; enabled; vend

  Active: active (running) since Sex 2018-05-04 14:46:14 WEST; 1h 2min ago

Anexo B – Instalação da *ELK Stack*

Antes de serem instalados os módulos da *ELK Stack*, deve ser instalada a versão 1.8 do OpenJDK:

 # rpm --nodeps -ivh jdk-8u221-linux-x64.rpm

 # apt install openjdk-8-jre-headless

A versão instalada deve ser verificada com o comando:

 # java –version

openjdk version "1.8.0_222"
OpenJDK Runtime Environment (build 1.8.0_222-8u222-b10-1ubuntu1~16.04.1-b10)
OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode)

Instalação do *Elasticsearch*, *Kibana*, *Logstash* e *Filebeat*

Os quatro ficheiros de instalação da *ELK Stack*, disponíveis no servidor da *Elastic*, devem ser descarregados e descomprimidos:

elasticsearch-6.3.2.tar.gz
kibana-6.3.2-linux-x86_64.tar.gz
logstash-6.3.2.tar.gz
filebeat-6.3.2-amd64.deb

Em seguida, deve ser aberto o ficheiro *config/kibana.yml* e editada a linha relativa ao endereço do *Elasticsearch* usado para efetuar as pesquisas:

```
# The URL of the Elasticsearch instance to use for all your queries.
elasticsearch.url: "http://localhost:9200"
```

O *Filebeat* integra vários módulos que podem ser listados usando o comando:

 # filebeat modules list

Inicialmente, os módulos estão desativados. Como interessa recolher eventos do *Syslog*, é ativado o módulo *System* usando o comando:

 # filebeat modules enable system

Em seguida, é configurado o ambiente inicial, através do comando:

 # filebeat setup -e

O comando **setup** instala o padrão de índice recomendado para as mensagens coletadas pelo *Filebeat* e disponibiliza um conjunto de painéis de visualização (*dashboards*). A *flag -e* serve para enviar a saída do comando para o *standard error*, em vez de o enviar para o *Syslog*.

Finalmente, o serviço é iniciado através do comando:

 # service filebeat start

Deve ser verificado que o serviço está a correr:

 # service filebeat status

- filebeat.service - filebeat

  Loaded: loaded (/lib/systemd/system/filebeat.service; disabled; vendor preset

  Active: active (running) since Sex 2019-09-06 14:19:47 WEST; 2s ago

Instalação do Módulo de Segurança do X-Pack

As versões mais antigas do *Kibana* não incluem a extensão *X-Pack*, pelo que é necessário instalá-la através da consola, usando o comando

 # bin/kibana-plugin install x-pack

Depois de instalado o X-Pack e ativada a licença *Trial*, é necessário ativar as opções de segurança nos ficheiros de configuração do *Elasticsearch* e do *Kibana*.

No ficheiro */config/elasticsearch.yml* são acrescentadas as linhas:

```
1 xpack.security.enabled: true
2 xpack.license.self_generated.type: trial
```

No ficheiro */config/kibana.yml* são acrescentadas as linhas:

```
elasticsearch.username: "elastic"
elasticsearch.password: "password_do_utilizador_elastic"
xpack.reporting.encryptionKey: "chave_secreta_extensa"
```

Finalmente, são definidas as palavras-passe do *Elasticsearch*, *Kibana*, *Logstash* e *Beats* na consola, usando o comando:

 # bin/elasticsearch-setup-passwords interactive.

Para que o comando seja processado, é necessário que o serviço do *Elasticsearch* esteja ativo.

Depois de executado o comando, são introduzidas e confirmadas as palavras-passe dos quatro componentes da *ELK Stack*:

```
Initiating the setup of passwords for reserved users elastic, kibana,
logstash_system, beats_system.
You will be prompted to enter passwords as the process progresses.
Please confirm that you would like to continue [y/N] y

Enter password for [elastic]:
Reenter password for [elastic]:
Enter password for [kibana]:
Reenter password for [kibana]:
Enter password for [logstash_system]:
Reenter password for [logstash_system]:
Enter password for [beats_system]:
Reenter password for [beats_system]:
Changed password for user [kibana]
Changed password for user [logstash_system]
Changed password for user [beats_system]
Changed password for user [elastic]
```

Anexo C – Padrões *Grok* de extração de campos

***Firewall* do Windows:**

%{DATESTAMP:timestamp} %{WORD:action} %{WORD:protocol} %{NOTSPACE:src_ip} %{NOTSPACE:dst_ip} %{INT:src_port} %{INT:dst_port} %{INT:size} %{NOTSPACE:tcpflags} %{NOTSPACE:tcpsyn} %{NOTSPACE:tcpack} %{NOTSPACE:tcpwin} %{NOTSPACE:icmptype} %{NOTSPACE:icmpcode} %{NOTSPACE:info} %{NOTSPACE:path}

UFW:

%{SYSLOGHOST:ufw_hostname} %{WORD:facility}: \[%{DATA}\] \[UFW %{NOTSPACE:ufw_action}\] IN=%{NOTSPACE:ufw_interface} OUT= (MAC|PHYSIN)=%{DATA:ufw_mac} SRC=%{IP:ufw_src_ip} DST=%{IP:ufw_dest_ip} %{GREEDYDATA:ufw_protocol_data}

Servidor *Web* da Apache:

%{IPORHOST:c_ip} %{HTTPDUSER:ident} %{USER:auth} \[%{HTTPDATE:timestamp}\] \"%{WORD:cs_method} %{NOTSPACE:request_page} HTTP/%{NUMBER:http_version}\" %{NUMBER:server_response} (?:%{NUMBER:bytes}|-)

Servidor *Web* Microsoft IIS:

%{DATESTAMP: timestamp} %{IP:s_ip} %{WORD:cs_method} %{URIPATHPARAM:cs_uri_stem} %{NOTSPACE:cs_uri_query} %{INT:s_port} %{NOTSPACE:cs_username} %{IP:c_ip} %{NOTSPACE:cs_user_agent} %{INT:sc_status} %{INT:sc_substatus} %{INT:sc_win32_status} %{INT:time_take}

Servidor *OpenSSH*

SSHD_USER_ACCESS

%{WORD:result}(?<String1> password for) %{USERNAME:sshd_user} (?<String2>from) %{IP:sshd_client_ip} port %{NUMBER:sshd_port} %{WORD:sshd_protocol}

Anexo D – Expressões regulares de extração

***Firewall* do Windows:**

^(?P<timestamp>[^]+\s[^]+)\s(?P<action>\w+)\s+(?P<protocol>\w+)\s+(?P<src_ip>[^]+)\s+(?P<dst_ip>[^]+)\s+(?P<src_port>\d*)\s+(?P<dst_port>\d+)\s+(?P<size>\d+)\s+(?P<tcpflags>[^]+)\s+(?P<tcpsyn>[^]+)\s+(?P<tcpack>[^]+)\s+(?P<tcpwin>[^]+)\s+(?P<icmptype>[^]+)\s+(?P<icmpcode>[^]+)\s+(?P<info>[^]+)\s+(?P<path>\w+)

UFW:

^(?P<timestamp>\w*+\s*+\d*+\s*+\d*:\d*:\d*)\s+(?P<ufw_hostname>[^]*)\s+(?P<facility>[^:]*):\s+\[+\s*+(?P<uptime>[^ \]]*)\]\s+\[(?P<ufw_action>\w*\s\w*)\]\s+IN=(?P<ufw_interface>\w*)\s+OUT=[^]*\s+MAC=(?P<ufw_mac>[^]*)\s+SRC=(?P<ufw_src_ip>[^]*)\s+DST=(?P<ufw_dst_ip>[^]*)\s+(?P<protocol_data>.*)

Servidor *Web* da Apache:

^(?P<c_ip>[^]+)\s+(?P<ident>[^]+)\s+(?P<auth>\-)\s+(?P<timestamp>\[\d+\/\w+\/\d+:\d+:\d+:\d+\s\+\d+\])[^"\n]*"(?P<cs_method>\w+)\s+(?P<request_page>[^]+)\s+HTTP\/(?P<http_version>\d+\.\d+)"\s+(?P<server_response>[^]+)\s(?P<bytes>\d+)

Servidor *Web* Microsoft IIS:

^(?P<timestamp>\d*-\d*-\d*\s\d*:\d*:\d*)\s+(?P<s_ip>[^]*)\s+(?P<cs_method>\w+)\s+(?P<cs_uri_stem>[^]*)\s+(?P<cs_uri_query>[^]*)\s+(?P<s_port>[^]+)\s+(?P<cs_username>[^]*)\s+(?P<c_ip>[^]*)\s+(?P<cs_user_agent>[^]*)\s+(?P<sc_status>\d*)\s+(?P<sc_substatus>\d*)\s+(?P<sc_win32_status>\d*)\s+(?P<time_take>\d*)

Servidor *OpenSSH*

SSHD_USER_ACCESS

^(?P<result>\w*)\spassword for\s+(?P<sshd_user>[^]*)\s+from+\s+(?P<sshd_client_ip>[^]*)\s+port+\s+(?P<sshd_port>\d+)\s+(?P<protocol>\w+)

Anexo E – Configuração de pipelines do *Graylog*

Pseudonymize Windows Firewall IPs

```
rule "Pseudonymize Windows Firewall IPs"
when
    has_field("dst_ip") AND has_field("src_ip")
then
    // Pseudonimiza campos identificativos
    let odst_ip = to_string($message.dst_ip);
    let hdst_ip = sha256(odst_ip);
    set_field("dst_ip", hdst_ip);
    let osrc_ip = to_string($message.src_ip);
    let hsrc_ip = sha256(osrc_ip);
    set_field("src_ip", hsrc_ip);

    // Elimina campos redundantes com valores identificativos
    remove_field("IPV6");
    remove_field("IPV4");

    // Cria campo mensagem com valores identificativos mascarados
    let message = concat(to_string($message.timestamp), " ");
    let message = concat(message, to_string($message.action));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.protocol));
    let message = concat(message, " src_ip dst_ip ");
    let message = concat(message, to_string($message.src_port));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.dst_port));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.size));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.tcpflags));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.tcpsyn));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.tcpack));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.tcpwin));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.icmptype));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.icmpcode));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.info));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.path));
    set_field("message", message);

    // Reencaminha mensagem para stream de mensagens pseudonimizadas
    set_field("Pseudonymised", "Yes");
    route_to_stream("Pseudonymised Messages");

    // Cria mensagem com par valor+resumo para a stream de identidades
    let msg = concat(to_string(hsrc_ip), " ");
    let msg = concat(msg, osrc_ip);
    let msg1 = create_message(msg, to_string($message.source));
```

```
    route_to_stream("Pairs", "", msg1);
    let msg = concat(to_string(hdst_ip), " ");
    let msg = concat(msg, odst_ip);
    let msg2 = create_message(msg, to_string($message.source));
    route_to_stream("Pairs", "", msg2);
end
```

Pseudonymize Winapache

```
rule "pseudonymize winapache"
when
    has_field("c_ip")
then
    // Pseudonimiza campos identificativos
    let oc_ip = to_string($message.c_ip);
    let hc_ip = sha256(oc_ip);
    set_field("c_ip", hc_ip);

    // Elimina campos redundantes com valores identificativos
    remove_field("IP");
    remove_field("IPV4");

    // Cria campo mensagem com valores identificativos mascarados
    let message = concat("c_ip ", to_string($message.ident));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.auth));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.timestamp));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.cs_method));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.request_page));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.http_version));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.server_response));
    let message = concat(message, " ");
    let message = concat(message, to_string($message.bytes));
    set_field("message", message);

    // Reencaminha mensagem para stream de mensagens pseudonimizadas
    set_field("Pseudonymised", "Yes");
    route_to_stream("Pseudonymised Messages");

    // Cria mensagens com pares valor+resumo para a stream de identidades
    let msg = concat(to_string(hc_ip), " ");
    let msg = concat(msg, oc_ip);
    let msg1 = create_message(msg, to_string($message.source));
    route_to_stream("Pairs", "", msg1);

end
```

Pseudonymize SSHD_USER_ACCESS

```
rule "Pseudonymize SSHD_USER_ACCESS"
when
    has_field("sshd_client_ip") && has_field("sshd_user")
then
    // Pseudonimiza campos identificativos
    let o_ip = to_string($message.sshd_client_ip);
    let h_ip = sha256(o_ip);
    let o_user = to_string($message.sshd_user);
    let h_user = sha256(o_user);
    set_field("sshd_client_ip", h_ip);
    set_field("sshd_user", h_user);

    // Elimina campos redundantes com valores identificativos
    remove_field("payload");
    remove_field("full_message");
    remove_field("IPV4");
    remove_field("IPV6");
    remove_field("IP");

    // Cria campo mensagem com valores identificativos mascarados
    let message = concat("sshd: ", to_string($message.result));
    let message = concat(message, " ");
    let message = concat(message, "password for $sshd_user from
$sshd_client_ip port $sshd_port ssh2");
    set_field("message",message);

    // Reencaminha mensagem para stream de mensagens pseudonimizadas
    set_field("Pseudonymised", "Yes");
    route_to_stream("Pseudonymised Messages");

    // Cria mensagem com par valor+resumo para a stream de identidades
    let msg = concat(to_string(h_ip), " ");
    let msg = concat(msg, o_ip);
    let msg1 = create_message(msg, to_string($message.source));
    route_to_stream("Pairs", "", msg1);
    let msg = concat(to_string(h_user), " ");
    let msg = concat(msg, o_user);
    let msg2 = create_message(msg, to_string($message.source));
    route_to_stream("Pairs", "", msg2);
end
```

Anexo F – Configuração da *pipeline* do *Splunk*

inputs.conf

```
[default]
host = Elastic-VirtualBox

[tcp://12201]
disabled = false
connection_host = dns
source = winevent
sourcetype = generic_single_line

[tcp://12202]
disabled = false
connection_host = dns
source = iis
sourcetype = generic_single_line

[tcp://12203]
disabled = false
connection_host=dns
source = Windows_Firewall
sourcetype =  generic_single_line

[tcp://12204]
disabled = false
connection_host = dns
source = WinApache
sourcetype = generic_single_line
```

props.conf

```
[source::Windows_Firewall]
TRANSFORMS-step1 = WF_raw_backup
TRANSFORMS-step2 = WF_src_ip_extration
TRANSFORMS-step3 = WF_src_ip_pseudonymization
TRANSFORMS-step4 = WF_raw_restore1
TRANSFORMS-step5 = WF_dst_ip_extraction
TRANSFORMS-step6 = WF_dst_ip_pseudonymization
TRANSFORMS-step7 = WF_raw_restore2
TRANSFORMS-step8 = WF_raw_transformation

[source::WinApache]
TRANSFORMS-step9  = WA_extration
TRANSFORMS-step10 = WA_raw_backup
TRANSFORMS-step11 = WA_c_ip_extraction
TRANSFORMS-step12 = WA_c_ip_pseudonymization
TRANSFORMS-step13 = WA_raw_restore
TRANSFORMS-step14 = WA_raw_transformation
```

transforms.conf

```
[WF_raw_backup]
INGEST_EVAL = _rawtmp=_raw

[WF_src_ip_extraction]
REGEX = ^(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+
(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)
FORMAT = $5
DEST_KEY = _raw

[WF_src_ip_pseudonymization]
INGEST_EVAL = src_ip=sha256(_raw)

[WF_raw_restore1]
INGEST_EVAL = _raw=_rawtmp

[WF_dst_ip_extraction]
REGEX = ^(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+
(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)
FORMAT = $6
DEST_KEY = _raw

[WF_dst_ip_pseudonymization]
INGEST_EVAL = dst_ip=sha256(_raw)

[WF_raw_restore2]
INGEST_EVAL = _raw=_rawtmp

[WF_raw_transformation]
REGEX = ^(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+
(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)
FORMAT = $1 $2 $3 $4 src-ip dst-ip $7 $8 $9 $10 $11 $12 $13 $14 $15 $16 $17
DEST_KEY = _raw

[WA_extration]
REGEX = ^(\S+)\s+(.*)
FORMAT = $1 $2
DEST_KEY = _raw

[WA_raw_backup]
INGEST_EVAL = _rawtmp=_raw

[WA_c_ip_extraction]
REGEX = ^(\S+)\s+(.*)
FORMAT = $1
DEST_KEY = _raw

[WA_c_ip_pseudonymization]
INGEST_EVAL = c_ip=sha256(_raw)
[WA_raw_restore]
INGEST_EVAL = _raw=_rawtmp

[WA_raw_transformation]
REGEX = ^(\S+)\s+(.*)
FORMAT = c-ip $2
DEST_KEY = _raw
```

Anexo G – Configuração da pipeline do *Logstash*

```
1
2  input {
3
4     tcp {
5        type => "winevent"
6        codec => json_lines { charset => CP1252 }
7        port => "12201"
8        tags => [ "tcpjson" ]
9     }
10
11    tcp {
12       type => "iis"
13       codec => json_lines { charset => CP1252 }
14       port => "12202"
15       tags => [ "tcpjson" ]
16    }
17
18    tcp {
19       type => "WinFirewall"
20       codec => json_lines { charset => CP1252 }
21       port => "12203"
22       tags => [ "tcpjson" ]
23    }
24
25    tcp {
26       type => "win_apache"
27       codec => json_lines { charset => CP1252 }
28       port => "12204"
29       tags => [ "tcpjson" ]
30    }
31
32    beats {
33       type => "syslog_filebeat"
34       port => "5044"
35    }
36
37 #  udp {
38 #    type => "syslog_udp"
39 #    codec => plain
40 #    port => "1514"
41 #  }
42
43 }
44
45 filter {
46
47    if [type] == "WinFirewall" {
48
49       clone {
50          clones => ["WinFirewall_pairs1"]
51       }
52
53       clone {
```

```
54          clones => ["WinFirewall_pairs2"]
55       }
56
57       grok {
58          match => ["ShortMessage", "%{DATESTAMP:datestamp} %{WORD:action}
    %{WORD:protocol} %{NOTSPACE:src_ip} %{NOTSPACE:dst_ip} %{INT:src_port}
    %{INT:dst_port} %{INT:size} %{NOTSPACE:tcpflags} %{NOTSPACE:tcpsyn}
    %{NOTSPACE:tcpack} %{NOTSPACE:tcpwin} %{NOTSPACE:icmptype}
    %{NOTSPACE:icmpcode} %{NOTSPACE:info} %{NOTSPACE:path}"]
59       }
60
61                  if "_grokparsefailure" in [tags] {
62          drop { }
63       }
64
65       if [type] == "WinFirewall" {
66
67          fingerprint {
68             method => "SHA256"
69             key => "HMAC_SECRET_KEY"
70             source => "src_ip"
71             target => "src_ip"
72          }
73
74    fingerprint {
75             method => "SHA256"
76             key => "HMAC_SECRET_KEY"
77             source => "dst_ip"
78             target => "dst_ip"
79          }
80
81
82         mutate {
83            replace => { "ShortMessage" => " %{datestamp} %{action} src-ip
    dst-ip %{src_port} %{dst_port} %{size} %{tcpflags} %{tcpsyn} %{tcpack}
    %{tcpwin} %{tcpack} %{icmptype} %{icmpcode} %{info} %{path}" }
84         }
85       }
86    }
87
88    if [type] == "win_apache" {
89
90       clone {
91          clones => ["win_apache_pairs"]
92       }
93
94       grok {
95          match => ["ShortMessage", "%{IPORHOST:c_ip} %{HTTPDUSER:ident}
    %{USER:auth} \[%{HTTPDATE:timestamp}\] \"%{WORD:cs_method}
    %{NOTSPACE:request_page} HTTP/%{NUMBER:http_version}\"
    %{NUMBER:server_response} (?:%{NUMBER:bytes}|-)"]
96       }
97
98       if "_grokparsefailure" in [tags] {
99          drop { }
100      }
101
102      if [type] == "win_apache" {
103
```

```
104           fingerprint {
105              method => "SHA256"
106              key => "HMAC_SECRET_KEY"
107              source => "c_ip"
108              target => "c_ip"
109           }
110
111          mutate {
112             replace => { "ShortMessage" => "c_ip %{ident} %{auth}
    [%{timestamp}] \"%{cs_method} %{request_page} HTTP/%{http_version}\"
    %{server_response} %{bytes}" }
113          }
114       }
115    }
116
117    if [type] == "WinFirewall_pairs1" {
118
119       fingerprint {
120          method => "SHA256"
121          key => "HMAC_SECRET_KEY"
122          source => "src_ip"
123          target => "resumo"
124       }
125
126       mutate {
127          add_field => { "valor" => "%{src_ip}" }
128       }
129
130       prune {
131          interpolate => true
132          whitelist_names => ["type", "valor", "resumo"]
133       }
134
135       fingerprint {
136          source => "resumo"
137          target => "fingerprint_id"
138          method => "MURMUR3"
139       }
140    }
141
142 if [type] == "WinFirewall_pairs2" {
143
144       fingerprint {
145          method => "SHA256"
146          key => "HMAC_SECRET_KEY"
147          source => "dst_ip"
148          target => "resumo"
149       }
150
151       mutate {
152          add_field => { "valor" => "%{dst_ip}" }
153       }
154
155
156       prune {
157          interpolate => true
158          whitelist_names => ["type", "valor", "resumo"]
159       }
160
```

```
161         fingerprint {
162            source => "resumo"
163            target => "fingerprint_id"
164            method => "MURMUR3"
165         }
166      }
167
168      if [type] == "win_apache_pairs" {
169
170         fingerprint {
171            method => "SHA256"
172            key => "HMAC_SECRET_KEY"
173            source => "c_ip"
174            target => "resumo"
175         }
176
177         mutate {
178            add_field => { "valor" => "%{c_ip}" }
179         }
180
181         prune {
182            interpolate => true
183            whitelist_names => ["type", "valor", "resumo"]
184         }
185
186         fingerprint {
187            source => "resumo"
188            target => "fingerprint_id"
189            method => "MURMUR3"
190         }
191      }
192   }
193
194   output {
195
196      if [type] == "WinFirewall_pairs1" {
197
198         elasticsearch {
199            hosts => localhost
200            index => "logstash-pairs"
201            document_id => "%{fingerprint_id}"
202         }
203        stdout { codec => rubydebug }
204      }
205
206      if [type] == "WinFirewall_pairs2" {
207
208         elasticsearch {
209            hosts => localhost
210            index => "logstash-pairs"
211            document_id => "%{fingerprint_id}"
212         }
213        stdout { codec => rubydebug }
214      }
215
216      if [type] == "win_apache_pairs" {
217
218         elasticsearch {
219            hosts => localhost
```

```
220             index => "logstash-pairs"
221             document_id => "%{fingerprint_id}"
222          }
223        stdout { codec => rubydebug }
224      }
225
226      if [type] == "winevent" {
227         elasticsearch {
228            hosts => localhost
229            index => "logstash-main"
230         }
231         stdout { codec => rubydebug }
232      }
233
234      if [type] == "iis" {
235         elasticsearch {
236            hosts => localhost
237            index => "logstash-main"
238         }
239         stdout { codec => rubydebug }
240      }
241
242      if [type] == "WinFirewall" {
243         elasticsearch {
244            hosts => localhost
245            index => "logstash-main"
246         }
247         stdout { codec => rubydebug }
248      }
249
250      if [type] == "win_apache" {
251         elasticsearch {
252            hosts => localhost
253            index => "logstash-main"
254         }
255         stdout { codec => rubydebug }
256      }
257
258   if [type] == "syslog_filebeat" {
259         elasticsearch {
260            hosts => localhost
261            manage_template => false
262            index => "syslog"
263         }
264         stdout { codec => rubydebug }
265      }
266
267   }
```

Anexo H – Configuração do *NXLog*

Windows Host

```
1
2  ##################################################################
3  ##########                  ######################################
4  ##########    DEFINITIONS   ######################################
5  ##########                  ######################################
6  ##################################################################
7
8  define ROOT     C:\Program Files (x86)\nxlog
9  define CERTDIR  %ROOT%\cert
10 define CONFDIR  %ROOT%\conf
11 define LOGDIR   %ROOT%\data
12 define LOGFILE  %LOGDIR%\nxlog.log
13
14 define GRAYLOG 192.168.43.35
15 #define SPLUNK 192.168.43.112
16 #define LOGSTASH 192.168.43.58
17
18 LogFile %LOGFILE%
19
20 Moduledir %ROOT%\modules
21 CacheDir  %ROOT%\data
22 Pidfile   %ROOT%\data\nxlog.pid
23 SpoolDir  %ROOT%\data
24
25 ##################################################################
26 ##########              ##########################################
27 ##########    MODULES   ##########################################
28 ##########              ##########################################
29 ##################################################################
30
31 <Extension _syslog>
32     Module      xm_syslog
33 </Extension>
34
35 <Extension _json>
36     Module      xm_json
37 </Extension>
38
39 <Extension _charconv>
40     Module      xm_charconv
41     AutodetectCharsets iso8859-2, utf-8, utf-16, utf-32
42 </Extension>
43
44 <Extension _exec>
45     Module      xm_exec
46 </Extension>
47
48 <Extension _fileop>
49
50     Module      xm_fileop
51
52     # Check the size of our log file hourly, rotate if larger than 5MB
53     <Schedule>
```

```
        Every   1 hour
        Exec    if (file_exists('%LOGFILE%') and \
                   (file_size('%LOGFILE%') >= 5M)) \
                    file_cycle('%LOGFILE%', 8);
    </Schedule>

    # Rotate our log file every week on Sunday at midnight

    <Schedule>
        When    @weekly
        Exec    if file_exists('%LOGFILE%') file_cycle('%LOGFILE%', 8);
    </Schedule>
</Extension>

<Extension _gelf>
    Module      xm_gelf
</Extension>


########################################################################
##########              ###############################################
##########    INPUTS    ###############################################
##########              ###############################################
########################################################################


<Input winevent>
   Module im_msvistalog
   Exec $ShortMessage = $Message;
</Input>

<Input iis>
    Module im_file
    File "C:\inetpub\logs\LogFiles\W3SVC1\u_ex*"
</Input>

<Input WinFirewallLog>
   Module im_file
   File "C:\Windows\system32\LogFiles\Firewall\pfirewall.log"
</Input>

<Input WinApache>
    Module im_file
    File "C:\Apache24\logs\access.log"
</Input>

########################################################################
##########              ###############################################
##########    OUTPUTS   ###############################################
##########              ###############################################
########################################################################

<Output winevent_graylog>
    Module om_tcp
    Host %GRAYLOG%
    Port 12201
    OutputType GELF_TCP
</Output>

<Output winevent_splunk>
    Module      om_tcp
```

```
    Host        %SPLUNK%
    Port        12201
</Output>

<Output winevent_logstash>
    Module      om_tcp
    Host        %LOGSTASH%
    Port        12201
    Exec        to_json();
</Output>

<Output iis_graylog>
    Module          om_ssl
    Host            %GRAYLOG%
    Port            12212
    CAFile          %CERTDIR%/rootCA.pem
    CertFile        %CERTDIR%/artur-TOSH.crt
    CertKeyFile     %CERTDIR%/artur-TOSH.key
    AllowUntrusted  TRUE
</Output>

<Output iis_splunk>
    Module      om_tcp
    Host        %SPLUNK%
    Port        12202
    Exec        $ShortMessage = $raw_event;
</Output>

<Output iis_logstash>
    Module      om_tcp
    Host        %LOGSTASH%
    Port        12202
    Exec        $ShortMessage = $raw_event;
    Exec        to_json();
</Output>

<Output WinFirewall_graylog>
   Module       om_tcp
   Host         %GRAYLOG%
   Port         12203
   Exec         $ShortMessage = $raw_event;
   OutputType   GELF_TCP
</Output>

<Output WinFirewall_splunk>
   Module       om_tcp
   Host         %SPLUNK%
   Port         12203
   Exec         $ShortMessage = $raw_event;
</Output>

<Output WinFirewall_logstash>
   Module       om_tcp
   Host         %LOGSTASH%
   Port         12203
   Exec         $ShortMessage = $raw_event;
   Exec         to_json();
</Output>

<Output Apache_graylog>
   Module      om_tcp
```

```
   Host        %GRAYLOG%
   Port        12204
   Exec        $ShortMessage = $raw_event;
   OutputType  GELF_TCP
</Output>


<Output Apache_splunk>
   Module      om_tcp
   Host        %SPLUNK%
   Port        12204
   Exec        $ShortMessage = $raw_event;
</Output>


<Output Apache_logstash>
   Module      om_tcp
   Host        %LOGSTASH%
   Port        12204
   Exec        $ShortMessage = $raw_event;
   Exec        to_json();
</Output>


########################################################################
##########              ##############################################
##########    ROUTES    ##############################################
##########              ##############################################
########################################################################

<Route 1>
   Path winevent => winevent_graylog
   #Path winevent => winevent_splunk
   #Path winevent => winevent_logstash
</Route>


<Route 2>
   Path  iis => iis_graylog
   #Path  iis => iis_splunk
   #Path  iis => iis_logstash
</Route>


<Route 3>
   Path  WinFirewallLog => WinFirewall_graylog
   #Path  WinFirewallLog => WinFirewall_splunk
   #Path  WinFirewallLog => WinFirewall_logstash
</Route>


<Route 4>
     Path  WinApache => Apache_graylog
     #Path  WinApache => Apache_splunk
     #Path  WinApache => Apache_logstash
</Route>
```

Graylog Server

```
1
2  ########################################
3  # GLOBAL DIRECTIVES                    #
4  ########################################
5
6
7  User nxlog
8  Group nxlog
9
10 LogFile /var/log/nxlog/nxlog.log
11 LogLevel INFO
12
13 define ROOT     /etc/nxlog
14 define CERTDIR  %ROOT%/cert
15
16
17 ########################################
18 # MODULES                              #
19 ########################################
20
21 <Extension _syslog>
22     Module      xm_syslog
23 </Extension>
24
25 <Extension _gelf>
26     Module      xm_gelf
27 </Extension>
28
29 <Extension _fileop>
30     Module      xm_fileop
31 </Extension>
32
33 <Extension _charconv>
34     Module      xm_charconv
35     AutodetectCharsets iso8859-2, utf-8, utf-16, utf-32
36 </Extension>
37
38 ########################################################################
39 ##########              ###############################################
40 ##########    INPUTS    ###############################################
41 ##########              ###############################################
42 ########################################################################
43
44 <Input Linux_Apache>
45     Module      im_file
46     File        "/var/log/apache2/access.log"
47 </Input>
48
49
50 <Input IIS>
51     Module      im_ssl
52     Host        0.0.0.0
53     Port        12212
54     CAFile      %CERTDIR%/rootCA.pem
55     CertFile    %CERTDIR%/varanda-VirtualBox.crt
56     CertKeyFile %CERTDIR%/varanda-VirtualBox.key
57 </Input>
58
```

```
59 ######################################################################
60 ##########              ###########################################
61 ##########    OUTPUTS   ###########################################
62 ##########              ###########################################
63 ######################################################################
64
65 <Output IIS_TCP_OUT>
66    Module       om_tcp
67    Host         127.0.0.1
68    Port         12202
69    Exec         $ShortMessage = $raw_event;
70    OutputType   GELF_TCP
71 </Output>
72
73 <Output Linux_Apache_TCP_OUT>
74    Module       om_tcp
75    Host         127.0.0.1
76    Port         12205
77    Exec         $ShortMessage = $raw_event;
78    OutputType   GELF_TCP
79 </Output>
80
81
82 ######################################################################
83 ##########              ###########################################
84 ##########    ROUTES    ###########################################
85 ##########              ###########################################
86 ######################################################################
87
88 <Route 1>
89     Path IIS => IIS_TCP_OUT
90 </Route>
91
92 <Route 2>
93     Path Linux_Apache => Linux_Apache_TCP_OUT
94 </Route>
```

Anexo I – Protótipo da *ELK Stack* na *Cloud*

Instalação

A versão em *cloud* do protótipo desenvolvido na *ELK Stack* está instalada num servidor da Hostwinds. Este servidor, com o endereço IP 192.236.160.120, corre num sistema operativo Ubuntu 16.04 e contempla um CPU, 1 GB de RAM e 30 GB SDD de *storage.*

Alguns detalhes devem ser tidos em consideração na instalação da *ELK Stack* em servidores de *cloud*:

1) A falta de dependências necessárias para a instalação do Java JDK 8 podem não permitir instalar a ferramenta. A solução mais fácil é fazer a instalação manualmente:

```
$ tar -xvf jdk-8-linux-x64.tar.gz
$ mkdir -p /usr/lib/jvm
$ sudo mv ./jdk1.8.0 /usr/lib/jvm/

$ sudo update-alternatives --install "/usr/bin/java" "java" "/usr/lib/jvm/jdk1.8.0/bin/java" 1
$ sudo update-alternatives --install "/usr/bin/javac" "javac" "/usr/lib/jvm/jdk1.8.0/bin/javac" 1
$ sudo update-alternatives --install "/usr/bin/javaws" "javaws" "/usr/lib/jvm/jdk1.8.0/bin/javaws" 1

$ sudo chmod a+x /usr/bin/java
$ sudo chmod a+x /usr/bin/javac
$ sudo chmod a+x /usr/bin/javaws
$ sudo chown -R root:root /usr/lib/jvm/jdk1.8.0

$ sudo update-alternatives --config java
$ sudo update-alternatives --config javac
$ sudo update-alternatives --config javaws
```

2) O servidor da *cloud* pode não ter RAM suficiente para executar o *Elasticsearch*. Para solucionar o problema, podem ser ajustados os parâmetros *-Xms1g -Xmx1g* do ficheiro *jvm.options*, para valores mais baixos, por exemplo, *-Xms512m -Xmx512m*.
3) É possível que a RAM se esgote com o arranque do *Kibana*. Uma possibilidade para ultrapassar esta questão é criar um ficheiro de *Swap*:

```
$ sudo fallocate -l 4G /swapfile
$ sudo chmod 600 /swapfile
$ sudo mkswap /swapfile
$ sudo swapon /swapfile
```

Para que o ficheiro *Swap* seja permanente, deve ser editado o ficheiro */etc/fstab* e acrescentada a linha: **/swapfile none swap sw 0 0**

4) O filtro *prune* do *Logstash* não costuma vir instalado por omissão. Par instalá-lo, deve ser executado o seguinte comando:

   $ sudo bin/logstash-plugin install logstash-filter-prune

5) Para que o exterior possa ter acesso ao servidor, é necessário configurar o ficheiro *kibana.yml* com a entrada:

   server.host: "0.0.0.0"

Instruções de Acesso e Operação

O acesso à interface *Web* do *Kibana* na *cloud* da Hostwinds é feito através do endereço:

http://hwsrv-611644.hostwindsdns.com:5601

A partir da opção *"Discover"*, no painel esquerdo, é possível aceder às funcionalidades de visualização e pesquisa. As pesquisas podem ser efetuadas em qualquer um dos três índices: *syslog*, *logstash-main* e *logstash-pairs*. O primeiro índice agrega as mensagens do tipo Syslog. O segundo índice reúne as mensagens provenientes de sistemas Windows e o último índice arquiva os pares compostos pelos valores identificativos e pelos respetivos pseudónimos.

É possível usar expressões de pesquisa pré-configuradas para aceder a cada tipo de mensagens: Syslog, Windows Event Log, IIS, Apache e Firewall do Windows. Esse acesso é feito através da opção *"Open"*, na barra superior.

Configuração do Envio de Mensagens para o Servidor

O servidor está preparado para receber mensagens de registos no formato JSON nos seguintes portos:

Mensagens do Syslog:	1514/UDP
Mensagens do Windows Event Log:	12201/TCP
Mensagens de Servidores IIS da Microsoft:	12202/TCP
Mensagens da Firewall do Windows:	12203/TCP
Mensagens de Servidores Web da Apache:	12204/TCP

As mensagens do Syslog podem ser enviadas como está explicado na secção 3.5, acrescentando ao ficheiro de configuração do *resyslog.d* a linha:

```
*.* @192.236.160.120:1514
```

As restantes mensagens podem ser enviadas a partir de um agente NXLog, configurado conforme está explicado na secção 5.3.1. Para mais detalhes sobre a configuração do agente, deve ser consultado o Anexo H – Configuração do NXLog.