INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Icarus - A Cloud Security Perspective

HENRIQUE NUNO MARQUES MACIEL
Junho de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Icarus

A Cloud Security Perspective

Henrique Nuno Marques Maciel

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Jorge Pinto Leite

Co-orientador: Paulo Proença

Júri:

Presidente:

António Barros, Professor Adjunto, Instituto Superior de Engenharia do Porto

Vogais:

Osvaldo Santos, Professor Adjunto, Instituto Politécnico de Castelo Branco

Porto, Junho 2021

Resumo

O atual interesse na computação em nuvem tem levado a uma maior adoção deste tipo de soluções por todo o tipo de utilizadores, desde entusiastas até empresas multinacionais. No que toca à implantação de soluções informáticas, este ambiente veio descer consideravelmente a barreira de entrada neste mercado.

Porém, com este novo ambiente o utilizador final tem um controlo muito reduzido sobre a sua infraestrutura e os seus recursos na rede, especialmente quando comparado a uma implantação tradicional *on-premise*. Sendo o perímetro de segurança intangível neste tipo de soluções, a manutenção e operação de um sistema informático na nuvem afugenta muitos potenciais utilizadores e coloca em risco a informação dos seus utilizadores atuais, frequentemente consequência de configurações fracas.

Através do trabalho desenvolvido nesta dissertação, pretendem-se identificar controlos de segurança a aplicar em soluções informáticas hospedadas na nuvem, tendo por base os documentos ISO/IEC 27001 e 27017. Com os controlos de segurança levantados, será feito um levantamento do estado da arte atual na manutenção e operação segura de aplicações e sistemas informáticos.

Com os controlos de segurança e as boas práticas identificadas no estado da arte claramente delineados, será desenvolvida uma aplicação para efeito de prova de conceito que será implantada em nuvem fundamentada nos controlos de segurança analisados.

Finalmente, a prova de conceito será avaliada através de ferramentas de análise de vulnerabilidades, de testes de penetração e o projeto como um todo será avaliado através de um inquérito de satisfação.

Palavras-chave: Computação em nuvem, Segurança informática, Segurança em profundidade, Segurança na nuvem, Amazon Web Services, Terraform

Abstract

A growing interest in cloud computing has led to a greater adoption of this type of solution by all kinds of users, from enthusiasts to multinational corporations. When it comes to deploying IT solutions, this environment has considerably lowered the entry barrier into this market.

However, within this new environment the end user has very little control over their infrastructure and their network resources, especially when compared to a traditional on-premise deployment. As the perimeter security of this type of solutions is intangible, the maintenance and operation of an information system in the cloud scares away many potential users and places the information of its current users at risk, often because of weak configurations.

Through the work in this dissertation, we intend to identify security controls to be applied in computer solutions hosted in the cloud, based on the ISO/IEC 27001 and 27017 standards documents. Having identified the security controls, a survey on the state of the art will be done on the maintenance and operation of secure IT applications and systems.

With the security controls and best practices identified in the state of the art clearly outlined, an application will be developed as a proof of concept, which will be deployed in the cloud founded on the security controls previously analyzed.

As the final step, the proof of concept will be evaluated through vulnerability analysis tools, penetration tests and the project as a whole will be evaluated through a satisfaction survey.

Keywords: Cloud computing, Information Security, Defense in depth, Cloud Security, Amazon Web Services, Terraform

Agradecimentos

Em primeiro lugar, agradeço à minha família e aos meus amigos por todo o apoio ao longo do meu percurso pessoal e académico.

Agradeço ainda aos professores Jorge Pinto Leite e Paulo Proença pela sua disponibilidade, suporte e discernimento durante a escrita deste documento.

Resta ainda agradecer à Celfocus e aos meus supervisores e colegas dentro da empresa que me ajudaram e que me permitiram explorar este tema.

Índice

1 Introdução 1

1.1 Problema 1

1.2 Contexto 1

1.3 Motivação 2

1.4 Objetivos 3
- 1.4.1 Levantamento do estado da arte 3
- 1.4.2 Consolidação dos processos e boas práticas do desenvolvimento de aplicações baseadas em nuvem 3
- 1.4.3 Elaboração de uma arquitetura de referência 3
- 1.4.4 Seleção de um Cloud Service Provider (CSP) e desenvolvimento de uma prova de conceito 3
- 1.4.5 Avaliação da prova de conceito 3
- 1.4.6 Dissertação 4

1.5 Estrutura do documento 4
- 1.5.1 Contextualização 4
- 1.5.2 Engenharia da solução 5
- 1.5.3 Avaliação da solução 5

1.6 Método de investigação 5
- 1.6.1 Pergunta de Investigação 5
- 1.6.2 Resultados da Investigação 6
- 1.6.3 Método de Investigação 6
- 1.6.4 Validação da investigação 6

2 Contexto 7

2.1 Conceitos teóricos 7
- 2.1.1 Defesa em profundidade 7
- 2.1.2 Computação em nuvem 8
- 2.1.3 Tipos de implantação em nuvem 8
- 2.1.4 Modelos de serviço e modelo de responsabilidade partilhada 9
- 2.1.5 Vulnerabilidades de aplicações em nuvem 9
- 2.1.6 Documentos ISO 27001/27002 e 27017 12

2.2 Restrições 13

3 Estado da arte 15

3.1 Controlos ISO 27001/27002 e 27017 15
- 3.1.1 Testes à segurança 17
- 3.1.2 Gestão de eventos de segurança 17
- 3.1.3 Controlo de acessos 18
- 3.1.4 Segurança ao nível operacional 19
- 3.1.5 Gestão da informação 21
- 3.1.6 Conformidade 22
- 3.1.7 Camada tecnológica 22

3.1.8 Camada de arquitetura 23
3.2 Segurança aplicacional 24
3.2.1 OWASP Web Security Testing Guide 24
3.2.2 Técnicas de teste 24
3.2.3 Análise manual 24
3.2.4 Threat Modeling 25
3.2.5 Análise do código-fonte 25
3.2.6 Testes de penetração 26
3.3 Segurança de *containers* e máquinas virtuais 26
3.3.1 Mitigação dos riscos associados a containers e ao host 27
3.4 Segurança da rede 28
3.4.1 Controlos de segurança na rede 28
3.5 DevOps 29
3.5.1 DevSecOps 29
3.5.2 Práticas DevSecOps 29
4 Análise de valor 31
4.1 Comparação processos 31
4.2 Identificação de oportunidade 33
4.3 Análise de oportunidade 33
4.3.1 Forças 33
4.3.2 Fraquezas 34
4.3.3 Oportunidades 34
4.3.4 Ameaças 34
4.4 Proposta de valor 34
4.5 Método de análise hierárquica (AHP) 35
4.5.1 Fase 1 - Construção da árvore hierárquica de decisão 35
4.5.2 Fase 2 - Comparação das alternativas e critérios 36
4.5.3 Fase 3 - Prioridade relativa de cada critério 37
4.5.4 Fase 4 – Avaliar a consistência das prioridades relativas 38
4.5.5 Fase 5 – Construção da matriz de comparação para cada critério 39
4.5.6 Fase 6 – Prioridade composta para as alternativas 40
4.5.7 Fase 7 – Conclusão AHP 41
4.6 Quality Function Deployment 41
5 Design da Arquitetura de referência 43
5.1 Requisitos da arquitetura de referência 43
5.2 Mapeamento controlos ISO 43
5.3 Decisões para o design 44
5.3.1 Arquitetura Multi-VPC 44
5.3.2 Sub-rede pública e privada 45
5.3.3 Comunicação entre VPC 45
5.4 Diagrama de componentes 46

5.5 Diagrama de implantação 47
5.6 Arquitetura de referência proposta 48
6 Prova de Conceito 51
6.1 Requisitos 51
6.2 Escolha das tecnologias 52
6.2.1 Componente Frontend 52
6.2.2 Componente Backend 53
6.2.3 Componente de Base de Dados 53
6.2.4 Implantação da Infraestrutura 54
6.3 Modelo de Domínio 55
6.4 Diagrama de componentes 56
6.5 Diagrama de sequência 57
6.6 Diagrama de implantação em AWS 59
6.6.1 Distribuição CloudFront 59
6.6.2 API Gateway 60
6.6.3 Instância EC2 – Componente backend 60
6.6.4 Bucket S3 – Componente frontend 61
6.6.5 Instância RDS – Componente de base de dados 61
6.6.6 CloudWatch e CloudTrail – Componentes de monitorização e logging 62
6.6.7 Identity and Access Management – AWS IAM 62
6.6.8 Key Management Service – AWS KMS 62
6.6.9 GuardDuty 63
6.6.10 Amazon Inspector 63
6.6.11 Amazon Detective 63
6.6.12 Security Hub 63
6.6.13 AWS Systems Manager 63
6.6.14 AWS Artifact 64
6.7 Resumo cobertura de controlos de segurança 64
6.8 Implementação aplicação 65
6.8.1 Componente backend 66
6.8.2 Componente frontend 67
6.9 Implementação infraestrutura 68
6.9.1 VPC 68
6.9.2 Subnets 69
6.9.3 Network Security Groups 71
6.9.4 Instância EC2 73
6.9.5 Key Pair para instância EC2 74
6.9.6 Instância RDS 75
6.9.7 API Gateway 76
6.9.8 Bucket S3 78
6.9.9 Distribuição CloudFront 79
6.9.10 CloudWatch 80
6.9.11 CloudTrail 82
6.9.12 GuardDuty 84

6.9.13 Amazon Inspector 84
6.9.14 Amazon Detective 87
6.9.15 Security Hub 87

7 Experimentação e Avaliação 89

7.1 Indicadores 89

7.2 Especificação da hipótese 89

7.3 Métodos de avaliação 90
7.3.1 Avaliação de vulnerabilidades 90
7.3.2 Testes de penetração e mecanismos de deteção 91
7.3.3 Inquérito de satisfação 91

7.4 Avaliação dos resultados 91
7.4.1 Avaliação de vulnerabilidades 91
7.4.2 Testes de penetração e mecanismos de deteção 94
7.4.3 Inquérito de satisfação 96

8 Conclusão 107

8.1 Objetivos alcançados 107

8.2 Limitações 108

8.3 Trabalho futuro 108

8.4 Apreciação final 108

Lista de Figuras

Figura 1– Modelo de Canvas para a proposta de valor 35
Figura 2 - Árvore de decisão hierárquica segundo o método AHP 36
Figura 3 - Árvore hierárquica resumo com os pesos relativos de cada critério e alternativa ... 40
Figura 4 - QFD House of Quality 42
Figura 5 - Diagrama de componentes 46
Figura 6 - Diagrama de implantação 47
Figura 7 – Design da arquitetura de rede 48
Figura 8 – Modelo de domínio para prova de conceito 56
Figura 9 – Diagrama de componentes da prova de conceito 57
Figura 10 – Diagrama de sequência da prova de conceito 58
Figura 11 – Diagrama de implantação da prova de conceito em AWS 59
Figura 12 - Diagrama de pacotes do componente *backend* 67
Figura 13 – Captura 1 do serviço GuardDuty 95
Figura 14 – Captura 2 do serviço GuardDuty 96
Figura 15 - Familiaridade dos inquiridos com o tema de computação em nuvem 98
Figura 16 - Familiaridade dos inquiridos com temas de segurança 99
Figura 17 - Avaliação da postura dos inquiridos quanto a soluções informáticas implantadas em nuvem 99
Figura 18 – Distribuição de respostas acerca da gestão centralizada de segredos 102

Lista de Tabelas

Tabela 1 – Métodos exemplificativos de uma abordagem de defesa em profundidade 7
Tabela 2 – Modelo das responsabilidades partilhadas entre cliente e fornecedor de acordo com modelo de serviço 9
Tabela 3 – Onze principais vulnerabilidades de soluções em nuvem em 2019 e os modelos de serviço afetados (Cloud Security Alliance, 2020) 10
Tabela 4 – Causas das vulnerabilidades de computação em nuvem (Cloud Security Alliance, 2020) 11
Tabela 5 - Controlos levantados de acordo com a área de atuação 16
Tabela 6 – Tabela de comparação de metodologias 32
Tabela 7 - Escala fundamental - Níveis de importância de comparações (Saaty, 1980) 37
Tabela 8 - Matriz de comparações par a par dos critérios 37
Tabela 9 - Pesos relativos de cada critério 37
Tabela 10 - Tabela de índices aleatórios de acordo com Thomas Saaty 38
Tabela 11 - Matriz comparativa das alternativas de acordo com configurabilidade 39
Tabela 12 - Matriz comparativa das alternativas de acordo com os custos 39
Tabela 13 - Matriz comparativa das alternativas de acordo com garantias de segurança 40
Tabela 14 - Tabela resumo de cobertura dos controlos de segurança 65
Tabela 15 - Especificação da hipótese 90
Tabela 16 - Métodos de avaliação 90
Tabela 17 - Distribuição de vulnerabilidades Metasploitable 92
Tabela 18 - Distribuição de vulnerabilidades CloudFront 92
Tabela 19 - Distribuição de vulnerabilidades EC2 93
Tabela 20 - Distribuição de alertas ZAP para CloudFront 93

Acrónimos e Símbolos

Lista de Acrónimos

AHP	*Analytic Hierarchy Process*
API	*Application Programming Interface*
CI/CD	*Continuous Integration / Continuous Delivery*
CIDR	*Classless Inter-Domain Routing*
CSP	*Cloud Service Provider*
CVE	*Common Vulnerabilities and Exposures*
DAST	*Dynamic Application Security Testing*
DMZ	*Demilitarized Zone*
DNS	*Domain Name System*
DoS	*Denial of Service*
IaaS	*Infrastructure as a Service*
IaC	*Infrastructure as Code*
IAST	*Interactive Application Security Testing*
IAM	*Identity and Access Management*
IDE	*Integrated Development Environment*
IDS	*Intruder Detection System*
IEC	*International Electrotechnical Commission*
IPS	*Intruder Protection System*
ISMS	*Information Security Management System*
ISO	*International Organization for Standardization*
KMS	*Key Management Service*
NGFW	*Next-Generation Firewall*
NSG	*Network Security Group*

OWASP	*Open Web Application Security Project*
PaaS	*Platform as a Service*
QFD	*Quality Function Deployment*
RDS	*Relational Database Service*
REST	*Representational State Transfer*
RGPD	Regulamento Geral da Proteção de Dados
SaaS	*Software as a Service*
SAST	*Static Application Security Testing*
SDLC	*Software Development Lifecycle*
SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
VM	*Virtual Machine*
VPC	*Virtual Private Cloud*
VPN	*Virtual Private Network*
WAF	*Web Application Firewall*
XSS	*Cross-Site Scripting*
ZAP	*Zed Attack Proxy*

1 Introdução

1.1 Problema

Com uma crescente adoção de soluções baseadas em *cloud* no setor empresarial, deixam de existir preocupações com a aquisição e manutenção do *hardware* e infraestrutura necessários para suportar um sistema informático. Contudo, a adoção destas soluções acarreta diversos riscos que exigem cuidados adicionais.

Atualmente, a informação tem um valor inestimável, e o facto de esta ser armazenada fora do perímetro físico de uma empresa poderá afetar negativamente o negócio. O acesso indevido à informação sensível detida pela empresa poderá levar a multas por quebras de privacidade, para além de afetar a sua imagem pública.

Como resultado de uma implantação em nuvem, uma empresa acaba por ter menor controlo sobre a sua infraestrutura física, uma vez que cabe ao serviço de hospedagem a sua operação e manutenção.

Caso as credenciais de acesso aos serviços de hospedagem sejam intercetadas ou roubadas, um ator malicioso poderá ter acesso completo aos recursos informáticos que compõem os sistemas e infraestrutura da empresa, possivelmente divulgando os dados sensíveis da empresa e dos seus clientes.

1.2 Contexto

Nos últimos anos tem existido um crescido interesse na informatização dos negócios e na flexibilidade trazida pelas soluções informáticas implantadas em nuvem. Este interesse foi ainda impulsionado no último ano pela necessidade de adaptar os negócios para um regime de trabalho remoto, consequência da pandemia COVID-19 (McAfee, 2020).

Acrescendo à flexibilidade deste tipo de implantação, deixa ainda de existir a necessidade de uma organização possuir um *data center*, o que baixa os custos de entrada para a

disponibilização de um produto ou serviço através da *internet*, o que torna este tipo de implantação bastante atrativa para empresas, independentemente da sua dimensão.

Porém, muitas organizações não seguem qualquer tipo de processo para a implantação segura destas soluções, deixando configurações por defeito e contas de acesso com credenciais fracas com acesso a componentes sensíveis do seu sistema, que consequentemente se manifestou num aumento considerável de ataques a soluções em nuvem no último ano (McAfee, 2020).

Através do levantamento do estado da arte pretendem-se identificar os processos documentados no ISO/IEC 27001:2013 para o desenvolvimento de um sistema de gestão da segurança da informação; e no ISO/IEC 27017:2015, para identificar preocupações específicas à segurança de aplicações implantadas em nuvem.

De seguida, será analisado o atual panorama da segurança de soluções informáticas implantadas em nuvem, as principais vulnerabilidades e respetivas mitigações.

Assim que tenham sido levantados os controlos relevantes de cada um dos documentos ISO, será feita uma análise ao estado da arte de como alguns destes controlos são postos em prática no panorama atual.

Para encerrar o capítulo do estado da arte serão ainda analisados controlos de rede que contribuem para a segurança dos componentes da infraestrutura informática de uma organização.

1.3 Motivação

A hospedagem em nuvem possibilitou empresas a possuírem um sistema informático com baixos custos de entrada, facilmente escalável e com uma configurabilidade simplificada (comparativamente a uma implantação *on-premises*).

Porém, como mencionado nas secções anteriores, muitos clientes destas soluções não tomam os devidos cuidados com a implantação em nuvem, o que leva a que quebras de privacidade ocorram diariamente em pequenas empresas e até em multinacionais.

Através do trabalho desta dissertação, pretende-se identificar um conjunto de controlos que, quando aplicados ao longo do *Software Development Lifecycle* (SDLC) de uma aplicação hospedada na nuvem, resulta num produto mais seguro e preparado contra potenciais atacantes maliciosos e quebras de privacidade.

O conjunto de controlos a identificar serão baseados nas diretrizes dos documentos ISO/IEC 27001 e 27017, de forma a transformar as diretrizes numa linguagem mais acessível e sucinta. Pretende-se também identificar como implementar estas diretrizes num contexto prático.

1.4 Objetivos

Nesta secção são identificados os objetivos a atingir através do trabalho desenvolvido durante a escrita da dissertação.

1.4.1 Levantamento do estado da arte

O primeiro objetivo da dissertação passa pela análise acerca do estado da arte da segurança da informação em soluções hospedadas em nuvem.

1.4.2 Consolidação dos processos e boas práticas do desenvolvimento de aplicações baseadas em nuvem

Para a concretização deste objetivo, serão analisados os documentos ISO/IEC 27001/27002 e 27017, de forma a identificar os processos envolvidos no desenvolvimento e manutenção de uma solução informática com elevados critérios para a segurança de informação.

1.4.3 Elaboração de uma arquitetura de referência

Com os processos e boas práticas de desenvolvimento consolidados, será elaborada uma arquitetura de referência da rede que incorpore os controlos de segurança levantados, visando proporcionar uma maior segurança em ambiente de nuvem.

1.4.4 Seleção de um Cloud Service Provider (CSP) e desenvolvimento de uma prova de conceito

Será selecionado um serviço de hospedagem em nuvem público. Seguindo as práticas levantadas da análise dos documentos ISO/IEC, a arquitetura de referência será materializada numa prova de conceito.

A prova de conceito irá consistir numa *web app* para a auditoria aos controlos de segurança levantados da análise dos documentos ISO/IEC.

1.4.5 Avaliação da prova de conceito

Com a prova de conceito materializada e implantada, serão realizadas análises de vulnerabilidades e testes de penetração de forma a avaliar a eficácia dos controlos de segurança utilizados. Será ainda desenvolvido um inquérito de satisfação quanto aos controlos adotados em nuvem como resposta aos principais riscos de soluções expostas à Internet.

1.4.6 Dissertação

Desenvolvimento de uma dissertação acerca dos controlos e processos de cibersegurança em torno do desenvolvimento, manutenção, operação e evolução segura de uma solução informática, incluindo modelos de resposta a incidentes de cibersegurança, baseado nos padrões de referência da indústria: ISO 27001/27017 (segurança da informação) e ISO 22301 (continuidade de negócio). Serão ainda apresentadas a análise e conclusão das descobertas realizadas ao longo da dissertação.

1.5 Estrutura do documento

O documento encontra-se separado em três principais fases, nomeadamente: a Contextualização, a Engenharia da solução e a Avaliação.

Cada uma destas fases agregam diversas secções da dissertação, que serão clarificadas abaixo.

Alguns dos termos encontrados ao longo do documento encontram-se em inglês uma vez que são frequentemente utilizados neste idioma na nomenclatura informática de qualquer país.

1.5.1 Contextualização

A Contextualização é composta pelas secções de Introdução, Contexto, Estado da arte, Método de investigação e Análise de valor.

Na Introdução é apresentada uma contextualização do problema abordado na dissertação, bem como os objetivos traçados para tentar responder ao problema definido.

No Contexto são introduzidos os principais conceitos de negócio para suportar o trabalho da dissertação.

No Estado da arte são analisados os documentos ISO para o levantamento dos controlos e boas práticas a aplicar no desenvolvimento de uma solução segura. São ainda analisados os métodos atualmente utilizados para suportar alguns dos controlos de segurança propostos nos documentos ISO.

No Método de investigação é apresentada a abordagem seguida para a investigação acerca dos temas do problema, como será produzida a resposta ao problema e como o seu resultado será validado.

Na Análise de valor é apresentada uma breve comparação com outras metodologias de desenvolvimento de *software* seguro, e é realizado um posicionamento do processo levantado no estado da arte, de forma a avaliar o valor que este trará para a resolução do

problema. É ainda realizada a seleção de um CSP utilizando o método AHP (*Analytic Hierarchy Process*), para garantir uma solução final que trará mais valor aos *stakeholders*.

1.5.2 Engenharia da solução

De seguida, inicia-se a fase de Engenharia da solução, que é composta pelas secções de Análise e Design e de Implementação.

Na Análise e Design, é feito o levantamento dos requisitos para o desenvolvimento da prova de conceito e são apresentados os artefactos documentais utilizados para o planeamento da solução, possíveis alternativas e são esclarecidas as principais decisões arquiteturais.

Na Implementação são apresentadas as principais decisões tomadas durante a fase de implementação e alguns detalhes para possível recriação do trabalho desenvolvido.

1.5.3 Avaliação da solução

Na fase de Avaliação é encerrada a dissertação, e é composta pelas secções de Avaliação da solução, Validação da hipótese e Conclusão.

Na Avaliação da solução é avaliada a adequação da solução como resposta ao problema da dissertação seguindo os métodos de avaliação propostos.

Na Validação da hipótese é feita a comparação entre os objetivos estabelecidos para o projeto e como os resultados atingidos resolveram (ou não) o problema proposto.

Por fim, na Conclusão são apresentadas as conclusões derivadas do trabalho desenvolvido ao longo da dissertação, as limitações enfrentadas e potencial trabalho futuro dobre o tema abordado.

1.6 Método de investigação

Esta secção pretende elucidar o método de investigação adotado para o trabalho desenvolvido na dissertação.

1.6.1 Pergunta de Investigação

O primeiro passo passa por definir a *research question* do trabalho. Uma vez que o trabalho se foca na identificação de um conjunto de controlos a utilizar no desenvolvimento de uma aplicação baseada na nuvem com elevados critérios para a segurança da informação, a *research question* enquadra-se na avaliação de uma instância (*evaluation of instance*).

1.6.2 Resultados da Investigação

No que toca aos resultados da investigação, é esperado fazer o levantamento de um processo repetível que suporte o desenvolvimento e manutenção de uma aplicação em nuvem segura. Assim que seja sintetizado o conhecimento obtido durante a investigação, este será materializado numa prova de conceito, logo o *research result* enquadra-se numa solução específica (*specific solution*).

1.6.3 Método de Investigação

Os problemas de segurança específicos a soluções baseadas em nuvem constituem um problema real, que será endereçado através do desenvolvimento de um produto exemplo usando todas as boas práticas, controlos e ferramentas identificados durante a dissertação. Uma vez que o objetivo do trabalho passa por contribuir para o desenvolvimento soluções em nuvem mais seguras, o método de investigação enquadra-se na investigação através da ação (*action research*).

1.6.4 Validação da investigação

Como mencionado anteriormente, as descobertas ao longo da investigação serão aplicadas no desenvolvimento de uma prova de conceito para efeitos de teste do seu nível de segurança. Posto isto, a validação da investigação será feita através de um exemplo sob a forma de prova de conceito.

2 Contexto

Nesta secção da dissertação, pretendem-se transmitir os conhecimentos base que suportem e facilitem a compreensão dos temas abordados durante o trabalho.

Serão clarificados os principais conceitos do negócio, bem como as restrições existentes para a aceitação da solução.

2.1 Conceitos teóricos

Nesta secção do documento são abordados os conceitos teóricos relevantes para a compreensão do negócio e dos temas abordados ao longo da dissertação.

2.1.1 Defesa em profundidade

Um dos principais conceitos no desenvolvimento de aplicações seguras passa pela defesa em profundidade. Segundo esta abordagem, a segurança da informação passa por um conjunto de diversas camadas que em conjunto oferecem maiores garantias acerca da segurança da informação de um sistema. Dentro deste conceito, poderão ser distinguidas três camadas: física, técnica e administrativa (Imperva, 2019).

Identificam-se na Tabela 1 apresentada abaixo alguns dos métodos de proteção utilizados em cada uma das camadas identificadas acima.

Tabela 1 – Métodos exemplificativos de uma abordagem de defesa em profundidade

Camada física	Uso de cartões, códigos ou biometria para acessos a zonas restritas
Camada técnica (software, hardware e rede)	*Software* antivírus, uso de encriptação, autenticação multifator, *scanners* de vulnerabilidades, *logging, Firewalls*, *Demilitarized Zone* (DMZ), *Web Application Firewall* (WAF), *Intruder Detection System / Intruder Protection System* (IDS/IPS)
Camada administrativa	Formação, políticas de segurança, procedimentos

Para o levantamento de controlos adequados para uma abordagem de defesa em profundidade, são analisados os documentos ISO 27001, 27002 e 27017. Os controlos selecionados no âmbito do projeto estão explícitos na secção 3.1 desta dissertação.

2.1.2 Computação em nuvem

O modelo de computação em nuvem passa pela acessibilidade a recursos informáticos como servidores de aplicação, máquinas virtuais e soluções de armazenamento via Internet (Amazon Web Services, 2021s).

Este modelo de computação possui certas características que o distingue de soluções tradicionais *on-premises*, nomeadamente:

- Disponibiliza recursos computacionais *on-demand*, onde os utilizadores podem requisitar diferentes recursos conforme a sua necessidade, sem qualquer interação humana com o serviço de hospedagem (Mell e Grance, 2011).
- Suporta a fácil escalabilidade dos recursos computacionais. Deixa de existir a necessidade de disponibilizar recursos em excesso para atender a picos no seu uso, uma vez que estes podem ser facilmente escalados ou mesmo automatizados de forma a responder a um aumento da procura (Amazon Web Services, 2021s).
- O aplicativo cliente para a implantação e manutenção dos recursos é independente da plataforma e dispositivo, sendo normalmente acedido via *browser* (Mell e Grance, 2011).
- Flexibilidade dos planos de pagamento, sendo que são calculados de acordo com o volume de recursos utilizados (Amazon Web Services, 2021s).
- Os recursos computacionais físicos dos serviços de hospedagem são partilhados entre os diversos utilizadores, sendo conhecido como uma arquitetura *multi-tenant* (Mell e Grance, 2011). Uma vez que são partilhados, isso possibilita a prática de custos mais baixos, facilita a escalabilidade de recursos e facilita ainda manutenção dos recursos e espaços físicos (IBM Cloud Education, 2020).

2.1.3 Tipos de implantação em nuvem

Em termos de implantação em nuvem são geralmente identificadas três categorias distintas (existindo autores que distinguem categorias adicionais (Mell e Grance, 2011)), nomeadamente: nuvem pública, nuvem privada e nuvem híbrida.

- Nuvem pública: Os recursos em nuvem são providenciados por um CSP externo à organização, sendo estes mantidos e operados por essa entidade. Os clientes partilham ainda o ambiente de execução com outros utentes. Alguns exemplos de CSP que se enquadram nesta categoria são Microsoft Azure, Amazon Web Services, Google Cloud Platform e IBM Cloud (Red Hat, 2018).
- Nuvem privada: O ambiente de execução é dedicado a uma única entidade ou grupo (Red Hat, 2018).
- Nuvem híbrida: Quando o sistema informático é composto por diversos componentes que comunicam entre si mas têm diferentes tipos de implantação, por exemplo, uma organização tem uma infraestrutura informática *on-premises* mas utiliza ainda um CSP para disponibilizar outro produto ou como medida de armazenamento remoto (Red Hat, 2018).

2.1.4 Modelos de serviço e modelo de responsabilidade partilhada

Atualmente, existem três tipos principais de modelos a que os serviços de computação em nuvem aderem, nomeadamente: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS).

Cada um destes modelos difere na separação das responsabilidades do cliente e do fornecedor. Um sumário da separação de responsabilidades está explícito na Tabela 2 baseada em (Lanfear, 2021), apresentada abaixo, usando como referência as responsabilidades num modelo *on-premises*.

Tabela 2 – Modelo das responsabilidades partilhadas entre cliente e fornecedor de acordo com modelo de serviço

On-Premises	**IaaS**	**PaaS**	**SaaS**
Dados e informação	Cliente	Cliente	Cliente
Controlo de acessos	Cliente	Cliente	Cliente
Aplicação	Cliente	Cliente	Fornecedor
Controlos de Rede	Cliente	Fornecedor	Fornecedor
Sistema Operativo	Cliente	Fornecedor	Fornecedor
Virtualização	Fornecedor	Fornecedor	Fornecedor
Infraestrutura	Fornecedor	Fornecedor	Fornecedor
Componentes Físicos	Fornecedor	Fornecedor	Fornecedor

2.1.5 Vulnerabilidades de aplicações em nuvem

A *Cloud Security Alliance*, uma organização que promove a adoção de boas práticas para soluções em nuvem mais seguras, publicou em 2020 um relatório (Cloud Security Alliance, 2020) a catalogar as onze principais vulnerabilidades que afetaram aplicações em nuvem em 2019, classificadas de acordo com o seu nível de severidade. As principais vulnerabilidades, os modelos de serviços afetados e os seus responsáveis (fornecedor, cliente ou ambos) encontram-se enumeradas na Tabela 3.

Tabela 3 – Onze principais vulnerabilidades de soluções em nuvem em 2019 e os modelos de serviço afetados (Cloud Security Alliance, 2020)

Vulnerabilidade	Modelos afetados	Responsável
Violação de dados	IaaS, PaaS, SaaS	Ambos
Erros de configuração	IaaS, PaaS, SaaS	Cliente
Falta de controlos na arquitetura em nuvem	IaaS, PaaS	Cliente
Controlo de acessos inadequado	IaaS, PaaS	Cliente
Roubo de credenciais	IaaS, PaaS, SaaS	Ambos
Ameaças internas	IaaS, PaaS, SaaS	Cliente
Interfaces e APIs inseguras	IaaS, PaaS, SaaS	Ambos
Fraca gestão da informação	IaaS, PaaS, SaaS	Cliente
Exposição de informação sensível	IaaS, PaaS, SaaS	Ambos
Visibilidade limitada dos recursos em nuvem da organização	IaaS, PaaS, SaaS	Ambos
Uso impróprio de serviços em nuvem	IaaS, PaaS, SaaS	Ambos

Na Tabela 4, apresentada abaixo, são apresentadas as diversas vulnerabilidades, acompanhadas das principais causas e potenciais ameaças associadas.

Tabela 4 – Causas das vulnerabilidades de computação em nuvem (Cloud Security Alliance, 2020)

Vulnerabilidade	Causas	Ameaça
Violação de dados	Ataque malicioso, vulnerabilidade aplicacional, erro humano	Divulgação de informação
Erros de configuração	Credenciais *default*, controlos de segurança desativados	Quebra não-repúdio; Divulgação de informação; Negação de serviço
Falta de controlos na arquitetura em nuvem	Ausência de *firewalls*, ausência de segmentação	Roubo de credenciais; Quebra não-repúdio; Divulgação de informação; Negação de serviço; Elevação de privilégios
Controlo de acessos inadequado	*Passwords* fracas, falta de autenticação multifator, falta de proteção de credenciais	Roubo de credenciais; Quebra não-repúdio; Divulgação de informação; Negação de serviço; Elevação de privilégios
Roubo de credenciais	Ataques de *phishing*, vulnerabilidades aplicacionais	Roubo de credenciais; Quebra não-repúdio; Divulgação de informação; Negação de serviço; Elevação de privilégios
Ameaças internas	Incidentes de colaboradores, ataques maliciosos	Roubo de credenciais; Divulgação de informação; Elevação de privilégios
Interfaces e APIs inseguras	Controlo de acessos fraco, falta de encriptação, falta de monitorização	Quebra não-repúdio; Divulgação de informação; Elevação de privilégios
Falta de controlo sobre os dados da aplicação	Seleção de um CSP sem controlos de segurança adequados	Divulgação de informação; Elevação de privilégios
Exposição de informação sensível	Rotas da API do CSP inseguras, implantação de aplicações sem considerar as interações com os recursos da nuvem	Roubo de credenciais; Quebra não-repúdio; Divulgação de informação; Negação de serviço; Elevação de privilégios
Visibilidade limitada dos recursos em nuvem da organização	Fraca gestão sobre os recursos que são implantados em nuvem, baixo nível de segurança nos seus recursos	Roubo de credenciais; Quebra não-repúdio; Divulgação de informação; Negação de serviço; Elevação de privilégios
Uso impróprio de serviços em nuvem	*Malware* hospedado em nuvem, ataques de negação de serviço, mineração de criptomoedas	Roubo de credenciais; Quebra não-repúdio; Divulgação de informação; Negação de serviço; Elevação de privilégios

2.1.6 Documentos ISO 27001/27002 e 27017

Para a identificação das boas práticas no desenvolvimento de uma aplicação com foco na segurança da informação, foram analisados os documentos ISO 27001 e 27002.

O documento ISO 27001 lista os requisitos para a implementação de um sistema de gestão da segurança da informação (ISMS) e obtenção da respetiva certificação ISO 27001. No âmbito do projeto, e uma vez que o objetivo não passa por obter certificação, foram antes analisados os controlos para a segurança da informação listados no anexo A deste documento.

O anexo A do documento ISO 27001 possui um total de 14 controlos, estando estes divididos num total de 114 subcontrolos (ISMS Online, 2021).

- Anexo 5.A – Políticas da segurança da informação
- Anexo 6.A – Organização da segurança da informação
- Anexo 7.A – Sensibilização
- Anexo 8.A – Gestão de recursos
- Anexo 9.A – Controlo de acessos
- Anexo 10.A – Uso de criptografia
- Anexo 11.A – Segurança física
- Anexo 12.A – Operações do negócio
- Anexo 13.A – Segurança nas comunicações
- Anexo 14.A – Aquisição, desenvolvimento e manutenção do sistema
- Anexo 15.A – Relações com fornecedores
- Anexo 16.A – Gestão dos eventos de segurança
- Anexo 17.A – Continuidade do negócio – Aspetos da segurança da informação
- Anexo 18.A – Conformidade com obrigações contratuais e legais

Acrescentando aos controlos do anexo A do ISO 27001, o ISO 27002 elabora e aprofunda cada um dos controlos, uma vez que estes são apenas brevemente resumidos no ISO 27001. Para além disto, o ISO 27002 não é certificável (Irwin, 2019).

Por sua vez, o documento ISO 27017 adapta e acrescenta novos subcontrolos aos estabelecidos no ISO 27001/27002, desta vez orientados a aplicações hospedadas em nuvem (International Organization for Standardization, 2019). São acrescentados sete novos subcontrolos aos estabelecidos pelo ISO 27001/27002 (Kosutic, 2015), nomeadamente:

- Anexo 6.3.1.A – Definição do modelo de responsabilidades partilhadas
- Anexo 8.1.5.A – Remoção dos recursos do serviço CSP
- Anexo 9.5.1.A – Segregação em ambientes virtuais
- Anexo 9.5.2.A – Uso de técnicas de *hardening* em máquinas virtuais
- Anexo 12.1.5.A - Controlos de segurança para operações administrativas
- Anexo 12.4.5.A - Monitorização de serviços em nuvem
- Anexo 13.1.4.A - Alinhamento da segurança entre redes virtuais e físicas

Assim que foi concluída a análise destes documentos, os seus controlos foram selecionados de acordo com a sua relevância para o projeto, sendo apresentados na secção 3.1 da dissertação.

2.2 Restrições

Os processos e práticas usados para o desenvolvimento da solução informática devem ser fundamentados nos controlos levantados dos documentos ISO 27001/27002 e 27017.

O desenho da arquitetura de referência deve ser independente de qualquer ferramenta ou tecnologia. Apenas depois de selecionar um CSP é que os controlos e ferramentas usados na arquitetura serão mapeados conforme as soluções fornecidas pelo CSP.

3 Estado da arte

Como primeiro ponto no levantamento do estado da arte, é analisado o documento ISO 27001 juntamente com os novos controlos propostos no documento ISO 27017. Através desta análise pretende-se identificar o principal conjunto de processos e boas práticas que contribuem para a defesa em profundidade de um sistema informático em nuvem.

Com os controlos devidamente analisados e sumarizados, segue-se uma análise sobre o estado da arte ao nível da segurança aplicacional.

Para concluir a secção do estado da arte, será feito o levantamento do estado de arte atual no que toca aos controlos utilizados numa infraestrutura em nuvem.

3.1 Controlos ISO 27001/27002 e 27017

De maneira a sintetizar os controlos levantados da análise dos documentos ISO, é apresentada abaixo a Tabela 5 sumário que agrupa os diversos controlos de acordo com a sua área de atuação. Os controlos estão ainda identificados com o respetivo código do anexo. Os novos controlos introduzidos no ISO 27017 encontram-se assinalados com um asterisco.

Tabela 5 - Controlos levantados de acordo com a área de atuação

Área de atuação	Controlos relevantes
Testes à segurança	14.2.8 e 14.2.9: Testes à segurança do sistema
Gestão de eventos de segurança	6.1.1 e A.6.1.2: Definição e separação de responsabilidades 16.1.2: Canais de comunicação de eventos de segurança 16.1.5: Resposta a incidentes de segurança 17.1.1: Planeamento da continuidade da segurança da informação 6.3.1*: Definição do modelo de responsabilidades partilhadas
Controlo de acessos	9.1.1: Política de controlo de acessos 9.2.2: Gestão rigorosa de permissões 9.2.3: Gestão de acessos privilegiados 9.2.4: Informação sobre autenticação 9.2.5: Atualização de acessos dos utilizadores
Segurança ao nível operacional	12.1.1: Documentação das operações 12.3.1: Política de *backup* 12.4.1: Registo de eventos (*logging*) 12.4.2: Proteção de *logs* 12.6.1: Gestão de vulnerabilidades 8.1.5*: Remoção dos recursos do serviço CSP 12.1.5*: Controlos de segurança para operações administrativas 12.4.5*: Monitorização de serviços em nuvem
Gestão da informação	8.1.1: Inventário dos recursos computacionais 8.2: Classificação da informação
Conformidade	14.1.1: Requisitos da segurança da informação 18.1.1: Identificação da legislação e requisitos contratuais
Camada tecnológica	12.2.1: Controlos contra *malware* 14.2.1: Política de programação segura
Camada de arquitetura	10.1.1: Política de uso de controlos criptográficos 13.1.1: Controlos de segurança na rede 13.1.3: Segregação na rede 14.2.6: Ambiente de trabalho seguro 9.5.1*: Segregação em ambientes de computação virtuais 9.5.2*: Uso de técnicas de *hardening* em máquinas virtuais 13.1.4*: Alinhamento da segurança entre redes virtuais e físicas

Abaixo encontram-se resumidos os diversos controlos levantados através da análise dos documentos ISO.

De acordo com os controlos levantados, é criada uma *framework* na forma de *checklist* para a validação de um sistema de acordo com os controlos ISO. A *framework* pode ser consultada no Anexo C da dissertação.

3.1.1 Testes à segurança

3.1.1.1 Testes à segurança do sistema

Devem ser realizados testes às medidas de segurança adotadas ao longo do desenvolvimento de um sistema. Estes testes devem ser realizados e aprovados por uma autoridade competente na área de segurança. Os resultados esperados devem ser registados antes de iniciar a fase de testes, servindo estes como medidores para a avaliação da adequação das funcionalidades testadas (ISMS Online, 2020d).

Devem ainda ser realizados testes de aceitação, que também devem ser definidos antes de começar a fase de teste. Os testes de aceitação devem ainda incluir testes à segurança (ISMS Online, 2020d).

3.1.2 Gestão de eventos de segurança

3.1.2.1 Definição e separação de responsabilidades

Todas as responsabilidades relacionadas com a segurança da informação devem ser devidamente definidas e atribuídas. Todos os atores que participam na segurança da informação devem ter as suas responsabilidades bem definidas, de forma a não existir ambiguidade nas funções que devem desempenhar (ISMS Online, 2020h).

Cada responsável pela segurança da informação deve ter conhecimento das ameaças relevantes ao seu trabalho desenvolvido, e deve receber formação frequente na matéria (ISMS Online, 2020h).

É necessário estabelecer uma linguagem comum, visando a separação clara de responsabilidades. Cada um dos atores que participam na segurança da informação deve ter os seus deveres claramente delineados.

3.1.2.2 Canais de comunicação de eventos de segurança

Devem existir canais apropriados para reportar quaisquer eventos de segurança de um sistema. Os colaboradores da organização devem ter conhecimento da sua responsabilidade para reportar tais eventos adequadamente (ISMS Online, 2020e).

3.1.2.3 Resposta a incidentes de segurança

Deve existir um indivíduo responsável por coordenar os esforços de resposta a um incidente de segurança. Para além disto, deve procurar recolher qualquer informação pertinente do incidente, por exemplo, qual foi a fonte do ataque e quais foram os atacantes envolvidos.

Deve ainda comunicar este incidente aos órgãos pertinentes (quadros administrativos, autoridades, etc.), conforme a dimensão e impacto do incidente (ISMS Online, 2020e).

3.1.2.4 Planeamento da continuidade da segurança da informação

A organização deve considerar os requisitos da segurança da informação em situações de risco previstas no plano de continuidade de negócio. Para cada tipo de evento de crise que a organização considere deve existir um plano traçado para a manutenção da segurança da informação nesse cenário (ISMS Online, 2020f).

Assim que estes requisitos tenham sido identificados, devem ser implementadas políticas e medidas que permitam responder a esses requisitos em situações de crise (ISMS Online, 2020f).

3.1.2.5 Definição do modelo de responsabilidades partilhadas

As responsabilidades partilhadas entre o cliente e o CSP devem estar claramente identificadas, documentadas e devem ser implementadas por ambos os participantes (Sysprove, 2020).

3.1.3 Controlo de acessos

3.1.3.1 Política de controlo de acessos

Deve ser estabelecida uma política para o controlo de acessos aos recursos do negócio da organização. Os controlos em vigor podem ser físicos, como por exemplo o uso de portas de fecho eletrónico que necessitam de um cartão ou código de acesso. Poderão ser ainda controlos informatizados, como a restrição das permissões dos utilizadores no sistema informático da organização (Disterer, 2013; ISMS Online, 2020j).

A política a desenvolver deve detalhar quem necessita de acesso a que informação, estando alinhado conforme a classificação da informação definida na secção da gestão dos recursos.

Esta política deve também considerar a gestão das contas dos utilizadores dos sistemas informáticos da empresa, as permissões atribuídas e o estabelecimento de um período de tempo para uma revisão das mesmas (Disterer, 2013; ISMS Online, 2020j).

Adicionalmente, quando um utilizador termina o seu contrato ou mesmo no caso de mudança do seu papel na organização, deve ter as suas permissões revistas e atualizadas de acordo com as suas novas responsabilidades (ISMS Online, 2020j).

A principal medida adotada para o controlo de acessos é a política de privilégio mínimo. De acordo com esta medida, cada utilizador de um sistema informático apenas terá acesso à informação necessária para realizar o seu trabalho (ISMS Online, 2020j).

3.1.3.2 Gestão rigorosa de permissões

Deve existir um processo bem definido para a atribuição e anulação de permissões para todos o tipo de utilizadores de todos os sistemas informáticos da organização. Os utilizadores devem ter permissões necessárias para realizar o seu trabalho, conforme a política de privilégio mínimo (ISMS Online, 2020j).

3.1.3.3 Gestão de acessos privilegiados

A atribuição de permissões com maiores privilégios devem ser controlados mais rigorosamente, dado o seu nível de acesso a componentes essenciais e informação mais sensível (ISMS Online, 2020j).

3.1.3.4 Informação sobre autenticação

Qualquer tipo de credenciais fornecidas por defeito, seja na adoção de um novo componente do sistema ou na criação de uma nova conta de utilizador, devem ser alteradas o mais rápido possível. Deve-se evitar o armazenamento impróprio de informação de autenticação, por exemplo, armazenar palavras-passe em texto num ficheiro facilmente acessível. Caso esta informação seja comprometida, os dados de autenticação devem ser imediatamente alterados (ISMS Online, 2020j).

3.1.3.5 Atualização de acessos dos utilizadores

As permissões de acesso dos utilizadores devem ser revistas em períodos regulares ou em situações em que um utilizador muda de funções da organização. Utilizadores com acessos mais privilegiados devem ter as suas permissões revistas mais frequentemente que os demais utilizadores (ISMS Online, 2020j).

3.1.4 Segurança ao nível operacional

3.1.4.1 Documentação das operações

Os procedimentos em vigor para a operação da infraestrutura informática devem estar devidamente documentados e facilmente acessíveis aos utilizadores relevantes. Entre estes procedimentos, encontram-se por exemplo o *startup* e *shutdown* do sistema e mecanismos de cópias de segurança. Quando estes sistemas se encontram em nuvem, este tipo de

operações é geralmente tratado pelo serviço de hospedagem. Nos casos de sistemas *on-premises,* esta documentação será muito mais relevante (ISMS Online, 2020b).

3.1.4.2 Política de *backup*

Deve ser estabelecida uma política para o *backup* de dados importantes para o negócio. Esta política deve traçar que dados devem ser guardados, métodos a utilizar (*backups* completos, incrementais ou diferenciais), a frequência dos *backups* e a frequência com que estes devem ser testados, de forma a garantir a sua integridade (ISMS Online, 2020b).

As políticas de *backup* de uma organização devem assentar no seu plano de continuidade do negócio, uma vez que a frequência e metodologia dos *backups* estarão proximamente relacionados com a disponibilidade do sistema e o tempo de recuperação (ISMS Online, 2020b).

3.1.4.3 Registo de eventos (*logging*)

Devem ser registados diversos eventos de um sistema sob a forma de *logs*, nomeadamente a atividade dos seus utilizadores, falhas no funcionamento e eventos de segurança. Estes *logs* devem ser armazenados e monitorizados regularmente de forma a existir um registo irrefutável em situações de incidentes de segurança (ISMS Online, 2020b).

3.1.4.4 Proteção de *logs*

O armazenamento de *logs* deve ser devidamente considerado, nomeadamente o uso de encriptação, uma vez que estes geralmente contêm informação sensível ou informação que poderá identificar os utilizadores do sistema (ISMS Online, 2020b). A aplicação de encriptação ajuda ainda a garantir a sua irrefutabilidade.

3.1.4.5 Gestão de vulnerabilidades

Devem ser geridas as vulnerabilidades dos componentes do sistema informático da organização. Para este fim, devem ser utilizadas fontes de informação para identificar novas vulnerabilidades técnicas nos componentes em uso (BreachLock, 2019; Infosavvy, 2020a). Uma das principais fontes deste tipo de informação é a lista *Common Vulnerabilities and Exposures* (CVE) da organização MITRE.

De acordo com o impacto que a vulnerabilidade terá no funcionamento do negócio, esta deve ser avaliada e, de acordo com a sua criticidade, deve ser endereçada atempadamente (ISMS Online, 2020b). Uma vulnerabilidade crítica poderá implicar o *shutdown* completo do sistema em questão, enquanto que uma vulnerabilidade média ou baixa poderá apenas implicar uma atualização do componente vulnerável. As medidas a adotar devem ser proporcionais ao risco

associado à vulnerabilidade, e devem ser previamente estabelecidas pela organização (BreachLock, 2019; Infosavvy, 2020a; ISMS Online, 2020b).

3.1.4.6 Remoção dos recursos do serviço CSP

Quaisquer recursos da organização devem ser devidamente removidos ou devolvidos caso termine o contrato de utilização desse serviço de hospedagem (Sysprove, 2020).

3.1.4.7 Controlos de segurança para operações administrativas

As operações administrativas da plataforma de hospedagem em nuvem devem estar claramente definidas, documentadas e devem ser monitorizadas (Sysprove, 2020).

3.1.4.8 Monitorização de serviços em nuvem

A plataforma de hospedagem deve fornecer a capacidade de monitorizar a atividade no ambiente nuvem da organização (Sysprove, 2020).

3.1.5 Gestão da informação

3.1.5.1 Inventário dos recursos computacionais

Quaisquer recursos informáticos detidos pela organização devem ser identificados e mantidos atualizados ao longo do seu funcionamento. Todos os recursos devem ter ainda um dono, que será responsável pela sua gestão. Para facilitar a manutenção do inventário, deve ser ainda mantido um registo dos recursos da empresa. Este registo deve ser mantido atualizado com quaisquer alterações aos donos do recurso ou outros eventos pertinentes (ISMS Online, 2020i).

3.1.5.2 Classificação da informação

A informação detida pela organização deve ser classificada conforme o seu valor, importância para o negócio e a sensibilidade dos seus dados. Devem ser estabelecidos diversos níveis de confidencialidade de forma a assegurar que a informação se encontra devidamente segregada. Com a informação devidamente classificada, será mais fácil fazer o controlo do seu acesso (ISMS Online, 2020i).

Esta classificação deve ter em consideração tanto os formatos eletrónicos como físicos em que a informação poderá estar detida (ISMS Online, 2020i).

3.1.6 Conformidade

3.1.6.1 Requisitos da segurança da informação

Quando se desenvolve um novo produto ou se modifica um produto existente, é necessário analisar os requisitos da segurança levantados. Isto deve ser feito antes de começar o desenvolvimento da solução de forma a identificar os requisitos necessários e reduzir os riscos e custos futuros (ISMS Online, 2020d).

3.1.6.2 Identificação da legislação e requisitos contratuais

As legislações e requisitos contratuais aos quais a organização está sujeita devem ser devidamente documentados, juntamente com as medidas em vigor para responder a essas restrições. A natureza destas restrições estarão proximamente relacionadas com o negócio da organização, os dados que esta trata, bem como a sua localização geográfica (ISMS Online, 2020g).

3.1.7 Camada tecnológica

3.1.7.1 Controlos contra *malware*

Devem existir controlos para a deteção e prevenção de *malware*. O uso de armazenamento amovível bem como a instalação de *software* pelos utilizadores deve ser devidamente gerido. Os dispositivos em uso devem ainda ser mantidos atualizados, uma vez que *malware* costuma abusar de vulnerabilidades presentes num sistema operativo ou aplicativo desatualizado (ISMS Online, 2020b).

3.1.7.2 Política de programação segura

Devem ser aplicadas e estabelecidas políticas para a programação e desenvolvimento de aplicações na organização. Estas políticas visam garantir que os ambientes de programação são seguros e que a implementação encoraja o uso de programação segura (ISMS Online, 2020d).

Estas considerações devem estar presentes ao longo de todo o processo de desenvolvimento de uma solução. Durante a fase de implementação, a seleção de uma linguagem de programação ou ferramenta poderá trazer diferentes vulnerabilidades, pelo que poderá ser necessário aplicar técnicas de *hardening*, como por exemplo, usar de ferramentas de proteção contra ataques de *buffer overflow* no caso da linguagem C (ISMS Online, 2020d).

3.1.8 Camada de arquitetura

3.1.8.1 Política de uso de controlos criptográficos

O uso de encriptação deve ser avaliado e aplicado conforme as necessidades do negócio, uma vez que a sua aplicação implica um processamento mais lento das transações. Assim, devem ser definidas políticas que permitam identificar os requisitos do negócio que exijam a aplicação de encriptação, e os critérios a que o algoritmo ou tecnologia criptográfica deve obedecer (ISMS Online, 2020a).

Devem ser ainda estabelecidas políticas que endereçam o uso e armazenamento de chaves criptográficas ao longo do seu ciclo de vida (ISMS Online, 2020a).

3.1.8.2 Controlos de segurança na rede

A rede da organização deve ser gerida de forma a proteger a informação transmitida pelos seus componentes. Alguns mecanismos tradicionais utilizados para a proteção da informação incluem o uso de *firewalls*, de sistemas de deteção de intrusão e a segregação dos recursos da rede (ISMS Online, 2020c).

3.1.8.3 Segregação na rede

Agrupamentos de diferentes serviços de informação e de utilizadores devem ser segregados na rede da organização. Poderá existir ainda uma segregação de acordo com os departamentos da organização. Esta segregação dos componentes da rede deve suportar as políticas de classificação da informação (ISMS Online, 2020c).

3.1.8.4 Ambiente de trabalho seguro

Os ambientes de trabalho devem estar devidamente protegidos contra acessos não autorizados de forma a evitar a introdução de vulnerabilidades acidental ou maliciosa num produto (Infosavvy, 2020b; ISMS Online, 2020d).

3.1.8.5 Segregação em ambientes de computação virtuais

Um dado serviço a correr numa plataforma em nuvem deve estar devidamente isolado e protegido de outros aplicativos que partilhem a mesma infraestrutura (Sysprove, 2020).

3.1.8.6 Uso de técnicas de *hardening* em máquinas virtuais

As máquinas virtuais em execução num ambiente de nuvem devem ser adequadamente protegidas contra vulnerabilidades conhecidas através de aplicação de técnicas de *hardening* de acordo com o seu sistema operativo, distribuição e versão (Sysprove, 2020).

3.1.8.7 Alinhamento da segurança entre redes virtuais e físicas

A rede virtual implantada deve seguir e obedecer às políticas da segurança da informação estabelecidas para a rede física da organização (Sysprove, 2020).

3.2 Segurança aplicacional

Conforme levantado na análise dos controlos dos documentos ISO, uma das principais medidas para a promoção e manutenção da segurança aplicacional passa pela elaboração de testes à segurança.

3.2.1 OWASP Web Security Testing Guide

A fundação OWASP é uma organização com o principal objetivo de promover e melhorar a segurança de aplicações. Entre os diversos projetos que mantêm para este fim, para o levantamento do estado da arte da segurança ao nível aplicacional será analisada a sua *framework* de testes à segurança, o OWASP *Web Security Testing Guide*, nomeadamente a versão 4.2.

3.2.2 Técnicas de teste

De acordo com a framework Web Security Testing Guide, são distinguidas quatro categorias utilizadas na elaboração de testes a uma aplicação web (OWASP Foundation, 2020):

- Análise manual
- *Threat Modeling*
- Análise do código-fonte
- Testes de penetração

3.2.3 Análise manual

A análise manual de código é feita através da revisão humana de código produzido, documentos arquiteturais bem como outros artefactos pertinentes ao *design* de um aplicativo. Uma vez que se trata de uma revisão manual, não é necessário recorrer a ferramentas externas, é promovido o trabalho em equipa e a difusão de uma cultura orientada à segurança. Porém, à medida que uma organização e o seu trabalho crescem, este processo de teste torna-se muito mais intensivo a nível de tempo e recursos. A eficácia do processo estará também bastante dependente dos conhecimentos e capacidades do testador (OWASP Foundation, 2020).

3.2.4 Threat Modeling

Uma das principais técnicas utilizadas para a identificação de ameaças externas e o seu impacto nas aplicações e sistemas da organização passa pela elaboração de um modelo *threat modeling*. Nesta abordagem, são identificados os riscos associados à aplicação e são traçadas técnicas de mitigação para potenciais vulnerabilidades que possam ser encontradas. Os modelos devem ser criados no início do SDLC e mantidos atualizados conforme a evolução da aplicação e dos seus componentes (OWASP Foundation, 2020).

O OWASP Web Security Testing Guide distingue cinco fases para a criação de um modelo threat modeling (OWASP Foundation, 2020):

- Decompor a aplicação de acordo com os seus componentes e canais de comunicação
- Definir os ativos da organização e classificá-los de acordo com o seu impacto no negócio
- Baseando-se nas descobertas dos pontos anteriores, identificar potenciais vulnerabilidades do sistema
- Identificar potenciais ameaças de acordo com potenciais vetores de ataque que possam ser identificados por atacantes externos
- Desenvolver estratégias de mitigação para as ameaças mais relevantes

A elaboração de um modelo *threat modeling* é uma ferramenta importante para o mapeamento de um sistema a identificação de potenciais vetores de ataque, porém não se manifesta diretamente num aumento da qualidade do *software* (OWASP Foundation, 2020).

3.2.5 Análise do código-fonte

Em termos de identificação de potenciais vulnerabilidades, a melhor abordagem passa pela análise do código-fonte. Muitas vulnerabilidades potencialmente perigosas são difíceis de identificar usando outros métodos de teste (OWASP Foundation, 2020).

A análise do código-fonte pode ser feita manualmente, porém também pode ser assistida por ferramentas automatizadas. Este tipo de ferramentas costumam classificar-se em três categorias, distinguidas de acordo com o seu método de operação (Koussa, 2018):

- *Static Application Security Testing* (SAST) – Realizam uma análise estática do código-fonte (teste *white box*). São utilizados durante a fase da implementação para a deteção de vulnerabilidades no código. Este tipo de ferramentas não necessitam de executar o código, detetam vulnerabilidades de acordo com *templates* e regras pré-definidas. Atualmente são utilizados como extensões de ambientes de desenvolvimento (IDEs).
- *Dynamic Application Security Testing* (DAST) – Procuram vulnerabilidades numa aplicação em execução (teste *black box*). Utiliza métodos de injeção de dados

maliciosos de forma a detetar a presença de vulnerabilidades comuns em aplicações *web*.

- *Interactive Application Security Testing* (IAST) – Visam endereçar funcionalidades que os SAST não possuem, como identificação de vulnerabilidades em aplicações que usem *frameworks* e bibliotecas externas. As ferramentas IAST conciliam os métodos das SAST como das DAST, pois tem mecanismos de análise de uma aplicação em execução que pode ser utilizado ao longo de todo o SDLC. Uma vez que funciona ao nível aplicacional (teste *grey box*), tem acesso a mais informação sobre o funcionamento de uma aplicação que as abordagens das SAST e DAST.

3.2.6 Testes de penetração

A última categoria de testes à segurança a abordar passa pelos testes de penetração. Tipicamente, estes testes são realizados num contexto *black box*, em que a equipa de teste tenta identificar vulnerabilidades numa aplicação em execução (OWASP Foundation, 2020).

Estes tipos de testes são suportados por diversos tipos de ferramentas que permitem explorar vulnerabilidades comuns. As vulnerabilidades encontradas são dependentes das tecnologias usadas na aplicação (OWASP Foundation, 2020).

Dada a grande variedade e disponibilidade deste tipo de ferramentas, os testes de penetração são mais rápidos que outras alternativas e não exigem um conhecimento tão aprofundado como no caso de análise manual do código-fonte. Porém, ao contrário de outras alternativas, como necessita de uma aplicação em execução, apenas poderá ser utilizado numa fase mais tardia do SDLC (OWASP Foundation, 2020).

3.3 Segurança de *containers* e máquinas virtuais

A virtualização é um dos pilares na qual a computação em nuvem assenta. É através da simulação de *hardware* que é possível a execução de diferentes sistemas operativos num único servidor físico. Graças à virtualização é possível suportar diversos utilizadores num só servidor, seguindo um modelo *multi-tenant* (Souppaya, Morello e Scarfone, 2017).

Existe ainda a virtualização de aplicações, onde o *kernel* de um sistema operativo *host* é partilhado entre diversas aplicações. Componentes do sistema operativo garantem que estas aplicações executam isoladas umas das outras, tendo apenas acesso ao sistema operativo (Souppaya, Morello e Scarfone, 2017).

Atualmente as soluções de virtualização de aplicações têm como objetivo fornecer uma maneira de executar aplicações de forma fácil, reutilizável e automatizada. Estas soluções são conhecidas como *containers*. São as imagens de *containers* que garantem a portabilidade desta solução, pois contêm os ficheiros necessários para correr um *container*, independentemente do ambiente de execução (Souppaya, Morello e Scarfone, 2017).

3.3.1 Mitigação dos riscos associados a containers e ao host

De acordo com os controlos levantados na análise dos documentos ISO 27001 e 27017, o principal controlo que se enquadra na segurança de *containers* é o controlo 9.5.2 do ISO 27017, que passa pelo uso de técnicas de *hardening* em máquinas virtuais (VMs).

Para responder a esta necessidade poderão ser usadas imagens de VMs *hardened* por entidades externas (Center for Internet Security, 2021), uma vez que o processo de *hardening* manual de um sistema operativo é bastante caro a nível de tempo e recursos. Outra medida passa pelo uso de um sistema operativo próprio para *containers*. Estas soluções encontram-se bastante limitadas em termos de funcionalidade de forma a reduzir a superfície de ataque do sistema operativo *host*.

3.4 Segurança da rede

De acordo com os controlos levantados na análise dos documentos ISO, no anexo 13.1.1 - "Controlos de segurança na rede", devem existir mecanismos para a proteção da informação, que serão identificados nesta secção.

3.4.1 Controlos de segurança na rede

3.4.1.1 DMZ

Uma vez que queremos restringir ao máximo o tráfego que chega ao nosso sistema, queremos começar por estabelecer uma DMZ no perímetro da nossa rede interna. De acordo com (Mitchell, 2020), ao usar uma DMZ criamos uma sub-rede entre a rede interna e a rede externa, onde podemos adicionar proteção adicional antes dos pedidos chegarem à rede interna.

Alguns componentes que podemos utilizar para construir uma camada de proteção na DMZ são (Varma, 2018):

- *Firewalls* – Realizam a filtragem dos pedidos de acordo com as regras estabelecidas. Esta funcionalidade também é frequentemente desempenhada por *Network Security Groups* numa implantação em nuvem.
- *Intruder Detection System* (IDS) – Componentes que identificam e reportam atividade potencialmente maliciosa. Quando têm capacidade de responder a esta atividade classificam-se também como IPS.
- WAF – Permitem realizar a filtragem de pedidos HTTP ao nível da camada aplicacional (camada 7 do modelo OSI). Permitem mitigar ataques comuns em aplicações *web* como *Structured Query Language* (SQL) *Injection* ou *Cross-Site Scripting* (XSS).
- Proxy – Permitem ocultar endereços IP e a *stack* tecnológica da rede interna; realizam balanceamento da carga de processamento; podem fazer *cache* de conteúdos estáticos para reduzir a carga na rede.
- *Load balancers* – Tratam do balanceamento da carga computacional pelas diversas instâncias que possam existir de um recurso computacional.

Atendendo ainda ao controlo 13.1.3 do ISO, devem ser criadas sub-redes virtuais para a segregação dos diversos componentes do sistema. Uma vez que se trata de uma implantação em 3 *tiers*, teremos uma sub-rede para cada o componente do cliente, do servidor e da base de dados.

Utilizando os controlos de segurança levantados, será criada na secção **Error! Reference source not found.** da dissertação uma arquitetura da rede conforme os objetivos estabelecidos para o projeto.

3.5 DevOps

Graças à hospedagem em nuvem e à rapidez da implantação trazida pela tecnologia, tem havido uma crescente adoção de processos de CI/CD (*continuous integration* /*continuous delivery*) de forma a obter *feedback* mais rápido dos utilizadores finais e a reduzir os riscos e custos associados à implantação. Estes processos utilizam ferramentas geridos por equipas operacionais, cuja colaboração e comunicação com as equipas de desenvolvimento resultou no conceito de DevOps (Myrbakken e Colomo-Palacios, 2017).

3.5.1 DevSecOps

Seguindo as práticas DevOps, as responsabilidades operacionais são integradas em todas as fases de desenvolvimento, que juntamente com a automação de processos permitem entregar *software* fiável mais rapidamente. A inclusão de práticas de segurança neste processo ficou conhecido como DevSecOps, que tem como objetivo promover a colaboração entre as equipas operacionais, de desenvolvimento e de segurança (Myrbakken e Colomo-Palacios, 2017).

Através da prática de DevSecOps, é promovida uma cultura onde a segurança é uma responsabilidade transversal a todas as equipas que participam na criação e manutenção de um produto (Myrbakken e Colomo-Palacios, 2017).

3.5.2 Práticas DevSecOps

As principais práticas DevSecOps adotadas atualmente enquadram-se nas diversas fases do SDLC (Myrbakken e Colomo-Palacios, 2017), nomeadamente:

- Planeamento: Elaboração de modelos de Threat Modeling de forma a identificar os principais vetores de ataque de um sistema a desenvolver.
- Desenvolvimento: Estabelecimento de políticas de programação, e investimento na formação de colaboradores nas práticas de segurança. Uso de ferramentas automáticas para a deteção de vulnerabilidades, por exemplo SonarLint (SonarLint, 2021), dentro dos ambientes de programação.
- *Build*: Uso de ferramentas de análise estática de código como SonarQube (SonarQube, 2021).
- Testes: Uso de ferramentas de análise dinâmica e elaboração de testes de penetração.
- Operação: Análises de vulnerabilidades e exercícios Blue-Team/ Red-Team.

- Monitorização: Uso de *logs* em todos os recursos do sistema e monitorização do inventário informático.

4 Análise de valor

A análise de valor tem como objetivo analisar um dado produto ou serviço de forma a atingir um valor aumentado com custos mais baixos. Para esse fim, este capítulo da dissertação passa pela identificação e análise da oportunidade e pela elaboração da proposta de valor.

4.1 Comparação processos

Como primeiro passo na elaboração da análise de valor é feita uma avaliação de outras metodologias/ processos para o desenvolvimento de aplicações seguras. Desta forma é possível identificar potenciais oportunidades para que a metodologia a desenvolver seja mais atrativa que outras alternativas.

Foram consideradas na análise as metodologias *Security Development Lifecycle* em Azure (Lanfear, Coulter e Baldwin, 2019) e *Open Software Assurance Maturity Model* da OWASP (Deleersnyder e De Win, 2020).

Na Tabela 6 são compilados os principais temas abordados em cada uma das metodologias.

Tabela 6 – Tabela de comparação de metodologias

	Metodologia projeto	**SDLC em Azure**	**OpenSAMM**
Independente da tecnologia (agnóstica)	Sim	Não	Sim
Aborda todo o SDLC	Sim	Sim	Sim
Cobre a gestão de eventos de segurança	Sim	Pouco	Sim
Cobre segurança da infraestrutura	Sim	Sim	Sim
Preparação de respostas a incidentes de segurança	Sim	Sim	Sim
Propostos controlos para a arquitetura da rede	Sim	Poucos	Não
Reforça importância da monitorização e alarmística	Sim	Sim	Sim
Propõe diversos métodos de teste da segurança (ex: SAST, DAST, PenTesting)	Sim	Sim	Sim
Reforça importância do controlo de acessos	Sim	Não	Não
Desenvolvido para um contexto de implantação em nuvem	Sim	Sim	Não
Acompanhado de framework para avaliação dos controlos propostos	Sim	Sim	Sim

Como podemos observar através da análise das diferentes metodologias, a principal oportunidade identificada passa pela definição de uma *framework* para o desenvolvimento de aplicações em nuvem independentes do CSP e ferramentas a utilizar.

Uma vez que o ISO 27001 se trata de um documento extenso e é mais relevante a organizações que pretendam obter a respetiva certificação, poderá existir interesse numa *framework* fundamentada nos controlos mais relevantes para a segurança levantados do documento ISO.

4.2 Identificação de oportunidade

As soluções em nuvem têm um papel cada vez mais importante nas vidas pessoais e profissionais das pessoas, desde aplicações de armazenamento de ficheiros como Dropbox ou iCloud, até à hospedagem de aplicações empresariais que suportam negócios que realizam centenas de milhares de transações diariamente.

Contudo, o facto das soluções em nuvem estarem fora do alcance ou perímetro físico das empresas, aumenta os níveis desconfiança para com o uso desta tecnologia. Para a avaliação da confiança que as empresas depositam nos serviços em nuvem, foi realizado um inquérito em 2019 (Thales Group, 2019). De acordo com os resultados, 49% das empresas inquiridas acreditam que as aplicações em nuvem constituem o maior alvo para ataques informáticos. Das empresas inquiridas, 94% atualizaram as suas políticas de segurança de acordo com quebras da privacidade nos últimos 12 meses (no ano do inquérito).

Por estes elevados níveis de cuidado e desconfiança para com soluções informáticas hospedadas em nuvem, podemos aferir que quanto maiores as garantias fornecidas para com a segurança em nuvem, mais confortável uma empresa ficará em relação à segurança da sua informação como da informação dos seus clientes.

Adicionalmente, com o surgimento da pandemia COVID-19, no ano de 2020 houve um aumento de 50% no uso empresarial de serviços em nuvem, com um aumento de 630% no número de ataques externos ao serviços em nuvem (McAfee, 2020).

Uma vez que o trabalho da dissertação passa pela identificação de um processo para o desenvolvimento de aplicações em nuvem mais seguras, quanto maiores as preocupações para com a segurança de um produto desenvolvido, maior será a confiança dos clientes nos produtos de uma organização.

4.3 Análise de oportunidade

Para a análise da oportunidade identificada no ponto anterior, será feito o enquadramento estratégico em relação ao mercado e negócio da organização, usando o modelo SWOT.

4.3.1 Forças

- Soluções em nuvem com elevada procura
- Componentes de segurança suportam a escalabilidade do sistema
- Principais causas de violação de dados podem ser mitigadas com práticas de segurança
- Tempos mais rápidos de lançamento de novas soluções

4.3.2 Fraquezas

- Maiores requisitos de gestão para a manutenção de elevados critérios de segurança
- Maior número de componentes para a segurança implica maiores custos
- Necessidade de conhecimentos sobre segurança para uma operação segura

4.3.3 Oportunidades

- Mercado considera soluções em nuvem mais inseguras que aplicações tradicionais
- Crescido interesse na flexibilidade das soluções em nuvem como resultado da pandemia
- Aumento de ataques a soluções em nuvem leva a uma procura de soluções mais seguras

4.3.4 Ameaças

- *Vendor lock-in* como consequência do uso de controlos de segurança específicos a um CSP

4.4 Proposta de valor

Para o enquadramento da proposta de valor a apresentar a possíveis *stakeholders*, foi usado o modelo Canvas.

De acordo com este modelo, começamos por identificar o perfil do cliente, nomeadamente, o trabalho que este quer realizar, e o que ele pretende atingir (*gains*) e o que pretende evitar (*pains*).

De seguida, enquadramos o nosso produto de acordo com a proposta de valor, indicando como pode gerar os *gains* desejados pelo cliente final, bem como pode aliviar as suas *pains*.

O modelo Canvas elaborado para o contexto do projeto pode ser consultado na Figura 1, apresentada abaixo.

Figura 1– Modelo de Canvas para a proposta de valor

4.5 Método de análise hierárquica (AHP)

Um dos componentes do projeto da dissertação passa pela seleção de um CSP para materializar a arquitetura em nuvem desenvolvida. Para selecionar o melhor serviço de hospedagem em nuvem, será utilizado o método AHP.

Como primeiro passo neste método de análise, é construída uma árvore hierárquica de decisão, onde estarão definidos o problema, os critérios a avaliar e as alternativas.

4.5.1 Fase 1 - Construção da árvore hierárquica de decisão

Definição do problema - Tendo em conta o contexto do projeto e o seu âmbito abrangente de todo o SDLC, o método AHP será utilizado para decidir sobre o seguinte problema: “Qual o CSP mais vantajoso para uma solução informática segura?”.

Critérios a avaliar - Tendo sido identificado o problema, teremos que distinguir os critérios a utilizar para comparar as alternativas. Os critérios escolhidos para a seleção do CSP mais vantajoso no contexto do problema passam por: configurabilidade, custos e garantias de segurança.

Seleção de alternativas - Uma vez que queremos ter controlo sobre a infraestrutura será utilizado um CSP com modelo de serviço IaaS. Vão ser considerados os maiores CSP desta categoria, nomeadamente: Amazon Web Services, Microsoft Azure e Google Cloud.

Tendo levantado os componentes necessários para a construção da árvore hierárquica de decisão, esta pode ser consultada na figura apresentada abaixo.

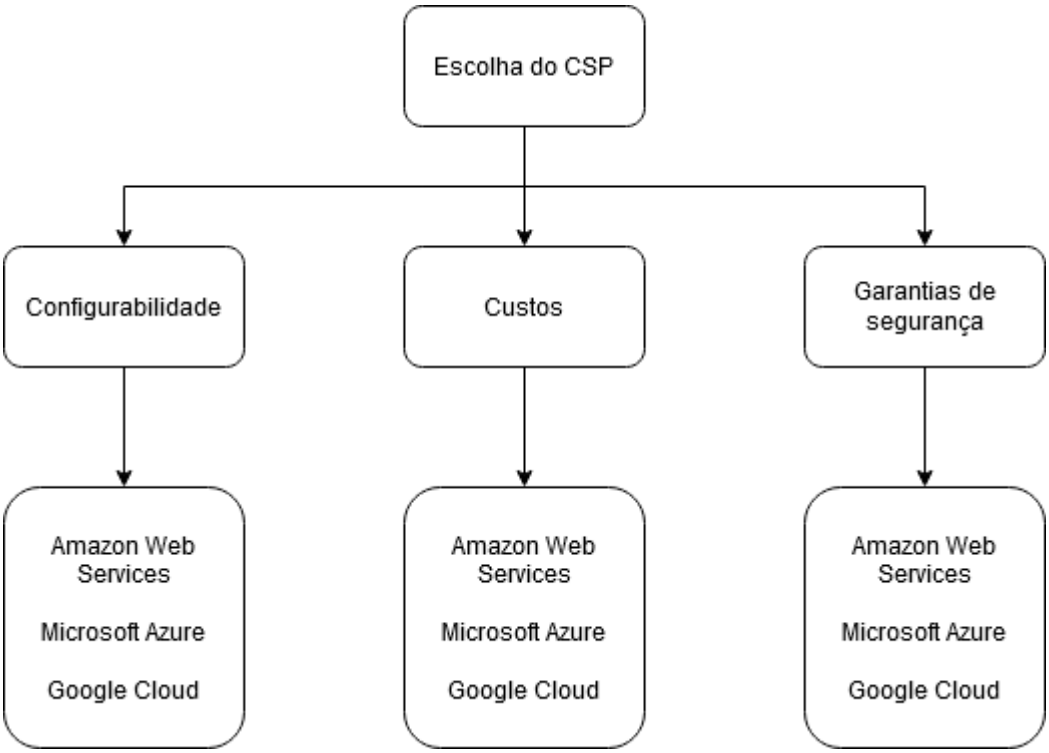


Figura 2 - Árvore de decisão hierárquica segundo o método AHP

4.5.2 Fase 2 - Comparação das alternativas e critérios

Para o estabelecimento das prioridades entre os elementos para cada nível da árvore hierárquica será utilizada uma matriz de comparação. Para este fim será utilizada a escala fundamental concebida por Thomas Saaty, apresentada abaixo.

Tabela 7 - Escala fundamental - Níveis de importância de comparações (Saaty, 1980)

Nível de importância	Definição	Explicação
1	Igual importância	As duas atividades contribuem igualmente para o objetivo
3	Fraca importância	A experiência e o julgamento favorecem levemente uma atividade em relação à outra
5	Forte importância	A experiência e o julgamento favorecem fortemente uma atividade em relação à outra
7	Muito forte importância	Uma atividade é muito fortemente favorecida em relação a outra
9	Importância absoluta	A evidência favorece uma atividade em relação a outra com o mais alto grau de certeza
2, 4, 6, 8	Valores intermediários	Quando se procura uma condição de compromisso entre duas definições

De acordo com os níveis de importância estabelecidos, foi criada uma matriz de comparação de forma a classificar a importância de cada critério. A matriz de comparação pode ser visualizada na tabela abaixo.

Tabela 8 - Matriz de comparações par a par dos critérios

Critério	Configurabilidade	Custo	Segurança
Configurabilidade	1	3	1
Custo	1/3	1	1/7
Segurança	1	7	1

4.5.3 Fase 3 - Prioridade relativa de cada critério

Para o próximo passo da análise AHP, é identificado a ordem de importância de cada critério. Para este fim, é calculada a média aritmética dos valores de cada linha da matriz de comparação após ser normalizada. Os pesos estimados obtidos são apresentados na tabela abaixo.

Tabela 9 - Pesos relativos de cada critério

Critério	Prioridade relativa
Configurabilidade	0,3893
Custo	0,1001
Segurança	0,5105

4.5.4 Fase 4 – Avaliar a consistência das prioridades relativas

Para o próximo passo, é calculada a razão de consistência (CR) para avaliar se os critérios foram consistentes em comparação com grandes amostras de juízos completamente aleatórios.

Para calcular CR é preciso saber os valores do índice de consistência (CI) e do índice aleatório (RI), de acordo com a seguinte fórmula:

$$CR = \frac{CI}{RI}$$

O CI pode ser obtido através da seguinte fórmula:

$$CI = \frac{\lambda_{max} - n}{n - 1}$$

Onde *n* corresponde ao número de critérios e λ_{max} pode ser obtido a partir do seguinte cálculo:

$$\begin{bmatrix} 1 & 3 & 1 \\ 1/3 & 1 & 1/7 \\ 1 & 7 & 1 \end{bmatrix} \times \begin{bmatrix} 0{,}3893 \\ 0{,}1001 \\ 0{,}5105 \end{bmatrix} = \begin{bmatrix} 1{,}2 \\ 0{,}33 \\ 1{,}6 \end{bmatrix}$$

$$\lambda_{max} = média\left\{ \frac{1{,}2}{0{,}3893} \; \frac{0{,}33}{0{,}1001} \; \frac{1{,}6}{0{,}5105} \right\} = 3{,}0809$$

Com o valor de λ_{max}, podemos proceder ao cálculo do CI:

$$CI = \frac{3{,}0809 - 3}{3 - 1} = 0{,}040474$$

Agora, para obter o valor do RI, é usada a tabela de índice aleatório de acordo com Saaty, para *n* = 3, apresentada abaixo.

Tabela 10 - Tabela de índices aleatórios de acordo com Thomas Saaty

N	1	2	3	4	5	6	7	8	9
RI	0,00	0,00	0,58	0,90	1,12	1,24	1,32	1,41	1,45

Podemos então calcular o valor da razão de consistência:

$$CR = \frac{0{,}040474}{0{,}58} = 0{,}069784$$

Podemos concluir que como o valor da razão de consistência é aproximadamente 0,07 e é inferior a 0,1, que os valores das prioridades relativas estão consistentes.

4.5.5 Fase 5 – Construção da matriz de comparação para cada critério

Para este passo do AHQ, é necessário construir matrizes de comparação paritárias de acordo com os critérios levantados, de acordo com cada uma das alternativas escolhidas.

Na Tabela 11 apresentada abaixo podemos observar a matriz de comparação para o critério de configurabilidade, que engloba a disponibilidade de ferramentas e controlo sobre os ambientes de acordo com cada CSP. Uma vez que a solução da Amazon se trata da mais popular e mais matura do mercado, providencia maior controlo e variedade de ferramentas em relação às alternativas (Mogull, 2019) (Dutta e Dutta, 2019) (Mufti, Mittal e Gupta, 2021).

Para facilitar a legibilidade das tabelas, os CSP serão representados pelas suas siglas.

Tabela 11 - Matriz comparativa das alternativas de acordo com configurabilidade

Alternativas	AWS	MA	GC	Peso relativo
AWS	1	3	7	0,6434
MA	1/3	1	5	0,2828
GC	1/7	1/5	1	0,0738

Na Tabela 12 apresentada abaixo podemos observar a matriz de comparação para o critério de custos de acordo com cada CSP. Como podemos observar, a plataforma Google Cloud oferece os melhores modelos de preços comparativamente aos outros CSP (Solanki, 2021) (Dutta e Dutta, 2019) (Mufti, Mittal e Gupta, 2021).

Tabela 12 - Matriz comparativa das alternativas de acordo com os custos

Alternativas	AWS	MA	GC	Peso relativo
AWS	1	1/3	1/5	0,1062
MA	3	1	1/3	0,2605
GC	5	3	1	0,6333

Na Tabela 13 apresentada abaixo podemos observar a matriz de comparação para o critério de segurança, que engloba os modelos de responsabilidade partilhada de cada CSP, bem como as garantias e controlos oferecidos por cada uma das alternativas. Uma vez que se tratam de três alternativas bastante competitivas e fiáveis, encontram-se todas bem posicionadas no que toca às garantias de segurança, porém, os serviços da Amazon Web Services mostram-se mais vantajosos dada a variedade de controlos de monitorização e proteção disponíveis (Mogull, 2019) (Dutta e Dutta, 2019) (Mufti, Mittal e Gupta, 2021).

Tabela 13 - Matriz comparativa das alternativas de acordo com garantias de segurança

Alternativas	AWS	MA	GC	Peso relativo
AWS	1	2	3	0,5390
MA	1/2	1	2	0,2973
GC	1/3	1/2	1	0,1638

De forma a resumir os pesos relativos dos critérios e das alternativas, é apresentada abaixo na Figura 3 a árvore hierárquica atualizada com as suas respetivas prioridades.

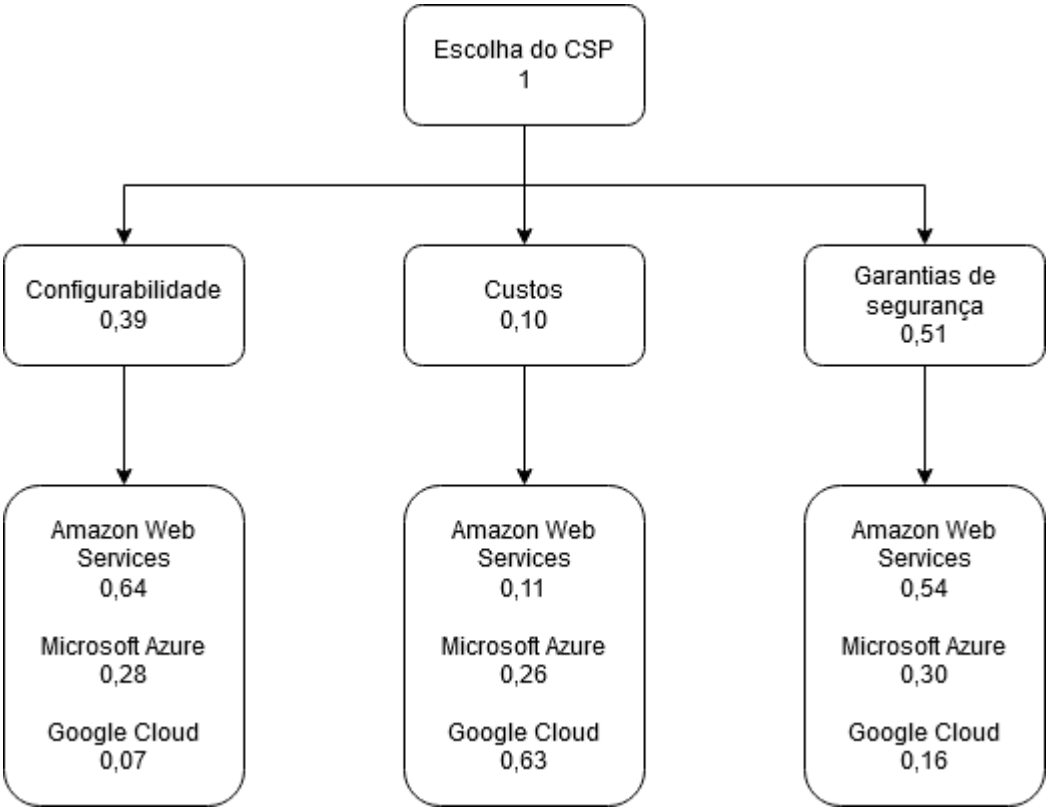


Figura 3 - Árvore hierárquica resumo com os pesos relativos de cada critério e alternativa

4.5.6 Fase 6 – Prioridade composta para as alternativas

Para concluir o modelo AHP, os pesos relativos obtidos nas matrizes comparativas da fase 5 são multiplicados pelos pesos de cada um dos critérios obtidos na fase 3. O maior valor obtido é a solução recomendada pelo modelo.

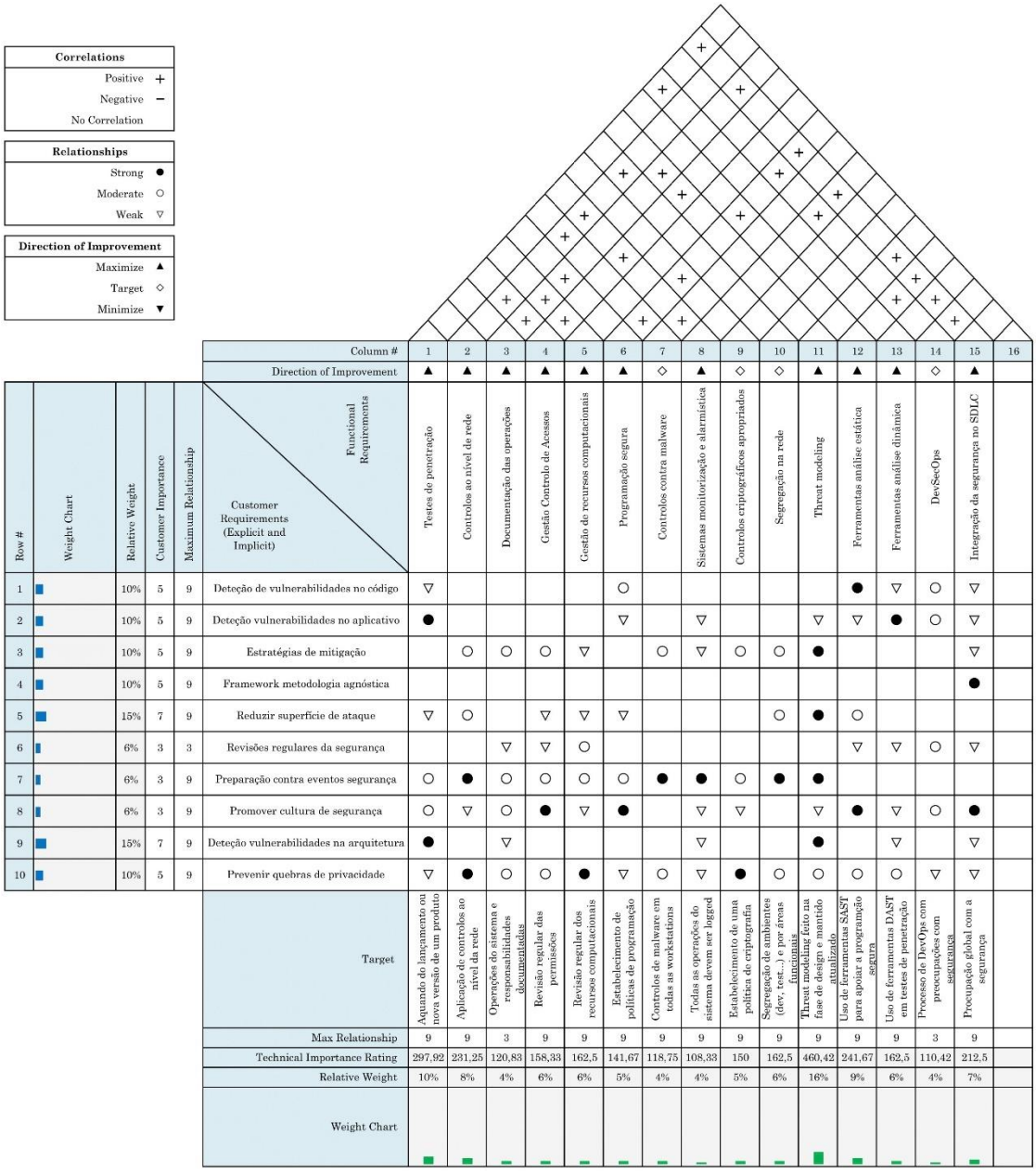
$$\begin{bmatrix} 0{,}6434 & 0{,}1062 & 0{,}5390 \\ 0{,}2828 & 0{,}2605 & 0{,}2973 \\ 0{,}0738 & 0{,}6333 & 0{,}1638 \end{bmatrix} \times \begin{bmatrix} 0{,}3893 \\ 0{,}1001 \\ 0{,}5105 \end{bmatrix} = \begin{bmatrix} 0{,}5363 \\ 0{,}2880 \\ 0{,}1758 \end{bmatrix}$$

4.5.7 Fase 7 – Conclusão AHP

De acordo com o método AHP a solução a escolher para a hospedagem de uma solução informática em nuvem com elevados critérios para a segurança da informação será o serviço da Amazon Web Services, que, apesar de se apresentar como a solução com maiores custos, também apresenta uma melhor configurabilidade e segurança.

4.6 Quality Function Deployment

Finalmente, como último componente da análise de valor, é feito o levantamento dos requisitos do cliente, juntamente com os métodos levantados para os cumprir, utilizando o diagrama de *house of quality*, apresentado na figura abaixo. O mesmo diagrama está incluído ainda no Anexo A da dissertação para melhor legibilidade.



Row #	Weight Chart	Relative Weight	Customer Importance	Maximum Relationship	Customer Requirements (Explicit and Implicit) \ Functional Requirements	1 Testes de penetração	2 Controlos ao nível da rede	3 Documentação das operações	4 Gestão Controlo de Acessos	5 Gestão de recursos computacionais	6 Programação segura	7 Controlos contra malware	8 Sistemas monitorização e alarmística	9 Controlos criptográficos apropriados	10 Segregação na rede	11 Threat modeling	12 Ferramentas análise estática	13 Ferramentas análise dinâmica	14 DevSecOps	15 Integração da segurança no SDLC	16
					Column #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
					Direction of Improvement	▲	▲	▲	▲	▲	▲	◇	▲	◇	◇	▲	▲	▲	◇	▲	
1		10%	5	9	Deteção de vulnerabilidades no código	▽					○						●	▽	○	▽	
2		10%	5	9	Deteção vulnerabilidades no aplicativo	●					▽		▽			▽	▽	●	○	▽	
3		10%	5	9	Estratégias de mitigação		○	○	○	▽		○	▽	○	○	●				▽	
4		10%	5	9	Framework metodologia agnóstica															●	
5		15%	7	9	Reduzir superfície de ataque	▽	○		▽	▽	▽				○	●	○				
6		6%	3	3	Revisões regulares da segurança			▽	▽	○							▽	▽	○	▽	
7		6%	3	9	Preparação contra eventos segurança	○	●	○	○	○	○	●	●	○	●	●					
8		6%	3	9	Promover cultura de segurança	○	▽	○	●	▽	●		▽	▽		▽	●	▽	○	●	
9		15%	7	9	Deteção vulnerabilidades na arquitetura	●		▽					▽			●		▽		▽	
10		10%	5	9	Prevenir quebras de privacidade	▽	●	○	○	●	▽	○	▽	●	○	○	○	○	▽	▽	
					Target	Aquando do lançamento ou nova versão de um produto	Aplicação de controlos ao nível da rede	Operações do sistema e responsabilidades documentadas	Revisão regular das permissões	Revisão regular dos recursos computacionais	Estabelecimento de políticas de programação	Controlos de malware em todas as workstations	Todas as operações do sistema devem ser logged	Estabelecimento de uma política de criptografia	Segregação de ambientes (dev, test...) e por áreas funcionais	Threat modeling feito na fase de design e mantido atualizado	Uso de ferramentas SAST para apoiar a programação segura	Uso de ferramentas DAST em testes de penetração	Processo de DevOps com preocupações com segurança	Procupação global com a segurança	
					Max Relationship	9	9	3	9	9	9	9	9	9	9	9	9	9	3	9	
					Technical Importance Rating	297,92	231,25	120,83	158,33	162,5	141,67	118,75	108,33	150	162,5	460,42	241,67	162,5	110,42	212,5	
					Relative Weight	10%	8%	4%	6%	6%	5%	4%	4%	5%	6%	16%	9%	6%	4%	7%	
					Weight Chart																

Figura 4 - QFD House of Quality

5 Design da Arquitetura de referência

O primeiro passo para o desenvolvimento da prova de conceito que permitirá avaliar a eficácia dos controlos levantados do documento ISO e da análise do estado da arte passa pelo *design* de uma arquitetura de referência.

Uma vez que a segurança da rede constitui um dos principais controlos de segurança propostos no ISO 27001, é importante aplicar os controlos ISO relevantes à arquitetura da rede, bem como as ferramentas levantadas na secção 3.4.1 desta dissertação de forma a garantir um sistema com defesa em profundidade.

Ao utilizar requisitos frequentemente encontrados em aplicações reais, pretende-se desenvolver uma arquitetura de referência com controlos genéricos que vão ao encontro dos controlos ISO, de forma a demonstrar como os aplicar na prova de conceito a desenvolver.

5.1 Requisitos da arquitetura de referência

Para o design da arquitetura de referência a desenvolver foram utilizados os seguintes requisitos:

- O aplicativo a desenvolver deve ser acessível via *smartphone* pelos utilizadores finais via aplicativo *mobile*. O componente back-end do aplicativo deve disponibilizar uma API para uso por aplicações externas.
- Irá existir ainda um aplicativo *backoffice* sob a forma de *web application* mas com acesso restrito via *Virtual Private Network* (VPN) para colaboradores internos.
- Aplicar os controlos pertinentes à camada de arquitetura levantados dos documentos ISO analisados.

5.2 Mapeamento controlos ISO

Nesta secção são identificados os controlos levantados do ISO na secção 3.1 da dissertação que serão seguidos para o desenho da arquitetura de referência.

- 9.1.1: Política de controlo de acessos – Devem existir controlos informatizados para a gestão de acessos de um utilizador aos ficheiros do sistema.
- 10.1.1: Política de uso de controlos criptográficos – Os dados em transporte e armazenamento serão encriptados usando métodos criptográficos adequados à natureza da sua sensibilidade.
- 12.2.1: Controlos contra *malware* – De forma a garantir um maior nível de proteção contra *malware*, será utilizado um componente *sandbox* ao nível de rede, que permita

executar ficheiros suspeitos numa VM isolada antes destes serem entregues aos utilizadores do sistema.

- 12.3.1: Política de *backup* – A política de *backup* será suportada por uma ferramenta ao nível de rede.
- 12.4.1: Registo de eventos – Serão usados controlos que suportam o *logging* das atividades dos utilizadores de um sistema.
- 12.4.2: Proteção de *logs* – Os *logs* devem estar devidamente armazenados e encriptados.
- 13.1.1: Controlos de segurança na rede – Para garantir este controlo serão utilizados os componentes levantados na secção 3.4.1 da dissertação.
- 13.1.3: Segregação na rede – Para a aplicação deste controlo as diferentes camadas (apresentação, aplicacional e armazenamento) do aplicativo terão componentes *firewall* nos seus perímetros, resultando num maior isolamento entre os componentes do sistema. Ainda no contexto deste controlo, os diferentes ambientes de programação (produção, teste, *quality assurance*) estarão segregados por diferentes *Virtual Private Clouds* (VPCs), que consistem em redes virtuais isoladas logicamente. Usando VPCs podemos definir o intervalo de endereços IP privados, criar sub-redes e configurar tabelas de roteamento dentro da rede (Amazon Web Services, 2021l).

5.3 Decisões para o design

Nesta secção estão explícitas as principais decisões tomadas durante a conceção da arquitetura de rede do sistema.

5.3.1 Arquitetura Multi-VPC

Conforme mencionado anteriormente, para garantir maior isolamento lógico dos componentes do sistema, os diferentes ambientes de programação (produção, teste, *quality assurance*) serão segregados por diversas VPC.

Adicionalmente, a própria sub-rede pública será colocada numa VPC à parte. Nesta sub-rede, serão colocados os controlos de WAF, Reverse Proxy e *Next-Generation Firewall* (NGFW) que desempenha os papéis de *Firewall* e IPS/IDS. Assim, esta VPC poderá servir de ponto de entrada para qualquer outra VPC do sistema e não terá informações sobre as VM que se encontrem nas VPC privadas (Diez, 2020).

Esta segregação por diferentes VPC ajuda a garantir que as máquinas de um ambiente não têm conhecimento das VM de um ambiente diferente.

5.3.2 Sub-rede pública e privada

No que toca às camadas de apresentação, aplicação e armazenamento, é importante garantir que a camada de apresentação é a única que interage com a sub-rede pública, que por sua vez terá acesso à *internet*. A camada de aplicação por sua vez interage com ambas as camadas de apresentação e armazenamento.

Estas poderão ser impedidas de interagir diretamente com a *internet* através do uso de sub-redes privadas, que serão complementadas com *Network Security Groups* (NSGs).

Os NSG têm um funcionamento semelhante a *firewalls* tradicionais, com capacidade para filtrar o tráfego de entrada e saída na rede com base na sua origem, destino, porta e protocolo (Microsoft Docs, 2020).

Por outro lado, utilizando sub-redes privadas, quaisquer recursos dentro destas não são diretamente acessíveis pela *internet,* o que nos oferecem maiores garantias para a segurança. Porém, ao utilizar sub-redes privadas introduzimos alguma complexidade na arquitetura, uma vez que para recursos dentro da sub-rede poderem comunicar com a *internet*, necessitam de o fazer através de um NAT Gateway (Services, 2020).

5.3.3 Comunicação entre VPC

Uma vez que vamos optar por uma arquitetura multi-VPC, teremos que considerar como se realizará a comunicação entre as diversas VPC.

Existem dois padrões principais utilizados na comunicação em arquiteturas multi-VPC, nomeadamente, *many-to-many* e *hub and spoke*.

No padrão *many-to-many*, a comunicação entre VPCs é gerido individualmente entre cada VPC, por isso, quantas mais VPC tiverem que comunicar entre si, mais difícil fica a manutenção e escalabilidade do padrão. (Amazon Web Services, 2020a)

No padrão *hub and spoke*¸ toda a comunicação entre VPCs é realizada através de um recurso centralizado, que transmite o tráfego de acordo com as regras estabelecidas. (Amazon Web Services, 2020a)

Ao adotar uma arquitetura multi-VPC, com diversos ambientes de desenvolvimento, e visando uma infraestrutura escalável, será seguido o padrão *hub and spoke* para a comunicação entre VPCs, sendo que o componente central da comunicação será um *Traffic Gateway*.

5.4 Diagrama de componentes

Nesta secção do relatório é apresentado o diagrama dos componentes que irão constituir a arquitetura de referência, de forma a atender aos requisitos levantados. O diagrama pode ser consultado na Figura 5, apresentada abaixo.

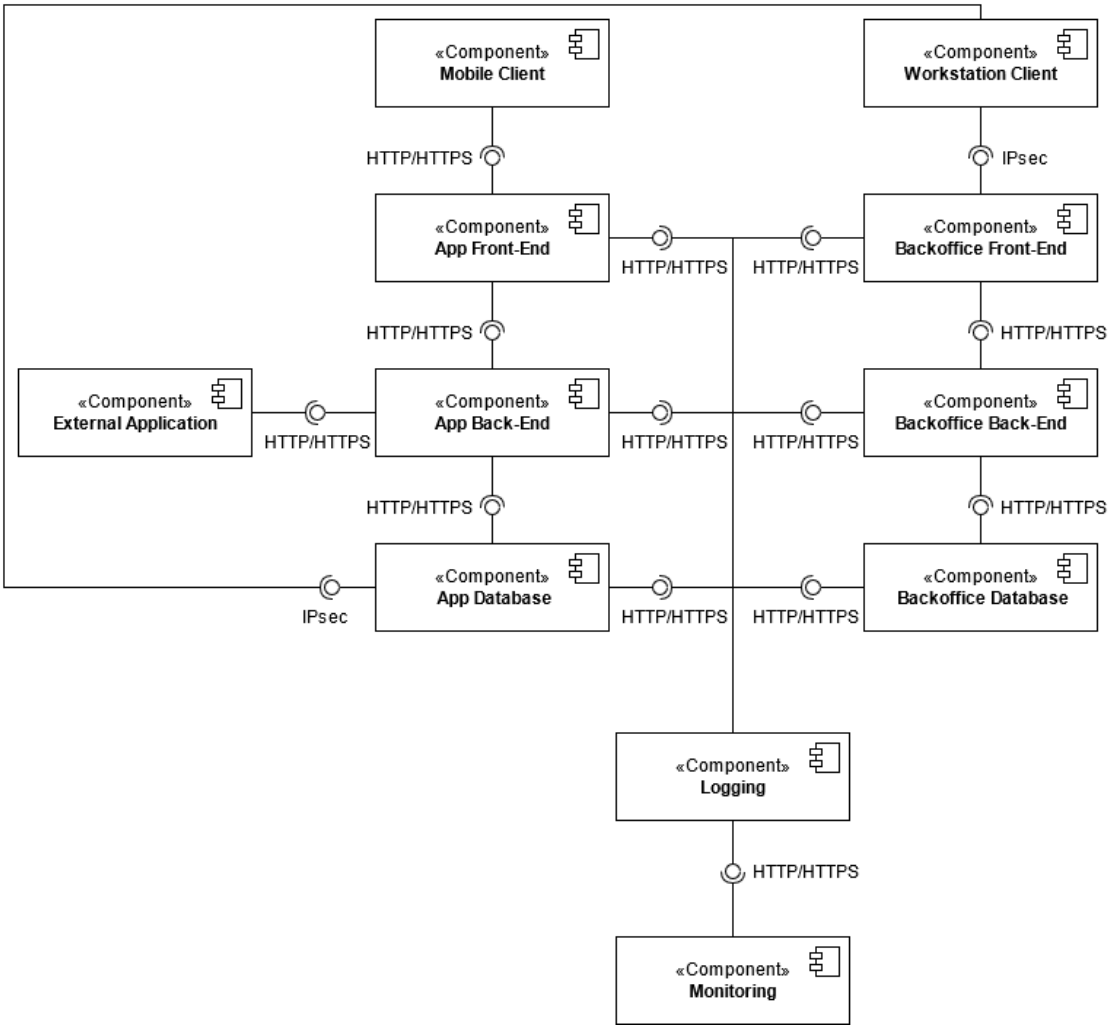


Figura 5 - Diagrama de componentes

Conforme os requisitos levantados, existem dois aplicativos, nomeadamente o aplicativo destinado aos utilizadores finais, acedido através de aplicativo *smartphone* (identificado como *Mobile Client*) e um aplicativo *backoffice*, acedido através um computador via VPN. O componente *back-end* do aplicativo para clientes disponibiliza ainda uma API para acesso por aplicações externas.

Todos os componentes de ambos os sistemas têm interface com um componente de *logging*, que por sua vez interage com um componente para a monitorização do desempenho e alarmística.

5.5 Diagrama de implantação

Nesta secção é apresentado o diagrama de implantação para a arquitetura de referência desenvolvida, sendo identificados os ambientes de implantação nos quais os componentes do sistema vão executar. O diagrama pode ser consultado na Figura 6, apresentada abaixo.

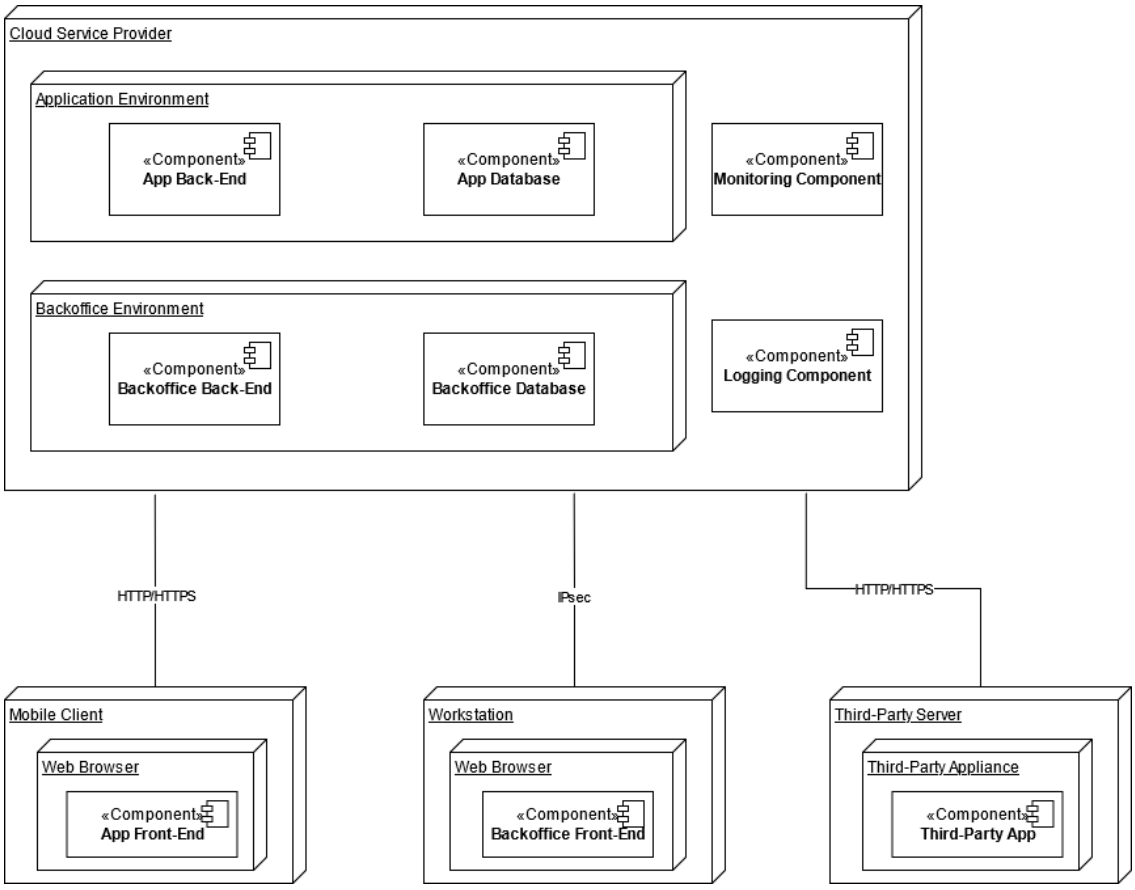


Figura 6 - Diagrama de implantação

No que toca à implantação do sistema, todos os componentes poderão ser hospedados num único CSP, de forma a facilitar a manutenção e monitorização do sistema. O CSP irá ainda hospedar ambos os aplicativos, bem como os componentes de *logging* e monitorização.

Para além disto, prevêem-se três tipos distintos de cliente, nomeadamente, os clientes finais que acedem via *smartphone*, os colaboradores que acedem via VPN, e entidades externas que façam uso da API disponibilizada pelo componente *back-end* da aplicação principal.

Alternativamente, poderão ser utilizados diversos CSP para hospedar diferentes componentes da aplicação de forma a obter uma maior segregação, ou mesmo como mecanismo de redundância. Porém a implementação de um ambiente *multi-cloud* exige maiores cuidados sobre a manutenção e visibilidade de todos os componentes de um sistema (Fortinet, 2018).

5.6 Arquitetura de referência proposta

Tendo levantado os controlos relevantes à camada de arquitetura e tendo ponderado sobre as principais decisões para o design, é apresentada na Figura 7 a arquitetura de referência criada. No Anexo D da dissertação está incluído o mesmo artefacto da arquitetura com melhor legibilidade.

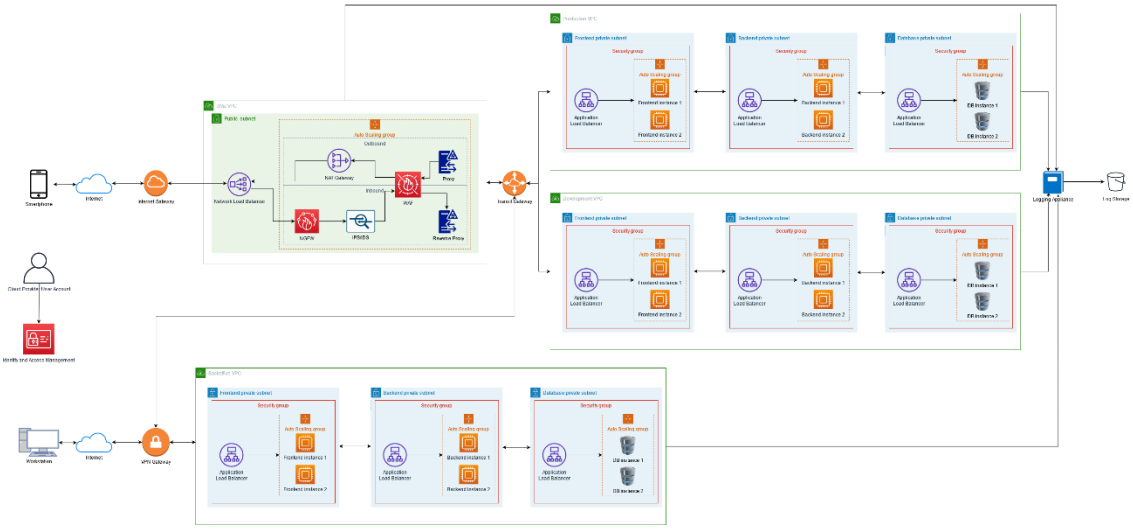


Figura 7 – Design da arquitetura de rede

Através do *design* da arquitetura de rede, pretendem-se identificar os fluxos da informação dentro do sistema, bem como identificar os controlos de segurança a utilizar na implantação em nuvem.

O aplicativo principal vai fazer uso de uma VPC pública que vai atuar como DMZ, onde serão colocados os principais componentes para a filtragem e monitorização do tráfego a entrar no sistema. Será usado um componente NGFW para filtragem de tráfego ao nível das camadas 3 e 4 do modelo OSI. Depois da NGFW, existe um componente IPS/IDS para a deteção e prevenção contra intrusões. De seguida é utilizada uma WAF para a deteção e proteção contra ataques ao nível aplicacional. Finalmente, existe um componente Reverse Proxy para ocultar a *stack* tecnológica do sistema. Quanto ao tráfego de saída da DMZ, existe um Proxy para ocultar a *stack* tecnológica do tráfego de saída. De seguida passa no componente WAF, e finalmente por um NAT Gateway, para possibilitar o contacto das sub-redes privadas com a *internet*.

O roteamento do tráfego entre VPC será assegurado pelo componente Transit Gateway, que terá conhecimento das diferentes VPC que dizem respeito a cada um dos ambientes que a aplicação será executada (desenvolvimento, produção, teste…).

Cada uma das instâncias do sistema encontram-se em Auto-Scaling Groups que, em conjunto com os componentes de Load Balancing, permitirão que os componentes acompanhem picos de uso do sistema uma vez que tratam da escalabilidade.

Na arquitetura está ainda representado um componente de gestão de acessos e identidades, que será um serviço fornecido pelo CSP que tratará de manter uma *whitelist* dos acessos dos utilizadores aos recursos implantados em nuvem.

6 Prova de Conceito

Nesta secção da dissertação é documentado o processo de design da prova de conceito a implementar de forma a colocar os controlos ISO levantados em prática.

Uma vez que o principal foco do projeto passa pela consolidação de um conjunto de práticas para o desenvolvimento de aplicações mais seguras em nuvem, será ao nível da infraestrutura e da conta AWS onde serão postos em prática os controlos para a prova de conceito.

Já que a segurança aplicacional é uma preocupação transversal a qualquer produto informático independentemente da sua implantação, este aspeto da segurança do sistema não será o foco da prova de conceito.

6.1 Requisitos

Para o desenvolvimento da prova de conceito foram considerados os seguintes requisitos:

Requisitos funcionais:

- A aplicação deve funcionar como uma *checklist* para realizar auditorias a um produto informático.
- A *checklist* deve ter por base a *framework* desenvolvida durante o projeto, apresentada no Anexo C da dissertação.
- Esta *checklist* deve ser carregada remotamente de um componente *backend.*
- O componente *backend* deve ainda guardar o progresso de um dado utilizador à medida que este vai completando tarefas da lista.

Requisitos não funcionais:

- A aplicação deve ser acessível via Internet.
- A aplicação deve ter uma interface de utilizador com um *design* amigável e intuitivo.
- A infraestrutura deve fazer uso de diversas camadas de segurança.
- A infraestrutura a implantar deve fazer apenas uso de componentes e serviços incluídos no plano gratuito do CSP.

Apesar da funcionalidade relativamente limitada do aplicativo, a implantação desta *web app* em nuvem permite pôr em prática os controlos levantados ao longo do projeto.

6.2 Escolha das tecnologias

Nesta secção do documento serão apresentadas as principais tecnologias consideradas para a implementação de cada um dos componentes, e a justificação para a sua escolha.

Não existem restrições nas tecnologias a utilizar para a implementação da aplicação para a prova de conceito, logo, é necessário selecionar tecnologias que suportem o desenvolvimento dos componentes que vão constituir o aplicativo a desenvolver, nomeadamente:

- Um componente *frontend* que servirá como interface gráfica para o utilizador interagir com o aplicativo.
- Um componente *backend* que consistirá numa API REST[1] que irá gerir a lógica de negócio do aplicativo.
- Um componente de base de dados que irá persistir o estado e a informação do aplicativo.

6.2.1 Componente Frontend

No que toca ao desenvolvimento de componentes *frontend* para aplicativos *web*, as tecnologias com maior adoção são ReactJS, Angular e VueJS (Dhaduk, 2021) (Figueiredo Ribeiro, 2020).

Uma vez que o único requisito relacionado com o componente de *frontend* passa pela apresentação de uma interface gráfica amigável, as principais motivações para a seleção da tecnologia serão:

- A facilidade de uso, de forma a facilitar o processo de implementação.
- A disponibilidade de documentação, de forma a facilitar a fase de aprendizagem e de implementação.
- Baixo consumo de recursos computacionais, já que irá melhorar o desempenho do aplicativo.
- A familiaridade com a tecnologia.

[1] API que segue as restrições arquiteturais REST. Entrega uma representação do estado do recurso ao qual um cliente acede via a API (Red Hat, 2020).

Atendendo a estes pontos, a tecnologia selecionada para a implementação do componente *frontend* foi ReactJS dada a familiaridade com o projeto, a maior comunidade de utilizadores, a ótima documentação e um desempenho comparável às restantes tecnologias (Figueiredo Ribeiro, 2020).

6.2.2 Componente Backend

Para o desenvolvimento de componentes *backend* que disponibilizam APIs REST, as tecnologias mais adotadas são Laravel, Django, Flask e Spring Boot (Bawe, 2021).

Uma vez que é da responsabilidade do *backend* gerir a lógica do aplicativo, as principais motivações para a seleção da tecnologia serão:

- A disponibilidade de documentação, de forma a facilitar a fase de aprendizagem e de implementação.
- Capacidade de integração com dependências externas, de forma a reduzir a chance de erros de implementação (embora a introdução de dependências *third-party* exija um maior cuidado na sua monitorização para a deteção de eventuais vulnerabilidades).
- Baixo consumo de recursos computacionais, já que irá melhorar o desempenho do aplicativo.
- A familiaridade com a tecnologia.

Tendo em conta estes pontos, foi selecionada a tecnologia Spring Boot, uma vez que utiliza Java como linguagem de programação que, sendo uma linguagem fortemente tipada, ajuda a reduzir a chance de introdução de erros na fase de implementação. Aliado a isto, esta *framework* na sua forma mais básica é bastante leve, com alta capacidade de integrar módulos externos que assistem na implementação de funcionalidades mais sensíveis como funções de encriptação e autenticação no aplicativo (Monocubed, 2021).

6.2.3 Componente de Base de Dados

Para a implementação do aplicativo resta selecionar o motor de base de dados mais adequado. Antes de analisar as possíveis tecnologias a utilizar, serão excluídos motores de bases de dados não relacionais uma vez que de modo geral oferecem piores garantias de segurança quando comparados a bases de dados relacionais (Mohamed, Altrafi e Ismail, 2014).

No que toca à escolha da tecnologia a utilizar, serão consideradas algumas das tecnologias com maior adoção, como Oracle, MySQL, SQL Server e PostgreSQL (DB-Engines, 2021).

As principais motivações para a seleção da tecnologia serão:

- A facilidade de uso e integração com o componente *backend*, de forma a facilitar o processo de implementação.

- A disponibilidade de documentação, de forma a facilitar a fase de aprendizagem e de implementação.

Atendendo às motivações sublinhadas e às integrações existentes com a *framework* Spring Boot e os serviços da AWS, o motor de base de dados selecionado para integrar o aplicativo foi MySQL.

6.2.4 Implantação da Infraestrutura

Sendo este ponto o principal foco da implementação da prova de conceito, será ainda utlizada uma ferramenta para a implantação da infraestrutura em AWS, seguindo as práticas de DevOps exploradas na secção 3.5 do documento. Para este fim será utilizada uma ferramenta para a implantação da infraestrutura através ficheiros de código, seguindo o conceito de *Infrastructure as Code* (IaS).

A implantação da infraestrutura está dependente de diversas variáveis, como por exemplo a seleção do CSP a utilizar, a configuração de máquinas virtuais ou *containers* e a configuração da rede que permita a comunicação entre os componentes implantados. Neste contexto, o conceito de DevOps promove o uso de código para a definição da infraestrutura de forma a automatizar a configuração dos componentes que compõem a infraestrutura (Artac *et al.*, 2017).

O objetivo desta abordagem passa pela aplicação das práticas normalmente utilizadas no desenvolvimento de *software* na implantação da infraestrutura, nomeadamente (Artac *et al.*, 2017):

- O uso de ferramentas de controlo de versões como *Git*.

- A melhor visibilidade sobre a infraestrutura.

- A aplicação de padrões de *design* de forma a responder a problemas comuns.

- Facilitar a testagem frequente do sistema dada a facilidade de implantação de uma infraestrutura semelhante para efeitos de teste.

No que toca às tecnologias a usar na implantação da infraestrutura, forma consideradas ferramentas como Chef, Puppet, Ansible, AWS CloudFormation e Terraform, que são frequentemente associadas ao tema de IaC.

Porém, existe uma distinção entre estas ferramentas IaC, uma vez que as tecnologias Chef, Puppet e Ansible tratam-se de ferramentas de gestão de configuração, cuja funcionalidade

passa pela instalação e gestão do *software* nas máquinas de uma infraestrutura pré-existente (Brikman, 2019).

Por outro lado, o CloudFormation e o Terraform tratam-se de ferramentas de provisionamento de infraestrutura, que tratam de definir e implantar a infraestrutura em si (Brikman, 2019).

Estas duas categorias de ferramentas, apesar de diferentes nas suas finalidades, podem ser utilizadas em conjunto, sendo que algumas possuem funcionalidades comuns a ambas as categorias. Porém, o principal objetivo para o uso de uma ferramenta de IaC para efeito da prova de conceito passa pela melhor visibilidade sobre a infraestrutura e pela facilidade de gestão do crescimento gradual da infraestrutura. Para este fim, foram consideradas apenas a ferramentas cujo principal foco é o provisionamento de infraestrutura.

Entre as ferramentas CloudFormation e Terraform, foi selecionado o Terraform uma vez que esta ferramenta suporta diversos CSP (Terraform Registry, 2021b) o que ajuda a reduzir ligeiramente a chance de *vendor lock-in*, enquanto que o CloudFormation se trata de uma ferramenta para a implantação de infraestruturas exclusivo à AWS.

6.3 Modelo de Domínio

Para suportar os requisitos funcionais levantados anteriormente, foi desenvolvido o modelo de domínio apresentado abaixo na Figura 8.

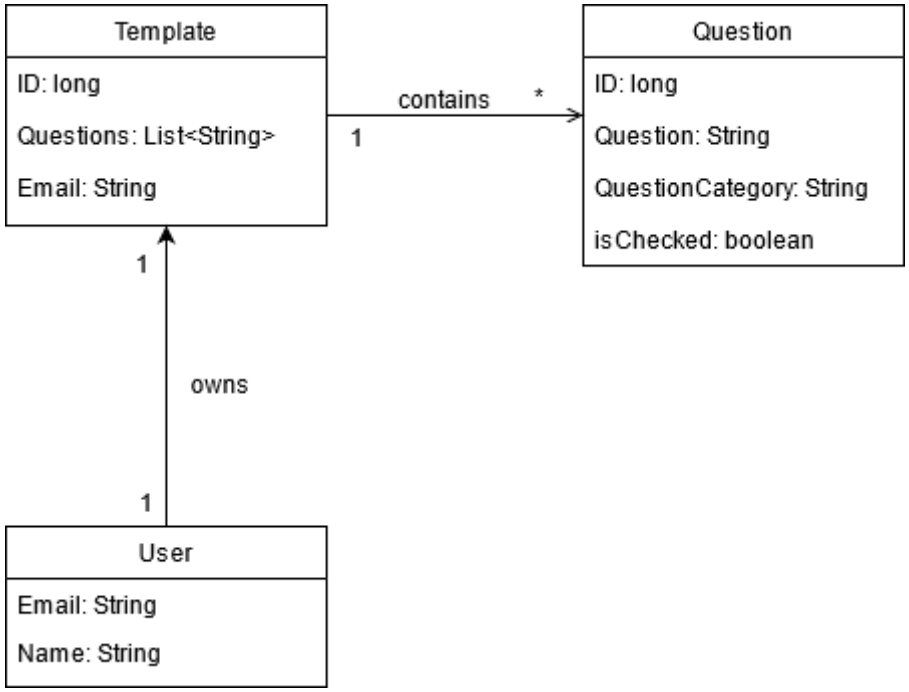


Figura 8 – Modelo de domínio para prova de conceito

Neste diagrama são introduzidos três novos conceitos que serão frequentemente referenciados ao longo desta secção da dissertação, nomeadamente: o *User*, a *Template* e a *Question*.

Um *User* corresponde a um utilizador do aplicativo, sendo identificado pelo seu *email*, e tendo ainda um campo para o seu nome próprio. Cada um dos *Users* possui uma *Template*.

Uma *Template* consiste apenas numa lista de *Questions*, que é utilizada por um único *User*.

Uma *Question* corresponde a um item da *checklist*, sendo a descrição do item guardada no campo *question*. Cada *Question* pode ser marcada como finalizada através do campo *isChecked*.

6.4 Diagrama de componentes

De forma a explicitar as principais interações entre os componentes do sistema, foi criado o diagrama de componentes apresentado abaixo na Figura 9.

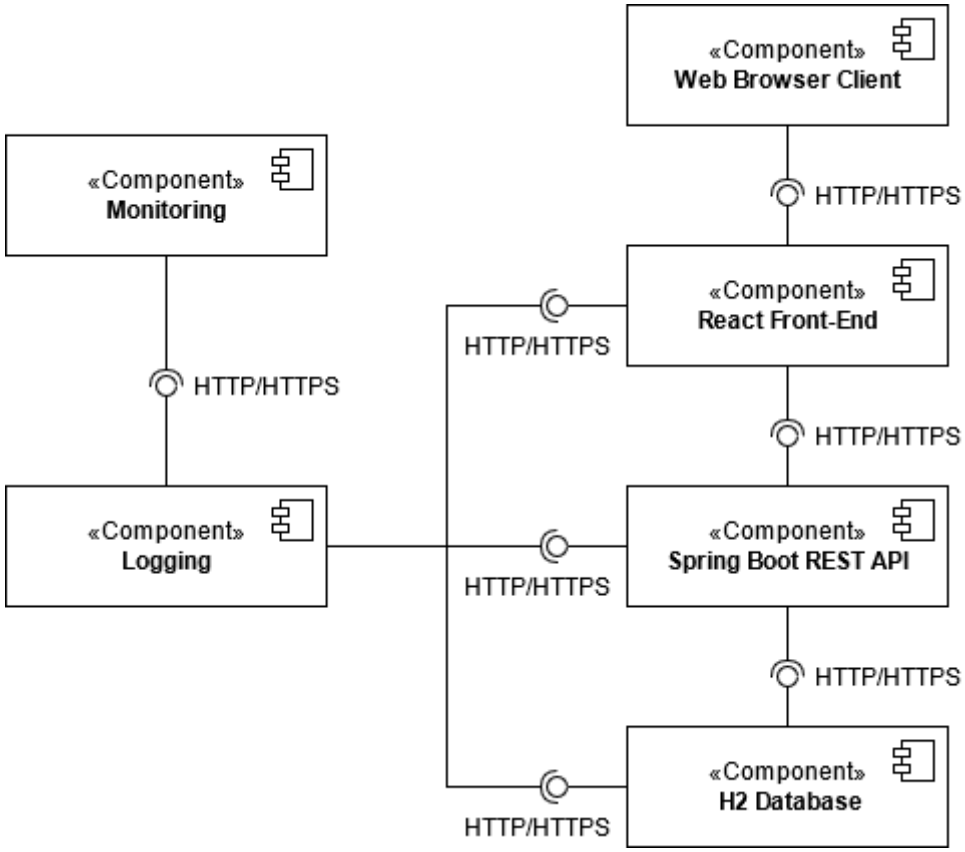


Figura 9 – Diagrama de componentes da prova de conceito

Como podemos observar, o aplicativo *frontend* será acessível via um cliente *web browser*. O componente *frontend* vai comunicar apenas com o componente *backend*, que por sua vez comunica com o componente da base de dados.

Cada um dos componentes aplicacionais disponibilizam ainda interfaces para exportar *logs* em formato JSON, que serão agregados e armazenados por um componente *Logging* central. Existe ainda um componente responsável pela monitorização dos *logs* que permite a configuração de alarmísticas e ações automatizadas sobre os *logs* recolhidos do sistema.

6.5 Diagrama de sequência

De forma a explicitar as interações entre os componentes do sistema e a obter uma melhor visibilidade sobre os fluxos da informação no aplicativo, foi criado o diagrama de sequência da Figura 10 a explicitar as principais funcionalidades do *User*.

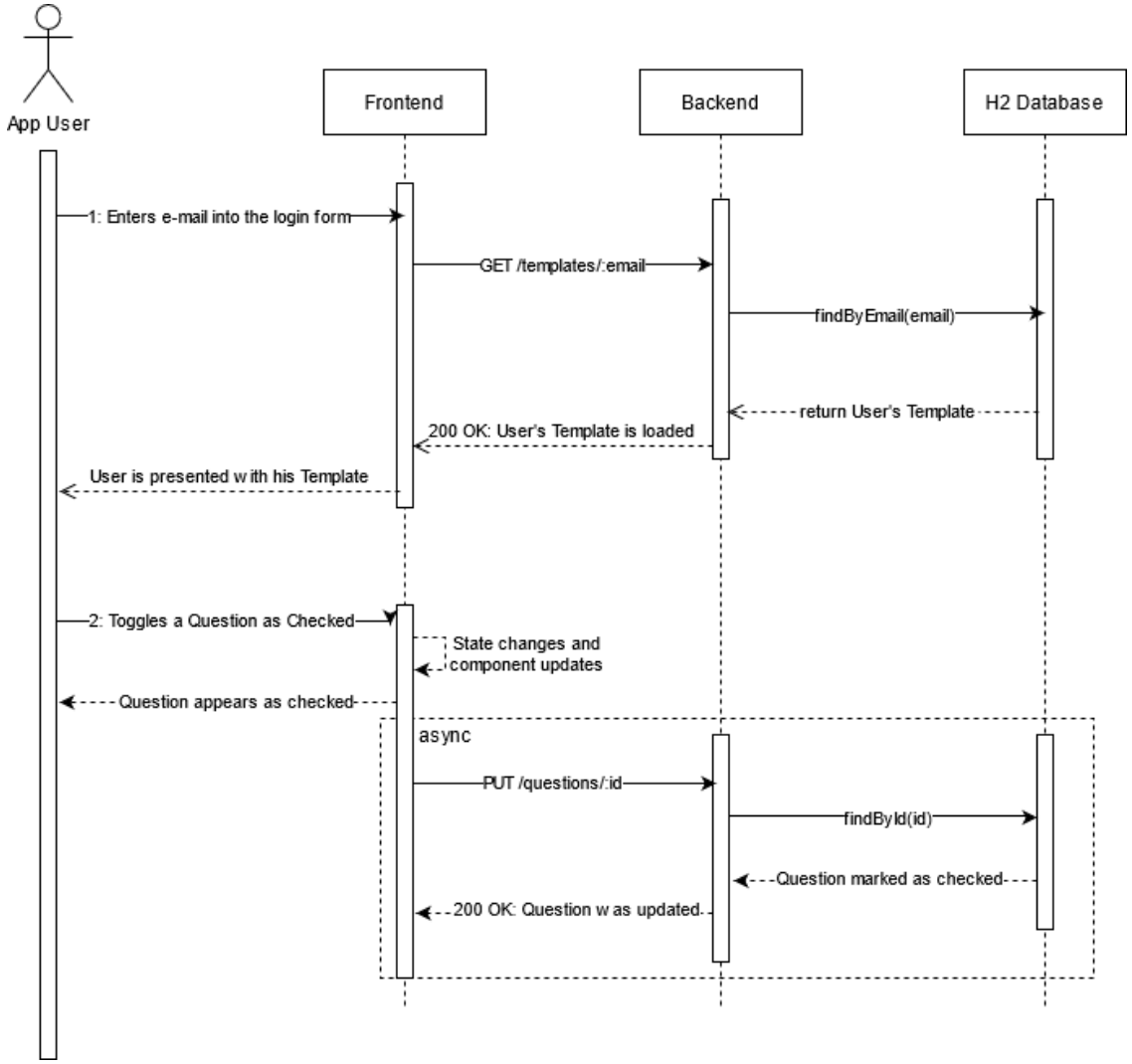


Figura 10 – Diagrama de sequência da prova de conceito

Existem duas funcionalidades representadas no diagrama, nomeadamente, o processo de *login* e o processo de marcar uma *Question* como finalizada.

Começando pelo *login*, o *User* introduz o seu *email* único no formulário de *login* apresentado pelo componente *frontend*. O componente *frontend* por sua vez consulta o *backend* acerca da *Template* deste *User*, passando o *email* como parâmetro numa *query* GET. O componente *backend* consulta a base de dados acerca da *Template* associada ao *email* do *User*, que por sua vez retorna a informação requisitada. Já no componente *frontend*, o *User* é redirecionado para a sua *checklist* tendo sido preenchida com as *Questions* que fazem parte da sua *Template*.

Agora na segunda funcionalidade, o *User* marca uma das *Questions* como finalizada, o que vai provocar uma alteração no estado mantido pelo componente *frontend*, que vai mostrar a *Question* como finalizada e assincronamente atualizar o estado da *Question* no *backend*. O *frontend* envia então uma *query* PUT com o identificador da *Question* como parâmetro. O

backend trata de atualizar a *Question* na base de dados como finalizada, e por sua vez envia um código de sucesso ao *frontend*.

6.6 Diagrama de implantação em AWS

Encerrando a fase de *design* da prova de conceito, resta apresentar a arquitetura da infraestrutura a ser implantada em AWS, explicita na Figura 11, apresentada abaixo. No Anexo E do documento encontra-se uma versão maior do diagrama apresentado para melhor legibilidade.

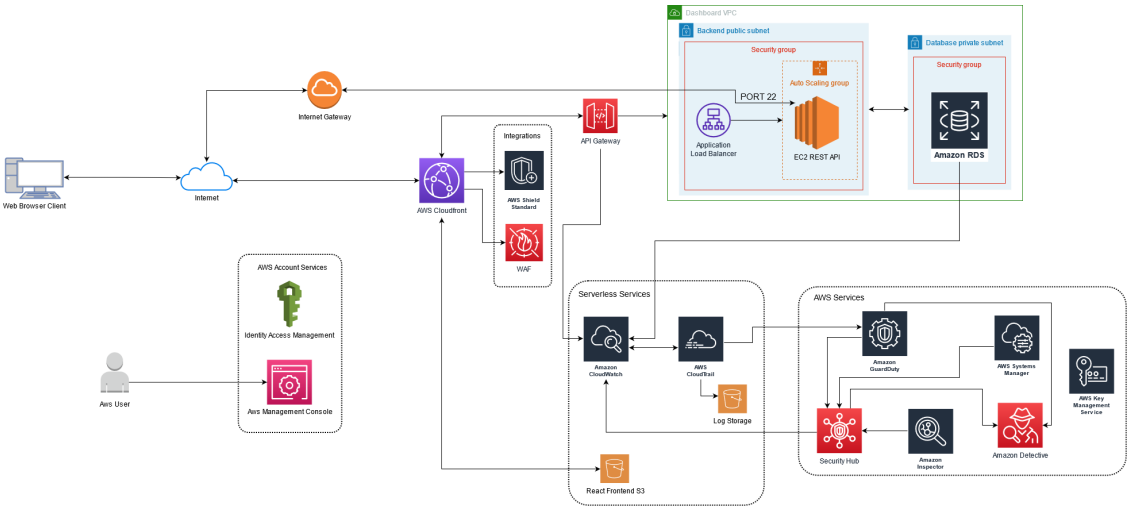


Figura 11 – Diagrama de implantação da prova de conceito em AWS

Para analisar a arquitetura apresentada, vamos detalhar a responsabilidade de cada um dos componentes envolvidos no tráfego da informação assim que esta entra no perímetro da infraestrutura.

Um utilizador do aplicativo, representado pela *workstation* intitulada como *Web Browser Client*, tem acesso via Internet a duas interfaces de interação com o nosso sistema, nomeadamente:

- Poderá comunicar com a distribuição CloudFront, que tratará de expor publicamente o nosso aplicativo e de gerir as chamadas à API do componente *backend*.
- Poderá comunicar com o porto 22 da máquina EC2 para proceder à sua eventual configuração, que é o único porto da máquina exposto à Internet.

6.6.1 Distribuição CloudFront

Atendendo à restrição do uso exclusivo de serviços incluídos no plano gratuito da AWS, a distribuição CloudFront irá constituir a nossa primeira camada de defesa ao nível da

arquitetura (idealmente, seria precedido por um componente *Network Firewall*). Uma vez que este componente se trata de um serviço gerido pela Amazon e não de uma instância gerida pelo utilizador, não é possível contê-lo numa VPC como na representação da arquitetura de referência da secção 5.6 do documento.

Para além das funcionalidades de *caching* e de rede de entrega de conteúdo, este componente permite-nos ainda assegurar algumas medidas de segurança (Amazon Web Services, 2021c):

- Garante que a comunicação com o componente *frontend* é realizada exclusivamente usando HTTPS, independentemente da implementação do mesmo.

- Integra com diversos componentes de segurança normalmente encontrados numa DMZ tradicional, nomeadamente o AWS *Shield* para proteção contra ataques DoS (*Denial of Service*) e a AWS WAF para proteção contra ataques ao nível da camada aplicacional.

6.6.2 API Gateway

O uso do componente API Gateway no perímetro da nossa VPC oferece algumas garantias quanto à segurança, nomeadamente (Amazon Web Services, 2021b) :

- Apenas aceita pedidos à API que forneçam uma chave de acesso válida.

- Permite estabelecer planos de uso com eventuais mecanismos *throttling* para cada chave de acesso gerada.

- Garante que a comunicação à API é realizada exclusivamente usando HTTPS, independentemente da implementação do componente *backend* hospedado na máquina EC2.

- Facilita a monitorização dos pedidos à API falhados e os recursos acedidos.

Uma vez que para o caso de uso deste aplicativo não estão previstos acessos diretos à API, é gerada apenas uma chave de acesso, que é partilhada com a distribuição CloudFront.

6.6.3 Instância EC2 – Componente backend

A instância EC2 terá a mínima exposição possível à Internet, de forma a reduzir a superfície de ataque deste componente. O porto 22 encontra-se aberto apenas para a configuração da instância através de uma conexão SSH, sendo que para iniciar esta conexão será necessário autenticar-se através de uma chave privada e de um endereço IP *whitelisted*.

Esta instância terá ainda o porto 8080 aberto apenas à API Gateway, onde servirá a API REST.

6.6.4 Bucket S3 – Componente frontend

Para hospedar o componente *frontend* poderia ser utilizada uma instância EC2 tal como no componente *backend*, porém como o componente *frontend* se trata de uma *web page* simples desenvolvida em ReactJS, optou-se por utilizar um *bucket* S3, que se trata de um serviço *serverless* garantido pela AWS (Amazon Web Services, 2020b).

Utilizando este serviço, perdemos algum controlo sobre a sua configurabilidade ao nível da rede quando comparado a uma instância EC2 (como por exemplo filtragem de tráfego por protocolos, portos e serviços), a qual é equiparável ao funcionamento de uma máquina virtual tradicional (Amazon Web Services, 2021g). Porém ao utilizar um *bucket* S3 permite-nos explorar outras medidas de segurança a utilizar na nuvem, uma vez que não é possível realizar filtragem do tráfego utilizando NSGs com serviços *serverless*.

Em termos de configuração, teremos de garantir que apenas a distribuição CloudFront tem permissões de leitura do *bucket*, de forma a garantir que qualquer tráfego oriundo da Internet passe pelos controlos de segurança assegurados pelo CloudFront. Para este fim, serão definidas políticas no sistema de gestão de identidades da AWS, o IAM (*Identity and Access Management*), que permitam apenas ao CloudFront aceder à *web page* hospedada pelo *bucket* S3.

6.6.5 Instância RDS – Componente de base de dados

Quanto ao componente de hospedagem da base de dados, uma vez que se optou por uma base de dados relacional, será utilizado uma instância do serviço Amazon RDS (*Relational Database Service*). Graças a este serviço temos algumas medidas de segurança asseguradas, nomeadamente (Amazon Web Services, 2021k):

- Automatização de *backups*.
- Provisionamento de instâncias em diversas regiões geográficas de forma a melhorar a disponibilidade.
- Replicação automática dos dados entre diferentes regiões.
- Encriptação dos dados em repouso e em trânsito.
- Isolamento ao nível da rede recorrendo a uma VPC.
- Mecanismos de monitorização e de *logging* integrados.

No que toca à sua configuração, apenas a instância EC2 poderá comunicar com a instância RDS, sendo completamente inacessível via Internet.

6.6.6 CloudWatch e CloudTrail – Componentes de monitorização e logging

Um dos principais controlos a assegurar ao nível da infraestrutura passa pelo uso de componentes de *logging* e monitorização para obter uma melhor visibilidade sobre o funcionamento do nosso sistema. Para uma infraestrutura implantada em AWS, os serviços CloudWatch e CloudTrail ajudam-nos a ir ao encontro destes controlos de segurança.

Através do uso do CloudWatch, conseguimos monitorizar métricas de *performance* e configurar alarmística para a nossa infraestrutura, aplicações e serviços, possibilitando a automação de respostas e alarmes para dados eventos (Amazon Web Services, 2021d).

E graças ao CloudTrail, temos um registo de todos os eventos ao nível da conta AWS, o que nos fornece maior visibilidade sobre quaisquer alterações à infraestrutura e aos nossos recursos (Amazon Web Services, 2021n).

Ambas estas soluções possuem integrações com o serviço *Key Management Service* que permite encriptar os *logs* gerados, o que vai ao encontro de mais um dos controlos de segurança levantados durante a análise do estado da arte (Amazon Web Services, 2021d) (Amazon Web Services, 2021n).

6.6.7 Identity and Access Management – AWS IAM

Para a gestão de identidades no ambiente nuvem, é utilizado o serviço de IAM da AWS. Este serviço permite-nos controlar o acesso e as permissões de utilizadores (ou grupos de utilizadores) aos serviços e recursos implantados na nuvem com um controlo granular específico a esse dado serviço (Amazon Web Services, 2021o).

O IAM possibilita ainda a definição de políticas de acesso, e disponibiliza ferramentas de análise que assistem na manutenção de uma política *least privilege* (Amazon Web Services, 2021o).

6.6.8 Key Management Service – AWS KMS

O serviço KMS fornece um controlo centralizado sobre as chaves utilizadas para a encriptação da informação detida nos vários recursos implantados em AWS. Diversos serviços da AWS disponibilizam integrações com o KMS de forma a facilitar a encriptação da informação (Amazon Web Services, 2021p).

Adicionalmente, este serviço permite a rotação automática de chaves geradas, que vai ao encontro dos controlos de segurança de aplicação de criptografia e da gestão de segredos (Amazon Web Services, 2021p).

6.6.9 GuardDuty

Através do serviço GuardDuty, são analisados eventos ao nível da conta AWS, bem como *logs* gerados pelo CloudTrail, de forma a identificar atividade potencialmente maliciosa ou mesmo recursos e contas de utilizador comprometidos (Amazon Web Services, 2021h).

Ao aplicar este serviço no nosso ambiente em nuvem, temos uma ferramenta adicional que nos ajuda a ir ao encontro do controlo de resposta a incidentes de segurança.

6.6.10 Amazon Inspector

O serviço Inspector executa avaliações à infraestrutura de forma a identificar pontos de entrada ou vulnerabilidades identificadas em instâncias EC2 implantadas. Este serviço permite ainda definir boas práticas aceites e validar se estão a ser adotadas no aplicativo, reforçando o controlo de política de programação segura (Amazon Web Services, 2021i).

6.6.11 Amazon Detective

O Detective consome os *logs* gerados por serviços como o Security Hub e GuardDuty e processa os dados de forma a sintetizá-los em informação pertinente para momentos de monitorização e de resposta a incidentes. Através do processamento desta informação, o Detective consegue correlacionar diversos eventos e centraliza os detalhes das suas descobertas num único local (Amazon Web Services, 2021e).

6.6.12 Security Hub

O Security Hub tem como principal função a agregação das descobertas feitas por serviços de segurança em funcionamento como a deteção de intrusos pelo GuardDuty ou avaliações de vulnerabilidades feitas pelo Inspector. A utilização deste serviço promove a visibilidade de eventos pertinentes à segurança da nossa infraestrutura e aplicações (Amazon Web Services, 2021q).

6.6.13 AWS Systems Manager

O principal motivo para a inclusão do serviço Systems Manager passa pela centralização da informação operacional dos nossos recursos implantados em nuvem, uma vez que podemos agregar recursos por diferentes grupos (por exemplo, ambiente de produção separado do ambiente de desenvolvimento) e obter uma melhor visibilidade sobre o funcionamento dos diversos componentes do sistema (Amazon Web Services, 2021r).

6.6.14 AWS Artifact

O serviço AWS Artifact não se encontra representado no diagrama da infraestrutura apresentado na Figura 11, uma vez que não se trata de um serviço que opere ao nível da infraestrutura. Porém, este serviço gratuito fornece acesso a diversos documentos da AWS relevantes aos controlos de conformidade e da gestão de eventos de segurança, como por exemplo o modelo de responsabilidade partilhada e outros documentos de certificações do CSP (Amazon Web Services, 2021m).

6.7 Resumo cobertura de controlos de segurança

Nesta secção do documento é sintetizada a cobertura dos controlos de segurança levantados no estado da arte de acordo com os serviços utilizados em AWS. A Tabela 14 apresentada abaixo explicita a cobertura atingida.

Tabela 14 - Tabela resumo de cobertura dos controlos de segurança

Área de atuação	Controlos relevantes	Serviços AWS
Testes à segurança	Testes à segurança do sistema	n/a (Testes de penetração e análise de código efetuados com recurso a ferramentas externas)
Gestão de eventos de segurança	Definição e separação de responsabilidades	IAM
	Resposta a incidentes de segurança	GuardDuty, Security Hub, Detective
Controlo de acessos	Política de controlo de acessos e gestão de permissões	IAM
	Gestão de segredos	KMS
Segurança ao nível operacional	Política de *backup*	RDS
	Registo de eventos	CloudWatch, CloudTrail
	Proteção de *logs*	CloudWatch, CloudTrail com KMS
	Gestão de vulnerabilidades	Security Hub
Gestão da informação	Inventário dos recursos computacionais	Systems Manager
	Classificação da informação	IAM
Conformidade	Requisitos da segurança da informação	CloudWatch, Inspector
	Identificação da legislação e requisitos contratuais	Artifact
Camada tecnológica	Política de desenvolvimento seguro	Uso de imagens de VMs *hardened*
Camada de arquitetura	Controlos criptográficos	KMS
	Controlos de segurança na rede	NSG, WAF, Shield, CloudFront, API Gateway
	Segregação na rede	VPCs, Subnets
	Ambiente de trabalho seguro	VPCs, Subnets

6.8 Implementação aplicação

Dando a secção de *design* como encerrada, nesta secção do documento será detalhado o processo de implementação dos componentes que constituem o aplicativo desenvolvido para efeito de prova de conceito.

6.8.1 Componente backend

Na implementação do componente *backend* foi adotada uma arquitetura de 3 camadas, composta por: a camada *Controller*, a camada *Model* e a camada *Repository*.

Cada uma destas camadas tem as suas responsabilidades claramente separadas, nomeadamente:

- *Controller*: Responsável por ser a interface com o componente *frontend*, compõe as respostas HTTP/HTTPS de acordo com a rota invocada no URL e traduz as entidades do negócio em objetos JSON.

- *Model*: Camada responsável por definir a lógica de negócio e efetuar validações aos dados.

- *Repository*: Camada responsável por servir de interface entre o componente *backend* e a base de dados e traduz as entidades de negócio em dados a armazenar.

Como referido na secção 6.3 deste documento, existem três entidade intervenientes ao negócio, sendo que cada uma delas terá um módulo correspondente em cada uma destas camadas. O diagrama de pacotes apresentado na Figura 12 apresentada abaixo permite explicitar a arquitetura adotada.

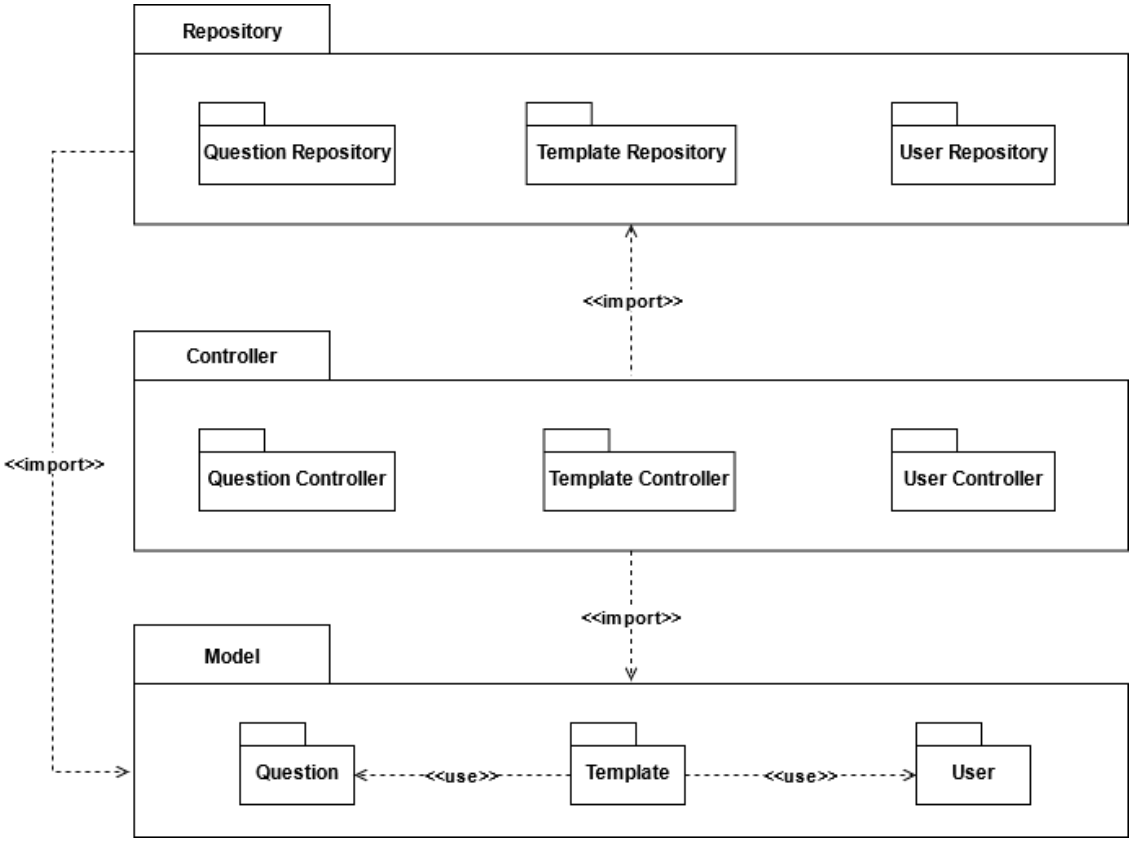


Figura 12 - Diagrama de pacotes do componente *backend*

Como também foi referido na secção 6.5 do documento, este componente vai expor apenas duas funcionalidades na API, nomeadamente:

- GET **/templates/{email}:** Onde o parâmetro *email* representa o email de um dado utilizador. Este pedido irá carregar da base de dados a *Template* do *User* identificado pelo email enviado como parâmetro.

- PUT **/questions/{id}:** Onde o parâmetro *id* representa o identificador único de uma *Question*. Este pedido irá marcar a *Question* com o dado *id* como completa ou incompleta.

Como este aplicativo serve apenas para efeito de prova de conceito, no momento de execução o componente *backend* realiza um *bootstrap* da base de dados com dados de teste.

6.8.2 Componente frontend

Para o componente *frontend* desenvolvido com React, foram criados apenas 2 ecrãs para interação do utilizador com o aplicativo, nomeadamente:

- Um primeiro ecrã para o *login* de um *User* através do seu email.

- Um segundo ecrã que apresenta a *Template* do respetivo *User* que realizou o *login.* Este ecrã trata-se de um componente React chamado de *Checklist*.

Assim que um *User* fornece um email com uma *Template* associada no ecrã de *login*, o componente *Checklist* carrega a *Template* para o seu estado.

Ao atualizar o estado do componente, o React despoleta uma atualização do ecrã (React Docs, 2021). Isto quer dizer que quando um *User* assinala uma das *Questions* carregada pela *Template* como completa, o React vai realizar o pedido assíncrono ao componente *backend* pela rota **/questions/{id}**, e vai ainda atualizar o estado do componente, que por sua vez vai fornecer *feedback* na interface gráfica que a *Question* ficou marcada como completa.

6.9 Implementação infraestrutura

Nesta secção do documento, é detalhado o processo de implementação da infraestrutura que vai suportar o aplicativo desenvolvido e servir como prova de conceito à exploração dos controlos de segurança levantados no estado da arte da dissertação.

Esta secção será dividida em subsecções de acordo com os componentes implantados em AWS.

6.9.1 VPC

O primeiro componente implantado foi a VPC, onde serão colocadas as instâncias EC2 e RDS e respetivas sub-redes. Como previamente mencionado na secção 5.2, uma das principais funcionalidades da VPC passa pela definição do intervalo de endereços IP privados de forma a obter uma segregação lógica da rede.

Para este fim, para a configuração da VPC será definido como intervalo IPv4 CIDR o bloco de endereços 10.0.0.0/16 (linha 2), o que nos disponibiliza o intervalo de IP desde o endereço 10.0.0.1 até ao endereço 10.0.255.254, resultando num total de 65536 endereços privados disponíveis (o que nos irá permitir o uso de várias redes dentro desta gama para funções específicas, como se verá de seguida).

Para além de definir o bloco CIDR, configuramos a VPC para permitir *hostnames* DNS (linhas 3 e 4), o que vai fazer com que as instâncias dentro desta VPC tenham *hostnames* DNS públicos atribuídos.

No excerto de Código 1 é apresentada a configuração da VPC usando Terraform.

```
1       resource "aws_vpc" "app-vpc" {
2         cidr_block = "10.0.0.0/16"
3         enable_dns_hostnames = true
4         enable_dns_support = true
5       }
```

Código 1 - Configuração da VPC

6.9.2 Subnets

Com a VPC criada, serão criadas as duas sub-redes onde serão implantadas as instâncias EC2 e RDS, sendo que a instância EC2 será colocada numa sub-rede pública, enquanto que a instância RDS será colocada numa sub-rede privada, uma vez que não necessita de qualquer acesso à Internet.

6.9.2.1 Sub-rede pública – EC2

Para a criação da sub-rede para a instância EC2, temos que definir um bloco de endereços CIDR, sendo utilizado o bloco de endereços 10.0.1.0/24, com 254 endereços IP privados disponíveis. Conforme definido pela AWS, para uma sub-rede se tornar efetivamente pública e acessível pela Internet, terá de ser associada a uma Route Table que possui uma rota para uma Internet Gateway (Amazon Web Services, 2021u).

Para além de configurar o bloco CIDR, temos que indicar a VPC à qual esta sub-rede pertence.

De forma a permitir que esta sub-rede consiga comunicar com a Internet, teremos que configurar componentes adicionais, nomeadamente:

- Uma Internet Gateway, de forma a permitir à nossa VPC que comunique com a Internet.

- Uma Route Table, com uma entrada que exponha a sub-rede à Internet. Assim, o tráfego da Internet que tenha como destino o endereço da instância EC2 é entregue com sucesso (desde que o tráfego tenha como origem um endereço IP *whitelisted*).

Para a configuração de uma Internet Gateway, apenas temos que indicar a VPC à qual esta ficará associada.

Para a configuração da Route Table, vamos indicar a VPC à qual esta tabela ficará associada. De seguida vamos adicionar uma regra onde indicamos os endereços IP públicos *whitelisted* que queremos que tenham acesso à instância EC2 via Internet, selecionando como *target* a Internet Gateway criada.

Com a Route Table configurada, resta associar a Route Table à sub-rede pública da instância EC2.

O bloco de código apresentado em Código 2 mostra a configuração resultante em Terraform, sendo que a variável **var.allowed_ips** na linha 11 armazena os endereços IP *whitelisted* que podem aceder à instância EC2 e a variável **var.subnet_backend_range** na linha 2 armazena o endereço desta rede, "10.0.1.0/24".

```
1       resource "aws_subnet" "subnet-backend" {
2         cidr_block = var.subnet_backend_range
3         vpc_id = aws_vpc.app-vpc.id
4       }

5       resource "aws_internet_gateway" "app-vpc-gw" {
6         vpc_id = aws_vpc.app-vpc.id
7       }

8       resource "aws_route_table" "vpc-route-table" {
9         vpc_id = aws_vpc.app-vpc.id

10        route {
11          cidr_block = var.allowed_ips
12         gateway_id = aws_internet_gateway.app-vpc-gw.id
13        }
14      }

15      resource "aws_route_table_association" "subnet-association" {
16        subnet_id = aws_subnet.subnet-backend.id
17        route_table_id = aws_route_table.vpc-route-table.id
18      }
```

Código 2 - Configuração da sub-rede pública

6.9.2.2 Sub-rede privada – RDS

De acordo com a documentação disponibilizada pela AWS, uma instância RDS não pode ficar associada a apenas uma sub-rede, tendo que ser colocada num Subnet Group para ser colocada numa VPC. Este Subnet Group será composto por duas sub-redes privadas, cada uma localizada numa zona de disponibilidade diferente (Amazon Web Services, 2021t).

Tendo esta restrição em conta, criamos uma primeira sub-rede com o bloco CIDR 10.0.2.0/24, associada à VPC criada e na zona de disponibilidade **euw1-az1.**

De seguida, criamos a segunda sub-rede com o bloco CIDR 10.0.3.0/24, associado também à VPC, mas desta vez com a zona de disponibilidade **euw1-az2**.

Com ambas as sub-redes criadas, resta criar um Subnet Group que as irá agregar.

O bloco de código apresentado em Código 3 demonstra a implementação da sub-rede privada usando Terraform, sendo que as variáveis **var.subnet_database_range** e **var.subnet_database_range2** nas linhas 2 e 7 armazenam os valores "10.0.2.0/24" e "10.0.3.0/24" respetivamente.

```
1       resource "aws_subnet" "subnet-database" {
2         cidr_block = var.subnet_database_range
3         vpc_id = aws_vpc.app-vpc.id
4         availability_zone_id = "euw1-az1"
5       }

6       resource "aws_subnet" "subnet-database-2" {
7         cidr_block           = var.subnet_database_range2
8         vpc_id               = aws_vpc.app-vpc.id
9         availability_zone_id = "euw1-az2"
10      }

11      resource "aws_db_subnet_group" "subnet-group-database" {
12        name = "database-subnet-group"
13        subnet_ids = [aws_subnet.subnet-database.id, aws_subnet.subnet-
database-2.id]
14      }
```

Código 3 - Configuração da sub-rede privada

6.9.3 Network Security Groups

Quando inseridos numa VPC, os NSG irão ter um funcionamento semelhante a uma *firewall* tradicional, podendo ser definidas regras *ingress* (tráfego de entrada) e *egress* (tráfego de saída) de forma a filtrar o tráfego de acordo com o endereço de origem e o porto de destino, entre outros possíveis filtros que serão abordados de seguida.

De momento, temos a instância EC2 inserida numa sub-rede pública exposta a tráfego da Internet, desde que origine de endereços IP *whitelisted*. Porém, não existe qualquer tipo de restrição ao nível da rede quanto aos portos que poderão ser alcançados pela Internet. Para isto, serão utilizadas as regras de filtragem ao nível dos NSG.

Resta ainda restringir o acesso à sub-rede privada da instância RDS apenas à sub-rede pública da instância EC2.

Para este fim, as instâncias EC2 e RDS terão os seus próprios NSG que irão aplicar estas restrições.

6.9.3.1 NSG EC2

Para a criação deste NSG, começamos por associá-lo à VPC criada, e são criadas 3 regras de *ingress*, que nos permitam restringir o tráfego à instância EC2 a apenas 3 portos[2]:

- Porto 22, para fazer ligação via SSH.

[2] Registo IANA dos portos: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

- Porto 8080, onde será servida a API REST do componente *backend.*

- Porto 3306, onde será feita a comunicação com a base de dados.

No que toca ao tráfego *egress*, não serão definidas quaisquer regras de filtragem.

Para este fim, foi criada a configuração em Terraform ilustrada no excerto Código 4.

```
1       resource "aws_security_group" "subnet-backend-sg" {
2         name   = "ec2-sg"
3         vpc_id = aws_vpc.app-vpc.id

4         ingress {
5           from_port   = 3306
6           to_port     = 3306
7           protocol    = "tcp"
8           description = "MySQL"
9           cidr_blocks = [var.subnet_database_range]
10        }

11        ingress {
12          cidr_blocks = [
13            var.allowed_ips
14          ]
15          from_port = 22
16          to_port   = 22
17          protocol  = "tcp"
18        }

19        ingress {
20          cidr_blocks = [
21            var.allowed_ips
22          ]
23          from_port   = 8080
24          to_port     = 8080
25          protocol    = "tcp"
26          description = "HTTP"
27        }

28        egress {
29          from_port   = 0
30          to_port     = 0
31          protocol    = "-1"
32          cidr_blocks = ["0.0.0.0/0"]
33        }
34      }
```

Código 4 - Configuração do NSG da instância EC2

Das linhas 4 a 10, definimos a regra *ingress* para permitir tráfego com origem (linha 5) e destino (linha 6) do porto 3306, desde que o tráfego venha da sub-rede privada da instância RDS (linha 9).

Das linhas 11 a 18 e das linhas 19 a 27, são criadas as regras de filtragem para os portos 22 e 8080, respetivamente. Sendo a comunicação com estes portos restrita aos endereços IP *whitelisted* e restrito ao protocolo TCP.

Finalmente, criamos uma regra *egress* nas linhas 28 a 33 que indica que todo o tráfego de saída é permitido, razão pela qual se observa na definição associada a **cidr_blocks** o valor "0.0.0.0/0".

6.9.3.2 NSG RDS

Para o NSG da instância RDS queremos garantir que apenas expomos o porto 3306, utilizado por omissão pelo motor MySQL, apenas ao NSG da instância EC2.

Para este fim necessitamos de criar apenas uma regra de *ingress* e uma regra de *egress*.

O bloco de código Terraform apresentado em Código 5 explicita a configuração necessária, sendo que nas linhas 5 e 12 restringimos o tráfego apenas ao NSG criado para a instância EC2.

```
1       resource "aws_security_group" "subnet-database-sg" {
2         name   = "rds-sg"
3         vpc_id = aws_vpc.app-vpc.id

4         ingress {
5           security_groups = [aws_security_group.subnet-backend-sg.id]
6           from_port       = var.rds_port
7           to_port         = var.rds_port
8           protocol        = "tcp"
9           description     = "MySQL"
10        }

11        egress {
12          security_groups = [aws_security_group.subnet-backend-sg.id]
13          from_port       = var.rds_port
14          to_port         = var.rds_port
15          protocol        = "tcp"
16          description     = "MySQL"
17        }
18      }
```

Código 5 – Configuração do NSG da instância RDS

6.9.4 Instância EC2

Com a infraestrutura necessária implantada, procedemos à implantação da instância EC2 onde será executado o componente *backend*.

No excerto Código 6 é apresentado o bloco de código Terraform utilizado para a configuração da EC2.

```
1       resource "aws_instance" "api_server" {
2         ami = "ami-063d4ab14480ac177"
3         associate_public_ip_address = true
4         instance_type               = "t2.micro"
5         key_name                    = "EC2Keypair"
6         security_groups             = [aws_security_group.subnet-backend-
sg.id]
7         subnet_id = aws_subnet.subnet-backend.id
8 }
```

Código 6 – Configuração da instância EC2

Para a criação da instância, começamos por indicar o identificador único da imagem a utilizar para a máquina (linha 2), sendo neste caso utilizada uma imagem baseada em Linux, incluída no plano gratuito da AWS. Num cenário ideal, a imagem a utilizar na instância EC2 seria profissionalmente *hardened* de forma a reduzir a sua superfície de ataque.

De seguida, na linha 3 do excerto, indicamos que queremos atribuir um endereço IP público à instância de forma a esta ser acessível via Internet.

Na linha 4, indicamos que queremos uma instância do tipo **t2.micro**, que influencia as características da máquina virtual, nomeadamente o tipo de armazenamento, a memória e poder de processamento. Foi selecionado este tipo de instância uma vez que é o único incluído no plano gratuito da AWS.

Na linha 5 do excerto, associamos à instância EC2 um par de chaves: uma pública e uma privada, identificadas pelo nome “EC2Keypair”. Isto permitirá a um utilizador identificar-se quando tenta ligar-se à máquina via SSH. Este mecanismo, aliado ao endereço IP *whitelisted* ajuda-nos a garantir que apenas utilizadores com permissões para tal poderão aceder à máquina EC2. A configuração desta chave será abordada de seguida.

Nas linhas 6 e 7 procedemos a associar a instância EC2 à infraestrutura implantada previamente, atribuindo-lhe o NSG e sub-rede respetivos.

6.9.5 Key Pair para instância EC2

Como previamente mencionado, para um utilizador aceder à instância EC2 via SSH, terá que se identificar com uma chave privada.

Para gerar este par de chaves pública e privada, é usado o seguinte excerto de código Terraform apresentado em Código 7.

```
1       resource "tls_private_key" "webserver_private_key" {
2         algorithm = "RSA"
3         rsa_bits  = 2048
4       }

5       resource "aws_key_pair" "webserver_key" {
6         key_name   = "EC2Keypair"
7         public_key = tls_private_key.webserver_private_key.
public_key_openssh
8       }
```

Código 7 – Configuração de um Keypair

Começamos nas linhas 1 a 4 por gerar um *resource* do tipo TLS Private Key, utilizando o RSA como algoritmo de encriptação com um bloco de 2048 bits para gerar o par de chaves.

De seguida, nas linhas 5 a 8, instanciamos um *resource* em AWS para identificar o par de chaves criado pelo nome "EC2Keypair".

6.9.6 Instância RDS

Para concluir a implantação dos componentes incluídos na VPC criada, resta apenas implantar a instância RDS que irá hospedar a base de dados do aplicativo.

Para a configuração da instância RDS foi utilizado o seguinte excerto de código Terraform apresentado em Código 8.

```
1       resource "aws_db_instance" "poc_db" {
2         allocated_storage               = 10
3         engine                          = "mysql"
4         instance_class                  = "db.t2.micro"
5         multi_az                        = false
6         name                            = "poc_db"
7         username                        = "username"
8         password                        = "password"
9         publicly_accessible             = false
10        vpc_security_group_ids          = [aws_security_group.
subnet-database-sg.id]
11        db_subnet_group_name            = aws_db_subnet_group.
subnet-group-database.name
12        enabled_cloudwatch_logs_exports = ["error", "slowquery"]
13      }
```

Código 8 – Configuração da instância RDS

Na linha 2 começamos por definir o tamanho máximo da instância em Gigabytes. Temos ainda que definir o tipo de motor de base de dados relacional que a instância RDS irá executar. Na linha 3 indicamos à instância para utilizar o motor MySQL.

Tal como com a instância EC2, temos que indicar as características da instância a utilizar, sendo que neste caso usamos o tipo **db.t2.micro*,*** incluído no plano gratuito.

Na linha 9 do excerto indicamos à instância que esta deve ser inacessível publicamente.

Nas linhas 10 e 11 procedemos a associar a instância à infraestrutura implantada previamente, nomeadamente o NSG e o Subnet Group.

Na linha 12, indicamos à instância RDS para exportar os *logs* do tipo *error* e *slow query* para o serviço CloudWatch, que será configurado futuramente.

Das linhas 6 a 8 procedemos à configuração do nome da base de dados e das credenciais do utilizador root. Este excerto trata-se de um exemplo de implementação, sendo que idealmente a palavra-passe a utilizar seria encriptada pelo serviço KMS, e não armazenada em texto no ficheiro de configuração.

Para encriptar as credenciais de acesso usando o KMS, pode ser usado o comando: **aws kms encrypt --key-id [KEY_ID] --plaintext [PASSWORD] --encryption-context foo=bar --output text --query CiphertextBlob**, onde o campo **KEY_ID** deve conter o identificador único da *key* gerada que irá encriptar as credenciais, o campo **PASSWORD** contém a credencial em *plaintext* ou o caminho para um ficheiro que contém a credencial e o parâmetro **encryption-context** fornece contexto para a encriptação do segredo (Terraform Registry, 2021a).

Da execução deste comando resulta uma *payload* no formato Base64 encriptada usando a *key* fornecida, que deve ser incluída no bloco de código Terraform em Código 9, substituindo o parâmetro **ENCODED_PAYLOAD** da linha 4.

```
1       data "aws_kms_secrets" "db_credential" {
2         secret {
3           name    = "master_password"
4           payload = "[ENCODED_PAYLOAD]"

5           context = {
6             foo = "bar"
7           }
8         }
9       }
```

Código 9 - Acesso a credenciais encriptadas usando KMS

Este segredo poderá depois ser utilizado na configuração do RDS através da seguinte linha de código: **password = data.aws_kms_secrets.db_credential.plaintext["master_password"]** (Terraform Registry, 2021a).

6.9.7 API Gateway

Com a VPC que contém o componente *backend* e a base de dados implantada com sucesso, teremos de garantir que o acesso à API será apenas realizado pelo componente CloudFront, que será a primeira camada de defesa ao nível da infraestrutura. Para este fim, será instanciada uma API Gateway, de forma a ter um maior controlo sobre o controlo de acessos à API do *backend*, e também de forma a obrigar o uso de HTTPS independentemente da implementação do componente *backend*.

No que toca à configuração deste componente, vamos tratar de disponibilizar as rotas descritas na secção 6.8.1 do documento, porém, vamos indicar à API Gateway para permitir o acesso a estas rotas apenas perante a apresentação de uma chave válida enviada nos *headers* do pedido HTTP. A definição OpenAPI apresentada em Código 10 exemplifica a implementação da rota **/templates/{email}** na API Gateway.

```
1       (var.get_api_path) = {
2         get = {
3           "parameters" : [
4            {
5             "name" : "email",
6             "in" : "path",
7             "required" : true,
8             "schema" : {
9               "type" : "string"
10             }
11           }
12          ],
13          "security" : [
14           {
15            "api_key" : []
16           }
17          ],
18          x-amazon-apigateway-integration = {
19            "uri" : "http://${aws_instance.api_server.
public_ip}:8080/templates/{email}",
20            "httpMethod" : "GET",
21            "requestParameters" : {
22              "integration.request.path.email" :
"method.request.path.email"
23            },
24            "type" : "http_proxy"
25           }
26        }
27      }
```

Código 10 - Configuração da rota GET Templates na API Gateway

Na linha 1 a variável **var.get_api_path** armazena a rota a ser usada na API Gateway, sendo mantida a rota **/templates/{email}**.

Das linhas 3 a 12 é declarado o parâmetro *email* a ser passado na rota e indicamos à API Gateway que se trata de um parâmetro obrigatório do tipo String.

Das linhas 18 a 31 é configurada a integração da API Gateway com o componente *backend*. Neste bloco é definido o URL onde a instância EC2 vai servir a API REST (linha 19), o método HTTP (linha 20), o parâmetro email que é incluído na rota (linhas 21 a 23) e o tipo de integração com o componente *backend* que usa HTTP (linha 24).

Das linhas 13 a 17 declaramos o uso da **api_key** no bloco de segurança, cuja configuração será apresentada em Código 11 conforme a especificação OpenAPI.

```
1       "components" : {
2         "securitySchemes" : {
3           "api_key" : {
4             "type" : "apiKey",
5             "name" : "x-api-key",
6             "in" : "header"
7           }
8         }
9       }
```

Código 11 - Configuração da API Key na API Gateway

Com este bloco definimos a necessidade de uma **apiKey** (linha 4) com o nome **api_key** (linha 3) que deve ser enviada nos *headers* (linha 6) com o nome **x-api-key** (linha 5) nos pedidos HTTP que cheguem à API Gateway.

O bloco de código Terraform para gerar a chave de acesso à API é apresentado em Código 12.

```
1       resource "aws_api_gateway_api_key" "cloudfront_api_key" {
2         name = "api_key"
3       }
```

Código 12 - Instanciação de uma API Key usando Terraform

A chave de acesso à API é gerada no momento de implantação e é partilhada apenas com a distribuição CloudFront, que trata de injetar a chave nos *headers* nos pedidos HTTP dirigidos à API, garantindo que qualquer pedido válido à API Gateway origine unicamente do CloudFront.

6.9.8 Bucket S3

No que toca aos componentes relativos à aplicação resta apenas implantar o componente *frontend* desenvolvido em ReactJS. Para este fim, será utilizado um *bucket* S3, capaz de hospedar um *website* estático, como o desenvolvido para a prova de conceito.

No que toca à configuração do *bucket* S3, uma vez que se trata de um componente *serverless*, a principal preocupação não será ao nível da rede como nas instâncias EC2 e RDS, mas sim no controlo de acesso ao *bucket* via as funcionalidades do IAM. O foco passa por garantir que o *bucket* poderá servir o componente *frontend* ao CloudFront, porém a gestão do *bucket* deve ficar restrita à conta AWS do criador do recurso. Para isto, podemos utilizar uma *canned* ACL privada que dê controlo total sobre o *bucket* apenas ao dono (Amazon Web Services, 2021a).

Para garantir que a distribuição CloudFront consiga servir a *web page*, temos que garantir permissões de leitura do *bucket*. No excerto Código 13 é apresentada a configuração Terraform utilizada para este fim.

```
1       data "aws_iam_policy_document" "s3_iam_doc" {
2         statement {
3           actions   = ["s3:GetObject"]
4           resources = ["${aws_s3_bucket.react_bucket.arn}/*"]

5           principals {
6               type        = "AWS"
7               identifiers = [aws_cloudfront_origin_access_identity.
cloudfront.iam_arn]
8           }
9         }

10        statement {
11          actions   = ["s3:ListBucket"]
12          resources = [aws_s3_bucket.react_bucket.arn]

13          principals {
14            type        = "AWS"
15            identifiers = [aws_cloudfront_origin_access_identity
.cloudfront.iam_arn]
16          }
17        }
18      }
```

Código 13 - Configuração de política IAM para garantir permissões de leitura de bucket S3 à distribuição CloudFront

É declarada uma política IAM a atribuir ao *bucket* S3 composta por dois *statements*, das linhas 2 a 9 e das linhas 10 a 17. Ambos os *statements* têm como alvo o *bucket* S3, e garantem à distribuição CloudFront as permissões **s3:GetObject** e **s3:ListBucket** nas linhas 3 e 11, necessárias para garantir acesso à *web page* hospedada no *bucket*.

6.9.9 Distribuição CloudFront

Com os componentes *frontend* e *backend* implantados, pode-se proceder à implantação da distribuição CloudFront, que ajudará a ocultar a *stack* tecnológica do aplicativo e servirá como ponto de integração com outros componentes de segurança, nomeadamente de WAF e de proteção contra ataques DoS (a integração com estes controlos não será explorada porque não estão incluídos no plano gratuito da AWS).

Sendo a primeira camada de defesa da aplicação, a distribuição CloudFront terá de servir de ponto de entrada tanto para o componente *frontend* como para o componente *backend*.

Para isto, teremos de configurar duas origens que o CloudFront vai servir, uma origem para o *bucket* S3 e outra para a API a ser servida pela API Gateway. Em Código 14 é apresentado o bloco de código Terraform utilizado para este efeito.

```
1       origin {
2         domain_name = aws_s3_bucket.react_bucket.bucket_
regional_domain_name
3         origin_id   = aws_s3_bucket.react_bucket.bucket

4         s3_origin_config {
5           origin_access_identity = aws_cloudfront_origin_
access_identity.cloudfront.cloudfront_access_identity_path
6         }
7       }

8       origin {
9         domain_name = "${aws_api_gateway_rest_api.api.id}.execute-api.eu-
west-1.amazonaws.com"
10        origin_id   = "api-gateway"

11        custom_origin_config {
12             http_port              = 80
13             https_port             = 443
14             origin_protocol_policy = "https-only"
15             origin_ssl_protocols   = ["TLSv1.2"]
16        }

17        custom_header {
18             name  = "x-api-key"
19             value = aws_api_gateway_api_key.cloudfront_api_key.value
20        }
21      }
```

Código 14 - Configuração das origens da distribuição CloudFront

Das linhas 1 a 7 é configurada uma origem para o *bucket* S3, enquanto que nas linhas 8 a 21 é configurada uma origem *custom* para a API Gateway. Para o CloudFront se identificar nos pedidos que realiza à API Gateway, é inserida a API Key gerada nos *headers* dos pedidos HTTP, como pode ser observado nas linhas 17 a 20.

6.9.10 CloudWatch

Agora que diversos dos componentes essenciais ao funcionamento da prova de conceito se encontram implantados em nuvem, poderá ser feita a sua integração com o CloudWatch, responsável pelos controlos de monitorização e alarmística da infraestrutura.

Idealmente, este serviço centraliza os dados de monitorização dos diversos componentes que constituem a infraestrutura, podendo ser configurados alarmes para deteção de comportamento anómalo (por exemplo, um elevado número de pedidos falhados à API num curto espaço de tempo). Para o efeito de prova de conceito, o CloudWatch será integrado com a instância RDS e com a API Gateway.

A exportação dos *logs* gerados durante a execução da base de dados hospedada no RDS para o CloudWatch permite identificar a origem de falhas que possam surgir no serviço e automatizar o lançamento de alarmes de acordo com as métricas do serviço. Para ativar a exportação de *logs* do RDS via Terraform, basta indicar ao parâmetro

enabled_cloudwatch_logs_exports, os tipos de *logs* a exportar, como foi feito na secção 6.9.6 do documento.

Para a exportação dos *logs* de execução da API Gateway, teremos de criar um *log group* onde estes serão agregados. O bloco de código Terraform apresentado em Código 15 demonstra como criar um *log group* para a agregação dos *logs* gerados pela API Gateway.

```
1       resource "aws_cloudwatch_log_group" "api-gateway" {
2         name              = "API-Gateway-Execution-
Logs_${aws_api_gateway_rest_api.api.id}/api"
3       }
```

Código 15 - Configuração do *Log Group* para a API Gateway

Com o *log group* criado, é necessário ainda definir uma política IAM que forneça à API Gateway as permissões necessárias para armazenar os *logs* gerados no CloudWatch (Terraform Registry, 2021c).

Conforme referido na secção 6.6.6 do documento, outra das principais funcionalidades do CloudWatch passa pela configuração de alarmística. Uma vez que temos uma API Gateway que aceita apenas pedidos HTTP que incluam uma chave válida, podemos recorrer ao uso de alarmes do CloudWatch para lançar notificações quando um dado número de pedidos sem uma chave válida chegam à API Gateway, uma vez que pode constituir uma tentativa de ataque ao aplicativo.

Para configurar um alarme no CloudWatch para este tipo de evento, pode ser utilizado o excerto de código Terraform em Código 16.

```
1       resource "aws_cloudwatch_metric_alarm" "api_gateway_alarm" {
2         alarm_name                = "api-forbidden-reqs-alarm"
3         comparison_operator       = "GreaterThanThreshold"
4         evaluation_periods        = "1"
5         metric_name               = "4XXError"
6         namespace                 = "AWS/ApiGateway"
7         period                    = "60"
8         statistic                 = "Sum"
9         threshold                 = "3"
10        alarm_description         = "This metric monitors forbidden
requests to the API Gateway"
11        insufficient_data_actions = []

12        dimensions = {
13          ApiName = var.rest_api_name
14          Stage   = "api"
15        }
16      }
```

Código 16 - Configuração de alarme CloudWatch para pedidos à API Gateway sem chave válida

Neste excerto definimos que quando o número de pedidos à API com código de resposta 4XX (linha 5) num dado período de tempo (linha 7) é maior (linha 3) que o valor *threshold* (linha 9), deve ser lançado um alarme. Este alarme funciona para o efeito descrito anteriormente

porque aceder a uma rota da API sem uma chave válida incluída nos *headers* retorna uma resposta HTTP com o código 403 Forbidden.

Nas linhas 12 a 15 configuramos que este alarme se destina à API Gateway criada anteriormente.

6.9.11 CloudTrail

Utilizado como complemento ao CloudWatch, o serviço CloudTrail fornece monitorização das ações ao nível da conta AWS, bem como de chamadas aos serviços e componentes, como por exemplo, pedidos à API Gateway. Trata-se de uma ferramenta valiosa para a identificação de potenciais atores maliciosos.

Para a sua configuração, precisamos de criar um novo *bucket* S3 onde serão armazenados os *logs* recolhidos durante a execução.

Em Código 17 é apresentado o código Terraform utilizado para a configuração do *bucket* destinado aos *logs*.

```
1       data "aws_caller_identity" "current" {}

2       resource "aws_s3_bucket" "cloudtrail-bucket" {
3         bucket        = "framework-poc-cloudtrail-logs-s3"

4         policy = <<POLICY
5       {
6          "Version": "2012-10-17",
7          "Statement": [
8              {
9                "Sid": "AWSCloudTrailAclCheck",
10               "Effect": "Allow",
11               "Principal": {
12                    "Service": "cloudtrail.amazonaws.com"
13               },
14               "Action": "s3:GetBucketAcl",
15               "Resource": "arn:aws:s3:::framework-poc-cloudtrail-logs-
s3"
16             },
17             {
18               "Sid": "AWSCloudTrailWrite",
19               "Effect": "Allow",
20               "Principal": {
21                    "Service": "cloudtrail.amazonaws.com"
22               },
23               "Action": "s3:PutObject",
24               "Resource": "arn:aws:s3:::framework-poc-cloudtrail-logs-
s3/CloudTrailLogs/AWSLogs/${data.aws_caller_identity.
current.account_id}/*",
25               "Condition": {
26                    "StringEquals": {
27                        "s3:x-amz-acl": "bucket-owner-full-control"
28                    }
29               }
30            }
31         ]
32      }
33      POLICY
34      }
```

Código 17 – Configuração das políticas IAM do *bucket* S3 que armazena os *logs*

Das linhas 2 a 32 é configurado o *bucket* S3 que vai armazenar os *logs* e é definida a política IAM que fornece permissões ao CloudTrail para armazenar os *logs*.

Na definição da política de acesso ao *bucket* temos dois *statements*, nas linhas 8 a 16 e nas linhas 17 a 30. Ambas estas permissões são necessárias para permitir que o CloudTrail armazene os *logs* no *bucket* S3 criado.

Na linha 1 do excerto de código vamos buscar a informação da conta AWS a ser utilizada de forma a na linha 24 ir buscar os *logs* gerados por esta conta, para que possam ser armazenados no *bucket* S3.

Com o *bucket* S3 configurado com a permissões necessárias, pode-se proceder à configuração do CloudTrail conforme em Código 18.

```
1       resource "aws_cloudtrail" "cloudtrail" {
2         name = "cloudtrail"
3         s3_bucket_name = aws_s3_bucket.cloudtrail-bucket.id
4         s3_key_prefix = "CloudTrailLogs"

5         event_selector {
6           read_write_type = "All"
7           include_management_events = true

8           data_resource {
9              type = "AWS::S3::Object"
10             values = ["${aws_s3_bucket.react_bucket.arn}/"]
11          }
12        }
13      }
```

Código 18 - Configuração do Cloudtrail

Na linha 3 do excerto indicamos ao CloudTrail o *bucket* S3 a utilizar para armazenar os *logs* no bucket recém criado para o efeito.

Das linhas 5 a 12 indicamos ao CloudTrail para registar os eventos do *bucket* S3 (linha 9) destinado ao componente *frontend* (linha 10).

6.9.12 GuardDuty

O GuardDuty não necessita de nenhuma configuração para além da ativação do serviço para começar a recolher informação e a fornecer *feedback* sobre como melhorar a postura de segurança da conta AWS. A ativação do serviço pode ser feita via Terraform usando o seguinte bloco de código em Código 19.

```
1       resource "aws_guardduty_detector" "MyDetector" {
2         enable = true
3       }
```

Código 19 - Configuração GuardDuty

6.9.13 Amazon Inspector

Para a identificação de possíveis vulnerabilidades da instância EC2 que hospeda o componente *backend*, será instalado um agente na máquina que, durante a sua execução, comunica as suas descobertas ao serviço Inspector, visando melhorar a postura de segurança deste recurso.

Para a configuração do serviço Inspector temos duas principais preocupações:

- Configurar a instância EC2 alvo e respetivas regras a utilizar durante a avaliação.

- Integrar a instância EC2 com o serviço Run Command do Systems Manager, que permitirá a instalação do agente que irá executar na instância EC2 e recolher informação sobre a sua postura de segurança.

Para a configuração da instância-alvo da avaliação do serviço Inspector foi utlizado o bloco de código Terraform apresentado em Código 20.

```
1       resource "aws_inspector_resource_group" "inspector" {
2         tags = {
3           Name = var.instance_name
4         }
5       }

6       resource "aws_inspector_assessment_target" "inspector" {
7         name = "assessment target"
8         resource_group_arn = aws_inspector_resource_group.inspector.arn
9       }

10      resource "aws_inspector_assessment_template" "inspector" {
11        name = "inspector"
12        target_arn = aws_inspector_assessment_target.inspector.arn
13        duration = 3600

14        rules_package_arns = [
15          "arn:aws:inspector:eu-west-1:357557129151:rulespackage/0-
ubA5XvBh",
16          "arn:aws:inspector:eu-west-1:357557129151:rulespackage/0-
sJBhCr0F",
17          "arn:aws:inspector:eu-west-1:357557129151:rulespackage/0-
SPzU33xe",
18          "arn:aws:inspector:eu-west-1:357557129151:rulespackage/0-
SnojL3Z6",
19        ]
20      }
```

Código 20 - Configuração Amazon Inspector

Das linhas 1 a 5 é criado um *resource group* que vai agregar as instâncias EC2 a serem incluídas na avaliação do Inspector, sendo que o *resource group* criado é composto pela instância EC2 criada na secção 6.9.4 do documento, identificada pelo seu nome armazenado na variável **var.instance_name** (linha 3).

Das linhas 6 a 9 é criado um recurso *assessment target* que vai ser utilizado para associar um conjunto de regras da avaliação ao *resource group* criado para a instância EC2, fornecendo apenas um nome na linha 7 e o ARN do *resource group* criado na linha 8.

Das linhas 10 a 20 é criada a *template* a ser utilizada na avaliação a ser feita pelo Inspector. Na linha 11 é fornecido um nome que vai identificar a *template*, na linha 12 é configurado o *target* criado anteriormente e na linha 13 é indicado ao Inspector que o agente na instância EC2 deve executar durante 3600 segundos, ou seja, 1 hora quando esta *template* for executada. Das linhas 14 a 19 indicamos à *template* do Inspector os conjuntos de regras a utilizar na avaliação da instância EC2, estando os conjuntos de regras identificados pelos seus códigos únicos (Amazon Web Services, 2021j).

Com a máquina-alvo e as regras do Inspector configuradas, resta garantir que o agente Inspector na instância EC2 consegue ser instalado e executado com sucesso. Para este fim, será criado um *role* IAM para a instância EC2 que a permita interagir com o Systems Manager. Para esta configuração foi utilizado o bloco de código Terraform apresentado em Código 21.

```
1       resource "aws_iam_role" "ssm_role" {
2         name = "ssm-ec2"
3         assume_role_policy = <<EOF
4       {
5         "Version": "2012-10-17",
6         "Statement": [
7             {
8             "Action": "sts:AssumeRole",
9               "Principal": {
10                "Service": "ec2.amazonaws.com"
11              },
12            "Effect": "Allow",
13            "Sid": ""
14            }
15        ]
16      }
17      EOF
18      }

19      resource "aws_iam_instance_profile" "ssm_profile" {
20        name = "ssm-ec2"
21        role = aws_iam_role.ssm_role.id
22      }

23      resource "aws_iam_policy_attachment" "ssm_attach1" {
24        name = "attachment"
25        roles = [aws_iam_role.ssm_role.id]
26        policy_arn = "arn:aws:iam::aws:policy/AmazonSSM
ManagedInstanceCore"
27      }

28      resource "aws_iam_policy_attachment" "ssm_attach2" {
29        name = "attachment"
30        roles = [aws_iam_role.ssm_role.id]
31        policy_arn = "arn:aws:iam::aws:policy/service-
role/AmazonEC2RoleforSSM"
32      }
```

Código 21 - Configuração *role* IAM para instância EC2

Neste excerto, das linhas 1 a 18 configuramos um *role* IAM destinado à instância EC2 (linha 10) que permitirá (linha 12) obter novas permissões (linha 8).

Das linhas 19 a 22 é criado um *instance profile* que ficará associado à instância EC2.

Das linhas 23 a 27 e 28 a 32 são criadas as políticas IAM necessárias para a instância EC2 interagir com o serviço Systems Manager e permite a instalação e execução do agente Inspector.

Resta apenas associar à instância EC2 o *instance profile* criado acima, o que pode ser feito adicionando a linha de código Terraform em Código 22 ao bloco de configuração da instância EC2.

```
iam_instance_profile = aws_iam_instance_profile.ssm_profile.id
```

Código 22 - Atribuição *role* IAM a instância EC2

6.9.14 Amazon Detective

No momento da escrita deste documento o serviço Detective não aparenta ser configurável via Terraform, uma vez que não existe qualquer documentação oficial sobre o seu uso, logo a implementação seria realizada pelo aplicativo *web* da AWS ou via a linha de comandos da AWS.

Porém, dada a restrição de uso do plano gratuito da AWS, não foi possível implementar este controlo, uma vez que necessita de ter o serviço GuardDuty em execução por mais de 48 horas (Amazon Web Services, 2021f), o que poderia incorrer em custos à conta AWS dado o tempo de funcionamento.

6.9.15 Security Hub

Através do uso deste serviço pretende-se centralizar as descobertas relacionadas à segurança num local de forma a melhorar a visibilidade sobre o funcionamento do nosso aplicativo.

Para a configuração do serviço Security Hub, foi utilizado o bloco de código em Código 23.

```
1       resource "aws_securityhub_account" "securityhub" {}

2       resource "aws_securityhub_standards_subscription" "cis" {
3         depends_on    = [aws_securityhub_account.securityhub]
4         standards_arn = "arn:aws:securityhub:::ruleset/cis-aws-
foundations-benchmark/v/1.2.0"
5       }

6       data "aws_region" "current" {}

7       resource "aws_securityhub_product_subscription"
"guardduty_findings" {
8         depends_on  = [aws_securityhub_account.securityhub]
9         product_arn = "arn:aws:securityhub:${data.aws_region.current
.name}::product/aws/guardduty"
10      }

11      resource "aws_securityhub_product_subscription"
"inspector_findings" {
12        depends_on  = [aws_securityhub_account.securityhub]
13        product_arn = "arn:aws:securityhub:${data.aws_region.current
.name}::product/aws/inspector"
14      }
```

Código 23 - Configuração SecurityHub

Na linha 1 do excerto é ativado o serviço na conta AWS que implanta a infraestrutura.

Das linhas 2 a 5 indicamos ao serviço para adotar as regras de segurança estabelecidas pela instituição CIS, que vão determinar os critérios de avaliação de cada vulnerabilidade identificada.

Na linha 6 é armazenada a região AWS em uso, necessária para a integração do Security Hub com os restantes serviços.

Das linhas 7 a 10 e das linhas 11 a 14 o Security Hub é integrado com os serviços GuardDuty e Inspector, respetivamente, de forma a centralizar as descobertas destes serviços.

7 Experimentação e Avaliação

Neste capítulo da dissertação é avaliada a adequação da solução desenvolvida através da definição dos indicadores utilizados na avaliação, as hipóteses a serem testadas, a metodologia preconizada e finalmente, os resultados atingidos.

7.1 Indicadores

Para se proceder à avaliação da solução, terão de ser definidos os indicadores que permitem medir a adequação do processo desenvolvido para responder ao problema apresentado na secção 1.1. Para este fim, foram definidos os seguintes indicadores tendo por base os objetivos do projeto:

- Número e criticidade de vulnerabilidades detetadas nas avaliações de vulnerabilidades.
- Capacidade de resposta e prevenção de ataques informáticos.
- Adequação dos controlos de segurança levantados ao longo do projeto.

7.2 Especificação da hipótese

Para a avaliação dos indicadores propostos devem ser definidas hipóteses que permitam avaliar se os objetivos propostos na dissertação foram atingidos.

Para cada um dos indicadores definidos acima foram formuladas duas hipóteses:

- A hipótese nula (H_0), que implica que os objetivos não foram atingidos.
- A hipótese alternativa (H_1) é a rejeição da hipótese nula, em que os objetivos foram atingidos com sucesso.

Na Tabela 15 apresentada abaixo estão descritas as hipóteses para cada um dos indicadores.

Tabela 15 - Especificação da hipótese

Indicador	H_0	H_1
Vulnerabilidades identificadas	A solução tem um número reduzido de vulnerabilidades identificadas e de baixa criticidade	A solução apresenta um número considerável de vulnerabilidades e/ou com elevada criticidade
Resposta e prevenção de ataques	Vulnerabilidade de criticidade elevada explorada com sucesso	Incapacidade de explorar de vulnerabilidades identificadas com maior impacto
Satisfação com o processo desenvolvido no projeto	A satisfação com o processo é inferior a 80%	A satisfação com o processo é igual ou superior a 80%

7.3 Métodos de avaliação

Nesta secção são apresentados os métodos a utilizar para avaliar cada um dos indicadores identificados. A Tabela 16 elenca os métodos de avaliação para cada um dos indicadores.

Tabela 16 - Métodos de avaliação

Indicador	Método
Vulnerabilidades identificadas	Avaliação de vulnerabilidades (*Scan* de vulnerabilidades)
Resposta e prevenção de ataques	Testes de penetração e mecanismos de deteção
Satisfação com o processo desenvolvido	Inquérito de satisfação

7.3.1 Avaliação de vulnerabilidades

Para a identificação e classificação de vulnerabilidades identificadas na prova de conceito desenvolvida serão usadas ferramentas de identificação e avaliação de vulnerabilidades, nomeadamente, a versão *essentials* do Nessus (Tenable, 2021), o Zed Attack Proxy (ZAP) (ZAP Dev Team, 2022) e o Legion (GoVanguard, 2020).

Através do uso de três ferramentas distintas, pretendemos obter uma melhor perspetiva sobre o nível de segurança aplicacional e da infraestrutura, uma vez que este tipo de ferramentas de análise de vulnerabilidades apresenta, frequentemente, um valor elevado de falsos positivos.

Usando o Nessus será realizado um *scan* avançado ao nível de rede à instância EC2 e um *scan* de vulnerabilidades *web* à distribuição CloudFront que expõe a aplicação.

Usando o ZAP será realizado um *scan* à distribuição CloudFront para a identificação de potenciais vulnerabilidades *web* na aplicação.

Usando o Legion serão analisadas a distribuição CloudFront e a instância EC2 para efeitos de reconhecimento das configurações ao nível de rede e potenciais vulnerabilidades na infraestrutura.

7.3.2 Testes de penetração e mecanismos de deteção

Para a avaliação da resposta e prevenção de ataques informáticos serão conduzidos testes de penetração à prova de conceito tomando como ponto de partida as descobertas feitas na avaliação de vulnerabilidades.

O objetivo passará por tentar realizar o *exploit* das vulnerabilidades identificadas na prova de conceito e determinar se os componentes de alarmística e monitorização permitem proteger destes ataques e assistir durante as fases de operação e numa futura fase de análise forense.

7.3.3 Inquérito de satisfação

O inquérito de satisfação desenvolvido tem como objetivo avaliar a capacidade de resposta dos controlos AWS utilizados na implementação da prova de conceito aos controlos de segurança levantados na secção do estado da arte.

7.4 Avaliação dos resultados

Nesta secção do documento os indicadores serão avaliados de acordo com as hipóteses definidas e utilizando os métodos descritos anteriormente.

7.4.1 Avaliação de vulnerabilidades

Conforme mencionado na secção 7.3.1, serão utilizadas três ferramentas distintas para esta fase da avaliação da solução.

7.4.1.1 Nessus - Metasploitable

Para fornecer ao leitor algum termo de comparação quanto aos resultados obtidos nas análises do Nessus, foi realizado um *scan* de rede básico a uma aplicação propositadamente vulnerável alojada localmente, distribuída sob o nome Metasploitable.

Da análise a esta aplicação resultou um total de 69 vulnerabilidades identificadas.

Deste total, as vulnerabilidades seguiram a distribuição apresentada na Tabela 17.

Tabela 17 - Distribuição de vulnerabilidades Metasploitable

Criticidade	Percentagem
Crítica	6%
Elevada	4%
Média	19%
Baixa	4%
Informativa	66%

Das vulnerabilidades críticas identificadas destacam-se por exemplo:

- Capacidade de um atacante iniciar uma *shell* remota.
- Uma *shell* em execução num porto sem necessitar de autenticação.
- Uma partição NFS encontra-se aberta sem qualquer autenticação, permitindo acessos de leitura (e possivelmente de escrita) a um atacante.
- Deteção de uma versão do SO desatualizada sem suporte atual.
- Uso de palavras-passe facilmente *brute-forceable* em alguns dos serviços disponibilizados pela máquina.

7.4.1.2 Nessus – Distribuição CloudFront

Para a análise de vulnerabilidades do aplicativo servido via CloudFront foi conduzido um *scan* de *web application*.

Desta análise resultou um total de 6 vulnerabilidades identificadas.

Deste total, as vulnerabilidades seguiram a distribuição apresentada na Tabela 18.

Tabela 18 - Distribuição de vulnerabilidades CloudFront

Criticidade	Percentagem
Informativa	100%

Tendo em conta a criticidade das vulnerabilidades identificadas, não foram exploradas tentativas de *exploit*.

7.4.1.3 Nessus – Instância EC2

Para a análise de vulnerabilidades da instância EC2 foi conduzido um *scan* avançado com as configurações por defeito.

Desta análise resultou um total de 14 vulnerabilidades identificadas.

Deste total, as vulnerabilidades seguiram a distribuição apresentada na Tabela 19.

Tabela 19 - Distribuição de vulnerabilidades EC2

Criticidade	**Percentagem**
Baixa	7%
Informativa	93%

Tendo em conta a criticidade das vulnerabilidades identificadas, não foram exploradas tentativas de *exploit*.

7.4.1.4 ZAP – Distribuição CloudFront

Para a identificação de vulnerabilidades do aplicativo servido via CloudFront foi ainda conduzido um *scan* usando o ZAP.

Desta análise resultou um total de 5 alertas diferentes identificados.

Deste total, os alertas seguiram a distribuição apresentada na Tabela 20.

Tabela 20 - Distribuição de alertas ZAP para CloudFront

Criticidade	**Percentagem**
Média	20%
Baixa	60%
Informativa	20%

Dos alertas apresentados pela análise do ZAP, o alerta de criticidade média foi lançado relativamente à inexistência do *header* **X-Frame-Options**, que assiste na proteção contra ataques de *clickjacking*.

7.4.1.5 Legion – Distribuição CloudFront

Uma vez que o CloudFront se trata de um serviço mantido e gerido pela AWS, não se esperam identificar quaisquer vulnerabilidades neste componente da infraestrutura. Ainda assim, foi realizado um *scan* que apenas identificou os portos 80 e 443 (HTTP e HTTPS) abertos, e informa com 98% de certeza que o SO em uso trata-se de uma instância Oracle Virtualbox.

7.4.1.6 Legion – Instância EC2

A ferramenta Legion foi utilizada para a identificação de vulnerabilidades ao nível da rede e da configuração da instância EC2 criada. Embora esta instância tenha sido descoberta com sucesso pelo Legion, dadas as camadas dos controlos de filtragem de tráfego em

funcionamento, o Legion não conseguiu identificar nenhum porto aberto na instância, o seu SO, nem qualquer outra informação potencialmente valiosa para um atacante.

7.4.2 Testes de penetração e mecanismos de deteção

Dadas as poucas descobertas de potenciais vetores de ataque na fase de avaliação e identificação de vulnerabilidades, não foram exploradas nenhuma das vulnerabilidades listadas para além da possibilidade de *clickjacking* detetada pelo ZAP, porém, não foi possível abusar desta vulnerabilidade com sucesso.

Porém, durante uma fase de testes *blackbox* à prova de conceito, foi identificado que um ator não autenticado consegue realizar pedidos à API Gateway via a origem configurada no CloudFront (acedida via **[idCloudfront].cloudfront.net/api**), podendo visualizar e manipular dados do aplicativo (através de técnicas de *brute force*). Esta vulnerabilidade seria facilmente colmatada através da implementação de um componente *identity provider*, como por exemplo o Amazon Cognito, que gerasse um *token* que permita autenticar um utilizador sempre que este realiza um pedido à API Gateway.

Outro potencial vetor de ataque do sistema passava pelo abuso do porto 22 aberto ao público na instância EC2, porém dado o uso conjunto do mecanismo de identificação via o endereço IP *whitelisted* e o mecanismo de autenticação via segredo (chave privada), não foi identificado um método para o abuso deste vetor.

7.4.2.1 Mecanismos de deteção e proteção

Através da configuração de alarmes da API Gateway para códigos de resposta HTTP das famílias 400 e 500, como demonstrado na secção 6.9.10, na implementação do CloudWatch, podemos ter uma melhor visibilidade sobre uma tentativa de *brute force* ou de enumeração das rotas da API Gateway por um ator malicioso.

Para além da alarmística configurada manualmente, temos ainda os serviços GuardDuty e Security Hub que fornecem sugestões para o fortalecimento da postura de segurança ao nível da infraestrutura e da conta AWS.

Abaixo são apresentadas capturas da interface do serviço GuardDuty que alerta para o uso de credenciais *root*, registando o *timestamp*, a chave utilizada (Figura 13), o endereço IP de origem, o local, o serviço manipulado e o tipo de operação realizada (Figura 14).

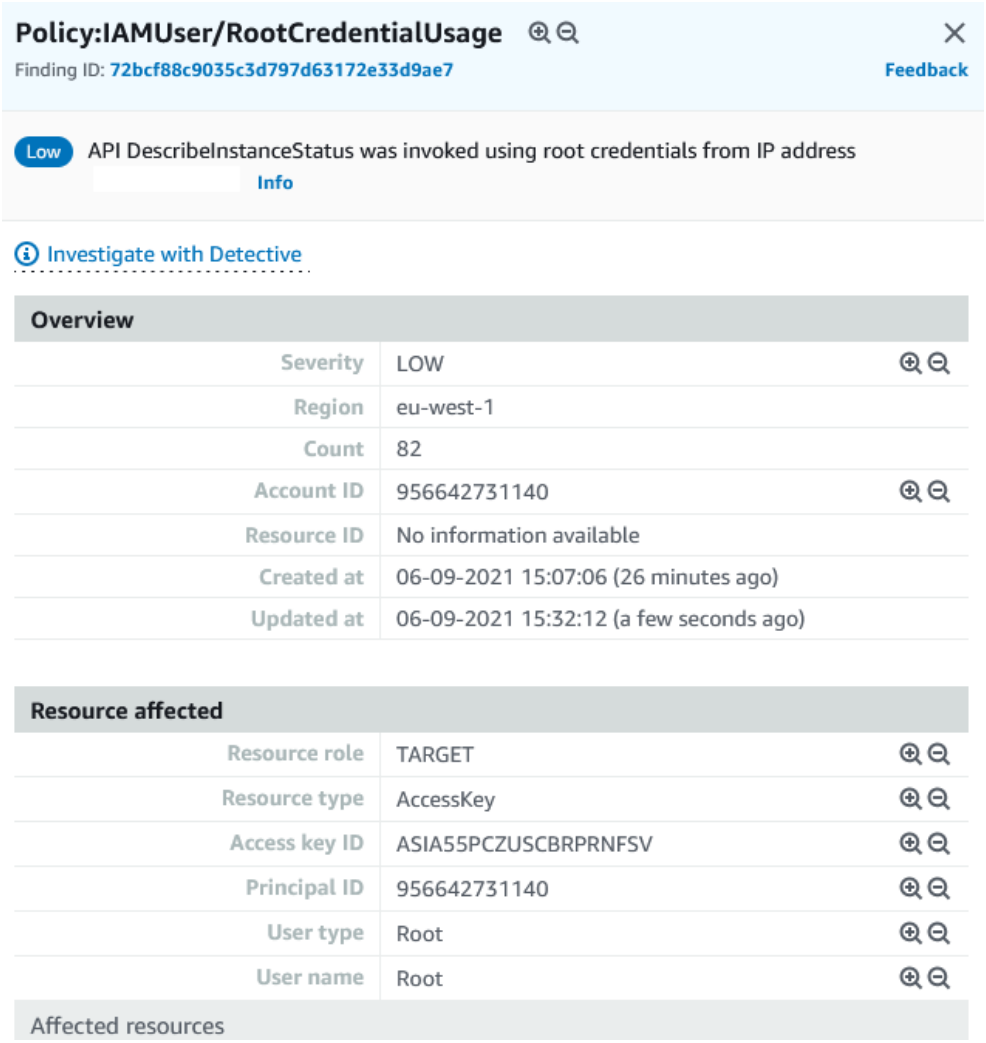


Figura 13 – Captura 1 do serviço GuardDuty

Figura 14 – Captura 2 do serviço GuardDuty

Consultando o Security Hub, são agregadas as descobertas do GuardDuty bem como recomendações ao nível da infraestrutura. Graças a este serviço foi identificada uma vulnerabilidade crítica, em que os *logs* gerados pelo CloudTrail estavam acessíveis publicamente devido à pobre configuração utilizada no *bucket* S3. Esta vulnerabilidade foi colmatada adicionando a linha **acl = private** no ficheiro de configuração Terraform do *bucket*, tornando-o acessível apenas ao criador do recurso.

No Anexo F do documento está apresentada a captura do Security Hub que permitiu a identificação desta vulnerabilidade e de outros problemas identificados.

7.4.3 Inquérito de satisfação

No anexo G do documento encontra-se o questionário de satisfação desenvolvido na plataforma Google Docs. O questionário foi distribuído por profissionais contextualizados com

o projeto desenvolvido e também por estudantes universitários que frequentam cursos na área das Tecnologias da Informação.

O questionário é composto por um total de 42 perguntas, estando dividido em dois principais temas:

- As primeiras duas secções pretendem traçar um perfil do conhecimento do inquirido quanto às áreas de computação em nuvem e de segurança.

- As restantes secções abordam os controlos de segurança levantados no estado da arte que são garantidos pela AWS e avaliam o grau de satisfação com a solução desenvolvida.

De forma a tornar o questionário mais acessível a inquiridos sem um extenso contexto sobre o projeto, cada um dos serviços e recursos AWS abordados no questionário são acompanhados por uma breve introdução e uma ligação externa a uma página que permita informar o inquirido.

Apenas as questões das secções de avaliação do perfil do inquirido encontram-se marcadas como obrigatórias, sendo que nas secções de avaliação dos controlos de segurança algumas questões poderão ser inadequadas de acordo com o perfil do inquirido.

Nas seguintes secções de análise das respostas, quando são apresentadas as perguntas do inquérito, estas aparecerão sublinhadas de forma a facilitar a legibilidade.

O inquérito contou com um total de 18 respostas, estando os resultados do mesmo clarificados abaixo.

7.4.3.1 Perfil dos inquiridos – Computação em nuvem

Todos os inquiridos indicaram que conhecem o termo ‘computação em nuvem’ e todos exceto 1 inquirido responderam que entraram em contacto com o tema durante o seu percurso académico ou profissional.

Entre a população de inquiridos, a plataforma AWS trata-se da mais popular, tendo sido usada por 12 dos inquiridos, seguida do Azure usado por 10 e finalmente a Google Cloud usada por 4 inquiridos.

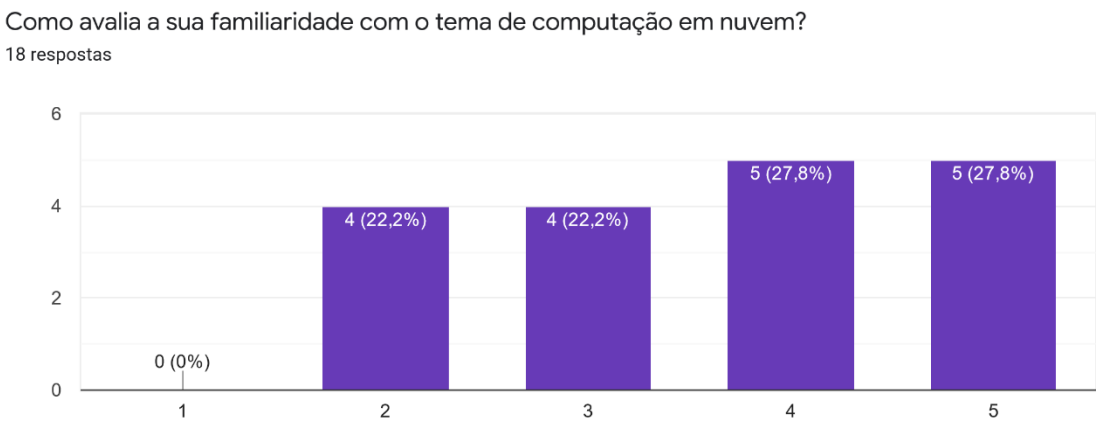


Figura 15 - Familiaridade dos inquiridos com o tema de computação em nuvem

No que toca à familiaridade com a computação em nuvem, existiu uma distribuição bastante equilibrada no nível de conforto dos inquiridos com o tema, sendo que 5 dos inquiridos utiliza regularmente este tipo de serviços (nível 5) e não existe nenhum inquirido que sem conhecimento deste tipo de serviços (nível 1).

Encerrando a primeira secção do inquérito, apenas 5 dos 18 inquiridos indicam nunca ter implantado uma solução informática em nuvem.

7.4.3.2 Perfil dos inquiridos – Segurança informática

No que toca à computação em nuvem, todos os inquiridos indicaram que consideram a segurança um aspeto importante neste tipo de ambiente.

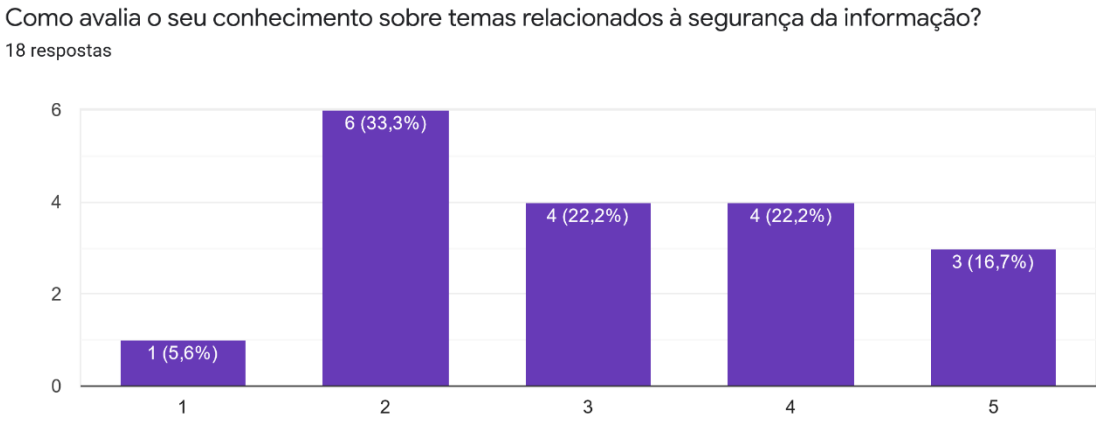


Figura 16 - Familiaridade dos inquiridos com temas de segurança

No que toca a temas relacionados à segurança a maioria dos inquiridos consideram-se medianos (níveis 2 a 4), e 3 dos inquiridos demonstram-se muito confortáveis com o tema (nível 5).

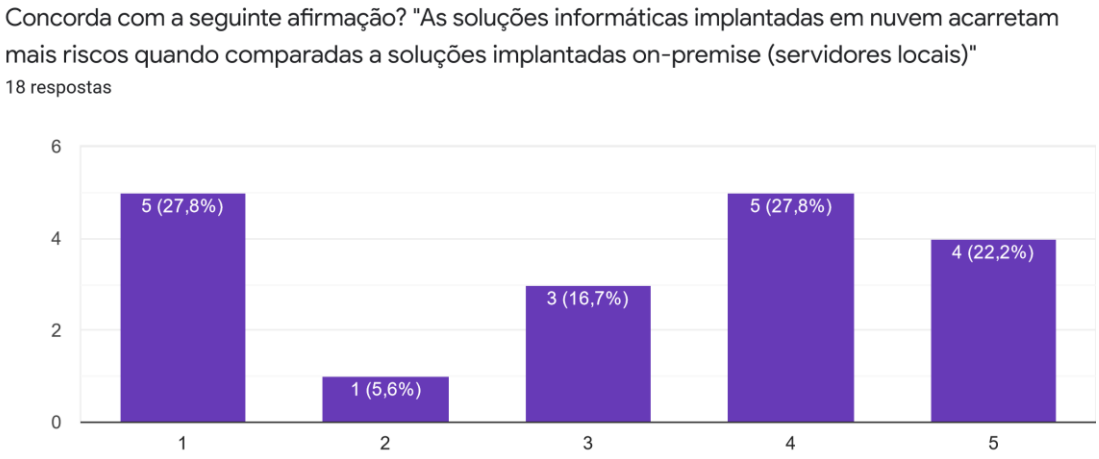


Figura 17 - Avaliação da postura dos inquiridos quanto a soluções informáticas implantadas em nuvem

Para avaliar a opinião dos inquiridos quanto ao tema de segurança enquadrado na implantação de soluções informáticas na nuvem, foi apresentada a questão da Figura 17, apresentada acima. Quanto a esta afirmação, 4 dos inquiridos afirmaram concordar sem qualquer dúvida (nível 5) e outros 5 também concordam, embora com menor grau de certeza (nível 4), que soluções implantadas em nuvem acarretam mais riscos quando comparadas a uma implantação tradicional. Porém existem ainda 5 inquiridos que discordam completamente com a afirmação.

Quanto a "segurança em profundidade", apenas 9 dos 18 inquiridos afirmam conhecer este termo.

Finalmente, para encerrar a avaliação do perfil dos inquiridos, verificou-se que apenas 7 dos 18 inquiridos implantaram uma solução informática em nuvem seguindo uma abordagem de segurança em profundidade.

Da informação derivada destas secções de perfil, podemos assumir que a maioria dos inquiridos têm alguns conhecimentos sobre as áreas de segurança e computação em nuvem, e existem ainda alguns especialistas entre os inquiridos (5 inquiridos 'nível 5' em computação em nuvem e 3 inquiridos 'nível 5' em segurança).

7.4.3.3 Avaliação de controlos de segurança

Para as seguintes secções do inquérito, foi utilizada a seguinte abordagem para avaliar o trabalho desenvolvido:

- Primeiro, é apresentada uma afirmação que o inquirido deve concordar ou discordar usando uma escala Likert com níveis de "1 – Discordo completamente" a "5 – Sem qualquer dúvida". Esta pergunta inicial pretende avaliar a adequação dos controlos de segurança levantados ao longo do trabalho da dissertação.

- Segundo, é feita uma introdução ao serviço ou conjunto de serviços AWS que permitam ir ao encontro do controlo de segurança apresentado anteriormente. O inquirido é então questionado se os serviços AWS selecionados vão (ou não) ao encontro do controlo de segurança apresentado primeiramente.

7.4.3.4 Avaliação de controlos de segurança – Testes à segurança

Para a avaliação deste controlo foi pedida a opinião dos inquiridos sobre a seguinte frase: "Com testes de penetração regulares a um sistema informático, são identificadas vulnerabilidades na sua configuração mais facilmente". A população de inquiridos pareceu concordar uniformemente, sendo que a maioria dos inquiridos responderam com nível 4 (44,4%) ou 5 (50%), existindo apenas um inquirido com nível 3 (5,6%).

Foi ainda pedida a opinião sobre um segundo controlo complementar: "Graças ao uso de ferramentas de análise de código, podemos evitar a introdução de vulnerabilidades aplicacionais na fase de implementação", onde a maioria dos inquiridos se inseriu nos níveis 4 (22,2%) e 5 (61,1%). Porém, três inquiridos responderam com nível 2 (16,7%).

Esta secção foi então encerrada com a seguinte questão: "Considera que recorrendo a testes de penetração regulares e a ferramentas de análise de código temos melhores garantias

quanto à segurança operacional de um sistema informático?", que obteve uma resposta afirmativa de 100% dos inquiridos.

7.4.3.5 Avaliação de controlos de segurança – Gestão de eventos de segurança

Para a avaliação deste controlo foi pedida a opinião dos inquiridos sobre a seguinte frase: "Após um incidente de segurança, é importante obter o máximo de informação e contexto possível acerca da origem do evento". A população de inquiridos pareceu concordar uniformemente, sendo que todos os inquiridos responderam com nível 4 (16,7%) ou 5 (83,3%).

Como resposta a este controlo de segurança, os inquiridos foram contextualizados acerca dos serviços GuardDuty e Security Hub, sendo depois apresentada as seguintes questões:

- "Considera que um serviço como o GuardDuty ajuda a prevenir contra ações indevidas e a identificar atores maliciosos no contexto da AWS?", que obteve apenas uma resposta negativa (5,9%).

- "Considera que um serviço como o Security Hub promove a visibilidade de eventos pertinentes à segurança de um sistema informático implantado em AWS?", que obteve 2 respostas negativas (11,8%).

- "Considera que aplicação de controlos como o GuardDuty e o Security Hub assistem na resposta e preparação para eventuais incidentes de segurança?", que obteve apenas uma resposta negativa (5,9%)

Cada uma das três questões apresentadas acima contaram cada uma com apenas 17 respostas, o que leva a deduzir que um dos inquiridos não se encontrava confortável o suficiente com estes serviços para responder.

7.4.3.6 Avaliação de controlos de segurança – Controlo de acessos

Para a avaliação de um dos controlos desta categoria foi pedida a opinião dos inquiridos sobre a seguinte frase: "O controlo rigoroso das permissões dos utilizadores num sistema informático ajuda a evitar incidentes de acessos indevidos e de erro humano". A população de inquiridos pareceu concordar uniformemente, sendo que todos os inquiridos responderam com nível 4 (33,3%) ou 5 (66,7%).

Para responder ao controlo apresentado acima, os inquiridos foram contextualizados acerca da ferramenta IAM da AWS e foi apresentada a seguinte questão: "Considera que um serviço como o IAM promove a manutenção de um sistema informático onde os utilizadores apenas têm acesso aos recursos que necessitam para realizar o seu trabalho (princípio least

privilege)?", à qual 16 inquiridos responderam que "Sim", apenas um respondeu que "Não" e um inquirido não respondeu.

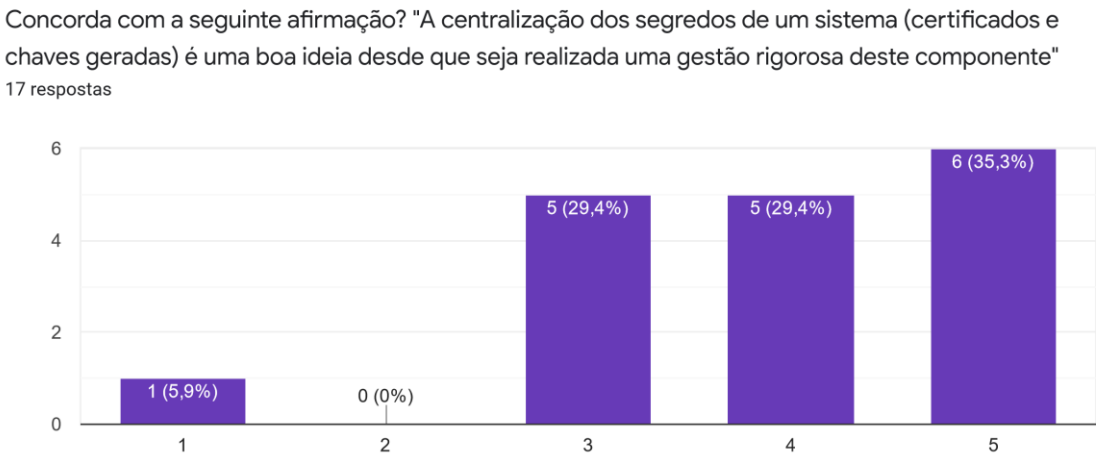


Figura 18 – Distribuição de respostas acerca da gestão centralizada de segredos

Foi ainda apresentada outra afirmação para avaliar um segundo controlo dentro da categoria de controlo de acessos: "A centralização dos segredos de um sistema (certificados e chaves geradas) é uma boa ideia desde que seja realizada uma gestão rigorosa deste componente", a qual recebeu respostas positivas, mas com um claro *outlier*, como pode ser comprovado na Figura 18 apresentada acima.

Em retrospetiva, este *outlier* estará provavelmente associado à questão formulada. A centralização de segredos, por um lado, facilita a sua gestão e rotação, mas por outro lado, pode constituir um foco para atacantes, podendo esta decisão estar dependente do contexto da solução a desenvolver.

De seguida, foi contextualizado o serviço KMS da AWS e foi colocada a seguinte questão: "Considera que um serviço como o KMS promove a manutenção de um ecossistema de gestão segura de segredos utilizados na encriptação e dos certificados?", a qual obteve "Sim" de 16 inquiridos, apenas um "Não" e um inquirido que não respondeu.

7.4.3.7 Avaliação de controlos de segurança – Segurança operacional

Para a avaliação deste controlo foi pedida a opinião dos inquiridos sobre a seguinte frase: "O logging de eventos do sistema aliado à monitorização contínua dos componentes do nosso sistema informático ajudam-nos a identificar atividade maliciosa e comportamentos inesperados que podem indicar a existência de uma vulnerabilidade". A população de inquiridos pareceu concordar uniformemente, sendo que todos os inquiridos responderam com nível 4 (33,3%) ou 5 (66,7%).

De seguida, foram introduzidos ao inquirido os serviços CloudWatch e CloudTrail da AWS, acompanhados da seguinte questão: "Considera que o uso conjunto dos serviços CloudWatch e CloudTrail fornecem uma melhor visibilidade sobre o funcionamento da infraestrutura, ajudam a identificar potenciais vulnerabilidades e constituem ferramentas importantes num momento de análise forense?", a qual obteve 16 respostas "Sim", um "Não" e um inquirido que não respondeu.

Foi ainda avaliado o controlo de gestão de vulnerabilidades através da seguinte questão: "É importante realizar uma gestão ativa das vulnerabilidades identificadas num sistema informático", onde a população de inquiridos pareceu concordar uniformemente, sendo que todos os inquiridos responderam com nível 4 (33,3%) ou 5 (66,7%).

Para ir ao encontro deste controlo na AWS, foi revisitado o Security Hub, acompanhado da seguinte questão: "Considera que o uso do Security Hub para a identificação de potenciais vetores de ataque permite uma melhoria incremental da postura de segurança de um sistema informático?", a qual obteve 16 respostas "Sim", um "Não" e um inquirido que não respondeu.

7.4.3.8 Avaliação de controlos de segurança – Gestão da informação

Para a avaliação do controlo de visibilidade do inventário informático foi apresentada a seguinte afirmação: "A visibilidade sobre o inventário informático e a sua manutenção é um componente importante da postura de segurança de uma organização". A população de inquiridos pareceu concordar uniformemente, sendo que a maioria dos inquiridos responderam com nível 4 (38,9%) ou 5 (55,6%), existindo apenas um inquirido no nível 3 (5,6%).

Foi de seguida introduzido o serviço Systems Manager, acompanhado da seguinte questão: "Considera que o uso do Systems Manager promove a visibilidade sobre o inventário informático e a sua manutenção?", a qual obteve 15 respostas "Sim", 2 "Não" e um inquirido que não respondeu. Ainda assim, as respostas positivas superaram o grau de satisfação de 80% com 88,2%.

7.4.3.9 Avaliação de controlos de segurança – Conformidade

Para a avaliação deste controlo foi apresentada a seguinte afrimação: "A definição de uma política de programação segura e a sua aplicação através da definição de mecanismos automáticos (ex: quality gates) ajuda a diminuir a chance de surgirem vulnerabilidades num sistema informático". A população de inquiridos pareceu concordar uniformemente, sendo que a maioria dos inquiridos responderam com nível 4 (33,3%) ou 5 (61,1%), havendo um inquirido que respondeu com nível 3 de certeza.

De seguida foi introduzido o serviço Amazon Inspector, acompanhado da seguinte questão: "Considera que o Inspector ajuda na identificação de riscos e vulnerabilidades que possam aparecer nos recursos computacionais implantados?", a qual obteve 100% de respostas positivas, com 3 inquiridos que não responderam.

Ainda nos controlos de conformidade foram colocadas as seguintes questões:

- "Qualquer organização que opte por implantar uma solução informática na nuvem deve ter conhecimento do modelo de responsabilidade partilhada da plataforma de hospedagem", que obteve um parecer positivo dos inquiridos, com 7 inquiridos inseridos no nível 4 (38,9%), 9 no nível 5 (50%) e 2 no nível 3 (11,1%).

- "Uma organização deve de ter conhecimento das legislações a que está sujeita e da sua aplicabilidade num ambiente de hospedagem em nuvem", que obteve um parecer positivo dos inquiridos, com 5 inquiridos inseridos no nível 4 (27,8%), 11 no nível 5 (61,1%) e 2 no nível 3 (11,1%).

Foi então introduzido o AWS Artifact como resposta a estas últimas duas questões, acompanhado da seguinte questão: "Considera que o Artifact permite a uma organização enquadrar as suas responsabilidades e requisitos legais num ambiente nuvem?", tendo obtido 13 respostas "Sim" (81,3%), 3 respostas "Não" (18,8%) e duas abstenções.

7.4.3.10 Avaliação de controlos de segurança – Camada tecnológica

Para a avaliação deste controlo foi apresentada a seguinte afirmação: "A gestão ativa da superfície de ataque constitui um controlo de segurança importante no desenvolvimento e operação de um sistema informático". A população de inquiridos pareceu concordar uniformemente, sendo que a maioria dos inquiridos responderam com nível 4 (47,1%) ou 5 (52,9%), existindo uma abstenção.

Como resposta a este controlo foi apresentada a seguinte questão: "Considera que o uso de imagens *hardened* numa solução informática que faça uso de máquinas virtuais ajudam a reduzir a superfície de ataque da nossa infraestrutura?", a qual obteve 15 respostas "Sim" (88,2%), 2 respostas "Não" (11,8%) e uma abstenção.

7.4.3.11 Avaliação de controlos de segurança – Camada de arquitetura

Para a avaliação do controlo da aplicação de encriptação foi apresentada a seguinte afirmação: "A encriptação de informação sensível é um dos pilares da segurança informática", a qual obteve um parecer positivo, com 6 inquiridos inseridos no nível 4 (33,3%) e 12 inseridos no nível 5 (66,7%) de concordância.

Após esta afirmação, e estando os inquiridos nesta fase já familiarizados com o KMS, foi colocada a seguinte questão: "Considera que o uso de um serviço como o KMS que facilita a integração de diversos outros serviços da AWS promove a aplicação de encriptação nos diversos componentes (ex: bases de dados, *logs*, armazenamento de objetos) de um sistema informático?", tendo obtido 15 respostas positivas, apenas 1 resposta negativa e com duas abstenções.

Para a avaliação do controlo de filtragem de tráfego foi apresentada a seguinte afirmação: "O uso de *firewalls* e a configuração de regras de filtragem permitem reduzir a superfície de ataque de um sistema", a qual obteve um parecer positivo, com 7 inquiridos inseridos no nível 4 (38,9%) e 11 inseridos no nível 5 (61,1%).

Após a afirmação, foram contextualizados os mecanismos de NSG, acompanhado da seguinte questão: "Considera que um NSG, quando devidamente configurado, reduz consideravelmente a superfície de ataque de um sistema informático?", com 16 respostas positivas, apenas 1 resposta negativa e uma abstenção.

Para a avaliação do controlo de segurança do perímetro foi apresentada a seguinte afirmação: "A segurança do perímetro de uma rede constitui uma camada importante para a deteção e proteção contra ataques à nossa infraestrutura", a qual obteve um parecer positivo, com 6 inquiridos inseridos no nível 4 (33,3%), 11 inseridos no nível 5 (61,1%), mas também um inquirido inserido no nível 3 (5,6%).

Com este controlo apresentado, foram introduzidos os serviços WAF, AWS Shield e CloudFront, acompanhados pela seguinte questão: "Considera que o uso conjunto dos serviços WAF, AWS Shield e CloudFront fortalece a segurança do perímetro de um sistema informático contra diversos tipos de ataques e tentativas de intrusão?", a qual obteve 15 respostas positivas, 1 resposta negativa e duas abstenções.

Finalmente, para a avaliação de isolamento de ambientes foi apresentada a seguinte afirmação: "A segregação de uma rede num conjunto de sub-redes fornece um melhor isolamento entre os diversos componentes da infraestrutura, dificultando o movimento de atacantes no sistema e prevenindo a propagação de ataques pela rede", a qual obteve um parecer positivo, com 7 inquiridos inseridos no nível 4 (41,2%), 9 inseridos no nível 5 (52,9%), mas também um inquirido inserido no nível 3 (5,9%). Existiu ainda uma abstenção.

Depois deste controlo, foram introduzidos os mecanismos de VPC e Subnet da AWS, acompanhados pela seguinte questão: "Considera que o uso de VPCs e Subnets oferece um melhor isolamento entre os componentes implantados em nuvem, contribuindo assim para uma melhor postura de segurança da infraestrutura?", que obteve 15 pareceres positivos, apenas 1 negativo e duas abstenções.

Deu-se assim como encerrado o questionário de satisfação com trabalho desenvolvido, podendo ser este consultado no Anexo G do documento.

7.4.3.12 Conclusão inquérito

Pode-se concluir pelos resultados obtidos do inquérito de satisfação elaborado que o trabalho desenvolvido ao longo do projeto foi satisfatório, uma vez que todos os controlos e serviços AWS abordados no inquérito obtiveram um nível de satisfação acima dos 80%.

8 Conclusão

Nesta secção do documento são apresentadas as conclusões retiradas do trabalho realizado ao longo do projeto e da materialização da prova de conceito. Esta secção encontra-se dividida em três subsecções, nomeadamente, os objetivos alcançados, as limitações do trabalho realizado e trabalho futuro.

8.1 Objetivos alcançados

A finalidade do trabalho desenvolvido ao longo do projeto de dissertação passa pela aplicação de controlos de segurança levantados do estado da arte numa arquitetura de referência que é materializada numa prova de conceito.

De acordo com os objetivos definidos na secção 1.4 do documento, podem ser retiradas conclusões quanto ao trabalho desenvolvido na duração do projeto:

- Levantamento do estado da arte e consolidação de boas práticas: Ambos objetivos foram cumpridos e desenvolvidos na secção do Estado da Arte.

- Elaboração de uma arquitetura de referência: Este objetivo foi abordado na secção de Design da Arquitetura de referência, e foi completado com sucesso graças ao trabalho desenvolvido no levantamento do estado da arte.

- Seleção do CSP: Abordado na secção da Análise de Valor, a plataforma de hospedagem AWS foi selecionada de entre as três alternativas consideradas.

- Desenvolvimento de prova de conceito: Abordado na secção Prova de Conceito do documento, aonde os requisitos traçados para o aplicativo e para a infraestrutura foram atingidos.

- Avaliação da prova de conceito: Abordada na secção Experimentação e Avaliação, os três métodos utilizados na avaliação da prova de conceito e do trabalho realizado apresentaram evidências satisfatórias que permitem indicar que a abordagem adotada foi uma resposta adequada ao problema proposto.

Concluindo, de acordo com os objetivos estabelecidos para o projeto da dissertação, a aplicação de controlos de segurança permitiu a implantação com sucesso de um aplicativo em nuvem com diversas camadas de defesa.

8.2 Limitações

Durante o desenvolvimento do trabalho surgiram naturalmente limitações quanto ao potencial valor ideal da dissertação.

Uma das principais limitações surgiu na fase de implementação da prova de conceito, uma vez que a impossibilidade de utilizar os serviços pagos disponibilizados pela AWS impediu a aplicação de todos os controlos de segurança previstos na arquitetura de referência.

Outra potencial limitação a apontar ao trabalho desenvolvido passa pela relativa falta de experiência do autor na condução formal de testes de penetração, os quais, quando conduzidos por um profissional, produzem descobertas mais valiosas quanto a potenciais lacunas na segurança da prova de conceito desenvolvida.

Finalmente, resta mencionar que o número relativamente baixo de inquiridos no inquérito de satisfação para a avaliação do trabalho pode ter levado a uma conclusão que possivelmente não reflita a experiência de uma população mais familiar com o tema de segurança na nuvem. Porém, após a entrega deste documento o inquérito continuará em aberto de forma a tentar recolher mais respostas, visando uma melhoria contínua do trabalho feito.

8.3 Trabalho futuro

No que toca a potencial trabalho futuro, poderão ser desenvolvidas provas de conceito semelhantes para as plataformas Azure e Google Cloud que permitirão uma melhor avaliação dos controlos de segurança garantidos por cada um destes serviços de hospedagem em nuvem em comparação à AWS.

Naturalmente, com a constante evolução dos temas de segurança e de computação em nuvem, poderão surgir novos serviços e controlos de segurança que permitam expandir sobre o trabalho realizado neste projeto.

Adicionalmente, existem novos tipos de soluções de computação em nuvem, como por exemplo serviços *serverless*, que continuam a gerar interesse dos utilizadores deste tipo de serviços. Porém, este novo tipo de computação que retira ao utilizador o controlo sobre a infraestrutura e configurabilidade dos seus aplicativos poderão trazer novos paradigmas da segurança em nuvem dignos de investigação.

8.4 Apreciação final

A exploração dos temas de segurança e de computação em nuvem abordados ao longo deste projeto provaram ser uma ótima oportunidade de crescimento a nível académico e profissional e os conhecimentos adquiridos irão sem dúvida ajudar-me ao longo do meu

percurso como engenheiro. A minha curiosidade por ambos os temas constituiu ainda uma importante fonte de motivação ao longo de todo o projeto e durante a escrita deste documento.

Os temas de segurança e de computação em nuvem estão em constante evolução, e o trabalho desenvolvido permitiu explorar o estado da arte de ambas as áreas, o que constituiu um valioso período de aprendizagem sobre estes temas cada vez mais relevantes.

Refletindo sobre os resultados atingidos, o trabalho desenvolvido foi ao encontro dos objetivos propostos e também ajudou na resposta ao problema de soluções informáticas em nuvem frequentemente inseguras. Logo, considero que o trabalho foi valioso não só como resposta a um problema bastante comum, mas também como ferramenta de aprendizagem.

Bibliografia

Amazon Web Services (2020a) *Building a Scalable and Secure Multi-VPC AWS Network Infrastructure*.

Amazon Web Services (2020b) *Serverless Computing*. Disponível em: https://aws.amazon.com/serverless/ (Acedido: 1 de Junho de 2021).

Amazon Web Services (2021a) *Access control list (ACL) overview - Amazon Simple Storage Service*. Disponível em: https://docs.aws.amazon.com/AmazonS3/latest/userguide/acl-overview.html#canned-acl (Acedido: 6 de Junho de 2021).

Amazon Web Services (2021b) *Amazon API Gateway*. Disponível em: https://aws.amazon.com/api-gateway/ (Acedido: 1 de Junho de 2021).

Amazon Web Services (2021c) *Amazon CloudFront*. Disponível em: https://aws.amazon.com/cloudfront/ (Acedido: 1 de Junho de 2021).

Amazon Web Services (2021d) *Amazon CloudWatch Product Features*. Disponível em: https://aws.amazon.com/cloudwatch/features/ (Acedido: 1 de Junho de 2021).

Amazon Web Services (2021e) *Amazon Detective features*. Disponível em: https://aws.amazon.com/detective/features/ (Acedido: 2 de Junho de 2021).

Amazon Web Services (2021f) *Amazon Detective prerequisites and recommendations*. Disponível em: https://docs.aws.amazon.com/detective/latest/adminguide/detective-prerequisites.html (Acedido: 10 de Junho de 2021).

Amazon Web Services (2021g) *Amazon EC2 Features*. Disponível em: https://aws.amazon.com/ec2/features/ (Acedido: 1 de Junho de 2021).

Amazon Web Services (2021h) *Amazon GuardDuty Features*. Disponível em: https://aws.amazon.com/guardduty/features/ (Acedido: 2 de Junho de 2021).

Amazon Web Services (2021i) *Amazon Inspector*. Disponível em: https://aws.amazon.com/inspector/ (Acedido: 2 de Junho de 2021).

Amazon Web Services (2021j) *Amazon Inspector ARNS for rules packages*. Disponível em: https://docs.aws.amazon.com/inspector/latest/userguide/inspector_rules-arns.html#eu-west-1 (Acedido: 13 de Junho de 2021).

Amazon Web Services (2021k) *Amazon RDS Features*. Disponível em: https://aws.amazon.com/rds/features/ (Acedido: 1 de Junho de 2021).

Amazon Web Services (2021l) *Amazon Virtual Private Cloud (VPC)*. Disponível em: https://aws.amazon.com/vpc/?vpc-blogs.sort-by=item.additionalFields.createdDate&vpc-blogs.sort-order=desc (Acedido: 3 de Junho de 2021).

Amazon Web Services (2021m) *AWS Artifact*. Disponível em: https://aws.amazon.com/artifact/ (Acedido: 2 de Junho de 2021).

Amazon Web Services (2021n) *AWS CloudTrail Features*. Disponível em:

https://aws.amazon.com/cloudtrail/features/ (Acedido: 1 de Junho de 2021).

Amazon Web Services (2021o) *AWS Identity & Access Management (IAM) features*. Disponível em: https://aws.amazon.com/iam/features/?nc=sn&loc=2 (Acedido: 2 de Junho de 2021).

Amazon Web Services (2021p) *AWS Key Management Service (KMS) features*. Disponível em: https://aws.amazon.com/kms/features/ (Acedido: 2 de Junho de 2021).

Amazon Web Services (2021q) *AWS Security Hub features*. Disponível em: https://aws.amazon.com/security-hub/features/ (Acedido: 2 de Junho de 2021).

Amazon Web Services (2021r) *AWS Systems Manager features*. Disponível em: https://aws.amazon.com/systems-manager/features/ (Acedido: 2 de Junho de 2021).

Amazon Web Services (2021s) *O que é cloud computing (computação em nuvem)? - Amazon Web Services*. Disponível em: https://aws.amazon.com/pt/what-is-cloud-computing/ (Acedido: 4 de Fevereiro de 2021).

Amazon Web Services (2021t) *Tutorial: Create an Amazon VPC for use with a DB instance - Amazon Relational Database Service*. Disponível em: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateVPC.html#CHAP_Tutorials.WebServerDB.CreateVPC.SecurityGroupDB (Acedido: 3 de Junho de 2021).

Amazon Web Services (2021u) *VPC with public and private subnets (NAT)*. Disponível em: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Scenario2.html (Acedido: 18 de Junho de 2021).

Artac, M. *et al.* (2017) *DevOps: Introducing Infrastructure-as-Code*, *2017 IEEE/ACM 39th IEEE International Conference on Software Engineering Companion*.

Bawe, B. (2021) *Top 9 Backend Frameworks all should use in 2021*. Disponível em: https://dev.to/brandonbawe/top-9-backend-frameworks-developers-and-big-tech-companies-are-using-in-2021-2hgh (Acedido: 29 de Maio de 2021).

BreachLock (2019) *Penetration Testing for ISO 27001 Control A.12.6.1 - BreachLock*. Disponível em: https://www.breachlock.com/penetration-testing-for-iso-27001-control-a-12-6-1/ (Acedido: 1 de Fevereiro de 2021).

Brikman, Y. (2019) *Why we use Terraform and not Chef, Puppet, Ansible, SaltStack, or CloudFormation*. Disponível em: https://blog.gruntwork.io/why-we-use-terraform-and-not-chef-puppet-ansible-saltstack-or-cloudformation-7989dad2865c (Acedido: 14 de Junho de 2021).

Center for Internet Security (2021) *CIS Hardened Images*. Disponível em: https://www.cisecurity.org/cis-hardened-images/ (Acedido: 7 de Março de 2021).

Cloud Security Alliance (2020) *Top Threats to Cloud Computing: Egregious Eleven*.

DB-Engines (2021) *DB-Engines Ranking - popularity ranking of database management systems*. Disponível em: https://db-engines.com/en/ranking (Acedido: 30 de Maio de 2021).

Deleersnyder, S. e De Win, B. (2020) *OWASP SAMM v2.0*.

Dhaduk, H. (2021) *Best Frontend Frameworks of 2021 for Web Development*. Disponível em: https://www.simform.com/best-frontend-frameworks/ (Acedido: 29 de Maio de 2021).

Diez, J. (2020) *A DMZ, what is that?* Disponível em: https://medium.com/google-cloud/a-dmz-what-is-that-acc3b21b9653 (Acedido: 26 de Fevereiro de 2021).

Disterer, G. (2013) «ISO/IEC 27000, 27001 and 27002 for Information Security Management», *Journal of Information Security*, 4, pp. 92–100. doi: 10.4236/jis.2013.42011.

Dutta, Pranay e Dutta, Prashant (2019) «Comparative Study of Cloud Services Offered by Amazon, Microsoft and Google», *International Journal of Trend in Scientific Research and Development*, Volume-3(Issue-3), pp. 981–985. doi: 10.31142/ijtsrd23170.

Figueiredo Ribeiro, R. de J. (2020) *Metodologia de avaliação e comparação de ferramentas de desenvolvimento de frontend para websites*.

Fortinet (2018) *A CISO GUIDE TO MULTI-CLOUD SECURITY*.

GoVanguard (2020) *Legion: About*. Disponível em: https://github.com/GoVanguard/legion (Acedido: 15 de Junho de 2021).

IBM Cloud Education (2020) *What is Multi-Tenant? | IBM*. Disponível em: https://www.ibm.com/cloud/learn/multi-tenant (Acedido: 4 de Fevereiro de 2021).

Imperva (2019) *What is Defense in Depth | Benefits of Layered Security | Imperva*. Disponível em: https://www.imperva.com/learn/application-security/defense-in-depth/ (Acedido: 26 de Fevereiro de 2021).

Infosavvy (2020a) *ISO 27001 Annex : A.12.6 Technical Vulnerability Management | Infosavvy Security and IT Management Training*. Disponível em: https://info-savvy.com/iso-27001-annex-a-12-6-technical-vulnerability-management/ (Acedido: 1 de Fevereiro de 2021).

Infosavvy (2020b) *ISO 27001 Annex : A.14.2.6 Secure Development Environment, A.14.2.7 Outsourced Development, A.14.2.8 System Security Testing & A.14.2.9 System Acceptance Testing | Infosavvy Security and IT Management Training*. Disponível em: https://info-savvy.com/iso-27001-annex-a-14-2-6-a-14-2-7-a-14-2-8-a-14-2-9/ (Acedido: 1 de Fevereiro de 2021).

International Organization for Standardization (2019) *ISO - ISO/IEC 27017:2015 - Information technology — Security techniques — Code of practice for information security controls based on ISO/IEC 27002 for cloud services*. Disponível em: https://www.iso.org/standard/43757.html (Acedido: 8 de Fevereiro de 2021).

Irwin, L. (2019) *Understanding the differences between ISO 27001 and ISO 27002 - IT Governance UK Blog*. Disponível em: https://www.itgovernance.co.uk/blog/understanding-the-differences-between-iso-27001-and-iso-27002 (Acedido: 8 de Fevereiro de 2021).

ISMS Online (2020a) *ISO 27001 Annex A.10 - Cryptography | ISMS.online*. Disponível em: https://www.isms.online/iso-27001/annex-a-10-cryptography/ (Acedido: 20 de Janeiro de 2021).

ISMS Online (2020b) *ISO 27001 Annex A.12 - Operations Security*. Disponível em: https://www.isms.online/iso-27001/annex-a-12-operations-security/ (Acedido: 22 de Janeiro

de 2021).

ISMS Online (2020c) *ISO 27001 Annex A.13 - Communications Security*. Disponível em: https://www.isms.online/iso-27001/annex-a-13-communications-security/ (Acedido: 1 de Fevereiro de 2021).

ISMS Online (2020d) *ISO 27001 Annex A.14 - System Acquisition, Development and Maintenance*. Disponível em: https://www.isms.online/iso-27001/annex-a-14-system-acquisition-development-and-maintenance/ (Acedido: 1 de Fevereiro de 2021).

ISMS Online (2020e) *ISO 27001 Annex A.16 - Information Security Incident Management*. Disponível em: https://www.isms.online/iso-27001/annex-a-16-information-security-incident-management/ (Acedido: 2 de Fevereiro de 2021).

ISMS Online (2020f) *ISO 27001 Annex A.17: Infosec Forms of Business Continuity Management*. Disponível em: https://www.isms.online/iso-27001/annex-a-17-information-security-aspects-of-business-continuity-management/ (Acedido: 2 de Fevereiro de 2021).

ISMS Online (2020g) *ISO 27001 Annex A.18 - Compliance*. Disponível em: https://www.isms.online/iso-27001/annex-a-18-compliance/ (Acedido: 2 de Fevereiro de 2021).

ISMS Online (2020h) *ISO 27001 Annex A.6 - Organisation of Information Security*, *ISMS Online*. Disponível em: https://www.isms.online/iso-27001/annex-a-6-organisation-information-security/ (Acedido: 12 de Janeiro de 2021).

ISMS Online (2020i) *ISO 27001 Annex A.8 - Asset Management*. Disponível em: https://www.isms.online/iso-27001/annex-a-8-asset-management/ (Acedido: 15 de Janeiro de 2021).

ISMS Online (2020j) *ISO 27001 Annex A.9 - Access Control*. Disponível em: https://www.isms.online/iso-27001/annex-a-9-access-control/ (Acedido: 19 de Janeiro de 2021).

ISMS Online (2021) *ISO 27001 Annex A Controls - Overview*. Disponível em: https://www.isms.online/iso-27001/annex-a-controls/ (Acedido: 9 de Fevereiro de 2021).

Kosutic, D. (2015) *ISO 27001 vs. ISO 27017 – Security controls for cloud services*. Disponível em: https://advisera.com/27001academy/blog/2015/11/30/iso-27001-vs-iso-27017-information-security-controls-for-cloud-services/ (Acedido: 9 de Fevereiro de 2021).

Koussa, S. (2018) *What do SAST, DAST, IAST and RASP mean to developers?* Disponível em: https://www.softwaresecured.com/what-do-sast-dast-iast-and-rasp-mean-to-developers/ (Acedido: 10 de Fevereiro de 2021).

Lanfear, T. (2021) *Shared responsibility in the cloud - Microsoft Azure | Microsoft Docs*. Disponível em: https://docs.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility (Acedido: 4 de Fevereiro de 2021).

Lanfear, T., Coulter, D. e Baldwin, M. (2019) *Secure development best practices on Microsoft Azure | Microsoft Docs*. Disponível em: https://docs.microsoft.com/en-us/azure/security/develop/secure-dev-overview (Acedido: 23 de Fevereiro de 2021).

McAfee (2020) *Cloud Adoption and Risk Report: Work from Home Edition*.

Mell, P. e Grance, T. (2011) *The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology*. doi: 10.6028/NIST.SP.800-145.

Microsoft Docs (2020) *Visão geral dos grupos de segurança da rede Azure*. Disponível em: https://docs.microsoft.com/pt-pt/azure/virtual-network/network-security-groups-overview (Acedido: 27 de Fevereiro de 2021).

Mitchell, B. (2020) *Demilitarized Zone in Computer Networking*. Disponível em: https://www.lifewire.com/demilitarized-zone-computer-networking-816407 (Acedido: 11 de Fevereiro de 2021).

Mogull, R. (2019) *AWS vs. Azure vs. GCP: A Security Pro's Quick Cloud Comparison*. Disponível em: https://disruptops.com/aws-vs-azure-vs-gcp-a-security-pros-quick-cloud-comparison/ (Acedido: 7 de Março de 2021).

Mohamed, M., Altrafi, O. e Ismail, M. (2014) *Relational vs. NoSQL Databases: A Survey*.

Monocubed (2021) *A Detailed Comparison of 10 Popular Web Development Framework*. Disponível em: https://www.monocubed.com/web-development-framework-comparison/ (Acedido: 30 de Maio de 2021).

Mufti, T., Mittal, P. e Gupta, B. (2021) «A Review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud Platform (GCP) Services». doi: 10.4108/eai.27-2-2020.2303255.

Myrbakken, H. e Colomo-Palacios, R. (2017) «DevSecOps: A multivocal literature review», em *Communications in Computer and Information Science*. Springer Verlag, pp. 17–29. doi: 10.1007/978-3-319-67383-7_2.

OWASP Foundation (2020) *The OWASP Testing Project - Introduction*. Disponível em: https://owasp.org/www-project-web-security-testing-guide/v42/2-Introduction/ (Acedido: 10 de Fevereiro de 2021).

React Docs (2021) *Component State*. Disponível em: https://reactjs.org/docs/faq-state.html (Acedido: 3 de Junho de 2021).

Red Hat (2018) *Types of cloud computing*. Disponível em: https://www.redhat.com/en/topics/cloud-computing/public-cloud-vs-private-cloud-and-hybrid-cloud (Acedido: 8 de Fevereiro de 2021).

Red Hat (2020) *What is a REST API?* Disponível em: https://www.redhat.com/en/topics/api/what-is-a-rest-api (Acedido: 14 de Junho de 2021).

Saaty, T. (1980) *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. McGraw-Hill.

Services, A. W. (2020) *VPC with public and private subnets (NAT) - Amazon Virtual Private Cloud*. Disponível em: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Scenario2.html (Acedido: 27 de Fevereiro de 2021).

Solanki, J. (2021) *Cloud Pricing Comparison 2021: AWS vs Azure vs Google Cloud*. Disponível

em: https://www.simform.com/compute-pricing-comparison-aws-azure-googlecloud/ (Acedido: 7 de Março de 2021).

SonarLint (2021) *SonarLint | Fix issues before they exist*. Disponível em: https://www.sonarlint.org/ (Acedido: 4 de Maio de 2021).

SonarQube (2021) *Code Quality and Code Security | SonarQube*. Disponível em: https://www.sonarqube.org/ (Acedido: 4 de Maio de 2021).

Souppaya, M., Morello, J. e Scarfone, K. (2017) «Application Container Security Guide». doi: 10.6028/NIST.SP.800-190.

Sysprove (2020) *ISO 27017 – Security Controls for Cloud Services | Sysprove Consulting*. Disponível em: https://sysprove.com/iso-27017-security-controls-for-cloud-services/ (Acedido: 5 de Fevereiro de 2021).

Tenable (2021) *Nessus Product Page*. Disponível em: https://pt-br.tenable.com/products/nessus (Acedido: 15 de Junho de 2021).

Terraform Registry (2021a) *Data Source: aws_kms_secrets*. Disponível em: https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/kms_secrets (Acedido: 14 de Junho de 2021).

Terraform Registry (2021b) *Providers*. Disponível em: https://registry.terraform.io/browse/providers (Acedido: 14 de Junho de 2021).

Terraform Registry (2021c) *Resource: aws_api_gateway_account*. Disponível em: https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/api_gateway_account (Acedido: 7 de Junho de 2021).

Thales Group (2019) *2019 Identity Access Management Index - Home | Thales*. Disponível em: https://cpl.thalesgroup.com/access-management-index (Acedido: 12 de Fevereiro de 2021).

Varma, A. (2018) *Security in the Public Cloud — Roll your own DMZ on AWS, Azure or Google Cloud*. Disponível em: https://medium.com/public-cloud-security/security-in-the-public-cloud-roll-your-own-dmz-on-aws-azure-or-google-cloud-d90598fbda28 (Acedido: 11 de Fevereiro de 2021).

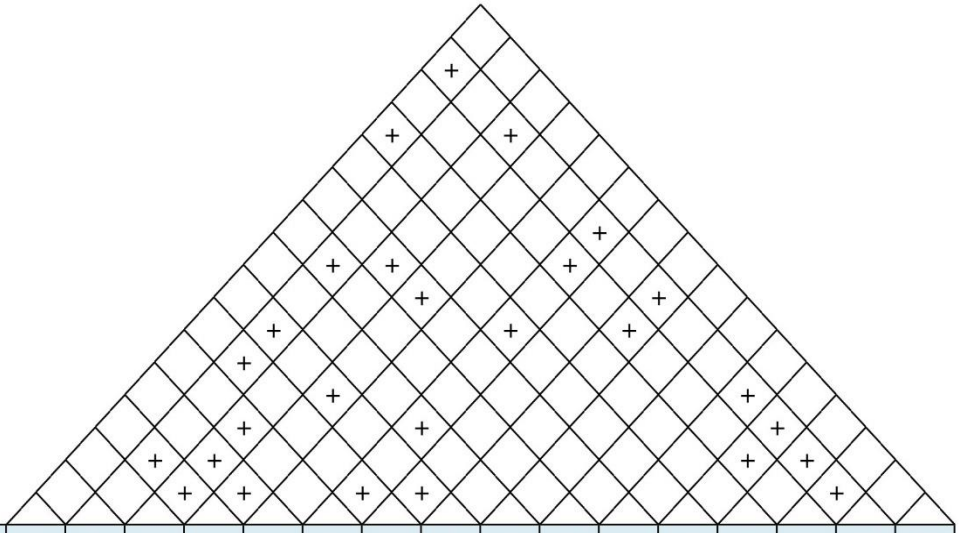
ZAP Dev Team (2022) *OWASP Zed Attack Proxy (ZAP)*. Disponível em: https://www.zaproxy.org/ (Acedido: 15 de Junho de 2021).

Anexo A: QFD House of Quality

Correlations	
Positive	+
Negative	−
No Correlation	

Relationships	
Strong	●
Moderate	○
Weak	▽

Direction of Improvement	
Maximize	▲
Target	◇
Minimize	▼



Row #	Weight Chart	Relative Weight	Customer Importance	Maximum Relationship	Functional Requirements / Customer Requirements (Explicit and Implicit)	1 Testes de penetração	2 Controlos ao nível de rede	3 Documentação das operações	4 Gestão Controlo de Acessos	5 Gestão de recursos computacionais	6 Programação segura	7 Controlos contra malware	8 Sistemas monitorização e alarmística	9 Controlos criptográficos apropriados	10 Segregação na rede	11 Threat modeling	12 Ferramentas análise estática	13 Ferramentas análise dinâmica	14 DevSecOps	15 Integração da segurança no SDLC	16
					Direction of Improvement	▲	▲	▲	▲	▲	▲	◇	▲	◇	◇	▲	▲	▲	◇	▲	
1	■	10%	5	9	Deteção de vulnerabilidades no código	▽					○						●	▽	○	▽	
2	■	10%	5	9	Deteção vulnerabilidades no aplicativo	●					▽		▽			▽	▽	●	○	▽	
3	■	10%	5	9	Estratégias de mitigação		○	○	○	▽		○	▽	○	○	●				▽	
4	■	10%	5	9	Framework metodologia agnóstica															●	
5	■	15%	7	9	Reduzir superfície de ataque	▽	○		▽	▽	▽				○	●	○				
6	■	6%	3	3	Revisões regulares da segurança			▽	▽	○							▽	▽	○	▽	
7	■	6%	3	9	Preparação contra eventos segurança	○	●	○	○	○	○	●	●	○	●	●					
8	■	6%	3	9	Promover cultura de segurança	○	▽	○	●	▽	●		▽	▽		▽	●	▽	○	●	
9	■	15%	7	9	Deteção vulnerabilidades na arquitetura	●		▽					▽			●		▽		▽	
10	■	10%	5	9	Prevenir quebras de privacidade	▽	●	○	○	●	▽	○	▽	●	○	○	○	○	▽	▽	
					Target	Aquando do lançamento ou nova versão de um produto	Aplicação de controlos ao nível da rede	Operações do sistema e responsabilidades documentadas	Revisão regular das permissões	Revisão regular dos recursos computacionais	Estabelecimento de políticas de programação	Controlos de malware em todas as workstations	Todas as operações do sistema devem ser logged	Estabelecimento de uma política de criptografia	Segregação de ambientes (dev, test...) e por áreas funcionais	Threat modeling feito na fase de design e mantido atualizado	Uso de ferramentas SAST para apoiar a programção segura	Uso de ferramentas DAST em testes de penetração	Processo de DevOps com preocupações com segurança	Procupação global com a segurança	
					Max Relationship	9	9	3	9	9	9	9	9	9	9	9	9	9	3	9	
					Technical Importance Rating	297,92	231,25	120,83	158,33	162,5	141,67	118,75	108,33	150	162,5	460,42	241,67	162,5	110,42	212,5	
					Relative Weight	10%	8%	4%	6%	6%	5%	4%	4%	5%	6%	16%	9%	6%	4%	7%	
					Weight Chart	■	■	▬	▬	▬	▬	▬	▁	▬	▬	■	■	▬	▁	▬	

Anexo B: Duração das tarefas

Tarefa	Data Início	Data Fim
Análise dos controlos ISO e estado da arte	12/01/2021	05/02/2021
Introdução testes de penetração	13/01/2021	15/04/2021
Levantamento arquitetura referência	08/02/2021	12/02/2021
Seleção do CSP	15/02/2021	17/02/2021
Desenho da arquitetura	18/02/2021	15/04/2021
Investigação dos serviços AWS	18/02/2021	11/06/2021
Materialização da prova de conceito	16/04/2021	11/06/2021
Testes à prova de conceito	09/06/2021	18/06/2021
Avaliação da solução	14/06/2021	30/06/2021
Escrita da dissertação	21/12/2020	30/06/2021

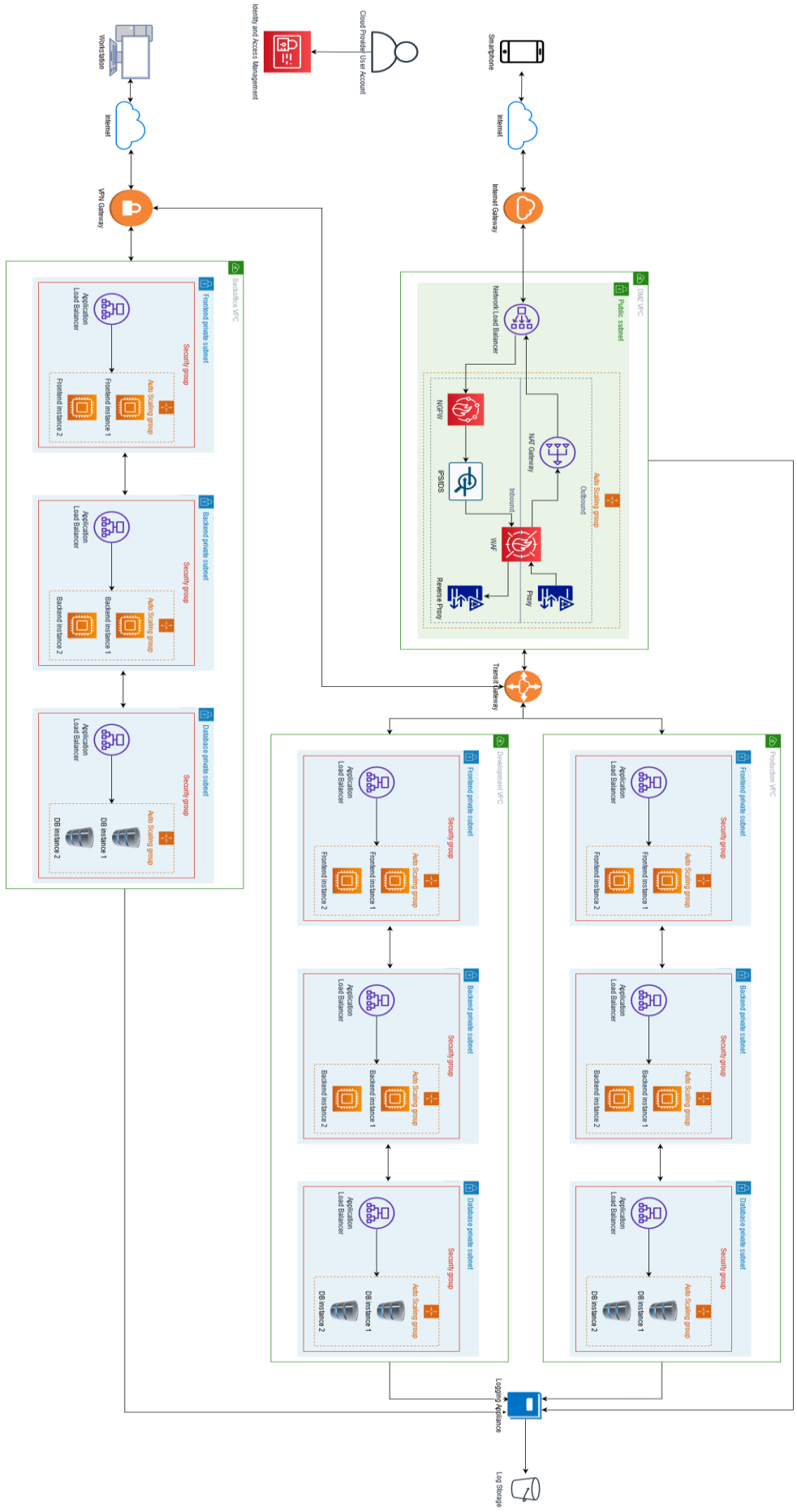
Anexo C: Framework de validação de controlos de segurança

Testes à segurança	Os componentes do sistema são alvo de testes de penetração periódicos? São utilizadas ferramentas que suportem a política de programação segura? (ex: SAST, DAST, SCA) Houveram revisões manuais do código? Foi adotado um método DevSecOps? Foram tidas em conta as boas práticas de programação para a segurança? (OWASP API, Mobile e Web Security) Foi feito um modelo Threat Modeling? Este modelo encontra-se atualizado?
Gestão de eventos de segurança	Existem canais para reportar eventos de segurança? Os eventos são categorizados de acordo com a sua criticidade? Os diferentes níveis de criticidade exigem tempos de atuação diferentes?
	Existe um documento a detalhar as respostas a incidentes de segurança? Cada incidente tem responsáveis atribuídos?
	Estão identificados os órgãos ou autoridades a notificar em caso de incidente? Existe um processo definido para a recolha de evidências para análise forense?
	Existe um plano de continuidade definido? O plano tem os eventos de crise claramente definidos? Os eventos têm uma estratégia de mitigação associada?
	Os responsáveis pela segurança estão devidamente identificados? As suas responsabilidades estão devidamente separadas e atribuídas?
Controlo de acessos	Existe uma política de controlo de acessos? Os controlos (físicos e informatizados) estão geridos centralmente? Todos os utilizadores têm os seus acessos documentados? Existe um período de revisão dos acessos estabelecido? Existem políticas para revisão e alteração dos acessos de um colaborador em caso de saída ou mudança de cargo/ área? Existe algum critério para atribuição de acessos (zero trust, least privilege)?
	As contas de utilizador com acessos mais sensíveis são revistas mais frequentemente?
	Um utilizador com conta de administrador tem uma outra conta para acessos mais comuns?
	Existem mecanismos de restrição do acesso / alarmistica? Existem mecanismos de lockout de login? Existem mecanismos de timeout de acesso?
	Controlos de acesso estão implementados?

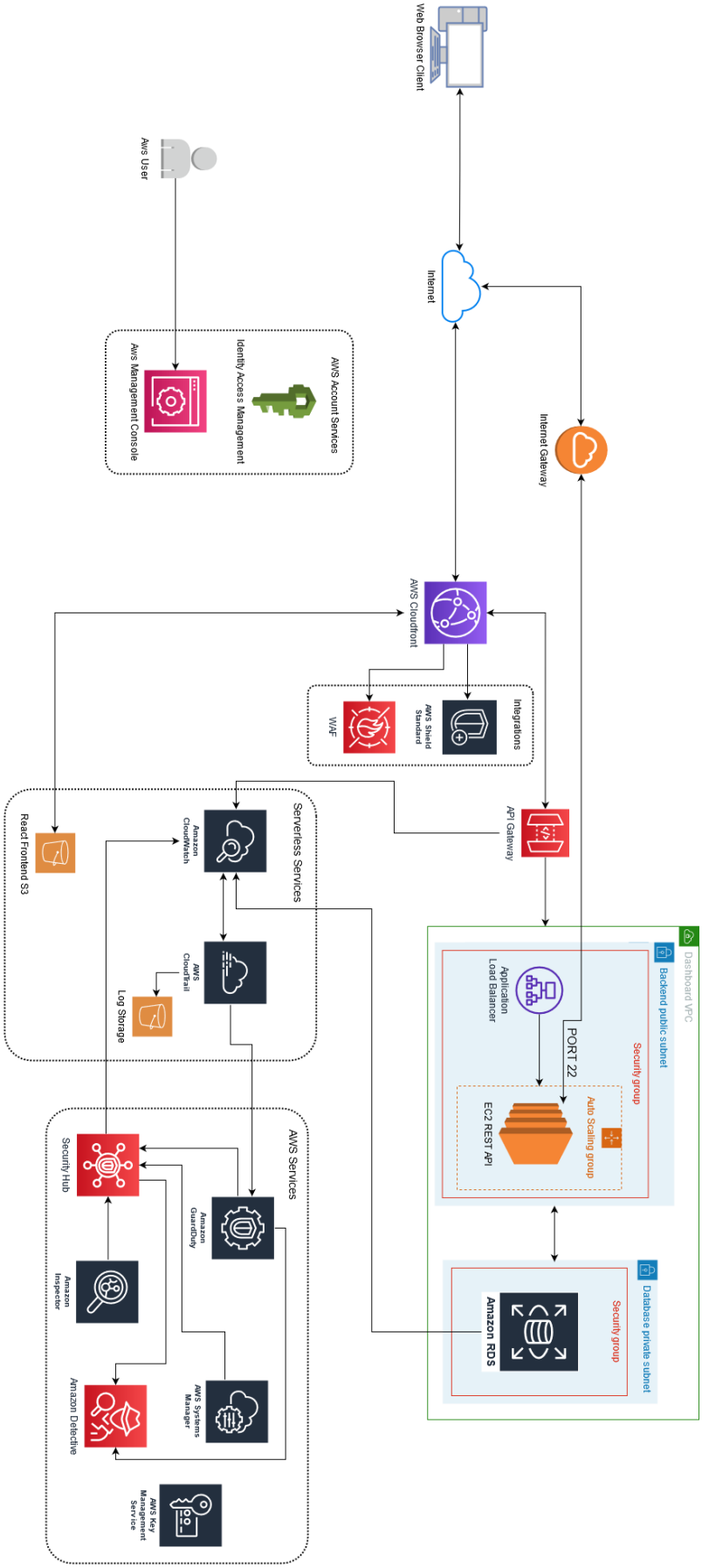
	É usado Single Sign On? É usado MFA?
Segurança ao nível operacional	Existe um documento com as operações do sistema (startup, shutdown)?
	Existe uma política de backup definida? A informação a ser incluida no backup está identificada? Estão definidos os tipos de backups? (Integral, incremental, diferencial) Estão definidos períodos de frequência para os backups? Os backups estão a ser encriptados? Estão armazenados num local distinto do sistema?
	É feita uma procura frequente de vulnerabilidades técnicas da aplicação? (ex: bibliotecas, frameworks)
	Existem logs a registar as atividades dos utilizadores? Existem logs a registar as atividades dos eventos do sistema? Existem logs de aplicação? Existem logs de comunicação? Os logs são encriptados? É garantida a integridade dos logs? Os logs são centralizados?
	São lançados alarmes para os logs?
Gestão da informação	Existe um inventário de ativos? Tanto de infraestrutura como de informação?
	A informação está classificada de acordo com diferentes níveis de sensibilidade? (Pública, Privada, Restrita, Confidencial)
Conformidade	A aplicação vai ao encontro das legislações a que está sujeita? (ex: RGPD, PCI DSS) A aplicação vai de encontro aos requisitos levantados? A aplicação está em conformidade com as políticas da empresa?
	É usado algum serviço e/ou produto third-party? Se sim, estamos salvaguardados contratualmente?
Camada tecnológica	Existe uma política de programação segura definida? Esta política é facilmente acessível pelas equipas de desenvolvimento? A política especifica linguagens de programação e frameworks aceites? A política especifica boas práticas a adotar?
	Os sistemas operativos estão atualizados? Foram utilizadas imagens *hardened* do sistema operativo? São procuradas com frequência vulnerabilidades do sistema operativo? Os patchs aplicacionais estão centralizados?
	Estão a ser utilizados controlos contra *malware* como software anti-virus? Está atualizado?
Camada de arquitetura	É feita a segregação de redes de acordo com os ambientes (ex: live, dev, test)? É feita a segregação de rede a cada ambiente? (camadas apresentação, aplicacional, dados, infraestrutura)

	É feita a segregação entre soluções internas e produtos expostos à internet?(Ex: BD,backend)
	A aplicação está exposta à Internet? Está implementada uma DMZ? Existe algum controlo para identificar atividade maliciosa? (ex: IPS/IDS, NGFW) Existe algum controlo para identificar e bloquear ataques aplicacionais? (ex: WAF, NGFW) Existe algum controlo para verificação de ficheiros potencialmente perigosos? (ex: Sandbox, NGFW) Existe algum controlo que permita ocultar a stack tecnológica do sistema? (ex: Reverse Proxy, Gateway)
	A aplicação é para uso interno? Não está exposta ao público?
	Existe algum controlo que realize filtragem de pacotes IP? (ex: firewalls, NSG, NGFW) As firewalls realizam filtragem *default deny?*
	Os dados estão a ser encriptados em repouso? E em trânsito? Os métodos de encriptação são adequados dada a sensibilidade dos dados? Quais são as cifras utilizadas? O que está a ser encriptado? (Base de dados, tabela, coluna, volume) Está a ser feita a gestão de chaves criptográficas e certificados?
	São utilizados protocolos de comunicação seguros? HTTPS, TLS…

Anexo D: Arquitetura de referência



Anexo E: Arquitetura da prova de conceito para implantação em AWS



Anexo F: Descobertas do Security Hub

☐	LOW	NEW	ACTIVE	AWS	Security Hub	3.1 Ensure a log metric filter and alarm exist for unauthorized API calls	Account 956642731140	FAILED	4 minutes ago
☐	MEDIUM	NEW	ACTIVE	AWS	Security Hub	2.5 Ensure AWS Config is enabled	Account 956642731140	FAILED	4 minutes ago
☐	LOW	NEW	ACTIVE	AWS	Security Hub	1.1 Avoid the use of the "root" account	Account 956642731140	FAILED	5 minutes ago
☐	CRITICAL	NEW	ACTIVE	AWS	Security Hub	2.3 Ensure the S3 bucket used to store CloudTrail logs is not publicly accessible	Account 956642731140	WARNING	5 minutes ago
☐	LOW	NEW	ACTIVE	AWS	Security Hub	2.6 Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket	Account 956642731140	WARNING	5 minutes ago

Anexo G: Questionário de satisfação

Icarus - A Cloud Security Perspective

Este questionário tem como objetivo avaliar a solução desenvolvida durante o trabalho no projeto da dissertação para a obtenção do Mestrado em Engenharia de Software no Instituto Superior de Engenharia do Porto.
O foco deste questionário passa por avaliar a adequação dos controlos de segurança em AWS explorados ao longo do trabalho na dissertação.

Este questionário destina-se a profissionais ou estudantes da área das Tecnologias da Informação.

O questionário tem uma duração aproximada de 7 a 10 minutos, devendo limitar-se a uma resposta.

Nota: Sendo este inquérito alojado na plataforma Google Forms, é instalado no seu browser um cookie para efeitos publicitários com uma validade de 6 meses. Caso a sua privacidade e o seu perfil online constituam uma preocupação sua, deverá considerar limpar os cookies do respetivo navegador que esteja a utilizar.

*Obrigatório

Avaliação do perfil

1. Conhece o termo 'computação em nuvem'? *

   Marcar apenas uma oval.

   ◯ Sim
   ◯ Não

2. Já entrou em contacto com este tema no seu percurso académico e/ou profissional? *

   Marcar apenas uma oval.

   ◯ Sim
   ◯ Não

3. Quais dos seguintes fornecedores de serviços de computação em nuvem já utilizou? *

Marcar tudo o que for aplicável.

- [ ] Amazon Web Services
- [ ] Microsoft Azure
- [ ] Google Cloud

4. Como avalia a sua familiaridade com o tema de computação em nuvem? *

Marcar apenas uma oval.

	1	2	3	4	5	
Nunca utilizei este tipo de serviços	◯	◯	◯	◯	◯	Utilizo regularmente

5. Alguma vez implantou uma solução informática em nuvem? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

Segurança na nuvem

6. Considera a segurança um aspeto importante da computação em nuvem? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

7. Como avalia o seu conhecimento sobre temas relacionados à segurança da informação? *

Marcar apenas uma oval.

	1	2	3	4	5	
Pouco confortável	◯	◯	◯	◯	◯	Muito confortável

8. Concorda com a seguinte afirmação? "As soluções informáticas implantadas em nuvem acarretam mais riscos quando comparadas a soluções implantadas on-premise (servidores locais)" *

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

9. Conhece o termo 'segurança em profundidade'? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

10. Alguma vez implantou uma solução em nuvem usando uma abordagem de segurança em profundidade? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

11. Concorda com a seguinte afirmação? "Com testes de penetração regulares a um sistema informático, são identificadas vulnerabilidades na sua configuração mais facilmente"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

12. Concorda com a seguinte afirmação? "Graças ao uso de ferramentas de análise de código, podemos evitar a introdução de vulnerabilidades aplicacionais na fase de implementação"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

13. Considera que recorrendo a testes de penetração regulares e a ferramentas de análise de código temos melhores garantias quanto à segurança operacional de um sistema informático?

Marcar apenas uma oval.

◯ Sim
◯ Não

14. Concorda com a seguinte afirmação? "Após um incidente de segurança, é importante obter o máximo de informação e contexto possível acerca da origem do evento"

Marcar apenas uma oval.

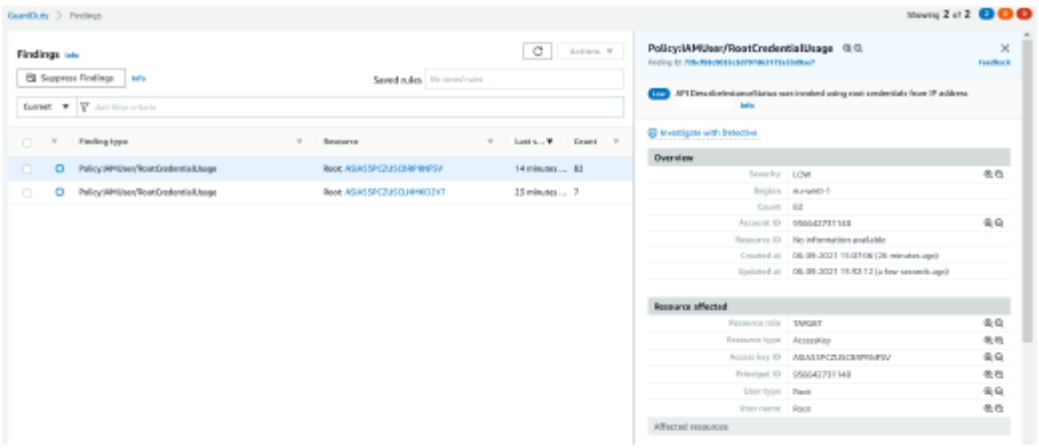
	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Introdução ao GuardDuty

Serviço que analisa eventos ao nível da conta AWS de forma a identificar atividade potencialmente perigosa e contas de utilizador comprometidas.

Mais informação: https://aws.amazon.com/guardduty/features/

Exemplo das descobertas feitas pelo GuardDuty (Abrir imagem num novo separador)



15. Considera que um serviço como o GuardDuty ajuda a prevenir contra ações indevidas e a identificar atores maliciosos no contexto da AWS?

Marcar apenas uma oval.

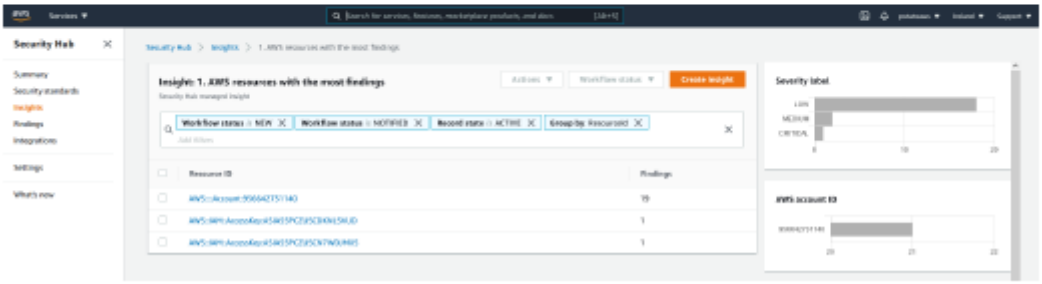
- ◯ Sim
- ◯ Não

Introdução ao Security Hub

Serviço que centraliza as descobertas feitas por serviços de segurança (ex: GuardDuty) e fornece sugestões quanto à melhoria da postura de segurança.

Mais informação: https://aws.amazon.com/security-hub/features/

Exemplos das descobertas feitas pelo Security Hub (Abrir imagem num novo separador)



16. Considera que um serviço como o Security Hub promove a visibilidade de eventos pertinentes à segurança de um sistema informático implantado em AWS?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

17. Considera que aplicação de controlos como o GuardDuty e o Security Hub assistem na resposta e preparação para eventuais incidentes de segurança?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

18. Concorda com a seguinte afirmação? "O controlo rigoroso das permissões dos utilizadores num sistema informático ajuda a evitar incidentes de acessos indevidos e de erro humano"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Introdução ao Identity and Access Management (IAM)

Serviço que permite controlar o acesso e as permissões de utilizadores (ou grupos de utilizadores) aos serviços e recursos implantados na nuvem com um controlo granular específico a esse dado serviço.

Mais informação: https://aws.amazon.com/iam/features/

19. Considera que um serviço como o IAM promove a manutenção de um sistema informático onde os utilizadores apenas têm acesso aos recursos que necessitam para realizar o seu trabalho (princípio least priviledge)?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

20. Concorda com a seguinte afirmação? "A centralização dos segredos de um sistema (certificados e chaves geradas) é uma boa ideia desde que seja realizada uma gestão rigorosa deste componente"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Introdução ao Key Management Service (KMS)

Serviço que fornece um controlo centralizado sobre as chaves utilizadas para a encriptação da informação detida nos vários recursos implantados em AWS e permite a rotação automática de chaves geradas.

Mais informação: https://aws.amazon.com/kms/features/

21. Considera que um serviço como o KMS promove a manutenção de um ecossistema de gestão segura de segredos utilizados na encriptação e dos certificados?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

Controlos de Segurança IV - Segurança operacional

22. Concorda com a seguinte afirmação? "O logging de eventos do sistema aliado à monitorização contínua dos componentes do nosso sistema informático ajudam-nos a identificar atividade maliciosa e comportamentos inesperados que podem indicar a existência de uma vulnerabilidade"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Introdução ao CloudWatch

Monitoriza métricas de performance e configura alarmística para a nossa infraestrutura, aplicações e serviços, possibilitando a automação de respostas e alarmes para dados eventos.

Mais informação: https://aws.amazon.com/cloudwatch/features/

Introdução ao CloudTrail

Logging de eventos ao nível da conta AWS, fornecendo uma maior visibilidade sobre quaisquer alterações à infraestrutura e aos recursos implantados.

Mais informação: https://aws.amazon.com/cloudtrail/features/

23. Considera que o uso conjunto dos serviços CloudWatch e CloudTrail fornecem uma melhor visibilidade sobre o funcionamento da infraestrutura, ajudam a identificar potenciais vulnerabilidades e constituem ferramentas importantes num momento de análise forense?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

24. Concorda com a seguinte afirmação? "É importante realizar uma gestão ativa das vulnerabilidades identificadas num sistema informático"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Revisitando o Security Hub

Conforme mencionado anteriormente, o Security Hub fornece sugestões de melhoria de segurança, nomeadamente, identifica potenciais vetores de ataque e classifica-os conforme a sua criticidade.

Mais informação: https://aws.amazon.com/security-hub/features/

Exemplos das sugestões feitas pelo Security Hub (Abrir imagem num novo separador)

25. Considera que o uso do Security Hub para a identificação de potenciais vetores de ataque permite uma melhoria incremental da postura de segurança de um sistema informático?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

Controlos de segurança V - Gestão da informação

26. Concorda com a seguinte afirmação? "A visibilidade sobre o inventário informático e a sua manutenção é um componente importante da postura de segurança de uma organização"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Introdução ao Systems Manager

Centraliza informação operacional dos recursos implantados em nuvem, agrega recursos por diferentes grupos (ex: produção vs desenvolvimento) e fornece uma melhor visibilidade sobre o funcionamento dos diversos componentes do sistema.

Mais informação: https://aws.amazon.com/systems-manager/features/

27. Considera que o uso do Systems Manager promove a visibilidade sobre o inventário informático e a sua manutenção?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

28. Concorda com a seguinte afirmação? "A definição de uma política de programação segura e a sua aplicação através da definição de mecanismos automáticos (ex: quality gates) ajuda a diminuir a chance de surgirem vulnerabilidades num sistema informático"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Introdução ao Amazon Inspector

Executa avaliações à infraestrutura de forma a identificar pontos de entrada ou vulnerabilidades identificadas em recursos computacionais. Permite a definição de boas práticas aceites e validar se estão a ser seguidas.

Mais informação: https://aws.amazon.com/inspector/

29. Considera que o Inspector ajuda na identificação de riscos e vulnerabilidades que possam aparecer nos recursos computacionais implantados?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

30. Concorda com a seguinte afirmação? "Qualquer organização que opte por implantar uma solução informática na nuvem deve ter conhecimento do modelo de responsabilidade partilhada da plataforma de hospedagem"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

31. Concorda com a seguinte afirmação? "Uma organização deve de ter conhecimento das legislações a que está sujeita e da sua aplicabilidade num ambiente de hospedagem em nuvem"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Introdução ao AWS Artifact

Serviço que fornece acesso a diversos documentos da AWS relevantes aos controlos de conformidade e da gestão de eventos de segurança.

Mais informação: https://aws.amazon.com/artifact/

32. Considera que o Artifact permite a uma organização enquadrar as suas responsabilidades e requisitos legais num ambiente nuvem?

Marcar apenas uma oval.

◯ Sim
◯ Não

Controlos de segurança VII - Camada tecnológica

33. Concorda com a seguinte afirmação? "A gestão ativa da superfície de ataque constitui um controlo de segurança importante no desenvolvimento e operação de um sistema informático"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

34. Considera que o uso de imagens hardened numa solução informática que faça uso de máquinas virtuais ajudam a reduzir a superfície de ataque da nossa infraestrutura?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

Controlos de segurança VIII - Camada de arquitetura

35. Concorda com a seguinte afirmação? "A encriptação de informação sensível é um dos pilares da segurança informática"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

36. Considera que o uso de um serviço como o KMS que facilita a integração de diversos outros serviços da AWS promove a aplicação de encriptação nos diversos componentes (ex: bases de dados, logs, armazenamento de objetos) de um sistema informático?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

37. Concorda com a seguinte afirmação? "O uso de firewalls e a configuração de regras de filtragem permitem reduzir a superfície de ataque de um sistema"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Introdução a Network Security Groups (NSG)

Firewall virtual para o controlo de tráfego de entrada e saída em instâncias de computação.

Mais informação: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html

38. Considera que um NSG, quando devidamente configurado, reduz consideravelmente a superfície de ataque de um sistema informático?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

39. Concorda com a seguinte afirmação? "A segurança do perímetro de uma rede constitui uma camada importante para a deteção e proteção contra ataques à nossa infraestrutura"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Introdução a WAF, AWS Shield e CloudFront

WAF: Firewall de aplicações Web que deteta e bloqueia os ataques aplicacionais mais comuns. (https://aws.amazon.com/waf/features/)

Shield: Deteção e proteção contra ataques DoS (https://aws.amazon.com/shield/features/)

CloudFront: Rede de distribuição de conteúdos global com capacidades de caching. Integra com os serviços WAF e Shield e torna-se o ponto de entrada na infraestrutura. (https://aws.amazon.com/cloudfront/features/)

40. Considera que o uso conjunto dos serviços WAF, AWS Shield e CloudFront fortalece a segurança do perímetro de um sistema informático contra diversos tipos de ataques e tentativas de intrusão?

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

41. Concorda com a seguinte afirmação? "A segregação de uma rede num conjunto de sub-redes fornece um melhor isolamento entre os diversos componentes da infraestrutura, dificultando o movimento de atacantes no sistema e prevenindo a propagação de ataques pela rede"

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo completamente	◯	◯	◯	◯	◯	Sem qualquer dúvida

Introdução a Virtual Private Cloud (VPC) e Subnets

VPC: Rede virtual isolada onde podem ser instanciados recursos da AWS. (https://aws.amazon.com/vpc/)

Subnet: Sub-rede virtual inserida numa VPC. (https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Subnets.html)

42. Considera que o uso de VPCs e Subnets oferece um melhor isolamento entre os componentes implantados em nuvem, contribuindo assim para uma melhor postura de segurança da infraestrutura?

Marcar apenas uma oval.

- Sim
- Não

Fim!

Obrigado pelo tempo despendido na resposta a este questionário!