Arquitetura segura no desenvolvimento de software: Abordagem à plataforma digital U.OPENLAB

Domingos Guedes Ferreira

Mestrado em Segurança Informática

Departamento de Ciência de Computadores

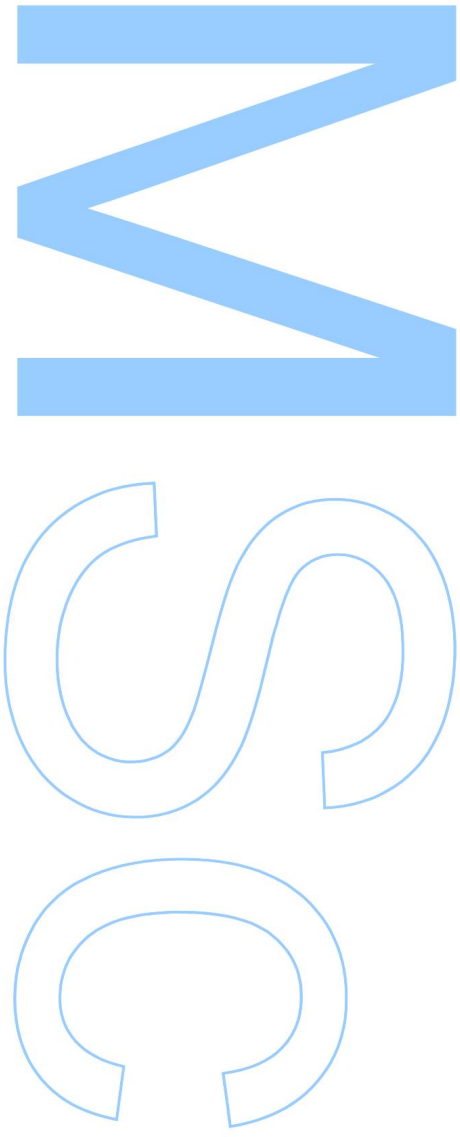
2019

Orientador

Luís Antunes, Professor Catedrático, DCC - Faculdade de Ciências da Universidade do Porto

Coorientador

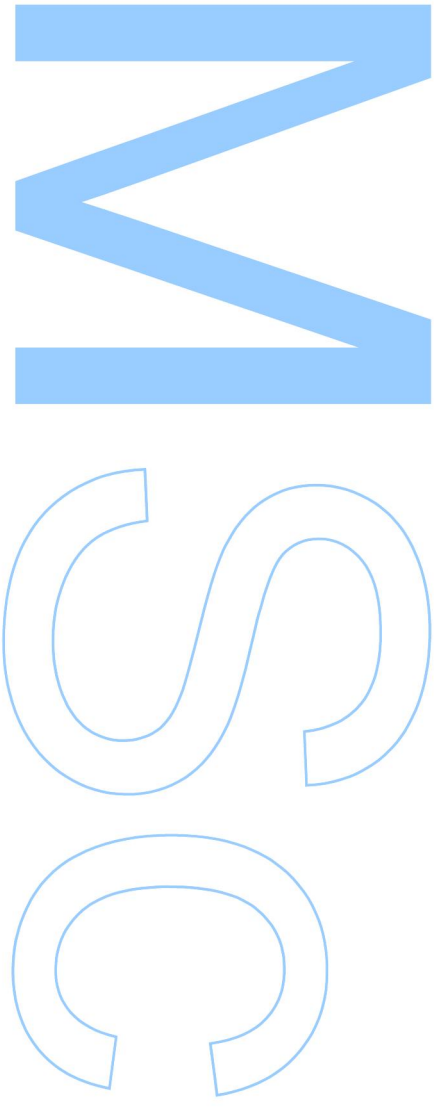
Maria Manuela Pinto, Professora Auxiliar, DCCI - Faculdade de Letras da Universidade do Porto



Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____



Agradecimentos

Agradeço sinceramente ao Professor Doutor Luís Antunes da Faculdade de Ciências da Universidade do Porto e à Professora Doutora Maria Manuela Pinto da Faculdade de Letras da Universidade do Porto pela orientação deste trabalho.

Apresento também os meus agradecimentos ao Professor Doutor Armando Sousa e Dr. Rodolfo Matos da Faculdade de Engenharia da Universidade do Porto que como responsáveis pelo desenvolvimento e implementação do projeto U.OPENLAB foram sempre muito solícitos no acompanhamento e desenvolvimento desta dissertação.

Resumo

Esta dissertação abarca questões relacionadas com a segurança informática em projetos de desenvolvimento de *software*, nomeadamente as tarefas necessárias para obter uma arquitetura segura numa plataforma digital. Desde os princípios gerais da segurança da informação em sistemas computadorizados, passando pela escolha de uma metodologia e modelos de desenvolvimento, as diversas etapas de análise da estrutura de funcionamento e objetivos da plataforma, a identificação dos ativos informacionais, a análise de vulnerabilidades e riscos, a especificação de requisitos de segurança, até à proposta de uma arquitetura segura e tecnologias a incorporar para o projeto U.OPENLAB. Incide-se, de forma particular, na identificação dos requisitos de segurança e respetiva sistematização.

Palavras chave: Segurança da informação, plataformas digitais, arquitetura segura, requisitos de segurança, U.OPENLAB, Universidade do Porto.

Abstract

This dissertation covers issues related to computer security in software development projects, namely the tasks required to achieve a secure architecture on a digital platform. From the general principles of information security in computer systems, through the choice of a methodology and development models and the various stages of analysis of the platform's operating structure and objectives, identification of informational assets, vulnerability and risk analysis, specification of security requirements, to the proposal of a secure architecture and technologies to incorporate into the U.OPENLAB project. It focuses particularly on the identification of security requirements and their systematization.

Keywords Information Security, Digital Platforms, Secure Architecture, Security requirements, U.OPENLAB, University of Porto.

Sumário

Índice de tabelas

Índice de imagens

Lista de abreviaturas, acrónimos e siglas

- AAA - Authentication, Authorization and Auditability - Autenticação, autorização e rastreabilidade
- CIA - Confidentiality, integrity and Availability - Confiabilidade, integridade e disponibilidade
- CMM - Capability Maturity Models
- CSP - Content security Policy - Política de segurança de conteúdos
- DMZ - DeMilitarized zone - Zona desmilitarizada
- DOS - Denial Of Service - Negação de serviço
- FISMA - Federal Information Security Management Act
- InfoSec - Information Security - Segurança da informação
- ISMS - Information Security Management System - Sistema de gestão de segurança da informação
- OTP – One Time Password
- OWASP - Open Web Application Security Project
- PIN – Personal Identification Number – Número de identificação pessoal
- PKI - Public Key infrastructure - Infraestrutura de chave pública
- SAMM - Software Assurance Maturity Model - Modelo de maturidade de garantia do software.
- SDLC- Software Development Life Cycle (SDLC)
- SI- Sistema de Informação - Information system
- SO - Sistema Operativo - Operating system
- SQL- Sctructured Query Language

- SSDLC - Secure Software Development Life Cycle - Ciclo de vida de desenvolvimento seguro de software
- SSO - Single Sign On - Autenticação única
- VPN - Virtual Private Network - Rede virtual privada
- WAF - Web Application Firewall - Firewall de aplicação Web
- XXE - External XML Entities
- XSS - Cross Site Scripting

1. Introdução

A segurança informática e da informação são questões fundamentais para os produtos e serviços de software. Nos dias de hoje em que para além das preocupações com o funcionamento das plataformas digitais também existem preocupações relacionadas com a privacidade e a confidencialidade, as organizações que desenvolvem software devem ter em conta todos os aspetos da segurança, introduzindo controles que minimizem potenciais danos que qualquer brecha possa causar, seja nos equipamentos, na informação e mesmo na reputação da própria organização. No caso de plataformas abertas e de acesso público, é ainda mais importante garantir todas estas questões relacionadas com a segurança informática, pois qualquer falha pode ganhar grandes proporções e prejudicar seriamente a organização, seja na sua imagem, seja por questões pecuniárias, designadamente por regulamentação como o RGPD (Regulamento Geral de Proteção de Dados).

Partindo da regra do senso comum de que é melhor e mais fácil prevenir do que remediar, em especial no que concerne à segurança, importa seguir uma metodologia que permita ter um rumo definido e poder ir fazendo retificações quando necessário evitando, assim, custos de desenvolvimento e prejuízos na reputação da organização. Será então importante que qualquer entidade que desenvolva e/ou disponibilize plataformas digitais consiga garantir que os cuidados relacionados com a segurança da informação foram e são devidamente acautelados durante a execução do projeto cujos resultados são disponibilizados aos utilizadores. Estas preocupações são ainda mais prementes quando a plataforma digital funciona, por exemplo, num regime de coautoria, onde, para além dos normais requisitos de segurança, acrescem questões relacionadas com direitos de autor, licenciamento de obras, patentes, etc., pois, sendo a maioria dos utilizadores também criadores da informação existente na plataforma, é importante manter registos inequívocos de quem são os contribuidores, assegurando um aspeto crítico relacionado com uma das propriedades da segurança da informação – a autenticidade –, para que, adicionalmente, se lhes possa prestar os devidos créditos e respetivos benefícios, como acontecerá no U.OpenLab.

1.1. Problemática, objetivos e questão de partida

A presente dissertação decorre da necessidade apresentada pelo projeto de desenvolvimento de uma plataforma digital colaborativa na U.Porto - o U.OPENLAB -, que visa suportar os processos decorrentes da oferta/submissão e licitação/aceitação de propostas de trabalhos a desenvolver no contexto do processo de ensino/aprendizagem na universidade ou instituições afins, apontando-se para uma solução que leve em conta o “Secure by design”.

Pretende-se, assim, garantir que nas diferentes fases do projeto e de forma precoce se seguem as melhores recomendações de desenvolvimento de software no que diz respeito à segurança, com uma metodologia que vai da identificação e análise das fragilidades que o próprio SI apresenta, de forma a poder mitigar os riscos, até à identificação e análise das vulnerabilidades, e consequente eliminação de forma eficiente e eficaz através da introdução dos respetivos controles na plataforma, e cálculo do risco.

Para este projeto, todas as especificações de segurança devem estar em conformidade com os seguintes requisitos funcionais base:

- A utilização desta plataforma deve utilizar credenciais baseadas em sistemas de SSO (*single sign on*):
  - Autenticação federada U.Porto para utilizadores internos;
  - Serviços que permitam esta funcionalidade (ex: Google, Facebook, Linked-In, etc.) para utilizadores externos.
- Cada utilizador deve ter um registo interno único e deve aceitar os termos de utilização.
- Os proponentes (por ex. serviços e instituições da universidade) podem registar propostas de trabalhos a realizar.
- Trabalho em grupo (ver Pinto, Medina, Matos, & Fontes, 2016).

Os tipos de utilização da plataforma incluem:

i. criar novos conteúdos (metadados, reproduções digitais de artefactos, documentários, etc.)

ii. criar conteúdos baseados em informação já existente no sistema (é necessário registar todos os contribuidores de forma indelével e unívoca). A criação será para regressar ao sistema

iii. visualização / utilização de conteúdos (de acordo com contrato, com ou sem custos) (necessário registar todos os contribuidores de forma indelével e unívoca) permite acesso, mas a eventual nova criação não regressa ao sistema, exceto, eventualmente, numa ligação externa

iv. os proponentes podem ser internos (U.Porto) ou externos, de qualquer forma, terão de estar sempre identificados."

Em todos os casos, o acesso é por intermédio de ligações seguras (API standard), devendo a plataforma ser "*Secure by design*".

As especificações devem incluir requisitos de segurança, designadamente: os de confidencialidade, os de integridade e disponibilidade, desde a validação de inputs, alteração de permissões, acessos não desejados, corrupção da informação quer a disponibilizada, quer a do próprio SI (Sistema de Informação), incluindo as cópias de segurança, etc. Deverá, ainda, atender a aspetos como: a implementação do sistema e perfis de utilizador; os requisitos RGPD; as potenciais vulnerabilidades; a criação de relatórios de análise de risco; pontos de controlo, etc.

Será, pois, interessante saber se incluir metodologias e tarefas específicas da segurança informática durante as fases iniciais do desenvolvimento da plataforma contribui para a melhoria da qualidade e redução de custos desta e ajudar a atingir a meta de ser *Secure by Design*, mesmo já tendo definidos alguns requisitos funcionais e de ambiente de funcionamento.

Em alinhamento com este questionamento, definem-se como principais objetivos desta dissertação:

- Especificar os requisitos de segurança para o desenvolvimento de software e de uma plataforma digital baseada na web;

- Proceder à análise de vulnerabilidades e cálculo de risco;

- Propor uma arquitetura segura para o sistema;
- Propor tecnologias para a fase de implementação do sistema.

1.2. Estrutura da Dissertação

Este relatório é composto por quatro capítulos principais: um primeiro capitulo introdutório, um de Enquadramento Teórico-metodológico, um com o Estudo de Caso e o que contém as Considerações Finais, seguindo-se as Referências Bibliográficas e Anexos.

Na Introdução, faz-se uma breve alusão à problemática, enunciam-se as especificações base com as quais se desenvolverá o restante trabalho e termina-se com a apresentação da estrutura da dissertação.

No Enquadramento Teórico-metodológico, sistematiza-se o estado da arte, descreve-se a abordagem metodológica e apresentam-se os princípios e conceitos utilizados para desenvolver o trabalho.

No Estudo de Caso, apresenta-se o projeto acolhedor – uma plataforma U.OPENLAB para a U.Porto –, analisam-se os casos de mau uso ou abuso, elaboram-se os requisitos de segurança, procede-se a uma análise qualitativa das vulnerabilidades e ao cálculo do risco, termina-se com a proposta de uma arquitetura segura para a plataforma e sugerem-se as tecnologias a utilizar para a fase de implementação do sistema.

Nas Considerações Finais, apresentam-se os resultados e respostas às questões que este trabalho apresentou.

2. Enquadramento teórico-metodológico

Para a conceção e execução do projeto, foram efetuadas reuniões com os responsáveis da equipa de desenvolvimento do protótipo da plataforma U.OPENLAB que serviram para a entrega de alguma documentação técnica já elaborada, tomar conhecimento das funcionalidades e objetivos gerais do SI e para o respetivo acompanhamento dos trabalhos. Fez-se a pesquisa bibliográfica para analisar o estado da arte e utilizaram-se

os conceitos que balizaram o trabalho, conforme se descreve nos subcapítulos que se seguem.

2.1. Estado da arte

Para a recolha de informação foram utilizadas diversas técnicas de pesquisa. O processo começou por efetuar uma pesquisa exploratória em todas as bases de dados disponíveis na EBSCO Industries, Inc por artigos relevantes para a temática desta dissertação. Após a seleção inicial dos artigos, fez-se uma pesquisa adicional nas referências bibliográficas neles mencionadas, de forma a aprofundar a investigação recorrendo às fontes primárias e alargar as fontes de informação utilizadas. Da mesma maneira, sempre que necessário incluíram-se monografias da área da segurança da informação e fizeram-se pesquisas adicionais na Internet e em sites relacionados com o tema, através de um motor de busca generalista para obter mais fontes e outros artigos de relevância.

Para a primeira fase da pesquisa, foram fatores de seleção os artigos terem o texto integral e a sua atualidade, sendo excluídos os artigos anteriores ao ano de 2014. Esta seleção da atualidade, deve-se sobretudo ao facto de se ter expandido a análise sobre as fontes bibliográficas neles mencionadas, o que amplia a antiguidade dos artigos incluídos no estudo e a obter indicações de orientação do estado da arte. O segundo fator de seleção, foi a pertinência dos artigos para o tema da dissertação. Para as fases subsequentes de recolha de informação, foram tomados em consideração a autoridade do domínio/autor, assim como a pertinência dos conteúdos para esta investigação. A primeira pesquisa foi realizada em português e inglês na EBSCO e obteve os seguintes resultados:

Termos de pesquisa	Resultados
Segurança da Informação AND Desenvolvimento de Software OR plataformas colaborativas OR protocolos criptográficos	86
Information security AND software development OR Collaborative platform OR cryptographic protocols	9186

Tabela 1 - Resultados recuperados na pesquisa inicial

Da recuperação de resultados em português, não foram encontrados artigos que satisfizessem os critérios de seleção inicial pois não versavam sobre o tema da

segurança, pelo que não foram incluídos neste estudo. Da recuperação de resultados no idioma inglês, foram selecionados os artigos que cumpriam com os critérios de seleção originais e após breve análise dos títulos, *abstracts* e textos foram selecionados os mais relevantes para incluir nesta investigação. Procedeu-se também à análise das fontes bibliográficas mencionadas nesses artigos, de forma a ampliar a base de pesquisa. Da mesma forma, sempre que foi necessário aprofundar um tema pertinente para o estudo, foram efetuadas pesquisas adicionais com palavras-chave mais específicas e direcionadas para o tema a desenvolver como por exemplo, “blockchain”, “secure protocols”, “Cryptographic protocol”, etc. Para a lista de referências bibliográficas mencionadas neste relatório, opta-se por apenas referir as obras e artigos que são explicitamente citadas ao longo do texto.

2.2. Conceitos e convenções

Devido à maioria das fontes de informação utilizadas neste trabalho estarem redigidas no idioma inglês, optou-se por, sempre que é apresentado um conceito ou um termo nesse idioma, utilizar a sua sigla quando esta existir, seguindo-se o termo por extenso e depois por uma tradução livre para o idioma português se esta fizer sentido. Em referências posteriores ao mesmo conceito, opta-se por utilizar a sigla em inglês ou o nome por extenso em português.

Prestado este esclarecimento, nos seguintes subcapítulos abordam-se conceitos e princípios relacionados com a infosec (*Information Security*) ou segurança da informação, assim como as tecnologias e protocolos associados ao tema que suportam a fundamentação teórica deste trabalho, relativas às diferentes fases de desenvolvimento do projeto U.OPENLAB.

Não obstante a segurança da informação compreender um ISMS (I*nformation Security Management System*) ou, em português, Sistema de Gestão de Segurança da Informação, completo e mais abrangente, dadas as necessidades específicas deste projeto e as características da organização – Universidade do Porto (U.Porto) – que o acolhe, que já tem em conta muitas das necessidades de segurança da informação, a abordagem foca-se mais sobre a fase de desenvolvimento de software seguro, quais são as vantagens de optar por esta metodologia e sobre as tecnologias que podem ser incorporadas para a sua realização.

Será importante, antes de mais, apresentar e discutir alguns dos conceitos abordados, começando pelo de Segurança da informação que, no seu sentido lato, envolve todas as questões relacionadas com a proteção de um sistema de informação, seja este computadorizado ou não, desde aspetos físicos a lógicos.

Segundo a norma ISO/IEC 27000:2018 a "*segurança da informação garante a confidencialidade, disponibilidade e integridade da informação. A segurança da informação envolve a aplicação e a gestão dos controlos apropriados que envolvem a consideração de uma ampla variedade de ameaças, com o objetivo de assegurar o sucesso e a continuidade sustentáveis dos negócios e minimizar as consequências dos incidentes de segurança da informação*" (ISO/IEC 27000, 2018).

Já a NIST na publicação especial SP 800-12 Rev.1, define-a como a "*proteção da informação e dos sistemas de informação contra o acesso não autorizado, uso, divulgação, interrupção, modificação ou destruição, a fim de garantir a confidencialidade, integridade e disponibilidade da informação"* (NIST SP 800-12, 2017).

Da análise da literatura ressalta o foco na tríade CID, Confidencialidade, Integridade e a Disponibilidade, ou, em inglês, CIA (Confidentiality, Integrity, Availability). Segundo a NIST (citado por Stallings & Brown, 2015, p. 1.1):

> O sentido de segurança da informação aqui utilizado está relacionado com sistemas computadorizados e pode ser definido como "*a proteção fornecida a um sistema de informação automatizado para atingir os objetivos aplicáveis de preservar a integridade, a disponibilidade e a confidencialidade dos recursos do sistema de informação (incluindo hardware, software, firmware, informações / dados e telecomunicações)*".

Para Stallings e Brown (2015, p. 1.1) o principal objetivo da segurança da informação é, assim, responder a três perguntas básicas:

1. *Quais são os ativos que necessitamos proteger?*
2. *Quais são as ameaças que existem para esses ativos?*
3. *O que podemos fazer para combater essas ameaças?*

O que está em causa e considerar que uma vulnerabilidade decorre então das ameaças e/ou ataques a que os ativos informacionais de um sistema estão expostos e dos riscos para todo o SI que advêm da correta exploração dessa brecha de segurança.

Independentemente da causa ou razão que a gera, isto é, se é uma ameaça, um ataque interno ou externo, uma vulnerabilidade é uma fragilidade no SI que deve antes de mais ser identificada e o risco relacionado ser mitigado, de forma a que se possa atingir o principal objetivo da segurança da informação.

Para se responder positivamente é necessário ter em conta outros tantos conceitos fundamentais da segurança da informação, como são a confidencialidade, a integridade e a disponibilidade. Todos estes conceitos e controlos serão discutidos no desenvolvimento desta dissertação dentro do contexto e significado que têm para a segurança da informação, assim como as propriedades e atributos associados a cada um deles como são os casos da autenticidade, da autorização, da rastreabilidade, do não-repúdio, de questões de privacidade e de direitos autorais.
Para alcançar estes objetivos, utiliza-se o novo paradigma *Secure by Design*, significando isto que a preocupação com a segurança entra na equação antes mesmo da implementação e desenvolvimento da solução, analisa-se se este tipo de abordagem traz alguma vantagem, ou não, para a qualidade do produto/serviço e para as organizações que os implementam e utilizam. Da mesma forma, analisam-se diferentes *frameworks* utilizadas pela indústria e as tecnologias e controles utilizados para alcançar o objetivo de tornar o *software* mais seguro.

2.3. Abordagem metodológica

O presente estudo assume a inclusão de uma abordagem da segurança da informação no início do ciclo de vida do desenvolvimento de *software* e adota o novo paradigma de *Secure by Design*, traduzindo-se isto na utilização de metodologias SSDLC (*Secure Software Development Life Cycle*) integrando as diferentes fases de desenho, desenvolvimento e implementação numa *framework* de segurança que permita entregar ao utilizador uma plataforma digital com *software* seguro.

Partindo de um estudo de caso, analisam-se as ameaças e vulnerabilidades, seguindo-se o desenho e aplicação dos casos de mau uso e abuso identificados, a especificação de requisitos de segurança e a análise de vulnerabilidades e matriz de risco, a proposta da arquitetura e tecnologias seguras, assim como se propõem testes de segurança para as fases posteriores do desenvolvimento da solução.

2.4. Novo Paradigma de *Secure by Design*

Embora o novo paradigma de *Secure by Design* seja mais amplo, estendendo-se às mais variadas áreas do conhecimento humano, no que diz respeito à engenharia de *software*, implica que a segurança é uma preocupação central do desenho da solução, é tomada em conta desde o início e que problemas com erros, falhas, vulnerabilidades e comportamentos maliciosos existem e acontecem. Por este motivo, quando se propõe uma solução, são levados em conta os princípios fundamentais do software seguro enunciados pelos autores Viega e McGraw (2006, pp 93-112):

1. *"Proteção do elo mais fraco*
   - *A segurança defensiva do sistema deve ser vista como uma corrente interligada, os atacantes procurarão por brechas no elo mais fraco dessa cadeia.*

2. *Defesa em profundidade*
   - *Gerir o risco defensivamente em todas as camadas e componentes, de tal forma que se um deles falhar, o outro tenha uma chance justa de poder funcionar.*

3. *Falhar com segurança*
   - *Lidar com falhas e erros corretamente e com segurança. O incorreto manuseio de falhas é frequentemente causa de quebras de segurança em termos de confiabilidade.*

4. *Mínimo privilégio*
   - *Executar o software com os privilégios mínimos necessários para reduzir o possível impacto de um ataque.*

5. *Compartimentar acessos e áreas*
   - *Tentar limitar os danos de ataques compartimentando as áreas utilizando VPNs, contentores, firewalls…*

6. *Manter a simplicidade Keep It Simple (KISS)*

- *Quanto mais complexo for um sistema, mais difícil será a sua manutenção e maiores serão os problemas para a segurança.*

7. *Promover a Privacidade*
   - *Proteger dados pessoais e do sistema é extremamente importante. A exposição de informação sensível é causa de muitos ataques e de quebra de confiança.*

8. *Esconder segredos é difícil*
   - *Mesmo os melhores algoritmos de segurança são quebrados e a confiança no secretismo é origem de muitas falhas de segurança.*

9. *Confiar desconfiando*
   - *A segurança é baseada na confiança de uma fonte, mas mesmo que a fonte seja fidedigna, nunca se deve confiar plenamente em nada nem ninguém e deve-se sempre tomar as devidas contramedidas para manter o software mais seguro.*

10. *Usar os recursos da comunidade*
    - *Se algo está bem concebido e já foi escrutinado pela comunidade, provavelmente é uma boa fonte de segurança.*

Observando estes 10 princípios do software seguro, todas as futuras opções tomadas para este projeto devem contribuir para que a solução proposta cumpra com os requisitos e as normas de segurança que a plataforma necessita.

2.5. CIA Triad Model

Nenhum SI elaborado com preocupações de segurança pode ser concebido sem levar em conta o modelo CIA (*Confidentiality, Integrity and Availability*) significando que a confidencialidade, a integridade e a disponibilidade são conceitos cruciais no que diz respeito à infosec.

Fig. 1 - Modelo CIA

Fonte: Sage Data Security - https://www.tylercybersecurity.com/blog/fundamental-objectives-of-information-security-the-cia-triad

O modelo CIA orienta a elaboração de políticas no domínio da segurança da informação e tem por objetivo unificar e interligar três conceitos fundamentais para a segurança num sistema de informação: confidencialidade, integridade e disponibilidade. De acordo com a Federal Information Security Management Act *(como citado em* Metivier, 2017), *a relação entre a segurança da informação e o modelo CIA envolvem:*

a) *Integridade, que significa proteção contra modificação imprópria ou destruição da informação e inclui a garantia de não-repúdio da informação, exatidão e autenticidade;*

b) *Confidencialidade, o que significa garantir restrições ao acesso e divulgação autorizados à informação, incluindo meios para proteger a informação privada, a privacidade* [diferente de confidencialidade] *e propriedade intelectual; e*

c) *Disponibilidade, o que significa assegurar a utilização confiável e o acesso à informação em tempo útil.*

Sendo que estes são os três principais objetivos do modelo, existem métodos ou características que lhes estão associados. Para alguns autores, estas características vão sendo incrementadas à medida das necessidades: "*As características, em número cada vez maior, com foco especial no controle, são as seguintes: zona privada, confidencialidade, disponibilidade, não repúdio, posse, prestação de contas, autenticidade, autenticação e auditoria*". (Moghaddasi, Sajjadi, & Kamkarhaghighi,

2016) Para a maioria dos autores (profissionais da área e académicos) são considerados os métodos dos três A para atingir os objetivos principais do modelo. Este triplo A significa métodos de Authentication (Autenticação), Authorization (Autorização) e Accounting (rastreabilidade), podendo ser definidos, segundo Nweke (2017) da seguinte forma:

> "*O primeiro A refere-se à autenticação, que é o processo de provar que uma pessoa é quem diz ser. Quando uma pessoa afirma ser alguém, isso chama-se identificação; mas quando essa pessoa prova ser quem diz, isso é autenticação.*
>
> *O segundo A no modelo AAA é autorização. Autorização significa fornecer o nível correto de acesso que um utilizador deve ter com base em suas credenciais. Isto está vinculado ao princípio do menor privilégio, que afirma que utilizadores, dispositivos, programas e processos devem receber apenas a permissão suficiente para executar as funções necessárias.*
>
> *O último A no modelo AAA é rastreabilidade, que acompanha o que os utilizadores fazem enquanto estão ligados a um sistema*".

Independentemente dos métodos ou características, este é um modelo basilar da segurança da informação e esta não se conseguirá obter sem a sua consideração, resta pois encontrar tecnologias, protocolos e metodologias para conseguir ajustar à proposta da solução, os princípios, os conceitos e o modelo atrás apresentados.

2.6. PKI (Public key infrastructure)

Embora não seja o tema desta dissertação, não é muito provável falar em segurança informática sem falar de criptografia. No caso das plataformas web, é quase impossível tratar de temas relacionados com a segurança sem falar de infraestruturas de chave pública que permitem muitas das funcionalidades que são caras ao setor da segurança, como por exemplo a autenticação, a autorização e a rastreabilidade. De uma forma geral uma infraestrutura de chave pública (PKI) pode ser definida como "... *o conjunto de hardware, software, pessoas, políticas e procedimentos necessários para criar, gerenciar, armazenar, distribuir e revogar certificados de chave pública ...*". (Kiran, Lareau, & Lloyd, 2002) Sem aprofundar muito as questões relacionadas com a criptografia de chave pública e a simétrica, as infraestruturas de chave pública permitem

utilizar estes dois métodos com implementações para usar protocolos de comunicação seguros, autenticação por serviços de SSO, assinar documentos e muitas outras características que imprimem ao SI a confiança e transparência que é necessária para um projeto desta natureza.

2.7. A tecnologia Blockchain

Por sua vez a tecnologia Blockchain teve origem no setor financeiro com as bitcoins, ou criptomoedas, mas as suas características permitiram expandir o seu uso por outras áreas, desde a saúde (Lavina, 2018) até às redes sociais. A utilização do blockchain apresenta algumas características apetecíveis e promissoras no que diz respeito à segurança da informação nas três vertentes do modelo CIA.
"*A tecnologia fornece uma maneira de registar transações ou qualquer outra interação digital de forma segura, transparente, altamente resistente a interrupções, auditável e eficiente*" (Piscini, Dalton, & Kehoe, 2017). Além destas características, a tecnologia permite passar aos objetos digitais propriedades que são essenciais quer para a transparência, quer para a confiança dos intervenientes como por exemplo a transferência de propriedade de forma segura e imutável.

"*O Blockchain fornece um instrumento para a criação de valor digital que pode ser transferido, trocado e negociado com proteção contra duplicação ilegal não controlada e falsificações*" (Aste & Tasca, 2017). Devido a estas propriedades de registos imutáveis e às características do próprio projeto U.OPENLAB, a tecnologia terá diversas aplicações que são bastante promissoras para os artefactos digitais gerados e administrados na plataforma.

2.8. Security Touch-points

Apesar dos 7 pontos de contacto do software seguro abrangerem todo o ciclo de vida de um projeto de software, desde a sua idealização até à sua descontinuação, com tarefas específicas para cada fase e o projeto U.OPENLAB ainda se encontrar na sua fase inicial, não é muito viável propor uma solução segura que não abarque desde logo estes pontos de contacto sugeridos por Gary McGraw em 1995, com posteriores

atualizações, o que implica uma série de boas práticas, assim como a inclusão das perspetivas de segurança ao longo de todo o ciclo de desenvolvimento de software.

Os sete pontos que a seguir se enunciam (McGraw, 2005) foram ordenados pela eficácia e quantidade de erros e falhas que podem detetar e mitigar durante as diferentes fases de um projeto, não refletindo a sequência de passos ou inclusão no mesmo:

1. *Análise estática e revisão de código fonte para detetar bugs e vulnerabilidades conhecidas;*

2. *Análise de risco para documentar tipos e padrões de ataque durante a fase de arquitetura do sistema;*

3. *Testes de penetração para testar as aplicações num ambiente real ou simulado;*

4. *Testes de qualidade que devem abranger os testes de funcionalidade padrão e testes de segurança baseados em padrões de ataque;*

5. *Construção de casos de abuso que permitam identificar potenciais maus usos do sistema, assim como as respostas deste;*

6. *Especificação de requisitos de segurança que devem ser desenhados a par dos restantes requisitos de engenharia e determinar as respostas do sistema;*

7. *Monitorização do software contra padrões de ataque de forma a desenvolver sistemas mais seguros.*

Tendo em conta que estes pontos de contacto têm tarefas específicas relacionadas com diferentes fases de um projeto, não são uma sequência linear que resolva todos os problemas quando na respetiva fase são executadas as tarefas indicadas para esse ponto. Quando é detetado qualquer problema, devem-se revisitar e atualizar as especificações de cada tarefa para que o resultado atingido seja o mais seguro possível. Neste sentido, todo o projeto U.OPENLAB e a solução proposta devem seguir

metodologias de desenvolvimento e utilizar *frameworks* que permitam enquadrar todos os princípios, conceitos e tecnologias abordados desde o início até ao momento.

2.9. Metodologias SSDLC

Qualquer projeto de software seja ele uma aplicação de *desktop*, uma plataforma digital, um Sistema Operativo (SO), uma aplicação Web, etc. deveria incluir um SDLC (*Software Development Life Cycle*) que em português pode ser traduzido por Ciclo de Vida de Desenvolvimento de Software. Este processo sistemático de desenvolvimento de software integra várias fases de trabalho e tem por principal objetivo obter soluções com a melhor qualidade a menor custo num menor espaço de tempo, visando a melhoria da qualidade, eficácia e eficiência das soluções de software. Dado os seus benefícios, a indústria do software tem utilizado diversas metodologias conhecidas, entre as quais se podem apontar a Agile, a Lean, a Waterfall, a Iterative, a Spiral e a DevOps. Independentemente de qual metodologia é a melhor, pois cada uma terá os seus prós e contras, estando a escolha de uma delas mais dependente das características da organização e da equipa ou projeto a realizar, para a segurança da informação, esta discussão está dependente da inclusão dos princípios da Infosec e, portanto, qualquer uma delas poderá ser utilizada desde que sejam bem implementados os seus próprios princípios.

Nestas metodologias de desenvolvimento costumam-se utilizar técnicas de modelagem que especificam processos de software e que podem ser definidos como o conjunto de passos necessários para realizar a ação determinada. Desta forma, um processo de software seguro pode ser definido como o conjunto de atividades realizadas para desenvolver, manter e fornecer uma solução de software segura. De notar que estas atividades não têm de ser necessariamente sequenciais, podendo ocorrer de forma concorrencial ou iterativa.

> "*Quando se modelam processos, especificam-se as características dos processos e fornecem-se um conjunto de práticas recomendadas que podem ser utilizadas para a melhoria e avaliação dos mesmos"*. (Davis, 2006)

Assim, a modelagem não obriga a seguir um algoritmo determinado, mas orienta de um ponto de vista de alto nível, as características que um processo de software deve ter, deixando a escrita do código independente deste processo de modelagem.

Só por si, a escolha de uma metodologia não garante que o software será seguro, mas seguir uma metodologia contribui em muito para a avaliação e melhoria da segurança da solução proposta.

Mesmo que seja consensual que nenhum sistema é totalmente seguro e o nível de segurança da informação que este consegue oferecer é uma equação que deve equilibrar a própria segurança e a usabilidade do mesmo, o sistema deverá ser tão seguro quanto as suas necessidades informacionais e a sua utilização for viável pelos seus utilizadores. Isto é, a segurança não deverá exceder, nem ficar aquém do nível necessário a ponto de tornar o sistema inviável, seja por falta de transparência, seja por excesso ou omissão de controles.

Como uma das especificações do projeto U.OPENLAB é ser uma plataforma *"Secure by Design"*, opta-se por seguir a metodologia/processo SSDLC (*Secure Software Development Life Cycle*) que permite incluir as questões relacionadas com a Infosec desde o começo até ao final do projeto e entregar uma solução segura, de qualidade com o melhor custo no menor espaço de tempo e que cumpra com as especificações funcionais do projeto.

Um SSDLC é uma metodologia que também contempla os princípios de segurança no ciclo de desenvolvimento do software, garantindo que as tarefas relacionadas com a segurança enunciadas nos sete pontos de contacto são implementadas corretamente durante os diferentes estágios do ciclo de desenvolvimento do software (Planeamento, análise, desenho, construção, testes e manutenção).

Independentemente da metodologia base que seja utilizada, alguns dos benefícios de utilizar um SSDLC são (Mougoue, 2016):

- *Software mais seguro e robusto, já que a segurança é uma preocupação desde o início do processo;*

- *Consciencialização dos stakeholders, sejam estes os utilizadores, desenvolvedores ou outra equipa de gestão, contribuindo para a transparência e segurança do processo;*

- *Deteção precoce de vulnerabilidades no sistema, contribuindo para a qualidade;*

- *Redução de custos devido à deteção e resolução de problemas;*

- *Redução de riscos intrínsecos à organização.*

Explicados os benefícios de seguir um SSDLC e estando já definidos os restantes conceitos, princípios e metodologias importantes para a Infosec, falta ainda enquadrar tudo num *framework* de segurança que seja suficientemente flexível para atender às necessidades de segurança do projeto em estudo e encaixe na cultura organizacional que o acolha.

2.10. *Frameworks* de Segurança

Ainda que a segurança da informação abranja mais áreas do que o desenvolvimento de software e deva estar presente do início ao fim de um projeto, ou que esta dissertação se foque mais na fase da arquitetura; especificação de requisitos; análise de vulnerabilidades e risco de uma plataforma web, não se pode ignorar que a segurança deve ser considerada como uma corrente com vários elos e, portanto, deverá ser incluída num quadro de projeto mais amplo, mesmo que o elo aqui em destaque esteja relacionado com uma fase específica do desenvolvimento de software. Desta forma, a solução a propor deverá estar incluída num quadro maior onde se possa encaixar todo o projeto.

Segundo Davis (2005) "… *para o desenvolvimento de soluções a indústria de software tem utilizado diversos modelos que permitem introduzir questões de segurança no SDLC. Desde os baseados nos Capability Maturity Models (CMM), passando por outras abordagens como as propostas por Microsoft's Trustworthy Computing Security Development Lifecycle ou Team Software Process for Secure Software Development (TSP) ou abordagens tipo Correctness by Construction ou Agile Methods ou ainda The Common Criteria, Software Assurance Maturity Model e Software Security Framework".*

Uma vez que fica fora do âmbito desta dissertação a discussão dos méritos ou benefícios de qual *framework* é melhor ou traz mais vantagens, pois cada um dos modelos terá os seus pontos fortes e fracos, a escolha de um modelo prende-se mais com a filosofia do próprio projeto de desenvolver uma plataforma digital utilizando

ferramentas *open source* que seja eficaz e atualizado e que possa proporcionar outras ferramentas para as diferentes tarefas do desenvolvimento seguro de uma forma consistente e coerente.

2.10.1. Software Assurance Maturity Model (SAMM)

O SAMM (*Open Web Application Security Project*, 2017) é um *framework* de segurança mantido pela OWASP *(Open Web Application Security* Project) desenvolvido em código aberto, com a sua primeira versão oficial lançada em 2009 e a última (Versão 1.5) data de abril de 2017, estando em desenvolvimento e em fase beta a versão 2.0, sendo então um *framework* já com dez anos de existência e que continua em atualização constante com introdução de melhorias de segurança que permitem mitigar riscos derivados de vulnerabilidades mais recentes.

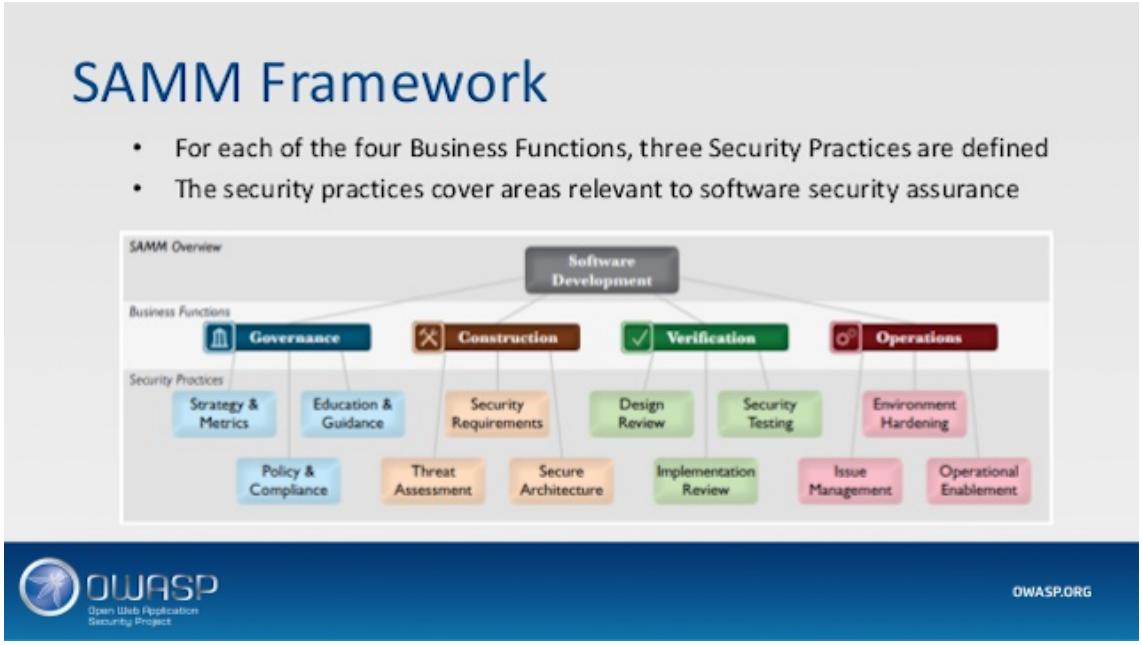


Fig. 2 - SAMM Framework

Fonte: OWASP - https://www.owasp.org/index.php/Main_Page

Este *framework* destina-se a ajudar organizações de qualquer dimensão (pequenas, médias ou grandes) a implementar programas de segurança, designadamente para o desenvolvimento de software seguro. Permite ajustes a um só projeto, ou estender o *framework* a toda a organização de acordo com as necessidades individuais de cada

projeto a desenvolver. Indica ainda o nível de segurança necessário para cada projeto de acordo com o risco, nivelando em três estágios diferentes as necessidades e níveis de conformidade com as normas de segurança.

Na visão geral do *framework representado na Figura 2*, observam-se as quatro áreas nucleares do desenvolvimento de software (governança, construção, verificação e operação) e as tarefas associadas a cada uma delas. De notar que, apesar de constituírem áreas distintas, não têm de ser necessariamente sequenciais, podem ser iterativas ou concorrenciais. Isto é, não é necessário terminar todas as tarefas de uma área nuclear para começar tarefas de outra ou, no caso de se verificar que alguma coisa não está em conformidade, não se possa e deva rever e retificar tarefas relativas a outra área, mesmo que o projeto já esteja noutro estágio.

Apesar de se apresentar aqui o *framework* completo, devido ao estágio em que se encontra o projeto em estudo, nesta dissertação o grosso do trabalho está orientado à área de construção do software focando portanto as tarefas de requisitos de segurança, avaliação de ameaças e desenho de arquitetura segura.

3. Estudo de caso

Apresentados os principais fundamentos teórico-metodológicos que sustentam esta dissertação e escolhidos as metodologias e modelos a seguir para propor uma solução para o desenvolvimento de uma plataforma digital segura, impõe-se, agora, aplicar todo o conhecimento abordado ao caso em estudo e às suas especificidades.

3.1. A plataforma U.OPENLAB na U.Porto

O projeto U.OPENLAB surge como uma forma de concretizar as necessidades e os conceitos identificados no artigo ASH.NET - ATLANTIC SCIENCE HERITAGE NETWORK INITIATIVE (Pinto, Ribeiro, Matos, & Medina, 2016) com uma abordagem metodológica que está descrita mais pormenorizadamente no artigo U.OPENLAB METHODOLOGY: A CONCEPTUAL MODEL AND FLOWCHART FOR THE DYNAMIC CO-PRODUCTION AND (RE)USE OF DIGITAL CONTENTS (Pinto, Medina, Matos, & Fontes, 2016), com o objetivo de desenvolver uma plataforma digital de licitações da

Universidade do Porto que funcionará em regime de coautoria, destinada a interligar as necessidades de conteúdos e produtos digitais da sociedade (instituições e outras organizações) com as competências e conhecimento da comunidade académica (estudantes, docentes e investigadores), particularmente aquando da realização de trabalhos práticos, assim como desenvolver as capacidades de cocriação e reuso de conteúdos/produtos por parte dos seus membros. Tem, por isso, por objetivos adicionais os de manter e preservar a longo prazo a informação/objetos digitais armazenados na plataforma recorrendo a software OpenSource. Configuram-se, assim, quatro vertentes principais de prestação de serviço:

1. Preservar o legado histórico, cultural, científico e identitário da U.Porto;

2. Incentivar a criação, cocriação, reutilização e melhoria de conteúdos por parte dos integrantes da comunidade académica;

3. Promover e divulgar as competências destes membros;

4. Promover e disseminar a ciência e o conhecimento com objetivos turísticos, culturais e científicos no seio da comunidade em geral.

A plataforma está pensada para ser operacionalizada em duas fases distintas:

- A primeira de recolha, incorporação e agregação da informação e metainformação já existente nos diversos repositórios da U.Porto;

- a segunda, de cocriação do material existente ou criação de novos conteúdos.

Em qualquer uma das fases será necessário manter os registos de todas as transações de todos os participantes dos projetos, de forma a assegurar os devidos créditos para os criadores da informação. Além das questões de confidencialidade, integridade e rastreabilidade que os artefatos gerados e geridos pela plataforma precisam de assegurar, é extremamente importante "*manter um registo que preserve a informação ao longo do tempo, já que muitos artefatos poderão ser trabalhados por diversas equipas em diferentes anos e por diferentes coautores"*. (Pinto, Medina, Matos, & Fontes, 2016).

Devido à dimensão, natureza, e objetivos da própria plataforma, assim como para proteção dos interesses e expectativas de todos os participantes, é extremamente importante que este sistema de informação seja totalmente transparente e garanta os níveis adequados de segurança e usabilidade nos serviços que prestará aos seus utilizadores e à instituição que o alberga.

3.2. Requisitos de Segurança U.OPENLAB

Para desenvolver os requisitos de segurança do projeto e concretizar o principal objetivo desta dissertação, a proposta de uma arquitetura segura para plataformas digitais, o primeiro passo consistiu em identificar as fragilidades que o próprio SI apresenta de forma a poder mitigar os riscos. Para este efeito, apresentam-se em seguida as tarefas realizadas relativas ao estágio em que se encontra o projeto. Começando pela análise da estrutura de funcionamento da plataforma, seguindo-se a tendência de ataques a aplicações Web, passando aos casos de abuso e especificação de requisitos de segurança e terminando com a análise das vulnerabilidades identificadas e o cálculo do risco.

3.2.1. Arquitetura U.OPENLAB

A estrutura do SI do U.OPENLAB foi pensada para ser uma plataforma digital a funcionar na web com um esquema cliente/servidor com armazenamento na nuvem que permita gerir o espaço necessário de uma forma mais eficiente.

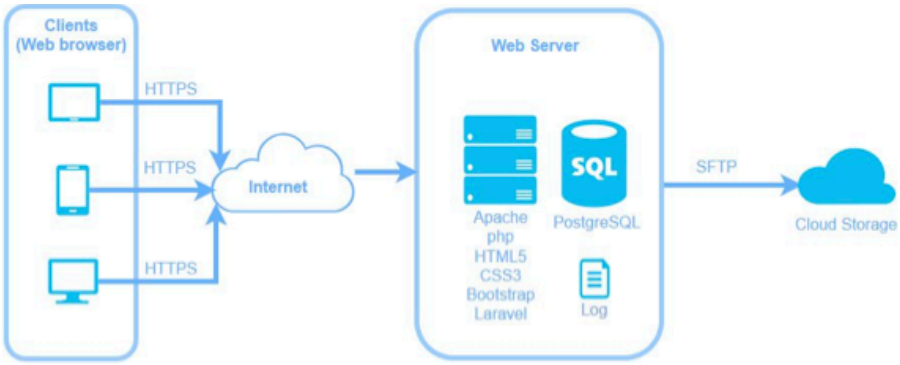


Fig. 3 - Arquitetura original U.OPENLAB

Fonte: (Pinto, Medina, Matos, & Fontes, 2016)

Conforme se pode observar na figura 3 que representa uma abstração de alto nível de funcionamento do SI, para a InfoSec, existem três componentes principais: o cliente, o servidor e o armazenamento. Além destes, existem ainda os canais de comunicação entre eles e os serviços. Não obstante os integrantes da componente servidor poderem ser diferentes dos indicados na figura, esta visão geral do sistema representa a superfície de ataque sobre a qual se fará a análise de ameaças, se desenvolverão os requisitos de segurança e se proporá uma arquitetura que leve em conta a segurança da informação.

3.2.2. OWASP - Top 10 de Vulnerabilidades em aplicações web

Para iniciar os trabalhos, começam-se por levar em conta os ataques a serviços e aplicações Web que ocorreram no ano de 2018 segundo a OWASP, o que não oferece um nível de segurança ideal, mas permite cruzar a informação das tendências de ataque para começar a avaliar as maiores probabilidades de risco e delinear estratégias com a abstração que se está a utilizar nesta fase do projeto. Na tabela 2 podem observar-se, de uma forma simplificada, a classificação da frequência de ataques (OWASP, 2018) que foram identificados no ano anterior e incidiram sobre aplicações Web e o nível de risco que poderiam causar ao SI no caso de as vulnerabilidades serem exploradas., assim como as contramedidas que ajudam a mitigar esses riscos.

Pos	Vulnerabilidades	Risco	Contramedidas
1.	Injeção (SQL, PHP, LDAP, OS)	Médio/Alto	Sanitização de Inputs, Whitelisting no servidor, API's Seguras e Querys parametrizadas
2.	Quebra de autenticação	Médio/Alto	Autenticação Multi-fator, Isolamento de sessão, Tempo limite para sessões inativas e Cookies seguros
3.	Exposição de dados sensíveis	Médio	Cifrar todos os dados, Usar protocolos e algoritmos seguros e Desabilitar Cache de dados sensíveis
4.	XML External Entities (XXE)	Médio	Evitar serialização de dados sensíveis, Whitelisting no servidor para carregamento de XML, WAF para bloqueio de XXE e revisão de código
5.	Quebra de controle de acesso	Alto	Invalidação de Cookies e Tokens após logout, Forçar Login/logout após alterações de credenciais, restrições no servidor e restrições a recursos baseado em perfis
6.	Erros de segurança	Alto	Assegurar que não se utilizam valores padrão no hardware e software, Instalar apenas componentes necessários e Rever configurações de segurança periodicamente
7.	Cross Site Scripting (XSS)	Médio/Alto	Codificação do Output e escapar caracteres não-confiáveis e utilização de CSP
8.	Serialização insegura	Médio/Alto	Cifrar dados serializados e execução com menos privilégios para deserializadores
9.	Componentes com vulnerabilidades conhecidas	Médio/Alto	Atualizações periódicas e últimos CVE e mitigações
10.	Monitorização e logging insuficientes	Médio	Monitorizar e analisar tráfego e eventos 24x7 e prática de resposta a incidentes eficaz

Tabela 2 - Top 10 Vulnerabilidades em aplicações Web 2018 da OWASP

3.2.3. Casos de Abuso

Embora semanticamente se possa diferenciar entre mau-uso e abuso, seja pela gravidade ou intencionalidade dos danos provocados, sejam estes limitados no tempo ou de carácter mais permanente, ambas as falhas representam uma subversão da segurança da informação. Será pois mais importante identificar as situações em que podem ocorrer vulnerabilidades que advêm das ameaças, ataques ou das contramedidas com a finalidade de fornecer aos desenvolvedores uma forma ágil de poder mitigar os riscos, do que estar neste momento a categorizar as próprias vulnerabilidades pela sua taxinomia(Abuso de API, autenticação, autorização, disponibilidade, permissão de código, qualidade de código, etc.) que podem ser consultadas em fontes que abordam o assunto em mais profundidade (OWASP, 2016) e cuja classificação está mais relacionada com outras atividades da infosec como por exemplo a análise estática, a revisão de código, os testes de qualidade e segurança, os testes de penetração e as auditorias de segurança.

Esta análise tem como intuito identificar onde e em que contexto existem brechas para especificar os requisitos de segurança. Todas as situações estão referenciadas com a respetiva especificação (RS) e numeração o que permitirá também à equipa de desenvolvimento perceber em que contexto a vulnerabilidade relacionada com o requisito foi identificada e como e onde implementar o respetivo controle.

Para esta análise foram utilizados o modelo da arquitetura do protótipo do SI, sobre a qual se faz uma análise geral das fragilidades inerentes ao modelo de funcionamento e o documento do relatório de especificações (U.OPENLAB, 2016) que descreve os atores, as personas, as *User Stories*, o *workflow* e as tabelas de permissões da plataforma, sobre o qual se aproveitou a seção de *User Stories* que descrevem o funcionamento do sistema e o fluxo da informação para cada persona.

a) Fragilidades do modelo de funcionamento

Importa antes de mais, verificar as fragilidades de segurança inerentes ao modelo de funcionamento do SI, baseado numa relação cliente-servidor através de ligações efetuadas em redes públicas, já que estas vulnerabilidades são de carácter geral afetando todo o sistema, os seus componentes e utilizadores.

Tendo em conta as necessidades do sistema sendo este uma plataforma em que a informação é gerada na modalidade de coautoria e os utilizadores/autores podem aceder à plataforma nos seus próprios equipamentos através de redes públicas (Internet) existem duas grandes vertentes que proporcionam uma grande área de ataque externa: as telecomunicações e os equipamentos cliente, estando estes últimos sujeitos aos gostos e necessidades dos próprios utilizadores.

No que concerne às telecomunicações é necessário evitar que terceiros capturem/alterem a informação trocada entre o cliente e o servidor RS1.1 com a finalidade de proteger dados sensíveis para o sistema ou para o utilizador. Por outro lado, é necessário garantir que se utilizam protocolos de comunicações seguros RS1.2 em que mesmo no caso de intercetação das comunicações, terceiros não consigam decifrar a informação em trânsito entre o cliente e o servidor, permitindo adicionar uma camada de confidencialidade para o serviço e para o utilizador.

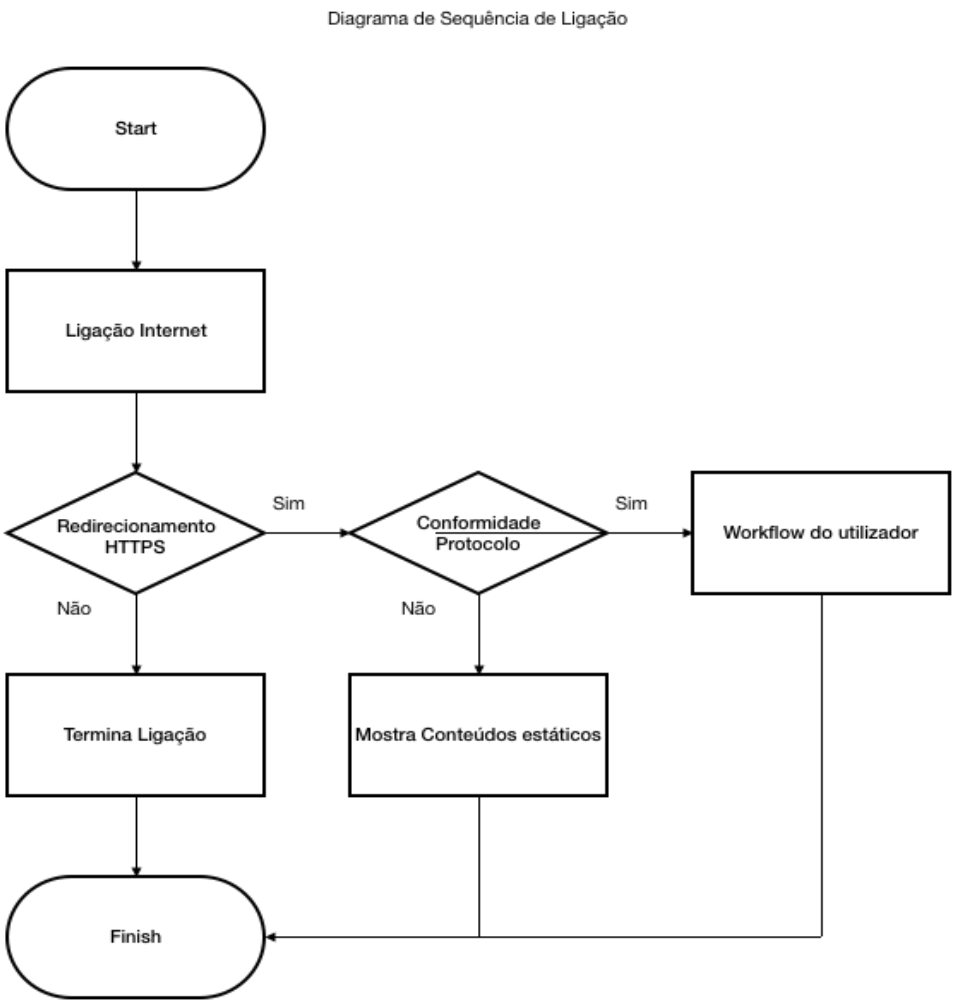


Fig. 4 - Diagrama de sequência de ligação

Na Figura 4, pode-se observar o esquema de ligação entre os clientes e o servidor, assim como as respostas do SI para estabelecer a ligação entre ambos. Para mitigar riscos de comunicações inseguras e evitar que informação sensível seja visualizada por terceiros, quando é feito um pedido de ligação ao Servidor que aloja a plataforma, o protocolo de comunicação deve ser redirecionado automaticamente para HTTPS. Em caso de não se conseguir estabelecer ligação segura, a ligação é terminada. No caso de se conseguir fazer a ligação HTTPS é efetuada uma verificação adicional para que a comunicação se faça com os protocolos TLS 1.1 ou TLS 1.2. No caso de se conseguir estabelecer um destes protocolos, o utilizador pode prosseguir com a visita e restantes tarefas proporcionadas pelo serviço. Caso não seja possível estabelecer um destes protocolos, apenas são mostradas páginas de conteúdo estático e informativo sobre a plataforma e requisitos de utilização. No que toca a comunicações, todos os pedidos efetuados pelo servidor a outros serviços (serviço de armazenamento, servidores de autenticação, bases de dados, etc.) devem também seguir os mesmos preceitos para que toda a informação em trânsito esteja protegida.

No lado cliente da equação, para além das verificações mínimas feitas no lado servidor pela aplicação sobre a compatibilidade, a conformidade com os padrões e protocolos estabelecidos para o correto funcionamento do SI e da consciencialização do utilizador, não se pode neste momento exigir maiores preocupações com a segurança, uma vez que o acesso será feito maioritariamente através do equipamento e tecnologias de preferência do utilizador, já que o desenvolvimento de aplicações móveis e desktop para todas as plataformas existentes tornaria o projeto inviável economicamente e quase impossível de manter. Portanto, para o lado cliente apenas se podem garantir comunicações seguras entre os dois lados do serviço com a verificação de compatibilidade com os requisitos mínimos de segurança.

Já no lado servidor da equação, em que a própria instituição tem muito mais controle do que sobre o equipamento e tecnologias dos clientes, devem-se estabelecer e observar certos princípios de segurança que, mesmo não estando diretamente relacionados com o desenvolvimento, funcionamento e manutenção da aplicação, dizem respeito a todo o ambiente em que a própria plataforma terá que funcionar.

Uma vez que este relatório está mais orientado à arquitetura lógica do SI, não serão abordadas questões de ordem física como por exemplo o controle de acesso aos locais onde estará instalada a máquina, a sua alimentação energética, equipamentos de redundância que permitam ter o SI sempre disponível, etc. que embora possam causar

diversos problemas em qualquer das três vertentes do modelo CIA são questões que terão de ser analisadas em mais detalhe pela instituição que alojará o serviço de acordo com as suas próprias políticas de segurança, os seus recursos e necessidades, pelo que qualquer abordagem aos aspetos físicos da segurança ficam algo fora do âmbito desta dissertação, principalmente pelo estágio em que o projeto se encontra neste momento. Resta então abordar as questões de ordem lógica do componente servidor do SI respeitantes ao ambiente que estão diretamente ligados à arquitetura de segurança da plataforma e das máquinas em que esta estará instalada, ainda que não estejam diretamente ligados ao desenvolvimento da plataforma do SI.

Começando pelo próprio sistema operativo da máquina que não deverá ter vulnerabilidades conhecidas e deve ser atualizado com os patches de segurança sempre que estes estejam disponíveis RS2.1 com a finalidade de evitar ataques que possam comprometer quer a confidencialidade, quer a integridade e/ou a disponibilidade do sistema, evitando vulnerabilidades que possam subverter qualquer uma destas vertentes. Da mesma forma, todos os componentes e *frameworks* de software necessários ao bom funcionamento da plataforma, devem ser alvo da mesma atenção. Será também desejável que todos estes componentes de software necessários, sempre que possível, não utilizem os valores padrão de configuração RS2.2, assim como também sigam o princípio do Menor Privilégio, atribuindo apenas as permissões necessárias ao correto funcionamento do SI, restringindo o acesso destes componentes a pastas e contas com privilégios especiais para o sistema operativo RS2.3 o que permite prevenir muitos dos ataques padronizados e restringir a propagação de danos no caso de um deles ser bem sucedido. Mesmo aceitando que a ofuscação não seja uma estratégia de confiança para a segurança da informação, não existe nenhum motivo para não ocultar mensagens que permitam a um adversário tomar conhecimento das versões do software e ganhar alguma vantagem quando planear um ataque, pelo que é desejável desativar todos os outputs RS1.3 que forneçam este tipo de informação a um atacante, como por exemplo a função phpinfo() da linguagem do PHP, ou diretrizes como o auto_index do Apache que permitiriam a um atacante listar os arquivos numa estrutura de pastas, mesmo àqueles que não deveria ter acesso, evitando desta forma, entre outras vulnerabilidades, as de “0 day”.

Continuando com princípios básicos de segurança, deve-se também observar o princípio de manter a simplicidade desativando no servidor todos os portos e serviços que não sejam necessários ao correto funcionamento da plataforma RS2.4 reduzindo-

se a superfície de ataque que um agente mal intencionado pode utilizar, ao mesmo tempo que se diminuem as tarefas necessárias com a manutenção de segurança.

Tendo em conta que será necessário efetuar tarefas de manutenção e atualização no equipamento recorrendo a ligações de rede, isto é, sem que o utilizador esteja na própria consola do servidor, será desejável utilizar métodos de restrição de acesso a contas especiais de administração a endereços IP internos ou por VPN RS2.5, o que dificultará o acesso a um pretenso atacante. Finalmente, será uma boa prática manter cópias de log de todo o sistema num servidor distinto RS2.6 o que permitirá manter sempre uma cópia fidedigna de todas as interações dos utilizadores e aplicações para efeitos de auditoria, pois ambos os logs podem ser comparados e para um atacante passar desapercebido, teria que invadir mais do que uma máquina.

Claro está que estes princípios de segurança não mitigam todos os riscos do sistema, mas em conjunto com os outros controles de segurança reduzirão muito a superfície e vetores de ataque além de dificultar o trabalho que um adversário terá para subverter a segurança do SI.

Apresentam-se de seguida os casos: 1) Proponente externo; 2) Proponente interno; 3) Gestor de plataforma (como Curador); 4) Desenvolvedor como Coordenador / Fornecedor do projeto terminado; 5) Desenvolvedor elemento da equipa / Investigador / Aluno / Professor.

b) Proponente Externo

"*Miguel Matos é o responsável de Marketing da Empresa ChocoBite, uma empresa portuense de 30 anos que se especializa no fabrico de porções pequenas de chocolate. A empresa sofreu recentemente uma grande redução no volume das vendas, devido a uma notícia imprecisa que apontava que a fonte do óleo de palma usada para produzir os chocolates poderia estar a por em risco orangotangos na Indonésia. Como responsável de Marketing, cabe ao Sr. Matos compreender como poderia limpar a imagem tornando-a mais ecológica e apelar novamente à demografia mais jovem. Apesar de não ser muito conhecedor de novas tecnologias, pretende que uma das estratégias seja a criação de uma aplicação para smartphones que permita utilizar o código dos chocolates para acumular pontos e com isso plantar árvores. Já comunicou diretamente com a Quercus sobre a disponibilidade de contribuir para este projeto e a organização mostrou interesse. No entanto, quer também promover o trabalho realizado por jovens pretendendo que esta aplicação seja realizada por estudantes universitários.*

É com isto em mente que concorre com o seu projeto para a plataforma de licitações da Universidade do Porto.

Visto que será a primeira interação da sua Empresa com a U. Porto como visitante da plataforma:

(a) necessita de esperar por aprovação para obter uma conta, fazendo o registo e autenticação na plataforma utilizando um dos serviços disponíveis, ou contactando o gabinete de inovação da U.Porto para pedir credenciais de acesso.

(b) necessita de esperar que o seu registo na plataforma seja aprovado por um administrador para que o seu perfil seja criado.

Após a obtenção da conta, o Sr. Matos faz login e tem acesso à sua homepage a partir da qual tem a opção de criar uma proposta para licitação. Escreve um título para a proposta, uma curta descrição, define que quer que a sua proposta seja apenas apresentada a estudantes, investigadores e professores autenticados e escolhe as áreas gerais da proposta (Aplicação Mobile, Ecologia) submete depois o formulário, aguardando a aprovação de um Decisor. O Decisor envia-lhe algumas correções para serem efetuadas na sua proposta. Depois do Sr. Matos efetuar as correções o Decisor coloca a proposta na plataforma, podendo ser vista e anunciada. Passado um certo período de tempo recebe duas propostas de licitação: de uma professora que pretende realizar o projeto no contexto da unidade curricular de Gestão de Espaços Exteriores da Licenciatura em Arquitetura Paisagística da Faculdade de Ciências e de um estudante do Mestrado Integrado em Engenharia Informática e Computação que pretende realizar como um projeto à parte da Faculdade.
O Sr. Matos considera ambas as propostas, vendo as competências que o perfil de cada equipa promete trazer ao projeto. Após essa análise, sente que as competências de gestão ambiental da equipa proposta pela professora já serão preenchidas graças ao apoio da Quercus. Assim, decide pela segunda opção. Após uma reunião presencial, o Sr. Matos envia à equipa os dados necessários para elaborar a aplicação. Na data acordada a equipa submete o projeto com um guia de utilização conforme o acordado. Satisfeito com o resultado Sr. Matos apresenta o projeto à Direção e põe em andamento o resto da estratégia para mudar a imagem da empresa".

Nesta User Storie que abarca todo o fluxo informacional do SI, podem-se encontrar diversas vulnerabilidades relacionadas com a segurança, desde logo as associadas com a informação do perfil, que abarcam questões de confidencialidade como a captura

de informação por terceiros e permissões de leitura e escrita que terão de ser atribuídas ao próprio e a outros ao longo de todo o processo da licitação até à criação e disponibilização do artefacto digital, passando por questões de autorização de utilização de dados pessoais. Existem ainda questões ligadas com a integridade e disponibilidade do sistema como por exemplo a manipulação direta de inputs, sem esquecer questões de rastreabilidade e autenticidade que são inerentes a todo o processo.

Considerando que alguns dos riscos relacionados com vulnerabilidades já foram abordados no caso das fragilidades gerais do sistema e, portanto, se consideram mitigados, como por exemplo as interceções de telecomunicações, observa-se que o fluxo normal será dividido em três fases: entrada como visitante, autenticação e/ou criação do perfil e só depois se entrará no *workflow* específico de cada utilizador. Pondo em prática o princípio do menor privilégio, o perfil de visitante apenas deverá poder aceder e ler a informação que estiver classificada como pública RS1.4. Ao mesmo tempo, uma vez que este visitante em particular vai criar um perfil, ou mesmo que não o queira fazer, terá que interagir com a plataforma fornecendo input's para serem processados, possam estes ser manipulados diretamente pelo utilizador, como por exemplo na barra de endereço ou valor de um campo de formulário, ou sejam resultado de uma manipulação do sistema como por exemplo dados de resposta de um servidor de autenticação, todos eles devem passar por um processo de sanitização que permita eliminar caracteres com significado especial para o SI RS2.7. Ainda tendo em conta o mantra da segurança que "Todos os input's são maliciosos", não significando isto que todos eles sejam mal-intencionados, o facto é que os input's que uma função recebe destinam-se a causar uma alteração de estado em algum lugar do sistema. Assim sendo, não faz muito sentido processar todos e qualquer input numa função sem o validar em primeira fase, pelo que todos devem ser verificados e validados antes de serem processados RS2.8. Continuando com input's fornecidos diretamente pelo utilizador, todas as interrogações SQL devem estar num formato que permita separar os dados da própria interrogação RS2.9 independentemente de quais privilégios o utilizador em causa possa ter para aceder às tabelas da base de dados, de forma a poder prevenir potenciais injeções SQL que possam causar mau funcionamento no SI.

Da mesma forma que todos os input's devem ser parametrizados e tratados, todos os output's devem ser objeto da mesma atenção por parte da aplicação. Assim sendo, para mitigar potenciais ataques de XSS entre outras vulnerabilidades conhecidas, todos os

output's, em especial os que confiam em input's do utilizador, devem escapar corretamente caracteres especiais RS2.10.

Continuando com o normal fluxo informacional, observe-se agora a fase de autenticação e/ou criação do perfil de utilizador da plataforma que é idêntica para todos os tipos de perfis de utilizador.

Sendo que quer para a criação, quer para a autenticação de um perfil de utilizador, será necessário processar e guardar dados pessoais, a plataforma deve estar em conformidade com o RGPD (Regulamento Geral de Proteção de Dados) informando o utilizador de quais são os seus direitos sobre os seus dados e o tipo de utilização que a plataforma deles fará. Nesse sentido, antes mesmo da criação de um perfil de utilizador, a plataforma deverá obter o consentimento informado por parte do utilizador do uso desses mesmos dados RS1.5.

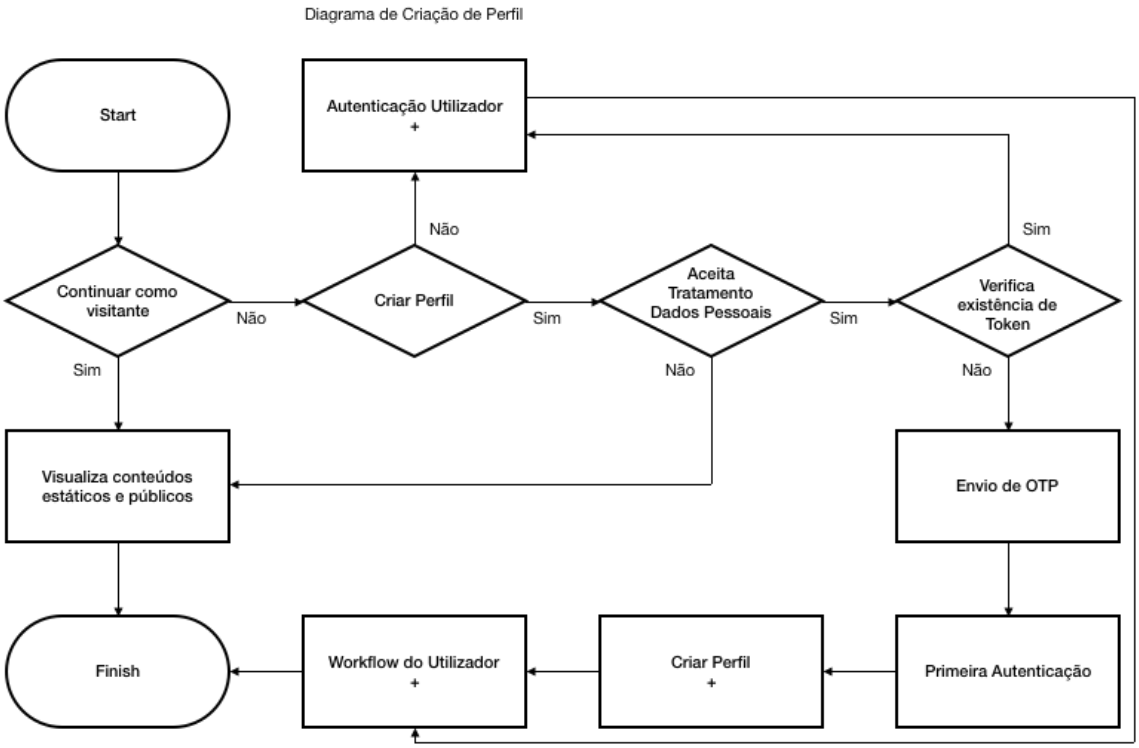


Fig. 5 - Diagrama de sequência de autenticação

No esquema da figura 5 pode-se observar o fluxo normal da informação em que o utilizador entra sempre com o perfil de visitante, pois, mesmo que previamente autenticado, todas as sessões devem ter um tempo limite de inatividade definido RS1.6 e todas elas devem ser cifradas individualmente com um algoritmo OTP (*One Time Password*) RS1.7, controles que juntamente com os *cookies* definidos para *http_only* e

http_secure RS1.8 ajudarão a mitigar ataques de “sequestro de sessão” e prevenir problemas associados como esquecimento de encerramento de sessão. Se o utilizador continuar como visitante, poderá visualizar os conteúdos públicos disponibilizados na plataforma. Caso o utilizador queira alterar o perfil de visitante para registado, passa para a zona de autenticação onde se pode autenticar se já tiver um perfil, ou criar um perfil de utilizador, onde pode visualizar os conteúdos personalizados para o seu tipo de perfil e de utilizador. No caso da criação, o utilizador terá primeiro de aceitar os termos de privacidade e tratamento de dados pessoais, caso contrário poderá continuar na plataforma com perfil de visitante, visualizando apenas os conteúdos públicos. Esta ação de aceitação de processamento de dados pessoais, contribui em muito para a transparência que uma plataforma deste tipo necessita e, portanto, para aumentar a confiança entre o utilizador e a instituição.

No caso da criação poderá utilizar um dos serviços de autenticação disponibilizados, através do qual a plataforma irá ler os dados necessários e verificar se já existe um *token* associado. Em caso afirmativo, redireciona o utilizador até ao processo de autenticação onde este se pode autenticar, evitando a duplicação de perfis de utilizador. No caso de não existir um *token* associado a um perfil na plataforma, é feita a validação dos dados e é enviado um código PIN (*Personal Identification Number*) aleatório para um dos contactos do utilizador, válido por um tempo limitado e de utilização única para que este se possa autenticar na plataforma. Este processo além de permitir validar os dados fornecidos pelo utilizador verificando se os contactos estão atualizados e funcionais, fornece uma forma de autenticação multi-fator na própria plataforma que não precisa de armazenar passwords, mitigando assim problemas de roubo de credenciais.

Só após a introdução do PIN é que a plataforma cria o perfil do utilizador, permitindo que este siga o seu *workflow*. Para o processo de autenticação, procede-se de forma similar à criação do perfil, variando apenas na seleção das ações do utilizador. Numa primeira fase, o utilizador seleciona o serviço de autenticação, isto é, se vai utilizar um dos servidores de autenticação ou se se vai autenticar na própria plataforma. Caso opte por um dos serviços de SSO (*Single Signing On*) é redirecionado para o serviço através de uma API segura, utiliza as credenciais do respetivo serviço e a plataforma verifica se a autenticação foi bem sucedida, verifica a existência de um *Token* associado a um perfil e em caso afirmativo, deixa o utilizador prosseguir o seu *workflow*, caso não exista um perfil associado ao *token*, redireciona o utilizador para o processo de criação de um perfil, conforme foi explicado no ponto anterior. Se o utilizador optar por fazer a

autenticação dentro da plataforma, apenas digita a sua identificação de utilizador, com o qual o sistema envia um PIN aleatório válido apenas para aquela sessão de autenticação e com um período de validade limitado. Apesar deste controle, para evitar ataques de força bruta em que podem ser enviados diversos códigos, após três tentativas falhadas o sistema deverá bloquear o código enviado e só permitir fazer a autenticação com novo código. De forma a prevenir ataques que possam ir fazendo várias tentativas, deverá também implementar um sistema retardador incremental que só permita fazer pedidos a certos períodos de tempo.

Com este sistema de autenticação/criação de um perfil em que se utiliza um serviço de autenticação externo ou autenticação multi-fator RS2.11, pretende-se para além das conveniências para a segurança como por exemplo prevenir problemas de roubo de credenciais, ou permitir a ao utilizador autenticar-se mesmo que o servidor de autenticação esteja indisponível, trazer funcionalidades que serão úteis a muitos perfis de utilizador entre as quais podemos enunciar o preenchimento automático de dados do perfil, o acesso por parte de utilizadores externos à instituição que alberga o serviço, a partilha de competências dos desenvolvedores noutras plataformas, etc.

Após a autenticação do utilizador, segue-se o fluxo de trabalho específico que neste caso vai compreender todo o processo de uma licitação desde a proposta até à disponibilização de um artefacto digital na plataforma. Pode-se dividir este processo em três grandes fases:

1. Proposta e aceitação, em que o utilizador faz a proposta e esta tem de ser aceite por um decisor;

2. Desenvolvimento do artefacto, em que será criado um artefacto digital pela equipa de desenvolvimento;

3. Armazenamento e disponibilização, altura em que o artefacto será guardado e disponibilizado.

Ao longo deste percurso existe a necessidade de ir alterando permissões de acesso e verifica-se que o esquema de permissões não pode estar apenas baseado em perfis de utilizador, sendo necessário que o esquema compreenda fases, perfis e utilizadores. Por exemplo, na primeira fase, quando o utilizador faz a proposta, por questões de confidencialidade, autenticidade e não-repúdio esta apenas deve poder ser escrita pelo próprio e lida pelos decisores e pelo próprio utilizador, logo não se pode nesta fase

atribuir permissões de leitura a outros perfis semelhantes de proponente. Ainda nesta fase, no momento de aceitação devem ser definidos os perfis que podem visualizar a licitação, o que não levanta grandes problemas à gestão por perfis, desde que se mantenha a restrição para os perfis equivalentes ao do próprio proponente. Já na fase do desenvolvimento, existem novamente problemas com as permissões apenas por perfis, uma vez que a licitação pode ser visualizada pelos utilizadores com os perfis indicados na aceitação, mas o artefacto digital a ser criado, ou reutilizado, apenas deve poder ser escrito pelas próprias equipas de desenvolvedores e a visualização desse artefacto fica dependente da sua natureza legal, isto é, se é público ou privado, se tem outras restrições aplicáveis como por exemplo patentes, direitos autorais, etc. Estes problemas não se encontram na fase do armazenamento e disponibilização, pois pode-se utilizar sem nenhum problema a gestão pelos perfis, conforme as condições anteriores, apenas se terá, por motivos de integridade, autenticidade e não-repúdio, restringir a escrita no artefacto a todos os utilizadores. Este tema de permissões será abordado em mais detalhe no capítulo “Proposta de arquitetura segura U.OPENLAB”.

c) Proponente Interno

“*Amélia Amorim faz parte da equipa de Curadores da plataforma de Licitações. É licenciada em Ciências Históricas pela Universidade Livre do Porto e pós-graduada em Gestão Cultural pela Escola de Gestão do Porto, University of Porto Business School. A Sra. Silva tem vindo a notar que o conhecimento dos estudantes em relação à oferta cultural que a U. Porto dispõe está aquém do que considera aceitável e quer, portanto, arranjar um modo de cativar os estudantes a terem mais interesse. Tendo assim como objetivo promover a cultura e dar maior visibilidade à plataforma de licitações, a Sra. Silva pretende dar continuidade a um projeto que consiste numa plataforma que permite a criação automática de códigos QR em pacotes de açúcar com as peças dos museus em 3D para o aniversário da U.Porto.*

Para isso, coloca a proposta pública na plataforma de licitação, com um titulo, uma descrição, algumas palavras chave da proposta (3D, QR Code, Aplicação Mobile) e uma data limite. Na descrição é de notar que adiciona um link para antigas iterações do projeto de modo a possibilitar a quem se queira propor uma análise do trabalho anteriormente realizado. Considerando que é curadora não necessita de esperar a aprovação da sua proposta.

Passado um certo tempo, recebe uma licitação da parte de uma equipa que lhe parece interessante. A Sra. Silva aprova a licitação e inicia o contacto com a equipa. No entanto, o projeto não é entregue no prazo acordado, portanto, apesar de aprovar a qualidade do projeto e o disponibilizar na plataforma de licitações não fará uso dele como cliente".

Nesta modalidade da plataforma em que o proponente é também decisor e vão ser utilizadas outras iterações de conteúdos já disponibilizados na plataforma, o principal problema é serem perdidas algumas das informações autorais das iterações anteriores, nomeadamente nos artefactos 3D que já existem na plataforma, uma vez que o novo objeto digital, neste caso uma aplicação móvel, terá de fazer uso daqueles artefactos. Além dos dados autorais, existe uma questão sobre o licenciamento do próprio artefacto original, isto é, se ele foi criado com um certo nível de permissões, por exemplo apenas para ensino em contexto de sala de aula, a aplicação móvel não deveria poder utilizá-lo, uma vez que será de uso público. Para evitar estes problemas que afetam principalmente a integridade no que diz respeito à autenticidade e a confidencialidade no que diz respeito à disponibilização de conteúdos, os artefactos criados e disponibilizados na plataforma devem ter associados obrigatoriamente metainformação com os dados autorais RS2.12 e de licença de utilização RS1.9, registados de forma indelével e inalterável, permitindo que qualquer acesso a esses artefactos dentro da plataforma possa obter, verificar e, caso necessário, importar esses dados para novos artefactos de forma automática.

d) Gestor de Plataforma (como Curador)

"*Maria Machado faz parte da equipa de Curadores da plataforma de Licitações. Licenciou-se na Faculdade de Economia da Universidade do Porto e tirou o Mestrado em Direção Comercial e Marketing no Instituto de Empresa em Madrid. É uma pessoa muito minuciosa no seu trabalho tendo brio em manter a qualidade dos projetos e atribuir os respetivos direitos de autor a quem os merece.*

A Sra. Machado faz login na plataforma e consegue ver uma lista de propostas de Proponentes para Aprovar e uma lista de projetos realizados a aprovar para a plataforma. Vai selecionando e analisando cada ponto da lista individualmente. Ao examinar a lista de propostas de proponentes, nota que a descrição de uma proposta efetuada pela empresa ChocoBite não é muito clara no objetivo final, rejeita-a com um

pequeno comentário. Quando passados alguns dias a proposta é reescrita, a Sra. Machado já aceita a proposta.

*Ao examinar a lista de projetos a aprovar para a plataforma, rejeita dois projetos: um projeto que submeteu um trabal*ho *sem anexo e outro que tinha como coordenador um professor que avaliou o trabalho em 12 valores. No caso do segundo, apesar de o rejeitar para a plataforma pública permite ao cliente aceder ao produto submetido como final e mantê-lo na plataforma para possíveis equipas que queiram licitar na mesma proposta*".

Nesta funcionalidade da plataforma em que a aprovação ou não de uma proposta ou disponibilização e publicação de um projeto está dependente de uma gestão baseada em perfis especiais, assim como o desenvolvimento está dependente de submissões da equipa de desenvolvimento, incorre-se no perigo de existirem propostas, ou projetos incompletos que acumulam informação na plataforma, fazendo com que após algum tempo de funcionamento esta fique carregada com informação desnecessária que ocupa recursos do sistema e tornam a manutenção uma tarefa mais penosa. Para tornar a plataforma mais eficaz e eficiente, todo o fluxo de trabalho e das licitações devem estar definidos e assentes numa filosofia de "*Smartcontract*" RS3.1 que faça a verificação automática da informação necessária antes de permitir que o processo passe à fase seguinte e informe os respetivos utilizadores e os perfis dos decisores para tomarem uma ação. Por exemplo, num processo duma proposta que não esteja em conformidade, ou que não seja retificada, os utilizadores com perfil de decisor devem ser informados ao fim de algum tempo para decidirem se a proposta continua na lista ou se será eliminada. Da mesma forma, quando uma licitação é aceite e não exista uma resposta para o desenvolvimento é necessário, ao fim de algum tempo, decidir se a licitação continua na plataforma ou que destino terá. Tal como, após ter sido atribuído um orientador e equipa de desenvolvimento, deve existir uma forma automática que verifique que toda a informação é inserida, antes de permitir enviar o projeto para submissão.

e) Desenvolvedor como Coordenador/Fornecedor do projeto terminado

"*Cristiano Costa é docente da cadeira de Álgebra Computacional na FCUP. Gosta de desafiar os seus alunos a encontrarem aplicações práticas na sua área de estudo. Tem alguns conhecimentos tecnológicos sendo particularmente proficiente no desenvolvimento e utilização de sistemas algébricos computacionais. Fazendo parte do corpo docente da U. Porto tem acesso automático à plataforma de Licitações. Costuma ver se há propostas de projetos novos todas as semanas e esta semana notou que o IBMC propôs uma otimização de um sistema algébrico computacional para o cálculo automático de cinética enzimática.*

Considerando que é do interesse dos seus alunos, propõe no âmbito da disciplina que a avaliação laboratorial seja realizada através deste projeto em vez dos trabalhos computacionais com uma regularidade quinzenal que habitualmente caracterizam este ponto da avaliação prática. Visto que os alunos mostram disponibilidade para aceitar este desafio, o professor analisa as competências dos participantes para formar e equilibrar a equipa de desenvolvedores. Observa que nas competências dos seus alunos se revela uma falta de pessoas com capacidade de programação. No entanto, alguns alunos mostram interesse em realizar essa parte do projeto, portanto o professor atribui-lhes essas competências para no final do projeto decidir se eles de facto revelam, ou não, essas competências.

O professor faz a licitação. Ao fim de alguns dias recebe a aprovação da parte do Proponente, iniciando assim, a comunicação entre ambas as partes. O Professor no papel de coordenador dá início ao projeto e acompanha o seu desenvolvimento com os alunos. Durante o desenvolvimento do projeto, os alunos que estão a lidar com a componente de programação encontram algumas dificuldades em realizar a sua parte recorrendo à ajuda de um colega de Ciência e de Computadores. Considerando a sua ajuda o professor convida-o para o projeto como desenvolvedor.

No final do semestre, o professor considera o projeto encerrado. Aprova, rejeita ou altera as competências de cada aluno conforme a avaliação que foi tendo ao longo do projeto. Como professor coordenador tem também a possibilidade de atribuir a avaliação do projeto numa classificação na escala de 0-20 valores. Dentro dos parâmetros de avaliação da disciplina, atribui ao projeto a avaliação de 18 valores.

O projeto é entregue ao cliente, aprovado para a plataforma e passa a fazer parte da página pessoal do professor e dos alunos na plataforma de licitações".

Nesta funcionalidade da plataforma em que existem perfis com privilégios especiais para o SI, como são os casos de curadores, professores e coordenadores que podem atribuir avaliações, alterar competências e estado das propostas, existe a possibilidade de tentativa para obter o acesso a estes perfis, já que permitem uma manipulação maior da informação por parte de um adversário mal intencionado. Para proteger potenciais ataques a este tipo de perfil, além de controles já abordados anteriormente como encerramento automático de sessão RS1.6, o encriptamento de sessão RS1.7 e a utilização de *cookies* seguros RS1.8 é recomendável limitar o acesso à plataforma através de endereços de IP internos ou através de VPN's (*Virtual Private Network*) fornecidos pelas próprias instituições RS2.13. De notar que embora semelhante à restrição de endereços IP para utilizadores especiais do SI RS2.5 esta restrição funciona a nível de aplicação e não a nível de sistema, resultando que os controles são configurados em locais diferentes, já que o que está em causa é a alteração e/ou visualização de informação que está dentro da plataforma e o perfil de utilizador da aplicação deve ser totalmente independente do utilizador do SO.
Nesta User Storie existe também a probabilidade do utilizador tentar carregar um arquivo malicioso na plataforma, mesmo que o faça inadvertidamente por estar a trabalhar num computador comprometido. Por este motivo, todos os arquivos que forem submetidos à plataforma devem ser verificados por softwares de segurança RS2.14, incluindo os arquivos de documentação que acompanhem um projeto. Da mesma forma, nenhum arquivo submetido deve ter permissões de execução na própria plataforma RS2.15, pois mesmo um simples arquivo de texto que utilize funcionalidades do sistema, como um script pode conter código que seja necessário no contexto de utilização desse artefacto, mas no contexto da plataforma pode causar mau funcionamento. Continuando com o princípio de evitar o mal funcionamento, é desejável que se evitem instruções que permitam a execução de código arbitrário RS2.16 como por exemplo o "eval" do PHP e quando seja necessário utilizar rotinas que façam chamada a comandos externos ou do SO, como por exemplo, cd, mkdir ou ls, é extremamente desejável indicar o caminho completo do binário RS2.17 em vez de apenas o nome, para prevenir ataques que possam usufruir de alguma manipulação de variáveis do sistema.

f) Desenvolvedor elemento da equipa/ Investigador/ Aluno / Professor

"*A Gabriela Gonçalves é uma estudante de Biologia da FCUP. É bastante proactiva e está neste momento à procura de projetos que a possam ajudar a desenvolver a sua versatilidade como futura profissional. Quer também reforçar o seu curriculum e por isso quer ter a certeza que tem algum modo de registar as suas contribuições em projetos. É por isso que participa na plataforma de licitações, e já concorreu a vários projetos, dois dos quais aguardam aprovação da sua candidatura. Nesses projetos, nos quais não foi aprovada a sua participação, consegue ver o perfil dos participantes do projeto. Costuma também comentar em projetos já terminados e aprovados e fazer download de alguns que considera interessantes para os trabalhos laboratoriais que tem de desenvolver no curso.*

Faz de momento parte de um projeto de conservação e taxonomia do Parque Biológico de Gaia, que está à espera de aprovação da parte de um Decisor para ser colocado na plataforma. Nesse trabalho de equipa, sente que alguns dos membros não cumpriram adequadamente o papel a que se tinham comprometido segundo as suas competências, obrigando outros a assumirem esses papeis. Tendo isso em mente realizou uma avaliação às competências dos seus colegas de trabalho. Após a aprovação do Decisor, serão atualizadas no seu perfil as seguintes competências: conservação vegetal e taxonomia, o que a Gabriela adivinha que a ajudará a concorrer a uma bolsa de estudos para investigação na Suécia. Recebeu um convite da parte da professora de Fisiologia Vegetal para participar num projeto sobre Fotossíntese, no entanto rejeitou-o, pois, a duração do projeto interferia com um estágio de Verão em que pensava participar".

Nesta funcionalidade do sistema em que o utilizador pode, potencialmente, visualizar diferentes conteúdos que podem ter diversos formatos multimédia, desde áudio, vídeo, texto, gráficos, entre outros, pode ocorrer um ataque de DOS (*Denial Of Service*), bastaria que para isso o utilizador pressionasse inadvertidamente várias vezes o comando de "refrescar página" do seu navegador. Para prevenir este tipo de vulnerabilidade a aplicação deve implementar controles que só permitam processar novos pedidos após o conteúdo anterior ter sido carregado RS3.2 e também possa disponibilizar versões reduzidas, com menor qualidade ou ainda alternativas dos artefactos RS3.3 que devem ser processadas automaticamente na submissão dos projetos. Sendo que a plataforma também poderá ter múltiplos formatos de ficheiros,

mesmo num só projeto, é extremamente recomendável que todas as páginas que exibam conteúdos implementem controles CSP (*Content Security Policy*) RS2.18, ou seja políticas de segurança de conteúdos que indiquem a proveniência e que formatos de conteúdos são permitidos naquela página.

Existem também nesta funcionalidade da plataforma questões relacionadas com a confidencialidade de alguns dados pessoais, uma vez que os utilizadores podem visualizar outros perfis. Por razões de ordem administrativa, a plataforma terá de ter acesso a esses dados para os mais variados fins, mas nem todos os utilizadores, ou visitantes, devem ter o mesmo nível de acesso a dados pessoais, ou poder-se-ia estar a expor informação sensível que seria fonte de SPAM ou facultaria informação relevante para efetuar ataques. Como forma de prevenção deste tipo de vulnerabilidade, cada perfil de utilizador deve ter classificado a informação que é pública (por exemplo: Nome e competências) e privada (por exemplo: Número de identificação fiscal, contactos de e-mail e telemóvel) e ter definido o nível de acesso para cada tipo de perfil RS1.10 de acordo com o papel que esse perfil tem na plataforma. Uma vez que esta informação é essencial para diversas funcionalidades e requisitos do sistema, como por exemplo autenticação e a confidencialidade que permitem algumas características de autenticidade e não repúdio, todos os dados pessoais privados só devem poder ser editados pelo próprio utilizador e devem estar cifrados RS1.11 de forma a manter o sigilo para utilizadores do sistema não autorizados, ou ocorra um ataque externo bem sucedido.

3.2.4. Especificação de requisitos de segurança U.OPENLAB

Os requisitos de segurança foram especificados de acordo com a análise geral do modelo de funcionamento da plataforma e das *User Stories* que demonstram as funcionalidades esperadas para cada perfil do utilizador da mesma. Para a especificação foram levados em consideração o TOP10 de vulnerabilidades em aplicações web da OWASP e a base de dados da CVE (Common Vulnerabilities and Exposures) (CVE security vulnerability database, 2019). Embora os requisitos estejam classificados separadamente nas três categorias do modelo CIA, a maior parte das vulnerabilidades de que estes requisitos são contramedidas, se bem exploradas permitem afetar mais do que uma das vertentes do modelo. A listagem dos requisitos

indica a ordem pela qual foram sendo identificados na análise dos cenários e não a prioridade de implementação, ou grau de relevância, ou ainda a frequência com que um ataque é executado.

Todos os requisitos de segurança destinam-se a fornecer aos desenvolvedores da aplicação linhas orientadoras na fase de construção da aplicação e devem ser incluídos com os restantes requisitos de engenharia, já que os problemas detetados na fase do desenho e desenvolvimento, quando bem implementados, permitem construir aplicações mais seguras e poupar tempo em futuras correções do código fonte.

Não obstante poderem ser seguidos diversos padrões para implementar os requisitos de segurança, como por exemplo os da família ISO 27K, por razões de coerência com o restante trabalho sugere-se a utilização do standard da OWASP *Application Security Verification Standard* 4.0 (*Open Web Application Security Project*, 2019).

- **Confidencialidade**
  - **RS1.1 - Comunicações seguras**.

    Para evitar ameaças de captura de informação em texto plano por terceiros, todas as comunicações com clientes e/ou outros serviços devem ser assentes no protocolo https.

  - **RS1.2 - Comunicações com a aplicação utilizam protocolos seguros sem vulnerabilidades conhecidas.**

    Com o objetivo de evitar ataques "*Man in the Middle*" em que terceiros possam visualizar e/ou alterar a informação em trânsito, a aplicação apenas deve utilizar protocolos considerados seguros e aos quais não se conhecem formas de ataque (à data da redação destes requisitos, os protocolos que fornecem estas garantias são o TLS (*Transport Layer Security*) versões 1.1 e 1.2).

  - **RS1.3 - Omitir informação do sistema**.

    Para evitar fornecer informação sobre o sistema a agentes mal intencionados que permitam explorar vulnerabilidades existentes nos componentes incluindo as de 0Day, todas as funcionalidades que forneçam

mensagens que informem sobre as versões de software instalado e mensagens de erro, ou passíveis de mostrar conteúdos confidenciais, como por exemplo o nível de display_error ou funcionalidades tipo phpinfo() ou a diretriz auto_index do Apache devem ser configuradas para ocultar essas mensagens e ou desabilitadas para não mostrar informação adicional sobre o sistema.

- **RS1.4 - Restrições a perfis de visitante.**

  Para mitigar potenciais riscos de fuga de informação e tentativas de subverter o sistema, utilizadores com o perfil de visitante apenas têm permissão de leitura e só podem aceder a conteúdos públicos.

- **RS1.5 - Conformidade RGPD.**

  Para estar em conformidade com o Regulamento Geral de Proteção de Dados, antes da criação de um perfil de utilizador na plataforma, o sistema deve obter o consentimento informado do utilizador de quais são os seus direitos de acesso aos seus dados pessoais, quem os pode visualizar, quais são os tratamentos e processamentos que lhes são aplicados, quais são os objetivos, destino e a duração em que os mesmos são armazenados, assim como as medidas de segurança executadas para proteção dos mesmos.

- **RS1.6 - Encerramento automático de sessão.**

  Para evitar que terceiros se possam fazer passar por um utilizador apossando-se do equipamento onde este iniciou sessão, visualizando e alterando as informações deste, todas as sessões devem ser terminadas automaticamente após um período de inatividade.

- **RS1.7 - Sessão encriptada**.
  Para evitar o uso de dados de sessão mais do que uma vez impedindo ataques de "sequestro de sessão", cada sessão de utilizador deve utilizar uma chave única de encriptação OTP (One Time Password), válida apenas para essa mesma sessão.

- **RS1.8 - Permissões de Cookies.**

Para evitar fuga de informação sensível do utilizador e da sessão que permitam explorar vulnerabilidades do sistema ou a utilização de informação sensível por terceiros, todos os cookies de sessão utilizados pelo navegador devem estar marcados como “http_secure” e “http_only”.

- **RS1.9 - Copyright de artefactos.**

  Para evitar quebras de confidencialidade expondo informação sensível através da plataforma, todos os artefactos disponibilizados devem ter a correspondente metainformação sobre o licenciamento.

- **RS1.10 - Classificação de informação em Perfis**.

  Para evitar quebras de confidencialidade divulgando informação sensível a terceiros, todos os perfis devem ter classificada qual é a informação pública e privada que cada utilizador pode visualizar de acordo com o seu perfil.

- **RS1.11 - Encriptação de dados pessoais privados**.

  Para evitar quebras de confidencialidade permitindo fugas de informação sensível fora da plataforma no caso de um ataque bem sucedido, todas as informações pessoais privadas de um perfil devem estar cifradas.

• **Integridade**

- **RS2.1 - Software seguro e atualizado.**

  Para prevenir ataques que possam comprometer o sistema, todos os softwares e *frameworks* necessários ao funcionamento da plataforma, incluindo o sistema operativo, devem estar atualizados com os correspondentes *patchs* de segurança e isentos de vulnerabilidades conhecidas.

- **RS2.2 - Configurações personalizadas.**

  Para reduzir ataques comuns e padronizados, todos os componentes de software necessários ao bom funcionamento da plataforma, devem ter configurações personalizadas, evitando quando possível os valores padrão.

- **RS2.3 - Permissões mínimas de software.**

  Para minimizar danos ao sistema em caso de ataques bem sucedidos, todas as contas, serviços e softwares necessários para o bom funcionamento da plataforma devem ter privilégios mínimos de execução e ter restrições de escrita às pastas do sistema.

- **RS2.4 - Desabilitação de portos e serviços.**

  Para reduzir a área de ataque, todos os portos de ligação, serviços e funcionalidades que não sejam necessárias ao bom funcionamento da plataforma devem estar desabilitados.

- **RS2.5 - Restrições de endereços IP**.

  Para restringir pontos de acesso ao sistema, todas as contas necessárias à manutenção do mesmo como por exemplo contas de admin, cópia de segurança, etc. devem ter o acesso restringido a endereços IP internos ou por VPN e pré-determinados.

- **RS2.6 - Logs de sistema independentes.**

  Para manter a integridade e a rastreabilidade dos logs de sistema e da plataforma, todas as interações dos utilizadores e visitantes com estes componentes do SI devem ter uma cópia remota que permita comparações e também permita estudar padrões de ataque.

- **RS2.7 - Sanitização de inputs.**

  Todos os campos passíveis de ser manipulados pelo utilizador como por exemplo campos de formulários e URL's, devem passar por funções de sanitização que limpem caracteres especiais com significado para o sistema e aplicações executadas por este, com a finalidade de evitar injeção de código malicioso que permita explorar vulnerabilidades no SI.

- **RS2.8 - Verificação e validação de inputs.**

  Todas as funções que recebam variáveis externas à mesma como input devem verificar a procedência, campos, tipo e tamanho da variável

esperada antes de executar o algoritmo de processamento para evitar um eventual mau-funcionamento no SI, incluindo dados de arquivos XML ou de autenticação. Por exemplo, uma função que espere um input de um número inteiro entre 1 e 1000, não deve ser processada se em vez do tipo esperado receber uma *string* ou se o input for 10000.

- **RS2.9 - Separação de dados e instruções em SQL.**

  Com o objetivo de evitar injeções de SQL, todas as *querys* devem estar no formato "*SQL_prepared_statement*" que faça a divisão entre a informação a processar e a própria *query*.

- **RS2.10 - Escapamento de output's.**

  Para evitar ataques comuns, entre os quais os de XSS (*Cross Site Scripting)* e outras vulnerabilidades conhecidas que possam expor informação ou causar mau funcionamento na plataforma, todos os output's gerados pela aplicação, especialmente os que confiam em input's do utilizador, devem ser parametrizados e escapar corretamente todos os caracteres com significado especial para o SI.

- **RS2.11 - Autenticação externa e/ou Multi-fator.**

  Para reduzir as vulnerabilidades que resultem na fuga de informação e quebras de autenticação, todos os perfis da aplicação devem utilizar servidores de autenticação externos confiáveis (ex: Autenticação Federada U.Porto, Autenticação.gov.pt, Google Authenticator, Linkedin, etc.) que proporcionem API's de ligação seguras e dos quais se possam utilizar os tokens de autenticação na plataforma. Adicionalmente, seja para quando um destes serviços esteja indisponível, ou para verificação da autenticidade dos dados de utilizador, a plataforma deve proporcionar um método interno de autenticação multi-fator que seja baseado em "User ID" + PIN para telemóvel ou correio eletrónico com controlo de tentativas e incremento temporal de pedidos.

- **RS2.12 - Registo de autores.**

Todos os artefactos digitais geridos e disponibilizados pela plataforma devem ter registado metainformação que identifique de forma indelével e inalterável os autores do mesmo.

- **RS2.13 - Restrições acesso Perfis especiais.**

  Para reduzir a superfície de ataque limitando os pontos de acesso, os perfis com papeis administrativos relevantes para o correto funcionamento da aplicação, tais como administradores, curadores, responsáveis de projeto, etc., devem ter o acesso limitado por WAF (Web Application Firewall) a uma gama de IP's internos e/ou por VPN da instituição.

- **RS2.14 - Verificação arquivos.**

  Para evitar propagar software malicioso que possa comprometer o sistema e equipamentos dos utilizadores, todos os arquivos submetidos à plataforma devem passar por um processo de verificação com antivírus e anti-malware antes de serem aceites e disponibilizados na plataforma.

- **RS2.15 - Restrição de execução de artefactos.**

  Para evitar vulnerabilidades que possam provocar mau funcionamento no SI, todos os artefactos que forem disponibilizados na plataforma não devem ter permissões de execução dentro da mesma.

- **RS2.16 - Prevenção de execução de código arbitrário.**

  Para reduzir as vulnerabilidades relacionadas com a execução de código arbitrário que possa subverter a segurança do SI, sempre que possível devem-se evitar no código fonte comandos tipo eval() ou exec() que permitem a execução de código. Quando não houver alternativa para este tipo de recurso, as instruções devem ser acompanhadas de lista de comandos válidos (white-listing) no lado do servidor.

- **RS2.17 - Especificação de caminhos**.

  Para prevenir ataques que permitam explorar vulnerabilidades no SI, todas as instruções no código fonte que façam uso de chamadas ao sistema operativo (ls, mkdir, rm, etc.) ou outros comandos existentes no SI, devem

utilizar caminhos completos dos binários a utilizar, por exemplo “/usr/bin/cd” ou “/bin/ls” em vez de “cd” ou “ls”.

- **RS2.18 - Controle de conteúdos.**

  Para prevenir ataques que explorem técnicas para subverter a segurança do SI, todas as páginas de conteúdos da aplicação devem ter “headers” que controlem a proveniência, tipo de arquivo e as suas permissões de execução.

- **Disponibilidade**
  - **RS3.1 - Controle de fluxo informacional.**

    Para evitar o excesso de acúmulo de informação desnecessária na plataforma e facilitar tarefas de manutenção, tornando o sistema mais eficiente, todo o fluxo de trabalho de uma licitação deve estar baseada em algoritmos de “Smartcontract” que permitam verificar o correto preenchimento da informação necessária e informar os utilizadores intervenientes no processo.

  - **RS3.2 - Controle de pedidos.**

    Para reduzir ameaças que possam deixar a plataforma indisponível temporariamente, a plataforma deve implementar mecanismos de controle de pedidos por utilizador que apenas sirva novos conteúdos após os anteriores estarem carregados.

  - **RS3.3 - Controle de conteúdos alternativos.**

    Para evitar problemas de sobre carregamento do SI, todos os artefactos devem ter versões reduzidas ou alternativas para visualização na plataforma, disponibilizando hiperligações para os conteúdos originais.

3.3. Avaliação de Risco e Análise de Vulnerabilidades U.OPENLAB

Com a identificação das vulnerabilidades que derivam das principais ameaças e ataques a que todo o SI está exposto, juntamente com as respetivas contramedidas associadas, procede-se de seguida a uma breve análise qualitativa das vulnerabilidades identificadas nos cenários anteriores. Para esta análise, utilizaram-se as orientações do SAMM v1.5 relativas às tarefas em questão (OWASP, 2017) e um template **(**Smartsheet**, 2019)** para avaliação de risco que foi adaptado ao estudo de caso.

Na Tabela 3, em que se faz uma análise qualitativa das vulnerabilidades, calcula-se a probabilidade de existir uma exploração das mesmas com base na frequência da ameaça ou ataque indicada pela tendência do Top 10 de ataques a aplicações Web de 2018 da OWASP. A esta variável está atribuído o valor qualitativo de Baixo, quando o risco da vulnerabilidade ser explorada não é frequente ou é de implementação muito difícil; atribui-se o valor de Médio quando a sua exploração for frequente ou com facilidade de implementação razoável e atribui-se o valor de Alto quando a exploração da vulnerabilidade é extremamente frequente, independentemente da facilidade de implementação.

Para o cálculo da severidade é considerado o impacto interno causado ao funcionamento do SI numa das vertentes do modelo CIA e o impacto causado ao utilizador da imagem e reputação da instituição, no caso de a vulnerabilidade conseguir ser explorada. Para esta variável, não foi considerada nesta fase o tipo de agente que explora a vulnerabilidade. Isto é, para o cálculo não foi considerado se é um agente externo, interno ou interno de confiança que, considerando as suas permissões, podem causar um nível diferente de estragos.

VID	CIA	Descrição	Severidade	Probabilidade	Impacto Interno	Impacto Utilizador	Contramedidas
V01	C	A falta de métodos de encriptação segura nas comunicações pode permitir a terceiros visualizar informação sensível ou alterações nessa informação em trânsito, levando a vulnerabilidades de confidencialidade e integridade.	Intolerável	Provável	Extremo	Extremo	RS1.1 e RS1.2
V02	I	A utilização de software com vulnerabilidades e sem as correspondentes atualizações de segurança, pode causar diversas brechas que permitem muitos dos ataques conhecidos (*Reverse shell,* Injeção, escalada de privilégios, etc.) que afetam o correto funcionamento do SI em todas as suas vertentes.	Intolerável	Possível	Extremo	Alto	RS2.1
V03	I	Componentes de software e serviços necessários ao correto funcionamento da plataforma que estejam mal configurados ou utilizem as configurações padrão, permitem a utilização de ataques comuns e padronizados que podem afetar o SI.	Indesejável	Possível	Alto	Alto	RS2.2 e RS2.3
V04	C	Exibir outputs com erros e versões de software utilizados pela plataforma permite a um atacante estudar e estruturar ataques ao SI.	Tolerável	Possível	Médio	Baixo	RS1.3
V05	I	Manter portos e serviços não necessários ao correto funcionamento do SI aumenta a superfície de ataque que um adversário pode utilizar, além de aumentar o esforço necessário para a monitorização de segurança e de recursos do SI.	Intolerável	Possível	Extremo	Alto	RS2.4
V06	I	Limitar os locais de acesso remoto de contas especiais do sistema permite manter um perímetro de segurança mais apertado e reduzir os pontos de acesso que um adversário pode utilizar.	Intolerável	Possível	Extremo	Alto	RS2.5

VID	CIA	Descrição	Severidade	Probabilidade	Impacto Interno	Impacto Utilizador	Contramedidas
V07	I	Manter cópias dos logs do sistema independentes das máquinas que alojam o SI, permite manter um registo íntegro para auditorias e para estudar padrões de ataque ao sistema, já que um adversário terá que quebrar a segurança em mais do que uma máquina.	Tolerável	Possível	Médio	Baixo	RS2.6
V08	C	Conceder privilégios de escrita e de acesso ao público à informação classificada pode abrir brechas no sistema de segurança quebrando a confidencialidade e/ou a integridade da plataforma.	Indesejável	Improvável	Médio	Baixo	RS1.4
V09	I	O incorreto tratamento dos inputs externos ao sistema pode resultar em diversas vulnerabilidades conhecidas e bastante frequentes como por exemplo, injeções SQL, injeções de sistema operativo, XSS, XXE, etc, pelo que todos os inputs devem ser sanitizados e verificados com *white-listing*.	Intolerável	Provável	Extremo	Alto	RS2.7, RS2.8 e RS2.9
V10	I	Não escapar corretamente e formatar os outputs da plataforma pode permitir uma grande variedade de ataques e fornecer informação sensível a um adversário sobre o sistema.	Intolerável	Provável	Extremo	Alto	RS2.10
V11	C	Por questões relacionadas com o RGPD, antes da plataforma processar ou guardar quaisquer dados pessoais do utilizador deve informar e obter o consentimento deste para tratar essa classe de dados.	Aceitável	Improvável	Baixo	Baixo	RS1.5
V12	C	Para mitigar vulnerabilidades relacionadas com sessões de utilizador, a plataforma deve implementar controles de sessão restritos quer ao nível de encerramento automático, encriptação de dados de sessão e de cookies.	Indesejável	Possível	Alto	Alto	RS1.6, RS1.7 e RS1.8
V13	I	Para mitigar fugas de informação com as credenciais de acessos do utilizador, a plataforma deve implementar técnicas de autenticação que não necessitem de armazenar essas credenciais internamente.	Intolerável	Possível	Alto	Alto	RS2.11

VID	CIA	Descrição	Severidade	Probabilidade	Impacto Interno	Impacto Utilizador	Contramedidas
V14	C.I	Para mitigar riscos relacionados com direitos autorais e de utilização, a plataforma deve ter meios automáticos de verificação de autores e licenciamento.	Indesejável	Improvável	Baixo	Baixo	RS1.9 e RS2.12
V15	A	Para otimizar a eficiência do SI a plataforma deve implementar formas de reduzir e remover informação desnecessária.	Aceitável	Provável	Baixo	Baixo	RS3.1
V16	I	Para mitigar problemas relacionados com a divulgação e adulteração de informação sensível, a plataforma deve implementar restrições a nível da aplicação para perfis com privilégios especiais.	Intolerável	Possível	Extremo	Alto	RS2.13
V17	I	Para evitar a propagação de software malicioso, todos os arquivos subidos à plataforma pelos utilizadores devem ser verificados por softwares de segurança antes de serem disponibilizados.	Intolerável	Possível	Extremo	Extremo	RS2.14 e RS2.15
V18	I	Para mitigar vulnerabilidades, o código fonte deve evitar instruções que permitam a execução de código arbitrário.	Intolerável	Possível	Extremo	Alto	RS2.16
V19	I	Para mitigar vulnerabilidades do sistema, sempre que for necessário fazer chamada a comandos externos, devem ser utilizados os caminhos completos dos binários.	Indesejável	Possível	Extremo	Alto	RS2.17
V20	A	Para mitigar vulnerabilidades relacionadas com negação do serviço, a plataforma deve implementar rotinas e controles que não permitam o excesso de pedidos e sobrecarga de informação.	Indesejável	Possível	Médio	Médio	RS3.2, RS3,3
V21	I	Para reduzir os vetores de ataque que um adversário pode utilizar, a plataforma deve implementar políticas de controle de conteúdos permitidos.	Indesejável	Possível	Alto	Médio	RS2.18
V22	C	Para evitar fuga de informação pessoal, o SI deve definir níveis de privilégio de visualização de dados pessoais para perfis e cifrar a informação privada de cada utilizador.	Indesejável	Possível	Médio	Alto	RS1.10 e RS1.11

Tabela 3 - Análise de Vulnerabilidades U.OPENLAB

Explicação dos valores encontrados na tabela 3:

- VID - Identificador interno da Vulnerabilidade.
- CIA - Indica as vertentes do modelo CIA em que estão classificadas as contramedidas.
- Descrição - Breve descrição da Ameaça e/ou ataque que contextualiza a análise da vulnerabilidade.
- Severidade - Valorização qualitativa distribuída por quatro escalões e calculada com base na probabilidade e impacto interno e no utilizador:
  - Aceitável - Indica vulnerabilidades identificadas, mas com as quais o SI pode funcionar sem grandes prejuízos.
  - Tolerável - Indica vulnerabilidades identificadas que não afetarão muito o funcionamento do SI, ou podem facilmente ser reparadas.
  - Indesejável - Indica vulnerabilidades que se devem evitar pois causam graves problemas no SI.
  - Intolerável - Indica que as vulnerabilidades afetam o SI de forma muito grave.
- Probabilidade - Variável calculada com base na frequência e/ou facilidade de ataque:
  - Improvável - Indica que a o ataque é pouco frequente ou de difícil implementação.
  - Possível - Indica que o ataque é frequente e de dificuldade média de implementação.
  - Provável - Indica que o ataque é frequente, independentemente da facilidade de implementação, mas também ligado ao valor do ativo

informacional.

- Interno e utilizador - Avaliação feita ao impacto que os danos podem causar quer no próprio SI e nos seus ativos informacionais ou na reputação da entidade que aloja a plataforma. Está distribuída nos seguintes valores:
  - Baixo - Significa que um ataque bem sucedido não causará grandes danos a qualquer uma das componentes avaliadas.
  - Médio - Significa que os danos não serão muito graves ou serão facilmente minimizados.
  - Alto - Significa que os danos causados poderão prejudicar muito qualquer um dos indicadores medidos.
  - Extremo - Significa que os danos causados poderão ser muito graves ou irreversíveis.
- Contramedidas - Indica quais são os requisitos de segurança associados à mitigação do risco da ameaça ou ataque em questão.

Com base nesta análise qualitativa foi elaborada uma matriz de risco presente na tabela 4 que permitirá aos desenvolvedores priorizar a implementação das respetivas contramedidas. A escala de prioridade está distribuída em “Baixa” quando o valor é inferior a 4, “Média” com valores entre 4 e 7 e “Alta” com valores entre 8 e 12. A valorização do risco é calculada com base do produto da “Severidade” a que foram atribuídos valores de 1 (Aceitável) a 4 (Intolerável) e “Probabilidade” aos quais se atribuiu os valores de 1 (Improvável) a 3 (Provável).

Para a elaboração da matriz de risco apenas se levou em conta as vulnerabilidades identificadas e o potencial risco que podem causar ao funcionamento da plataforma. Questões de ordem pecuniária, ou de custo de desenvolvimento, devem ser consideradas em futuras reuniões que abranjam equipas mais multidisciplinares e as possam avaliar com mais propriedade.

Com base nesta matriz de risco, a equipa de desenvolvedores pode iniciar os seus trabalhos sabendo quais são as funcionalidades que necessitam mais urgência. Esta análise não indica que os riscos classificados como baixos não devam ser também alvo da sua atenção, nem que, apesar da análise, não existam outras vulnerabilidades ainda não identificadas, nem outras questões que devam ser alvo de ponderação.

VID	Contra-medidas	Severidade	Probabilidade	Prioridade
V01	RS1.1, RS1.2	4	3	Alta
V02	RS2.1	4	2	Alta
V05	RS2.4	4	2	Alta
V06	RS2.5	4	2	Alta
V09	RS2.7, RS2.8 e RS2.9	4	3	Alta
V10	RS2.10	4	3	Alta
V13	RS2.11	4	2	Alta
V16	RS2.13	4	2	Alta
V17	RS2.14 e RS2.15	4	2	Alta
V18	RS2.16	4	2	Alta
V08	RS1.4	3	1	Baixa
V11	RS1.5	1	1	Baixa
V14	RS1.9 e RS2.12	3	1	Baixa
V15	RS3.1	1	3	Baixa
V03	RS2.2, RS2.3	3	2	Média
V04	RS1.3	2	2	Média
V07	RS2.6	2	2	Média
V12	RS1.6, RS1.7 e RS1.8	3	2	Média
V19	RS2.17	3	2	Média
V20	RS3.2, RS3,3	3	2	Média
V21	RS2.18	3	2	Média
V22	RS1.10 e RS1.11	3	2	Média

Tabela 4 - Matriz de risco U.OPENLAB

3.4. Proposta Arquitetura Segura U.OPENLAB

Com base no trabalho de análise realizado no capítulo anterior, propõe-se, agora, uma arquitetura segura para a plataforma, começando por uma vista de alto nível para a estrutura do SI, seguindo-se uma exposição das tecnologias que se podem utilizar para alcançar os objetivos propostos para este projeto.

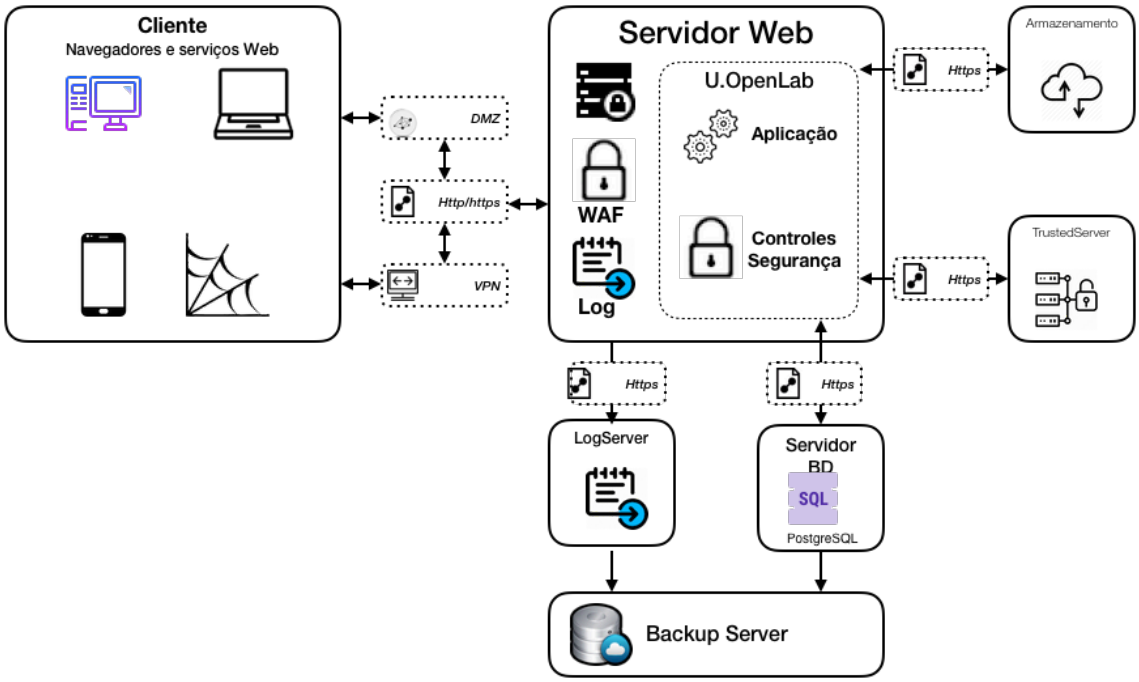


Fig. 6 - Proposta arquitetura Segura U.OPENLAB

Conforme se pode observar no esquema da Figura 6, sugerem-se algumas alterações à ideia inicial do SI sem que este perca funcionalidades e já com a segurança da informação em vista.

Neste esquema já se contemplam as diferentes ligações dos clientes através de uma DMZ (DeMilitarized zone) ou através de endereços internos ou por VPN das próprias instituições, o que, além de representar melhor o mundo real, permite implementar alguns controles e verificações de segurança independentes, nomeadamente a nível do tipo de perfil e serviços que o utilizador está a usar, mediante o tipo de ligação efetuada, isto é, com uma firewall de infraestrutura que pode filtrar serviços e portos e uma WAF que filtrará e analisará os próprios pacotes a nível de aplicação.

A própria vertente do servidor está segmentada entre os seus componentes funcionais (servidor da aplicação, base de dados, servidores de autenticação e armazenamento

remoto) o que facilita a manutenção e atualizações individuais de cada componente, permitindo ao mesmo tempo uma análise mais refinada de brechas e vulnerabilidades de forma isolada, o que permite aos outros componentes continuar a funcionar com relativa segurança, no caso de qualquer outro estar comprometido.

Esta proposta de arquitetura prevê também a utilização de log's remotos, garantindo desta forma mecanismos que permitem ter mais do que um exemplar desses log's para comparação, contribuindo com a autenticidade e fiabilidade dos mesmos, já que para um atacante passar desapercebido, seja ele externo ou interno, teria de comprometer mais do que uma máquina. Também prevê um sistema de cópias de segurança que ajudará a repor o sistema no caso de alguma vulnerabilidade ser explorada com sucesso, contribuindo desta forma para a disponibilidade de toda a informação existente na plataforma.

Apesar da abstração que é possível com esta vista de alto nível e com a análise realizada nos pontos anteriores, ainda existem alguns obstáculos relacionados com a segurança que seriam interessantes ultrapassar, nomeadamente a distinção entre projetos de licitação e artefactos, que medidas poderiam tomar-se para a preservação da informação e que tipo de tecnologias poder-se-iam incorporar para obter estes mesmos resultados.

3.5.Tecnologias para gestão de informação

Qualquer sistema mais ou menos complexo terá de ter uma forma eficiente e eficaz de gerir a informação. No caso específico das aplicações web, essa gestão costuma estar a cargo de bases de dados tradicionais em SQL (*Structured Query Language*), pois estas permitem uma grande flexibilidade e, quando bem implementadas, um nível razoável de segurança. Estas bases de dados, também se costumam distinguir entre o tipo de gestão a que estão sujeitas entre centralizadas e distribuídas. Os dois modelos têm vantagens e desvantagens do ponto de vista da segurança. Se por um lado as bases de dados centralizadas assentam num modelo de gestão mais fácil de gerir e manter sob controlo, do ponto de vista da segurança esta centralização da informação torna-as num alvo preferencial de atacantes, pois basta encontrar uma brecha para subverter o sistema e poder ter acesso a toda a informação. Pelo lado do modelo distribuído, um atacante teria de encontrar brechas em mais do que uma máquina para

subverter o sistema e controles de segurança, tornando o ataque mais difícil, mas sendo este modelo mais complexo para gerir.

Para o projeto U.OPENLAB que conta com os seus próprios requisitos funcionais e de segurança, a escolha de um modelo de gestão não é muito linear, mas terá de ser feito pensando nos objetivos do próprio projeto. Se por um lado a gestão das licitações e interações dos utilizadores com o sistema não é muito problemática podendo ser feito por qualquer um dos modelos tradicionais de base de dados, a questão dos artefactos já tem outras considerações a ponderar. Sendo que os novos artefactos que forem administrados pela plataforma terão natureza de licenciamento e utilização distintos, além de ficarem armazenados na nuvem em um sistema de armazenamento independente da plataforma, a gestão torna-se extremamente complexa. Para administrar estes objetos digitais é necessário empregar tecnologias que permitam transferir as propriedades de autenticidade e integridade, quer dos próprios artefactos, quer da metainformação associada, relativa aos autores, licenciamento e autorizações de acesso. Além de toda esta informação que deve estar registada de forma inequívoca e inalterada, existe a questão de a preservar e disponibilizar, mesmo que a plataforma seja descontinuada, ou esteja por qualquer motivo, indisponível. Será, portanto, desejável armazenar estes artefactos com garantia de que todas estas propriedades não são perdidas e a informação se conserva ao longo do tempo, independentemente da plataforma.

Para a gestão dos objetos digitais administrados pela plataforma U.OPENLAB, a tecnologia *Blockchain* é bastante promissora.

No que concerne à confidencialidade, assim como num sistema de base de dados tradicional, uma implementação correta de *Blockchain* pode atribuir permissões de quem pode ler e escrever registos utilizando esquemas de PKI, permitindo assim mitigar alguns riscos de confidencialidade associados aos artefactos. Sendo ainda possível utilizar criptografia de ponta-a-ponta, dificultando ataques de “man-in-the-middle”, desde que se utilizem protocolos de comunicação seguros.

No que diz respeito à integridade, a tecnologia *Blockchain* que faz registos incrementais cifrados, permite guardar um rastro das operações e objetos digitais, sendo virtualmente impossível alterar registos passados. Desta forma, pode-se garantir quais foram os utilizadores que interagiram com o objeto, permitindo atribuir os créditos da criação de um artefacto aos seus criadores. Isto torna todas as interações com determinado objeto auditáveis, sendo possível determinar quando e quem fez o quê de forma independente

da gestão central da plataforma. Esta característica de imutabilidade dos registos passados, permite por si só, também criar outros sistemas de informação confiáveis, apenas com os dados registados nos artefactos. Todavia, esta imutabilidade também levanta obstáculos a alguns direitos dos utilizadores, nomeadamente ao "direito ao esquecimento", pelo que a implementação do *blockchain* deve levar em conta a encriptação de dados pessoais e estar assente num esquema de PKI que permita revogar as respetivas chaves dos utilizadores e torne impossível ler a sua informação pessoal, mantendo assim a sua privacidade.

Em relação à disponibilidade a tecnologia *blockchain*, devido à sua natureza descentralizada, é mais resiliente a ataques de negação de serviço do que um esquema servidor-cliente, não significando isto que é imune a este tipo de ataque, apenas que não existe um ponto único de disrupção. Da mesma forma, é possível guardar o registo do *blockchain* fora do servidor, tornando a sua existência tão longa quanto os utilizadores o quiserem. Isto é, mesmo que o serviço onde a informação foi guardada seja desativado, o *blockchain* continua a fazer prova de autoria, integridade, e não repúdio que pode ser verificado por qualquer pessoa com acesso ao *blockchain* de forma independente.

Ainda no que concerne ao modelo CIA e aos seus atributos, um ataque bem sucedido a um sistema *blockchain*, teria proporções inferiores do que um ataque a um sistema com gestão de autoridade centralizada, pois afetaria apenas os nós envolvidos. Para não mencionar que o trabalho que um atacante teria para efetuar um ataque bem sucedido a um sistema baseado em *blockchain* seria muito maior e com recompensa menor do que a um sistema de autoridade centralizada, pois teria que efetuar o ataque em mais máquinas no caso do primeiro, devido à sua natureza descentralizada e no caso de ser bem sucedido num ataque com gestão de confiança centralizada, poderia alastrar a corrupção da informação a todo o sistema.

Na tabela que se segue, retirada de Agrawal (2019) pode-se observar uma comparação das diferenças entre a tecnologia *Blockchain* e Bases de dados.

Blockchain	**Base de Dados**
Não existem administradores ou responsáveis	Administradores e controlo central
Acesso aberto (Público)	Apenas entidades com permissões podem aceder
Escrita para quem tem direitos de trabalho	somente entidades autorizadas a ler e/ou escrever podem fazê-lo
Blockchains são lentos	BDs são rápidas
Histórico de registos e propriedade digital dos mesmos	Sem histórico dos registos e propriedade digital dos mesmos

Tabela 5 - Comparação Blockchain versus Bases de dados

Mesmo existindo implementações de *blockchain* em bases de dados como o BigchainDB (McConaghy, T. & ALL, 2016) não significa que a tecnologia *blockchain* possa ou deva substituir os sistemas tradicionais de bases de dados, até porque estes últimos permitem uma flexibilidade e têm uma facilidade de implementação maior. Independentemente desta discussão, ambos os sistemas de gestão da informação têm características, propriedades e funcionalidades próprias, podendo todos esses atributos ser utilizados convenientemente para as funções que lhes correspondem. Assim sendo, para a plataforma do projeto U.OPENLAB que vai necessitar de respostas ágeis às interrogações dos utilizadores, sugere-se usar um sistema de base de dados tradicional, seja ele centralizado ou distribuído, para as funções mais operacionais da plataforma.

Para a gestão e armazenamento dos objetos digitais que a plataforma disponibilizará pode-se utilizar o sistema *Blockchain*, mesmo que utilize a implementação em base de dados, já que cumpre com todos os requisitos necessários de confidencialidade, de integridade e de disponibilidade. Claro fica que, por razões legais, nomeadamente o direito ao anonimato, o sistema *Blockchain* terá de estar assente numa infraestrutura de chave pública que permita ao utilizador exercer esse direito.

3.6. Perspetivas futuras

Mesmo para um projeto que ainda está na fase inicial, em que a sua existência se limita ao desenho no papel e a um protótipo funcional que fez a prova de conceito, não se podendo especificar com muito mais detalhe todas as tarefas relacionadas com a segurança da informação, não se pode esquecer que este projeto pretende ser *Secure by design*. Isto significa que muitas tarefas ainda estão por realizar e que eventualmente as já realizadas terão de ser revistas.

Não sendo o objetivo desta dissertação aprofundar a discussão sobre os passos necessários para a produção de uma plataforma digital segura, nem detalhar e listar todas as ferramentas possíveis para alcançar esse efeito, até porque o projeto ainda está numa fase muito precoce, ainda que de forma conceptual, é necessário validar os requisitos de segurança, seja de forma manual, seja através de ferramentas que ajudem a alcançar este objetivo. Pode-se, assim, começar pelas especificações de ambiente que requerem configurações personalizadas e necessitam de confirmação humana. Em seguida, utilizar ferramentas como o Network Mapper (Lyon G., 1997) que é uma ferramenta *open source* que continua a ser atualizada e é utilizada para os mais diversos fins e que pode ajudar a validar algumas das especificações de requisitos, tal como portos e serviços; output de versões de software; etc. ou ferramentas da classe do Wireshark (Combs, G., 1998) que serve para analisar o trânsito em rede e permite verificar as especificações relacionadas com as comunicações.

Outras especificações de segurança, como por exemplo a sanitização e a validação dos input's, já necessitam de outras atividades, começando pela própria modelagem dos processos que permitirá perceber melhor onde existem vulnerabilidades, classificá-las e onde implementar os respetivos controles. Seguindo-se para estas especificações as tarefas de revisão de código e análise estática, já que ambas permitem descobrir e classificar muitas vulnerabilidades. Estas tarefas que aliás devem ser estendidas a outras especificações, como seria o caso da recomendação de evitar instruções que permitam a execução de código arbitrário e quando não seja possível fazê-lo, se estas estão acompanhadas por validação *white-listing*.

Ainda na fase de construção, mas já quando a plataforma estiver minimamente funcional, é necessário executar todos os testes de qualidade e segurança necessários para cada funcionalidade da plataforma. Testes estes que deverão garantir que os processos fazem apenas aquilo para que foram desenhados e nada mais. Por exemplo,

testes que verifiquem as funções de *login* e *logout*, ou que verifiquem as funcionalidades de subida de artefactos, etc.

Após estas tarefas estarem executadas e terem sido retificados os erros encontrados, deve-se proceder a testes de penetração, tanto em regime de *White-box* como em regime de *Black-box*, para garantir que todas as especificações são cumpridas e o risco a que a plataforma está exposta é mitigado. Para estes *pentesting's*, em conjunto com outras ferramentas que ajudam a obter informações do sistema, podem-se utilizar ferramentas como o ZAP (OWASP, 2011), que funciona com funcionalidades de *Man in the Middle* e permite identificar e classificar muitas vulnerabilidades em plataformas web, e pode ser utilizado para verificar se existe informação sensível em trânsito, ou se os *cookies* estão corretamente definidos. Existem também ferramentas que permitem explorar potenciais vulnerabilidades no sistema, tais como o Metasployt (Moore, H. D. (2003) que é uma *framework* de segurança, neste momento da responsabilidade da companhia Rapid7 e é muito útil para identificar fragilidades no sistema. No final dos estágios de desenvolvimento podem-se ainda realizar torneios do género CTF (Capture The Flag) entre diferentes unidades orgânicas da U.Porto, com o intuito de detetar alguma vulnerabilidade que tenha passado desapercebida nos outros testes e até utilizar os padrões de ataque dessas competições para futura monitorização.

Após a aplicação entrar em fase de produção e manutenção, não se pode esquecer a constante monitorização e as auditorias de segurança que são indispensáveis para assegurar que a plataforma é segura enquanto estiver em funcionamento.

Em suma, embora as tarefas relativas à infosec devam ter o seu início, a adoção do paradigma *Secure by design*, obriga a uma constante atualização e monitorização.

4. Considerações finais

Conforme foi abordado ao longo deste trabalho, o desenvolvimento de uma plataforma digital segura não é uma tarefa simples nem de sequência linear. A inclusão do paradigma *Secure by Design* num projeto abarca a junção de diversos princípios, conceitos, metodologias e tecnologias. Fazer a proposta de uma arquitetura segura para uma plataforma digital, implica conhecer as funcionalidades que esta deve ter, qual é o ambiente onde vai funcionar, escolher f*rameworks* e metodologias de desenvolvimento seguras, assim como a escolha de tecnologias a empregar. A escolha de um f*ramework* de segurança que se adapte às necessidades do projeto e da organização permite dividir as tarefas necessárias em etapas distintas que permitirão alcançar os objetivos pretendidos de uma forma mais eficiente.

A análise do ambiente de funcionamento da plataforma e das suas funcionalidades garantem que são identificadas as vulnerabilidades a que o SI está exposto e permite às equipas de desenvolvimento saber onde e que tipos de controles deve implementar, traduzindo-se isto num ganho de qualidade e de tempo de desenvolvimento. Permite, também, fazer uma avaliação do risco que pode orientar a gestão e onde as equipas de desenvolvimento devem ter mais atenção.

Seguir um *framework* de segurança, garante que o projeto, mesmo que ainda na sua fase inicial, segue linhas orientadoras e que todas elas têm a segurança como preocupação central. Embora um *framework* envolva mais do que as atividades de arquitetura, análise e avaliação do risco que compõem o grosso deste trabalho, estas atividades contribuem para o sucesso do projeto. Desde logo, responder às três perguntas a que qualquer sistema de segurança deve dar resposta, identificando quais são os ativos informacionais que se querem proteger; quais são os riscos a que a plataforma e os seus ativos estão expostos; definindo de que forma estes riscos podem ser mitigados, permite alcançar os objetivos de uma forma mais rápida e eficiente, tornando todo o processo mais transparente e confiável.

Em suma, seguir o novo paradigma “Secure by Design” significa que as atividades relacionadas com a segurança da informação têm sempre um início marcado, mas estas nunca terminam. Enquanto durar um projeto, existem sempre tarefas relacionadas com a segurança que devem ser uma preocupação de todos os intervenientes.

Referências Bibliográficas

Agrawal, H. (2019). *Understanding The Difference Between A Database & Blockchain* (CoinSutra, Ed.). https://coinsutra.com/blockchain-vs-database/ Acedido em: 2019/09/15.

Aste, T., & Tasca, P. (2017). *Blockchain Technologies: The Foreseeable Impact on Society and Industry*. https://discovery.ucl.ac.uk/id/eprint/10043048 Acedido em: 2019/12/26.

Combs, G. (1998). *Wireshark*. https://www.wireshark.org Acedido em: 2019/12/30.

CVE Database. (2019). *Security vulnerabilities, exploits, references and more*. https://www.cvedetails.com Acedido em: 2019/04/05.

Davis, N. (2005). *Secure Software Development Life Cycle Processes | CISA*. https://www.us-cert.gov/bsi/articles/knowledge/sdlc-process/secure-software-development-life-cycle-processes Última revisão: 31/01/2013, Acedido em: 2019/02/21.

Davis, N. (2006). *Secure Software Development Life Cycle Processes*. https://www.us-cert.gov/bsi/articles/knowledge/sdlc-process/secure-software-development-life-cycle-processes Última revisão: 31/01/2013, Acedido em: 2019/02/21.

ISO/IEC 27000. (2018). *ISO/IEC 27000 - Information Technology - Security Techniques - Information Security Management Systems - Overview And Vocabulary* [Standard].

Kiran, S., Lareau, P., & Lloyd, S. (2002). *PKI Basics-A Technical Perspective*. http://www.pkiforum.org IN PKI-Forum Acedido em: 2019/12/20.

Lavina, M. E. (2018). *Validação do uso da tecnologia Blockchain para o tráfego seguro de dados na área da saúde*. https://www.riuni.unisul.br/handle/12345/5544 Acedido em: 2019/06/01.

Lyon, G. (1997). *NMAP - The Network Mapper*. https://nmap.org Acedido em: 2019/12/30.

McConaghy, T., Marques, R.,ˈller, A. M., Jonghe, D. D., McConaghy, T., McMullen, G., … Granzotto, A. (2016). *BigchainDB: A Scalable Blockchain Database*. https://mycourses.aalto.fi/pluginfile.php/378362/mod_resource/content/1/bigchaindb-whitepaper.pdf Acedido em: 2019/09/03.

McGraw, G. (2005). *The 7 Touchpoints of Secure Software*. http://www.drdobbs.com/the-7-touchpoints-of-secure-software/184415391 Acedido em: 2019/02/22.

Metivier, B. (2017). *Fundamental Objectives of Information Security: The CIA Triad*. https://www.sagedatasecurity.com/blog/fundamental-objectives-of-information-security-the-cia-triad Acedido em: 2019/02/19.

Moghaddasi, H., Sajjadi, S., & Kamkarhaghighi, M. (2016). *Reasons in Support of Data Security and Data Security Management as Two Independent Concepts: A New Model*. https://www.ncbi.nlm.nih.gov/pubmed/27857823 Acedido em: 2019/12/18.

Moore, H. D. (2003). *METASPLOYT FRAMEWORK* (Rapid7, Ed.). https://github.com/rapid7/metasploit-framework Acedido em: 2019/12/30.

Mougoue, E. (2016). *Secure SDLC 101*. https://www.synopsys.com/blogs/software-security/secure-sdlc/ Acedido em: 2019/02/22.

National Institute of Standards and Technology - Special Publication 800-12 Revision 1. An Introduction to Information Security [Standard]. (2017). US: NIST SP 800-12; https://doi.org/10.6028/NIST.SP.800- 12r1 Acedido em: 2019/10/22.

Nweke, L. O. (2017). *Using the CIA and AAA Models to explain Cybersecurity Activities Commentary*. https://pmworldlibrary.net/wp-content/uploads/2017/05/171126-Nweke-Using-CIA-and-AAA-Models-to-explain-Cybersecurity.pdf Acedido em: 2019/12/20.

Open Web Application Security Project (Ed.). (2016). *OWASP- Category: Vulnerability*. https://www.owasp.org/index.php/Category:Vulnerability Acedido em: 2019/12/28.

Open Web Application Security Project (Ed.). (2017). *OWASP- Security Assurance Maturity Model*. https://www.owasp.org/index.php/OWASP_SAMM_Project Acedido em: 2019/06/15.

Open Web Application Security Project (Ed.). (2018). *OWASP- Top 10 Vulnerabilities in web applications*. https://www.greycampus.com/blog/information-security/owasp-top-vulnerabilities-in-web-applications Acedido em: 2019/04/03.

OWASP. (2011). *OWASP Zed Attack Proxy*. https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project Acedido em: 2019/12/30.

Pinto, M., Medina, S., Matos, R., & Fontes, P. (2016). *U.OpenLab methodology: a conceptual model and flowchart for dynamic co-production and (re)use of digital contents* (I. Publications, Ed.). Seville, SPAIN: 9th International Conference of Education, Research; Innovation. DOI: 10.21125/iceri.2016.2140

Pinto, M., Ribeiro, A., Matos, R., & Medina, S. (2016). *ASH.NET - ATLANTIC SCIENCE HERITAGE NETWORK* (I. Publications, Ed.). Seville, SPAIN: 9th

International Conference of Education, Research; Innovation. DOI: 10.21125/iceri.2016.2054.

Piscini, E., Dalton, D., & Kehoe, L. (2017). *Blockchain & Cyber Security. Let's Discuss* (Deloitte, Ed.). https://www2.deloitte.com/content/dam/Deloitte/us/Documents/financial-services/us-blockchain-and-cyber-security-lets-discuss.pdf Acedido em: 2019/01/17.

Project, O. W. A. S. (Ed.). (2019). *OWASP- Application Security Verification Standard 4.0*. https://www.owasp.org/images/d/d4/OWASP_Application_Security_Verification_Standard_4.0-en.pdf Acedido em: 2019/06/19.

Smartsheet. (2019). *IT RISK ASSESSMENT MATRIX TEMPLATE*. https://www.smartsheet.com/sites/default/files/IC-IT-Risk-Assessment-Matrix-Template-8849_PDF.pdf Acedido em: 2019/04/01.

Stallings, W., & Brown, L. (2015). *Computer Security: Principles and Practice, 3rd Edition*. New Jersey, USA: Pearson Education, Inc.

U.OPENLAB. (2016). *Relatório de Especificação de Requisitos*.

Viega, J., & McGraw, G. (2006). *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley.