UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Elaboração de Fluxos e Guias para uma Ferramenta de Auxílio à Engenharia de Segurança baseada em Texto

Édi Marques Aires

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Pedro R. M. Inácio

Covilhã, outubro de 2019

Agradecimentos

Concluindo esta grande etapa da minha vida, etapa esta que simboliza anos de estudo, esforço, trabalho e, sobretudo, o ganho de muito conhecimento, gostaria de expressar a minha gratidão a algumas pessoas especiais, reconhecendo assim todo o apoio que me foi facultado, de forma a que fosse possível alcançar este grande objetivo pessoal.

Agradeço ao meu orientador, Professor Doutor Pedro R. M. Inácio, por todo o apoio prestado, por me dar a oportunidade de trabalhar, compartilhar ideias e aprender com ele durante a realização desta dissertação.

Um agradecimento especial aos meus pais e à minha família pela possibilidade de evoluir nos meus estudos, pela força e apoio incondicional que me deram durante todos estes anos, servindo de suporte não só financeiramente mas também emocionalmente, auxiliando-me na tomada de decisões importantes da minha vida.

Sem esquecer da minha segunda família que fiz durante os anos universitários, quero deixar um enorme agradecimento a todos eles, nomeadamente ao José Pascoal, Nelson Fonseca, Paulo Neto, Raul Barbosa e Guilherme Boino, por toda a ajuda, força e paciência, e que ao longo desta grande caminhada estiveram sempre presentes em todos os bons e menos bons momentos, sempre a apoiar e a acreditar nas minhas capacidades.

Por fim, agradeço aos restantes amigos ubianos pela amizade e ajuda e que, de uma forma ou de outra, entraram na minha vida, e agradeço também a todos os professores que fizeram parte do meu percurso académico, e que enriqueceram o meu conhecimento na área da informática.

Mais uma vez, um grandioso obrigado a todos aqueles por estarem presentes na minha vida e que sempre me acompanharam na minha caminhada!

Resumo

Atualmente, *security by design* tornou-se um desafio notável, ou seja, a criação de sistemas de software que sejam seguros desde as etapas iniciais do desenvolvimento apresenta-se como uma tarefa difícil. Infelizmente, muitos dos sistemas e software disponíveis são desenvolvidos sem preocupações acerca da segurança ou apenas seguem um processo de engenharia de segurança fraco, nomeadamente sistemas pertencentes à área de *Internet of Things (IoT)*.

Este novo mundo inovador que começa a surgir e a ganhar destaque nas tecnologias do futuro possibilita que muitos dispositivos, alguns deles sendo retratados como objetos físicos da vida quotidiana com computadores embutidos, apresentem uma conexão entre eles através da *Internet*, transmitindo dados automaticamente. Contudo, a conexão de todos estes modernos dispositivos e sistemas tem um efeito colateral: um maior risco no que diz respeito à existência de potenciais ataques e vulnerabilidades.

A implementação destes tipo de sistemas é complexa e árdua e, devido a estes fatores, a integração da segurança por vezes é ignorada ou adicionada no final do desenvolvimento como um recurso extra. A consequência destas ações traduz-se na existência de erros / *bugs* desde as fases iniciais, que podem levar entidades maliciosas a explorar as vulnerabilidades existentes, provocando assim falhas de seguranças, algumas das quais podem gerar um impacto muito negativo. De forma a colmatar este problema, existe uma clara necessidade de encontrar uma solução que permita guiar arquitetos de sistema, fabricantes e programadores a integrar e implementar segurança no sistema desde o início do desenvolvimento.

Deste modo, esta dissertação descreve o trabalho direccionado para o *design* e desenvolvimento de uma ferramenta, na linguagem de programação *Python* que, através de perguntas simples e intuitivas, produza um relatório contendo boas práticas de segurança sobre vários temas a adotar durante a implementação de um sistema. O trabalho inclui a identificação das perguntas e a definição do fluxo de execução. Esta ferramenta possibilita o fornecimento de documentação necessária para aplicar um nível considerado de segurança no sistema a implementar desde as etapas iniciais, auxiliando assim arquitetos de sistema e *developers*, mesmo aqueles que têm pouco ou nenhum conhecimento de segurança. De uma forma rápida e simplista, esta ferramenta permite que o utilizador responda a uma série de questões sobre o sistema a desenvolver e, em troca, receba documentação estruturada sobre boas práticas seguras a adotar, possibilitando assim a redução de vulnerabilidades, e servindo como um utensílio de apoio no que diz respeito à integração de segurança de um sistema de *software*.

Palavras-chave

Boas Práticas, Engenharia de Segurança, Guias de Segurança, Internet das Coisas, Modelação de Ameaças, Segurança de Sistemas

Abstract

Security by design has now become a remarkable challenge, which means that creating software systems that are secure from early stages of development is a daunting task. Unfortunately, many of the available software systems are always developed without security concerns or just follow a weak security engineering process, namely systems that belong to the Internet of Things (IoT) area.

This new world that is emerging and gaining prominence in technologies of the future enables many devices, some of them being portrayed as physical objects of everyday life with embedded computers, to have a connection between them over the Internet and transmit data automatically. However, the connection of all these modern devices and systems has one side effect: a greater risk for potential vulnerabilities.

Implementing these types of systems is complex and arduous, and because of these factors, security integration is sometimes ignored or added at the end of development as an extra feature. The consequence of these actions is the existence of errors / bugs from the early stages, which may lead to malicious entities exploiting existing vulnerabilities, thus leading to security breaches, some of which may have a very negative impact. In order to address this problem, there is a clear need to find a solution that will guide system architects and developers to integrate security into a system to be implemented from the beginning of development.

Thus, this dissertation describes the work directed to the design and development of a tool, in Python, which, through simple and intuitive questions, produces a report containing good security practices on various topics, to be adopted during implementation of a system. This work included the identification of the questions and the definition of the execution flow. By assisting system architects and developers, even those with little or no security knowledge, this tool provides the necessary documentation to apply a considered level of security into the system from the earliest stages of development. Quickly and simply, this tool allows the user to answer a set of questions about the system to be developed, and in return, receive structured documentation of best practices to adopt, thereby reducing vulnerabilities. and serving as a support tool for security integration in a software system.

Keywords

Best Practices, Internet of Things, Security Engineering, Security Guides, Systems Security, Threat Modeling,

Conteúdo

Lista de Figuras

Listings

Lista de Tabelas

Acrónimos

ACM Association for Computing Machinery

AEGIS Appropriate and Effective Guidance in Information Security

API Application Programming Interface

ASF Application Security Frame

CCS Computing Classification System

CLASP Comprehensive Lightweight Application Security Process

CPE Common Platform Enumeration

CVE Common Vulnerabilities and Exposures

CVSS Common Vulnerability Scoring System

DDoS Distributed Denial of Service

DVD Digital Versatile Disc

GPS Global Positioning System

HTML HyperText Markup Language

IoT Internet of Things

IP Internet Protocol

OVVL Open Weakness and Vulnerability Modeler

OWASP Open Web Application Security Project

PDF Portable Document Format

PNG Portable Network Graphics

SDL Security Development Lifecycle

STRIDE Spoofing Tampering Repudiation Information disclosure Denial of service Elevation of privilege

SVG Scalable Vector Graphics

UBI Universidade da Beira Interior

UML Unified Modeling Language

Wi-Fi Wireless Fidelity

Capítulo 1

Introdução

Este documento descreve o trabalho realizado no âmbito de um projeto de dissertação para obtenção do grau de mestre em Engenharia Informática na Universidade da Beira Interior (UBI). Esta dissertação aborda a segurança no desenho de um sistema de software, em que é proposta uma solução que tem como objetivo identificar o sistema e produzir informações relativas a boas práticas de segurança, de modo a auxiliar *developers*, mesmo que estes não tenham experiência na área de segurança. A secção que se segue apresenta a motivação e o enquadramento deste trabalho. As secções 1.2 e 1.3 retratam a definição do problema, objetivos e a solução proposta. A secção 1.4 retrata as principais contribuições e, por fim, a última secção descreve a estrutura desta dissertação.

1.1 Enquadramento e Motivação

A próxima geração da Internet traduz-se numa grande conectividade em rede entre diversas aplicações e dispositivos nos vários domínios da sociedade humana - transporte, telecomunicações, entretenimento, saúde, militar, educação, entre outros. Estes sistemas modernos tornam-se cada vez mais críticos e complexos e, devido a esta elevada complexidade da elaboração destes sistemas, a integração de segurança tornou-se um grande desafio. Embora seja importante garantir que os sistemas de software sejam desenvolvidos de acordo com as necessidades do utilizador, é igualmente importante garantir que esses sistemas sejam também seguros.

Nos dias de hoje, a segurança desempenha um papel central no desenvolvimento de aplicações em larga escala. No entanto, uma análise aos processos de desenvolvimento de software revela que a engenharia de segurança no desenho geral do sistema é frequentemente negligenciada. Esta questão prende-se, em grande medida, com a forma como os requisitos são divididos durante o desenvolvimento de um software, em duas partições: requisitos funcionais e requisitos não funcionais, ou seja, funcionalidades e características. A segurança é classificada como uma característica e, como no desenvolvimento do software há uma maior prioridade nos requisitos funcionais relativamente a não funcionais, a segurança não é completamente integrada no produto final [Bay11]. O foco principal passa por começar a desenvolver e adicionar funcionalidades conforme for necessário, e só posteriormente é que a segurança é implementada no processo de desenvolvimento como um recurso extra, ou é apenas ignorada devido a condições de custo ou eficiência [GRMNT10]. Durante o desenvolvimento do software, a implementação da segurança é frequentemente tratada como um acréscimo, o que implica que existe uma falta de apoio na aplicação de segurança no processo de engenharia do sistema. Esta ausência é geralmente vista como consequência de [SR+19]:

- A integração da segurança no processo de desenvolvimento de software não é bem compreendida;
- Haver uma falta de ferramentas que suportam engenharia de segurança;
- Falta de experiência dos especialistas em segurança.

Internet of Things (IoT), ou Internet das Coisas, é agora um dos tópicos mais falados no contexto das tecnologias da informação e comunicação. A visão da IoT visa a interconexão de objetos físicos na vida quotidiana com a Internet de uma maneira efetiva e prática. Envolvendo a predominância de objetos e entidades com a capacidade de transferir dados pela rede automaticamente, o principal problema na criação sistemas na área de IoT está também relacionado com a fraca integração de mecanismos segurança [LL17].

Ao construir este tipo de sistemas, a segurança deve ser um dos aspetos mais importantes a ter em conta, e quando esta não é considerada com a respetiva importância, muitos dispositivos apresentam vulnerabilidades. Na área de IoT, uma implementação fraca no que diz respeito à integração de um nível de segurança poderá levar a violações de dados e incidentes, tais como alguns que são descritos a seguir:

- ***The Mirai Botnet*** – Em Outubro de 2016, ocorreu um dos maiores ataques de *Distributed Denial of Service (DDoS)* lançado num fornecedor de serviços (*Dyn*), usando uma *botnet* IoT, fazendo com que grandes plataformas como o *Twitter*, *Netflix*, *Reddit* e *CNN* ficassem em baixo. Uma vez infetados pelo *malware Mirai*, os computadores procuravam continuamente na *Internet* por dispositivos IoT vulneráveis (e.g., *routers*, câmaras digitais e *players* de *Digital Versatile Disc (DVD)*) e, em seguida, usavam nomes de utilizadores e senhas padrão de forma a ter acesso aos equipamentos [HKZ18, KKSV17, KKS17];
- ***The Hackable Cardiac Devices from St. Jude*** – Dispositivos cardíacos do hospital St. Jude, que permitiam a monitorização e controlo de funções cardíacas dos pacientes e preveniam ataques cardíacos, apresentavam uma vulnerabilidade no transmissor que lia os dados do dispositivo e os partilhava remotamente com o médico. A exploração desta vulnerabilidade permitia o acesso ao dispositivo cardíaco, em que depois era possível o *hacker* provocar o esgotamento da bateria, ou até mesmo administrar um ritmo incorreto de choques, o que poderia levar a um resultado catastrófico [CNN17];
- ***The TRENDnet Webcam Hack*** – A empresa *TRENDnet* comercializava câmaras de segurança para vários usos, desde segurança doméstica até monitorização de bebés, afirmando que eram seguras. No entanto, o software tinha uma falha de segurança que permitia que qualquer pessoa que obtivesse o endereço *Internet Protocol (IP)* de uma câmara conseguisse obter as credenciais de *login* do equipamento, visto que estas eram transmitidas em texto limpo pela *Internet* [Wir12];
- ***The Jeep Hack*** – Uma equipa de investigadores conseguiu assumir o controlo de um *Jeep SUV* usando uma rede interna de controlo do veículo. Ao explorar uma vulnerabilidade na atualização do *firmware*, os investigadores descobriram que podiam acelerar, travar o veículo, e até mesmo fornecer indicações *Global Positioning System (GPS)* para mudar de direção para fora da estrada [Kas15].

Através destes exemplos, verifica-se que a segurança implementada apresentava um nível baixo mas também, com alguma facilidade, se corrigiam estas vulnerabilidades se, por exemplo, existissem guias contendo práticas de segurança a adotar durante a implementação do sistema, ou até mesmo uma ferramenta que produzisse este tipo de informação. A forma/conteúdo dessa informação poderia ser semelhante ao seguinte: os dispositivos que não podem ter o software, *passwords* ou *firmware* atualizados nunca devem ser usados; a alteração do nome de utilizador e *password* padrão deve ser obrigatória; as *passwords* dos dispositivos IoT devem ser exclusivas por dispositivo; ou proteja os dispositivos IoT com as atualizações mais recentes de software e *firmware* para atenuar as vulnerabilidades. Se a entidade responsável pela elaboração do sistema tivesse na sua posse este tipo de documentação que auxiliasse a integração de segurança no sistema, estas falhas não estariam presentes no produto final, evitando assim ataques ao sistema.

Este projeto encontra-se enquadrado na interseção das áreas das redes (nomeadamente IoT), da segurança e da engenharia de sistemas e software. Usando a versão 2012 do sistema de classificação da *Association for Computing Machinery (ACM)* (conhecido por *Computing Classification System (CCS)*), o âmbito do projeto de mestrado refletido nesta dissertação enquadra-se nas seguintes categorias:

- **Security and privacy∼Systems security**;
- **Security and privacy∼Network security**;
- **Software and its engineering∼Software notations and tools**;
- *Networks*.

1.2 Definição do Problema e Objetivos

O problema principal abordado nesta dissertação está relacionado com a integração de segurança em sistemas de software logo no início de desenvolvimento do mesmo. Infelizmente, tem sido demonstrado que, cada vez mais, muitos sistemas e softwares são desenvolvidos sem preocupações de segurança, ou seguem um fraco processo de engenharia de segurança durante a sua elaboração. O foco principal na implementação dos sistemas de software refere-se às funcionalidades que estes apresentam, sendo a segurança uma característica que não é aplicada em estágios iniciais do desenvolvimento do sistema. Ao não serem aplicadas inicialmente, as medidas de segurança relevantes acabam por ser implementadas *a posteriori*, o que significa um acréscimo de custo e dificuldade.

"*Security by design*" tornou-se um grande desafio na atualidade, visto que a criação de novos sistemas modernos envolve uma maior complexidade, como é o caso de sistemas pertencentes à área de *IoT*, em que muitos dispositivos e entidades se encontram ligados à *Internet*, transmitindo dados. Constituindo desafios adicionais, sistemas nestas áreas inovadoras implicam que os *developers* necessitem de duplicar os seus esforços, não só na própria construção do sistema, mas também na segurança do mesmo. Infelizmente, devido à omnipresença e importância de

alguns destes sistemas modernos, não existem muitas ferramentas que auxiliem a integração de segurança, o que leva a que a implementação de segurança seja, por vezes, fraca. Desta forma, estes sistemas são um alvo atraente para invasores e, portanto, é necessário ter técnicas e ferramentas que permitam a análise e incorporação de segurança no sistema para que se possam abordar potenciais vulnerabilidades.

Muitas das entidades responsáveis pela elaboração deste tipo novo de sistemas não manifestam um conhecimento profundo de segurança. Deste modo, o objetivo principal desta dissertação passa por criar uma ferramenta simplista e intuitiva capaz de fornecer documentação que facilite arquitetos de sistemas, mesmo aqueles que não têm conhecimentos em segurança informática, na criação de sistemas de software com uma segurança mais significativa. Para alcançar este grande objetivo é necessário identificar um conjunto de questões simples com determinadas opções, bem como a recolha de documentação referente a boas práticas de segurança e, transformar este agrupamento de informações num software. A utilização desta ferramenta permite, através das respostas selecionadas, a produção de documentação sobre boas práticas de segurança a ter conta no desenvolvimento do sistema. Assim, com base nas informações fornecidas pela ferramenta, é garantido que o sistema é seguro a partir das primeiras etapas da elaboração do mesmo.

1.3 Abordagem Adotada para a Resolução do Problema

A escolha da abordagem para solucionar o problema mencionado começou com um estudo ao processo de engenharia de segurança do software abordando as metodologias existentes. De seguida procedeu-se à procura de ferramentas e técnicas existentes que fazem parte da fase de *design* do desenvolvimento do software, como por exemplo ferramentas que auxiliam na modelação de ameaças de um sistema. Após a conclusão deste estudo, investigaram-se e recolheram-se boas práticas de segurança sobre diversos temas. O próximo passo traduziu-se na identificação de questões e respetivas opções, de modo a que seja possível identificar os componentes e características do sistema a desenvolver, e também que a formulação das questões seja de fácil compreensão. Após realizada a investigação necessária, deu-se início à implementação de um protótipo da ferramenta, que através das opções que o utilizador selecione às perguntas colocadas, é produzido um relatório sobre práticas de segurança a adotar. Finalmente, efetuaram-se testes à ferramenta aplicando alguns cenários de utilização e verificaram-se os resultados obtidos.

1.4 Principais Contribuições

A ideia por detrás desta dissertação está relacionada com o desenvolvimento de uma ferramenta que, através de questões simples e intuitivas sobre características de um sistema colocadas ao utilizador, forneça documentação sobre práticas de segurança a adotar no desenvolvimento. Deste modo, o trabalho descrito ao longo deste documento pode contribuir para a segurança de

sistemas de software, em que as principais contribuições obtidas com a investigação e desenvolvimento desta ferramenta podem ser enumeradas da seguinte forma:

- O desenho, desenvolvimento e validação de um software simples, baseado a partir de um conjunto de perguntas, que produz guias sobre práticas de segurança abordando vários temas dependendo das opções seleccionadas pelo utilizador;
- A aplicação desta ferramenta em etapas iniciais de desenvolvimento do software, nomeadamente a fase de desenho, pode auxiliar arquitetos de sistemas e programadores, mesmo aqueles que não possuem conhecimentos de segurança informática, a implementar sistemas de software mais seguros;
- A inserção desta ferramenta num conjunto de outras ferramentas que estão a ser desenvolvidas no âmbito de um projeto de investigação e desenvolvimento denominado SECUR IoT ESIGN, permite que este agrupamento de ferramentas forneça um apoio considerável no que diz respeito à integração de segurança em sistemas de software, nomeadamente sistemas IoT.

1.5 Organização da Dissertação

Esta dissertação está organizada em seis capítulos e nove anexos que são resumidamente descritos a seguir:

- O Capítulo 1 – **Introdução** – introduz o enquadramento e a motivação do trabalho descrito neste documento, bem como a exposição do problema, os objetivos principais e a solução proposta. Inclui também as principais contribuições e a organização da dissertação;
- O Capítulo 2 – **Revisão do Estado da Arte e Trabalhos Relacionados** – aborda metodologias referentes ao processo de engenharia de segurança do software, técnicas e ferramentas existentes que fazem parte da fase de *design* do desenvolvimento, nomeadamente na área da modelação de ameaças. Engloba também a descrição de tabelas resumo (*cheatsheets*, do inglês) sobre segurança que contém boas práticas seguras;
- O Capítulo 3 – **Desenho do Fluxo da Ferramenta** – identifica as questões e as opções que fazem parte da ferramenta, bem como a respetiva estruturação das perguntas durante a execução da ferramenta;
- O Capítulo 4 – **Implementação da Ferramenta** – engloba a descrição das entradas e saídas da ferramenta, a descrição dos detalhes da implementação e a demonstração e validação da ferramenta;
- O Capítulo 5 – **Cenários de Utilização da Ferramenta** – retrata a identificação de alguns cenários de utilização, a aplicação da ferramenta aos cenários e a discussão dos resultados obtidos;
- O Capítulo 6 – **Conclusões e Trabalho Futuro** – apresenta as conclusões principais desta dissertação de mestrado bem como uma lista de possíveis melhoramentos futuros que podem ser incorporados neste trabalho;

- O Anexo 1 – **Fluxo da Ferramenta** – demonstra todo o fluxo de questões e respectivas respostas durante a execução da ferramenta;
- O Anexo 2 – **Função que Permite a Validação de *Input*** – retrata a função que permite efetuar a validação de entrada de dados por parte do utilizador;
- O Anexo 3 – **Algoritmo para Associar as Respectivas Respostas às Opções Escolhidas** – traduz o código que permite a associação das opções escolhidas pelo utilizador às respetivas respostas das questões;
- O Anexo 4 – **Excerto da Secção sobre Práticas Seguras sobre Autenticação** – engloba um excerto da secção de práticas seguras sobre autenticação que pertence ao relatório final produzido pela ferramenta;
- O Anexo 5 – **Excerto da Secção sobre Práticas Seguras sobre Validação de Entradas** – apresenta um excerto da secção de práticas seguras sobre validação de entradas que fazem parte do relatório final;
- O Anexo 6 – **Excerto da Secção sobre Práticas Seguras sobre *Cross Site Scripting*** – exibe um excerto da secção de práticas seguras sobre *Cross Site Scripting*;
- O Anexo 7 – **Excerto da Secção sobre Práticas Seguras sobre *Logging* e Gestão de Erros** – mostra um excerto da secção de práticas seguras sobre *Logging* e Gestão de Erros;
- O Anexo 8 – **Excerto da Secção sobre Práticas Seguras sobre *IoT (Embbebed Systems)*** – expõe um excerto da secção de práticas seguras sobre sistemas embutidos (dispositivos IoT); e
- O Anexo 9 – **Exemplo de um Ficheiro com as Respostas e Instruções de como Elaborar este Ficheiro** – apresenta o formato do ficheiro gerado com as opções seleccionadas pelo utilizador bem como instruções de como elaborar este ficheiro caso o utilizador assim o desejar.

Capítulo 2

Revisão do Estado da Arte e Trabalhos Relacionados

2.1 Introdução

Existem algumas metodologias e ferramentas que abordam o desenho de sistemas seguros mas, na maioria dos casos, oferecem pouca orientação sobre como a segurança pode ser integrada no desenvolvimento do software. Deste modo, este capítulo aborda diversos trabalhos relacionados sobre o assunto retratado nesta dissertação, *security by design*. A secção 2.2 apresenta metodologias que fazem parte da engenharia de segurança do software, a secção 2.3 refere-se a ferramentas existentes que se focam na etapa de desenho do ciclo de vida do desenvolvimento do software e, por fim, a secção 2.4 retrata alguns projetos e documentação que se relacionam com boas práticas de segurança a serem aplicadas em várias áreas.

2.2 Segurança em Engenharia de Software

A engenharia de software descreve o processo de desenvolvimento e gestão de sistemas de uma forma sistemática e disciplinada em que o trabalho é dividido em quatro áreas principais, análise de requisitos, desenho do sistema, implementação e testes. Tratando-se de um processo difícil e consumidor de tempo, é comum ocorrerem erros durante o desenvolvimento do software [Vli08], e erros que dizem respeito a falhas de segurança podem causar consequências fatais.

Sendo o principal objetivo a introdução de segurança como parte do processo de desenvolvimento de software, ao longo de vários anos diferentes modelos foram propostos, de modo a estruturar e gerir o ciclo de vida do desenvolvimento integrando aspetos de segurança. Exemplos de alguns modelos mais comuns e conhecidos são: modelo em cascata, modelo em espiral, métodos ágeis, modelo de desenvolvimento seguro da Microsoft (*Security Development Lifecycle (SDL)*) e o processo de segurança de aplicações da *Open Web Application Security Project (OWASP) — Comprehensive Lightweight Application Security Process (CLASP)*.

O modelo em cascata, proposto por Winston Royce [Roy87], descreve um método de desenvolvimento linear que é frequentemente considerado uma abordagem clássica do ciclo de vida de desenvolvimento do software. A ideia principal deste modelo reside na interação entre as diversas fases de uma forma sequencial, o que significa que uma fase de desenvolvimento deve ser concluída antes da próxima começar. A figura 2.1 representa a visão esquemática deste método de desenvolvimento, e este engloba as seguintes fases: requisitos de sistema, requisitos

de software, análise, desenho, codificação, testes e operação. De modo a desenvolver software seguro, aspetos de segurança foram considerados nas diferentes fases do modelo. Estes aspetos referem-se a atividades de engenharia de segurança como requisitos gerais de segurança do sistema, definição de casos de abuso ou casos de uso indevido — casos de uso nos quais as interações são causadas por um atacante e não por um utilizador legítimo — mecanismos de autenticação, de proteção de confidencialidade e integridade de dados, produção de cenários de ataque e deteção de vulnerabilidades.

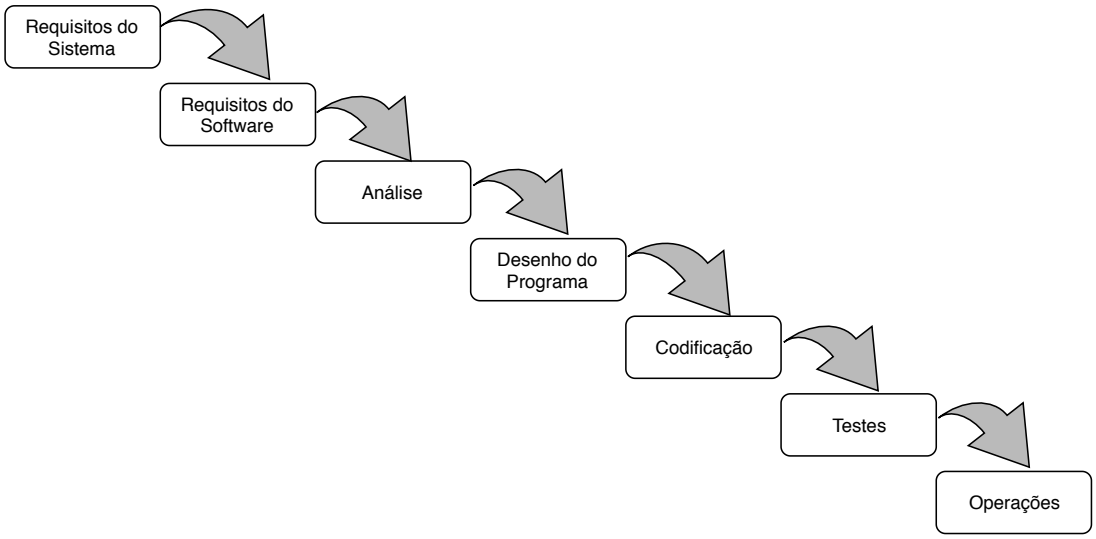


Figura 2.1: Esquema do modelo de desenvolvimento de software conhecido por *modelo em cascata* obtido de [Roy87].

O modelo em espiral surge como um aprimoramento do modelo em cascata, tentando superar as falhas deste, e foi proposto por Barry Boehm [Boe88]. Este modelo parte da constatação de que raramente o software é desenvolvido de uma forma linear, como proposto pelo modelo descrito anteriormente. Pelo contrário, durante o processo de desenvolvimento é habitual surgirem sucessivos protótipos, que vão sendo melhorados até se chegar ao produto, ou sucessivas versões do produto final que vão evoluindo. De forma a integrar aspetos de segurança, esta abordagem de desenvolvimento iterativo é baseada nas seguintes atividades: identificação de objetivos, alternativas e restrições, análise de risco, onde é efetuada a modelação de ameaças, desenvolvimento e avaliação do produto, e por fim a produção do protótipo. A figura 2.2 demonstra as fases deste método de desenvolvimento de software.

No desenvolvimento de software, muitas empresas estão cada vez mais a adotar a metodologia de desenvolvimento ágil. Devido a este facto, a *Microsoft* apresentou uma extensão do SDL para este tipo de desenvolvimento: SDL-Agile [Mic09]. Estes processos de desenvolvimento ágil concentram-se no conceito de *sprints*, ou seja, de um período de 2 a 4 semanas, no qual é desenvolvido um conjunto de funcionalidades designadas por *stories*. O desenvolvimento do produto traduz-se numa sequência de *sprints* que concretizam as *stories* que se encontram num repositório do produto (*product backlog*). Visto que cada *sprint* é curto e não existe a sequência de passos do modelo em cascata, é necessária uma adaptação ao SDL original de modo a integrar segurança no processo de desenvolvimento do software. No SDL-Agile são realizadas tarefas com foco em segurança que são colocadas no repositório, em que a execução destas tarefas pode variar entre uma só vez, regularmente ou em todos os *sprints*. Exemplos destas tarefas são:

requisitos de segurança, tarefas de planeamento, revisão do desenho, tarefas de verificação, análise de ameaças, análise estática do código e validação e codificação de entradas e saídas.

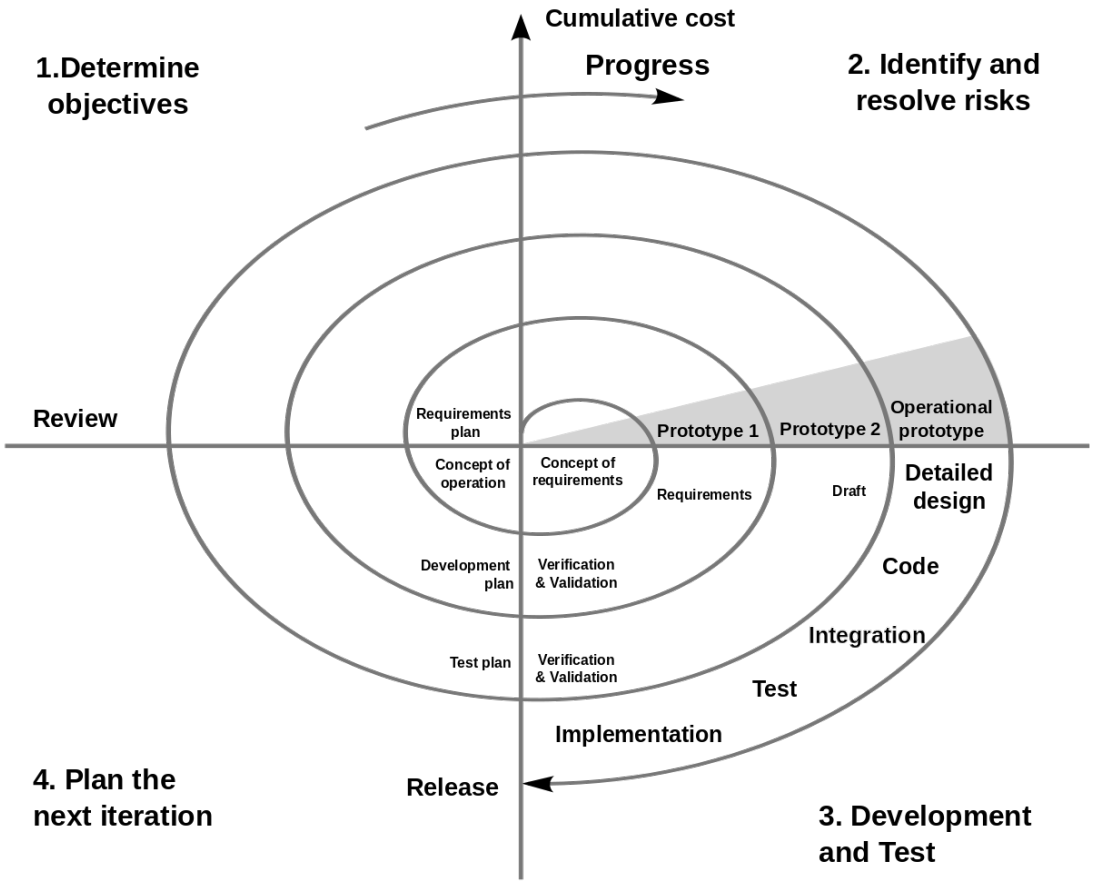


Figura 2.2: Esquema do modelo de desenvolvimento de software conhecido por *modelo em espiral* retirado de [Boe88].

Entre todos os modelos de desenvolvimento de software seguro, o mais conhecido na atualidade é, provavelmente, o Microsoft SDL. Este modelo desenvolvido pela Microsoft é direcionado para o desenvolvimento de software com o objetivo de reduzir o número de vulnerabilidades no software [Micb]. O SDL engloba um conjunto de atividades que abordam aspetos de segurança, agrupadas em sete fases que podem ser mapeadas para as fases padrão de desenvolvimento de software. Estas fases traduzem-se nas seguintes:

- **Pré-fase - Formação** — preparação dos membros da equipa de desenvolvimento de software para lidarem com os aspetos de segurança, incluindo o estudo de conceitos e técnicas de modo a integrar segurança no software;
- **Fase de Requisitos** — compreensão das necessidades do software a desenvolver, em que algumas das atividades desta fase correspondem à identificação das pessoas responsáveis pela coordenação de questões de segurança do projeto, assegurar a existência de uma ferramenta de seguimento de *bugs* adequada à gestão de vulnerabilidades e definição dos requisitos mínimos de segurança;
- **Fase de *Design*** — elaboração de um plano para a concretização da segurança durante o resto do desenvolvimento do software. Esta fase apresenta atividades como a projeção

de mecanismos de segurança a implementar, seguindo recomendações e boas práticas de modo a evitar vulnerabilidades no projeto, a realização de uma análise de risco de segurança e uma análise de ameaças, de forma a identificar as mais críticas sob o ponto de vista da segurança e privacidade;

- **Fase de Codificação** — implementação do software seguindo boas práticas de codificação, evitando assim a criação de vulnerabilidades comuns, produzindo também documentação sobre como os utilizadores devem configurar o software de forma segura;

- **Fase de Verificação** — consiste em garantir que o software forneça as propriedades de segurança que foram propostas. Esta fase engloba as abordagens de revisão manual do código e de realização de testes;

- **Fase de Publicação** — disponibilização do software aos utilizadores. Do ponto de vista da segurança do software, esta fase consiste no planeamento de medidas a tomar quando vulnerabilidades forem descobertas;

- **Pós-Fase - Resposta** — trata-se da recolha de informação sobre incidentes de segurança no projeto, criação de relatórios de vulnerabilidades e, por fim, corrigir as respetivas falhas de segurança.

Tratando-se de um processo leve com finalidade de criar softwares seguros, o modelo CLASP fornece uma metodologia estruturada para derivar requisitos de segurança de software [Vie05]. Este modelo define práticas essenciais e recomendadas, tais como: instituir programas de formação, realizar avaliações de aplicações, identificação de requisitos de segurança, implementar práticas de desenvolvimento seguras e criar procedimentos de correção de vulnerabilidades. Além destas práticas, este modelo também define um conjunto de atividades que devem ser integradas no processo de desenvolvimento do software.

Outro modelo que também incorpora atividades de segurança no desenvolvimento de software é o modelo *Appropriate and Effective Guidance in Information Security (AEGIS)*, desenvolvido por investigadores da *University College London*, que utiliza um modelo em espiral que visa dar apoio à equipa de desenvolvimento do software na abordagem aos requisitos de segurança e usabilidade no projeto [FMS07]. Neste modelo, as principais atividades de segurança para o design do software são descritas como se segue: identificação de recursos (do inglês *assets*) e requisitos de segurança, análise de risco e identificação de vulnerabilidades e ameaças ao sistema. Essas atividades resultam num documento de desenho que contém a arquitetura do sistema com todas as contra-medidas identificadas, sob o ponto de vista de aspetos de segurança. A principal desvantagem do AEGIS refere-se ao facto dos especialistas em segurança não se encontrarem envolvidos no processo de desenvolvimento do software.

McGraw salienta que a segurança pode ser intercalada em processos de desenvolvimento existentes e propõe as melhores práticas de segurança de software [McG06] em sete pontos de contacto, como se mostra na figura 2.3. Este sete pontos demonstram o que os engenheiros de software podem aplicar no processo de desenvolvimento. Encontram-se presentes atividades de segurança como análise de requisitos de segurança, análise de risco, revisão de código e testes de penetração.

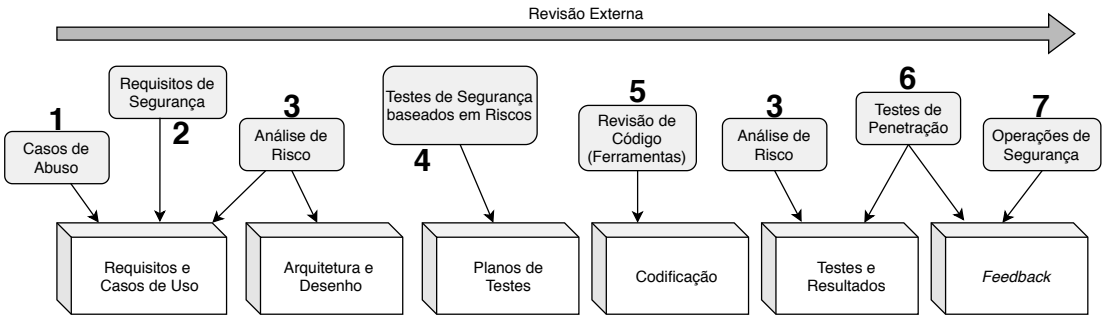


Figura 2.3: Sete pontos de McGraw que permitem a integração de segurança no processo de desenvolvimento do software obtido em [McG06].

Após a análise de vários modelos de desenvolvimento de software que integram a segurança durante todas as etapas do ciclo de vida, podem definir-se as seguintes atividades para cada fase do ciclo:

- **Fase de Pré-Estudo** — a fase de pré-estudo do desenvolvimento do software inicia o projeto consistindo na determinação da natureza do problema ou necessidade do cliente e engloba atividades como: identificação de *stakeholders* (partes interessadas), identificação dos objetivos e requisitos dos *stakeholders*, identificação de documentos e guias de transporte, definição do nível de segurança do software e identificação dos *assets*;

- **Fase de Requisitos** — esta fase diz respeito à elucidação dos requisitos do software de forma segura, ou seja, enunciar requisitos que se referem às capacidades de um software reter propriedades de segurança. As atividades que se encontram nesta fase são as seguintes: análise de requisitos, modelação de ameaças, geração de possíveis cenários de ataques, análise de riscos e, por fim, verificação e validação dos requisitos de segurança;

- **Fase de Desenho Seguro** — a fase de desenho seguro descreve a identificação dos elementos de software, como protocolos, sistemas operativos e aplicações. Incorpora considerações de segurança no processo de desenvolvimento do software a nível do desenho que envolve uma arquitetura de segurança física, lógica e de componentes (entidades, serviços de segurança, modelos de dados e mecanismos de segurança);

- **Fase de Implementação Segura** — atividades como codificação seguindo boas práticas de segurança e a análise da segurança do código do software fazem parte desta fase, e permitem a identificação de *bugs* de segurança bem como falhas de segurança no software;

- **Fase de Testes de Segurança** — por fim, esta última fase engloba vários testes de segurança, tais como testes de funcionalidade, testes de penetração, testes de validação e testes de aceitação. Esta fase de testes que aborda aspetos de segurança do software e compreende vários cenários negativos, isto é, enfatiza o que o software não deve fazer e determina o comportamento deste durante um ataque.

2.3 Ferramentas Disponíveis

Antes de se dar início à produção de uma nova ferramenta, é necessário efetuar uma pesquisa sobre ferramentas existentes que se identificam com o auxílio de integração de segurança durante o desenvolvimento do software bem como a modelação de ameaças ao sistema. O estudo destas ferramentas relacionadas com o tema desta dissertação, integração de segurança aquando o desenho de sistemas de software, é descrito nos seguintes parágrafos desta secção.

Unified Modeling Language (UML) é uma linguagem para especificar, construir, visualizar e documentar um desenho do software [MRRR02]. É sobretudo adequada para o paradigma orientado a objetos. Sendo uma notação padrão em engenharia de software, esta linguagem pode ser usada de três formas diferentes: criar rascunhos iniciais dos principais componentes do sistemas durante os estágios iniciais do projeto; gerir grandes sistemas de software mantendo o projeto e a implementação sincronizados; e documentar extensivamente o software após a sua implementação. De forma a modelar e a integrar aspetos de segurança no desenvolvimento do software, surgiram ferramentas de apoio com base na linguagem UML.

UMlet [ATB03] é descrita como uma ferramenta UML gratuita, de código aberto (*open-source*, do inglês) e com uma interface de utilizador simples que permite a criação de diagramas UML. Os elementos UML que se criam podem ser modificados e ser usados como modelos. Deste modo, os utilizadores podem adaptar facilmente a criação de diagramas conforme as suas necessidades de modelação, como por exemplo, a modelação de aspetos de segurança no desenvolvimento de um software.

Jürjens propôs uma extensão para a UML [Jü02], denominada UMLsec, com o propósito de modelar propriedades de segurança de um software como confidencialidade, integridade e controlo de acesso. Esta ferramenta fornece requisitos básicos de segurança, como confidencialidade e integridade, permite considerar diferentes cenários de ameaça e incorpora mecanismos de segurança, como por exemplo, controlos de acesso.

SecureUML [LBD02] descreve-se como outra extensão da UML e é focado em políticas de controlo de acesso no desenvolvimento do software. Esta extensão introduz a possibilidade criar diagramas em UML com conceitos referentes ao controlo de acesso tais como, o utilizador, a sua função e a sua permissão bem como os relacionamentos entre eles.

Padrões de segurança (*security patterns*, do inglês) são descritos como soluções reutilizáveis que consistem num conhecimento especializado na área da segurança, ou seja, visam ser práticas recomendadas para evitar falhas do desenho relacionadas com a segurança do software. Os padrões são definidos como uma descrição de um problema específico dentro de um contexto com uma solução genérica aprovada, resultando numa forma de encapsular o conhecimento acumulado sobre o desenho de sistemas seguros [SFHB05]. Estes padrões são incluídos em aspetos da segurança, tais como autenticação, autorização, controlo de acesso, *firewalls*, segurança de serviços *Web*, entre outros. Durante anos, os *padrões de segurança* comprovaram a sua utilidade e demonstraram que o seu uso, especialmente a nível da arquitetura do software, oferece vantagens importantes. Contudo, a crescente complexidade e heterogeneidade na criação de softwares e sistemas modernos provocou uma degradação na utilidade e aplicabilidade dos *pa-*

drões de segurança. Atualmente, os *padrões de segurança* fornecem um suporte limitado na integração dos sistemas em desenvolvimento, resultando em diferentes problemas que podem até reduzir a propriedade de segurança pretendida que deveria ser fornecida pelo padrão.

A modelação de ameaças é considerada a abordagem fundamental na identificação de falhas de segurança em softwares durante a fase de desenho no processo do ciclo de vida do desenvolvimento do software. Uma ameaça de segurança é a possibilidade de causar danos a um sistema devido à existência de uma vulnerabilidade, e quando uma entidade maliciosa encontra essa vulnerabilidade e a explora, pode ocorrer um ataque. Portanto, é importante implementar medidas de segurança de modo a corrigir as vulnerabilidades e impedir que entidades maliciosas causem danos ao sistema. Várias técnicas e ferramentas foram publicadas na área da modelação de ameaças que permitem a implementação de medidas de segurança, e em seguida serão apresentadas algumas delas.

Microsoft Threat Modeling Tool descreve-se como uma ferramenta de modelação de ameaças desenvolvida pela Microsoft que é um elemento central no ciclo de vida de desenvolvimento seguro de software [Pot09]. Apresentando uma componente gráfica fácil de usar para criar e analisar modelos de ameaças, esta ferramenta auxilia a captura de avaliações de impacto de mitigações propostas produzindo relatórios, permitindo assim que os arquitetos de software identifiquem e mitiguem problemas de segurança antecipadamente [WY15]. No contexto de modelação de ameaças, esta solução criada pela Microsoft segue uma própria metodologia STRIDE para gerar automaticamente uma lista de ameaças para o sistema que o utilizador está a construir. STRIDE refere-se a uma mnemónica em que cada letra corresponde à primeira letra de uma classe de vulnerabilidade: *Spoofing, Tampering, Repudiation, Information Disclosure, Denial of service e Elevation of privilege*, como se demonstra na tabela 2.1, que contém mais informações sobre esta metodologia.

	Ameaça	**Propriedade Violada**	**Definição da Ameaça**
S	*Spoofing*	Autenticação	Pretender ser algo ou alguém que não sejas tu próprio
T	*Tampering*	Integridade	Modificar algo no disco, rede, memória ou outro lugar
R	*Repudiation*	Não-Repúdio	Alegar que tu não fizeste algo ou não foste responsável
I	*Information Disclosure*	Confidencialidade	Fornecer informações a alguém que não tenha autorização
D	*Denial of Service*	Disponibilidade	Esgotar os recursos necessários que fornecem serviços
E	*Elevation of Privilege*	Autorização	Permitir que alguém faça algo que não esteja autorizado a fazer

Tabela 2.1: Metodologia STRIDE proposta pela *Microsoft* que permite a modelação de ameaças de um sistema.

Enquanto a metodologia STRIDE é usada como um meio de obtenção da visão geral das ameaças do sistema, o facto de se saber mais sobre vulnerabilidades sobre o software explícito permite um maior aprimoramento do modelo de ameaças do sistema. Elementos num diagrama podem ser definidos com maior clareza se for especificada uma certa marca de software, para a qual informações sobre vulnerabilidades são armazenadas e disponíveis em base de dados públicas. Sabendo informações mais detalhadas sobre as vulnerabilidades do software em causa, é possível que estas possam ser mitigadas antes de serem exploradas. Uma categorização de ameaças como a STRIDE é útil na identificação de ameaças e classificação de objetivos de entidades maliciosas.

Focando-se na prevenção de vulnerabilidades em oposição à deteção destas, a *SD Elements* é uma plataforma de gestão do desenvolvimento de aplicações seguras baseando-se numa base de conhecimento de segurança que abrange diferentes fases do ciclo de vida do desenvolvimento do software [EAS+11]. De forma a incorporar a segurança em fases iniciais do desenvolvimento, esta ferramenta permite a identificação de ameaças e riscos através de um questionário em que no final produz automaticamente uma lista de requisitos de segurança a serem implementados, bem como guias seguros sobre o *design* da arquitetura do sistema, como se demonstra na figura 2.4. Adaptando diretrizes de segurança para diferentes projetos de acordo com as especificações do utilizador, *SD Elements* também permite a integração dos requisitos de segurança com aplicações de gestão do ciclo de vida do desenvolvimento de software, tais como *Jira Software*, *CA Agile Central IBM Rational Team Concert Jenkins*, *Microsoft Azure Pipelines*, entre outras.

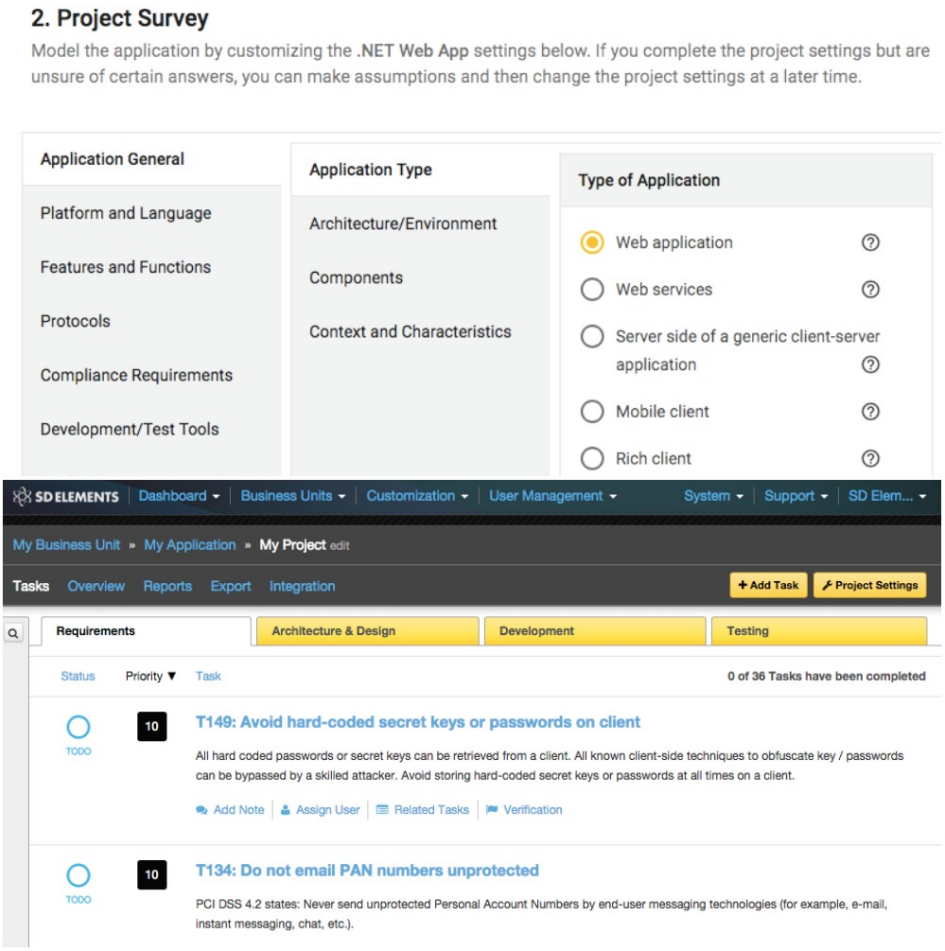


Figura 2.4: Apresentação da plataforma *SD Elements*.

IriusRisk apresenta-se como uma solução integrada para modelação de ameaças e especificações de requisitos de segurança, sendo capaz de fornecer informações sobre todas as fases de desenvolvimento de software [Con]. Este tipo de solução é o que seria considerado ideal para fornecer aos engenheiros ferramentas para integrar a segurança pelo *design*, permitindo assim, a criação de aplicações ou softwares seguros, mesmo sem se ter conhecimento detalhado de segurança.

Common Platform Enumeration (CPE) é descrito como um esquema de nomenclatura estruturada para sistemas de tecnologia da informação, software e pacotes [Nata]. Nomeando produtos de software de uma maneira padronizada, cada CPE pode ser vinculado apenas a um software, e para cada CPE são encontradas vulnerabilidades conhecidas numa forma específica denominada de *Common Vulnerabilities and Exposures (CVE)*. O CVE [Natb] trata-se de um sistema normalizado com o objetivo de referenciar vulnerabilidades de software conhecidas, em que referências CVE incluem um resumo da vulnerabilidade, a data em que a vulnerabilidade foi publicada, uma ou mais referencias CPE e uma pontuação associada dada por um sistema — o *Common Vulnerability Scoring System (CVSS)*. O CVSS fornece uma maneira de capturar as principais características de uma vulnerabilidade e produz uma pontuação numérica refletindo a gravidade associada [Fir]. Esta pontuação descreve o impacto da vulnerabilidade, em que esta é dada consoante a uma análise à complexidade do ataque e ao impacto à integridade e confidencialidade. Assim cada referência CVE inclui uma pontuação CVSS, o que permite elaborar um *ranking* de vulnerabilidades consoante o seu impacto no software.

TEAM Mentor é uma biblioteca interativa sobre segurança de aplicações que contém guias de segurança sobre o desenvolvimento do software, boas práticas de programação, implementação de controlos de segurança em diferentes plataformas e informações sobre vulnerabilidades [TEA]. Este repositório de artigos que descrevem como implementar controlos de segurança em aplicações, permite examinar vulnerabilidades comuns, bem como a leitura de guias que descrevem como evitar essas vulnerabilidades. Deste modo, após a implementação correta dos mecanismos de segurança recomendados, esta biblioteca torna-se útil na integração de segurança em aplicações, apresentando assim, uma redução do risco da inserção de vulnerabilidades no desenvolvimento do software.

O projeto OWASP *Threat Dragon* é a designação de uma aplicação *Web* de modelação de ameaças *on-line* desenvolvida pela OWASP [OWAd]. Sendo uma ferramenta *open-source* e independente, esta ferramenta pode ser integrada no ciclo de vida de desenvolvimento de software sem muito esforço. Apresenta um *design* limpo, e através da construção do sistema com diagramas ajuda a fornecer uma visão geral sobre arquiteturas de variedades complexas. Como este projeto ainda está na fase inicial de desenvolvimento, apresenta a desvantagem de não efetuar uma análise automática das ameaças nem geração de relatórios.

Por fim, a ferramenta *Open Weakness and Vulnerability Modeler (OVVL)*[1] [Res18] apresenta-se como uma estrutura de software livre, em contexto de aplicação *Web*, para modelação de ameaças. Baseada em ferramentas já existentes, OVVL oferece a possibilidade da criação do modelo do sistema baseado em diagramas, associa as ameaças aos componentes presentes na arquitetura do sistema e no software e sugere atenuações sobre as ameaças existentes.

[1] O VV significa a letra W e V. "**O**pen **W**eakness and **V**ulnerability Mode**l**er" [Res18]

2.4 Tabelas Resumo

Como também o contexto desta dissertação engloba documentação de boas práticas de segurança a adotar, torna-se essencial que se faça uma investigação sobre guias de segurança. Estes guias podem ser denominados de tabelas resumo (*cheatsheets*, do inglês) e contêm práticas a adotar sobre vários temas de modo a implementar uma melhor segurança no sistema a desenvolver. O conteúdo abordado nesta secção engloba a apresentação de *cheatsheets*, projetos e documentação que se relacionam com boas práticas de segurança a adotar no desenvolvimento de um sistema de software.

O projeto OWASP *Cheat Sheet Series* [OWAb] foi criado com o objetivo de fornecer uma coleção concisa de informações de alto valor sobre tópicos específicos sobre segurança em aplicações. Oferendo uma excelente orientação de segurança num formato de fácil leitura, esta documentação aborda guias de desenvolvimento do software seguro englobando autenticação, autorização, criptografia, validação de dados, proteção contra ataques de injeção, entre outros temas.

Produzido também pela OWASP, o guia de referência de práticas seguras de codificação [OWAc] define um conjunto de práticas gerais de programação segura de software englobando temas como validação de dados, autenticação e gestão de palavras-passes, controlo de acesso, práticas de criptografia, ou comunicação seguras, entre outros. Este documento apresenta um conjunto de práticas num formato de lista de verificação, e pode ser integrado no ciclo de vida de desenvolvimento do software. A correta implementação dessas práticas permite atenuar as vulnerabilidades de software mais comuns. Este guia fornece práticas de codificação que podem ser traduzidas em requisitos de codificação sem a necessidade de as equipas de desenvolvimento terem uma compreensão aprofundada de vulnerabilidades e conhecimento em segurança. Assim, com a utilização deste documento de referência é possível avaliar a maturidade de segurança do software a ser desenvolvido.

Tratando-se de documentação produzida pela Microsoft, o *Application Security Frame (ASF)* contém informação relacionada com ameaças, vulnerabilidades e contra-medidas associadas [Mica] em diversas categorias, tais como: autenticação, autorização, gestão de configurações, gestão de erros e exceções, proteção de dados, validação de dados e auditoria.

No final do ano de 2018, *IoT Security Foundation* publicou um documento denominado *Secure Design Best Practice Guides* [IoT], contendo informações sobre melhores práticas a nível de desenho seguro na área de desenvolvimento de produtos e sistemas IoT. Este guia aborda tópicos relacionados com classificação de dados, segurança física, inicialização segura dos dispositivos, segurança a nível do sistema operativo, da aplicação e da conexão a redes, gestão de credenciais, encriptação, segurança em atualizações do software e, por fim, práticas seguras no contexto de *logging*. No contexto de desenvolvimento de sistemas na área de IoT, esta documentação apresenta utilidade para equipas de desenvolvimento na medida em que o sistema ou software contenha um nível de segurança considerável.

No âmbito de aplicações orientadas à *Web*, a *Mozilla* [moz] elaborou documentação sobre guias seguros de codificação (*WebAppSec/Secure Coding Guidelines*), de modo a estabelecer uma abordagem concisa e consistente no desenvolvimento de aplicações e serviços *Web*. Este con-

junto de informações presentes numa só página *Web* engloba diretrizes de segurança direcionadas para temas relacionados com o desenvolvimento *Web*, tais como autenticação, gestão de sessões, controlo de acesso, validação de dados, prevenção de ataques de *scripting*, transmissão segura de dados, administração de páginas de administradores e tratamento de erros.

Securing Web Application Technologies Checklist [SAN] traduz-se numa lista de verificação que fornece um conjunto de práticas recomendadas que pode auxiliar as equipas de desenvolvimento a criar aplicações mais seguras, identificando o padrão mínimo necessário de segurança de modo a neutralizar as vulnerabilidades mais comuns. Esta lista de práticas seguras concentra-se em conteúdos referentes a proteção de dados, configurações, gestão de erros e *logs*, autenticação, gestão de sessões, controlo de acesso e validação e verificação de dados de entrada e saída.

Para finalizar, a *Cheatgraphy* [Che] refere-se a um repositório de *cheatsheets* que agrega diversa informação de consulta rápida relacionada com vários conteúdos. Neste agrupamento de informação é possível encontrar-se material referente à utilização de ferramentas, linguagens de programação, ou boas práticas na prevenção de vulnerabilidades, que servem como um auxílio no desenvolvimento seguro de software.

2.5 Conclusões

Este capítulo focou-se na apresentação de diferentes metodologias que fazem parte da engenharia de segurança do software, de ferramentas que auxiliam a análise e integração de segurança no desenvolvimento do software e de documentação relativa a práticas de segurança. Após o estudo das diferentes metodologias de engenharia de segurança, conclui-se que o modelo da Microsoft (SDL) apresenta mais popularidade e é o mais completo em termos de integração de segurança.

Relativamente à investigação de ferramentas que se inserem no desenho de sistemas de software, a maior parte destas identificam-se com o desenho manual de componentes do sistema através de diagramas e, que no final, produzem possíveis vulnerabilidades e ameaças ao sistema elaborado. Durante este estudo, foram encontradas ferramentas empresariais, como a *Irius Risk* e *SD Elements* que possibilitam responder a uma série de perguntas sobre detalhes do sistema e que, no final, produzem guias que apoiam a análise e integração de segurança.

Algumas das ferramentas estudadas inserem-se em contexto empresarial, o que implica que o acesso ao produto é restrito bem como a respetiva documentação, enquanto que outras que também permitem a modelação de ameaças, requerem o desenho do sistema com base em diagramas, o que por vezes, pode-se tornar uma tarefa complexa. Devidos a estes fatos, surgiu a ideia de implementar uma ferramenta de código aberto (*open-source*, do inglês) que permitisse a identificação do sistema e que produzisse um relatório contendo informações que ajudasse os arquitetos de sistemas a integrar segurança no sistema a desenvolver. Esta ferramenta é descrita nos próximos capítulos deste documento e consiste na apresentação de um questionário ao utilizador, de forma a identificar componentes e propriedades do sistema, e que o resultado final corresponde a documentação sobre guias de boas práticas de segurança a adotar.

De modo a elaborar esta documentação, foi realizada uma pesquisa sobre repositórios / documentação contendo boas práticas de segurança. No decorrer desta pesquisa, notou-se que não existia muita informação relativa a este tema, no entanto foi possível, ainda assim, retirar algum conteúdo significativo de repositórios encontrados, de forma a auxiliar a produção de um relatório contendo boas práticas de segurança no desenvolvimento de um sistema. Alguns destes conteúdos foram adaptados e incluídos em partes dos relatórios produzidos pela ferramenta adiante descrita.

Capítulo 3

Desenho do Fluxo da Ferramenta

3.1 Introdução

Antes de se iniciar a implementação da ferramenta é necessário identificar quais as questões e respetivas respostas a serem apresentadas ao utilizador, visto que a primeira parte da ferramenta corresponde a um questionário. Este questionário possibilita a identificação de componentes e características do sistema a implementar e que no final, dependendo das opções escolhidas, um relatório será produzido correspondendo assim à segunda parte da ferramenta. Este capítulo irá abordar a identificação e estruturação do questionário, sendo que a secção 3.2 discute a identificação de questões, a secção 3.3 se debruça sobre a identificação das opções, e por fim, a secção 3.4 descreve a estruturação das questões no fluxo da ferramenta.

3.2 Identificação das Questões

De modo implementar uma ferramenta baseada em texto, é necessário formular questões que irão ser propostas ao utilizador com a intenção de obter informações úteis sobre o desenvolvimento e características do sistema. Abordando vários temas, estas questões permitem a recolha de detalhes do sistema a desenvolver, ou do sistema já implementado, caso já exista, e da maior parte dos componentes que integram e interagem com o sistema.

Inicialmente, a primeira questão a ser apresentada refere-se ao modo de como irá ser utilizada a ferramenta — *"Which way do you want to run this tool?"* — permitindo assim a escolha de responder manualmente ao resto do questionário ou utilizar um ficheiro adicional com as respostas já previamente escritas. A segunda questão que se propõe ao utilizador relaciona-se com o estado de desenvolvimento do sistema — *"What is the status of development of the system?"* — questionando assim o utilizador se o sistema irá ser desenvolvido ou já se encontra implementado.

Após responder às perguntas introdutórias, dá-se continuação ao questionário, com detalhes sobre o desenvolvimento do sistema. A questão que introduz a implementação do sistema a desenvolver questiona o utilizador sobre o tipo de arquitetura ou modelo do sistema global em que se insere — *"Which will be the architecture of the system?"*. O tipo de arquitetura pode variar entre aplicações cliente-servidor, aplicações nativas ou *mobile*, aplicações orientadas à *Web*, sistemas embutidos, entre outros. Na próxima secção irão ser abordadas todas as opções desta pergunta.

Como a maior parte dos sistemas a desenvolver envolvem o tratamento e armazenamento de dados, faz sentido apresentar questões ao utilizador sobre esse tema. Questiona-se então se é utilizada uma base de dados para armazenamento dos dados — *"The system will use a Database?"* — qual o tipo de armazenamento que é usado — *"Which will be type of data storage?"* — e qual a base de dados, em específico, que é usada — *"Which Database will be used?"*. Para finalizar esta secção de perguntas referentes aos dados envolventes no sistema, questiona-se o utilizador acerca da sensibilidade dos dados que são armazenados e processados — *"Which type of data will be stored?"*.

Sendo necessária a autenticação de entidades e componentes, de forma a verificar a respetiva identidade, questiona-se ao utilizador qual o tipo de autenticação que melhor se adequa ao sistema — *"Which type of authentication will be implemented ?"*.

No desenvolvimento de alguns sistemas é comum ter utilizadores que interagem diretamente com o sistema, bem como com a análise e processamento de dados. Para que haja esta interação de utilizadores com o sistema é, por vezes, fundamental ocorrer um registo desses mesmos utilizadores. De modo a identificar estes aspectos, questiona-se se há registo de utilizadores — *"Will the system include a user registration process?"* — e, em caso afirmativo, é necessário perguntar como é que os utilizadores são registados — *"Which means of user registration will be used?"*.

A implementação de um sistema envolve muitos aspetos, sendo que um dos mais importantes corresponde à identificação de linguagens de programação a serem usadas durante o desenvolvimento. Neste contexto, é formulada uma questão de modo a identificar as linguagens de programação — *"Which programming languages will be used in the implementation of the system?"* .

Em seguida são apresentadas ao utilizador perguntas que se referem ao facto do sistema permitir a inserção manual de dados — *"The system will allow user input forms?"* —, o envio de ficheiros — *"The system will allow file uploads?"* e se o sistema apresenta um registo *logs* — *"The system will produce logs?"*. Uma breve descrição destas três questões mencionadas apresenta-se a seguir:

- De forma a fornecer práticas seguras de desenvolvimento no que diz respeito a validação de dados, questiona-se o utilizador se o sistema possibilita a inserção de dados através campos de entrada de dados ou, por exemplo, através do preenchimento de formulários;

- Em alguns sistemas, uma funcionalidade que estes demonstram é o envio de ficheiros. De modo a colmatar ataques que envolvam ficheiros maliciosos é preciso ter em conta práticas seguras na verificação dos ficheiros, daí questionar se o sistema permite o envio de ficheiros;

- Durante o desenvolvimento de sistemas, é comum haver um registo de *logs* de operações efetuadas, informações transmitidas, tentativas de autenticação, erros, entre outros. Por vezes os *logs* fazem parte do produto final, e se o sistema apresentar um registo de *logs* é necessário consciencializar o utilizador sobre que informações devem ser registadas nos *logs*.

Sistemas implementados utilizam, por vezes, uma componente de hardware mais específica, como acontece no caso de sistemas de IoT. Deste modo, questiona-se o utilizador sobre detalhes de implementação respeitantes ao hardware. É exibida uma pergunta inicial que interroga o utilizador se deseja especificar detalhes sobre o hardware — *"Do you want to further specify hardware details concerning the system?"* — e se este responder afirmativamente surgem mais duas novas perguntas. Estas perguntas permitem ter conhecimento sobre o tipo de autenticação implementada no hardware — *"What will be the type of authentication implemented in hardware? "* — e sobre quais as tecnologias de comunicação presentes — *" What will be the wireless tecnologies present in the hardware?"*.

De um modo resumido, a seguinte tabela 3.1 apresenta todas as questões que são abordadas pela ferramenta e apresentadas ao utilizador.

Questões da Ferramenta
Which way do you want to run this tool?
What is the status of development of the system?
Which will be the architecture of the system?
The system will use a Database?
Which will be type of data storage?
Which database will be used?
Which type of data will be stored?
Which type of authentication will be implemented?
There will be a user registration?
Which way of user registration will be used?
Which programming languages will be used in the implementation of the system?
The system will allow user input forms?
The system will allow file uploads?
The system will produce logs?
Do you want to further specify hardware details concerning the system?
What will be the type of authentication implemented in hardware?
What will be the wireless tecnologies present in the hardware?

Tabela 3.1: Tabela resumo das questões da ferramenta.

3.3 Identificação das Opções

Com o objetivo final de produzir um relatório de práticas seguras, este é adaptado conforme as respostas dadas durante o questionário. Nesta secção serão apresentadas as opções que o utilizador pode escolher nas questões exibidas pela ferramenta. Como já foi referido anteriormente, a primeira questão aborda o modo como a ferramenta irá ser usada, se o utilizador quer responder o questionário pergunta a pergunta ou se quer usar um ficheiro auxiliar com as respostas já escritas. No caso da escolha desta última opção, é pedido ao utilizador que escreva o nome do respetivo ficheiro que contém as respostas, entre aspas e com a extensão. A questão a seguir refere-se ao estado de desenvolvimento do sistema, isto é, se o sistema irá ser desenvolvido ou se já se encontra em desenvolvimento.

Iniciando então a apresentação de questões relacionadas com detalhes de implementação do sistema, a primeira aborda o tipo de arquitetura ou modelo geral do sistema. As opções desta pergunta englobam diversos tipos de modelos de sistema desde aqueles orientados à *Web*, aplicações cliente-servidor, até sistemas embutidos. Existe também a possibilidade de o utilizador introduzir manualmente a resposta caso escolha a última opção. De modo a conferir todas as opções, estas são demonstradas na pergunta:

- *Web Application*;
- *Web Service*;
- *Desktop Application*;
- *Mobile Application*;
- *Client-Server > Client Component*;
- *Client-Server > Server Component*;
- *API Service*;
- *Embedded System*;
- *Others*.

As próximas questões a serem exibidas correspondem ao armazenamento de dados. O utilizador é questionado se o sistema implementa uma base de dados, e caso a resposta seja afirmativa segue-se uma série de perguntas relacionadas ao tema de armazenamento de dados. Questiona-se qual o tipo de armazenamento que é usado, e as opções compreendem as seguintes:

- *SQL*;
- *NoSQL*;
- *Local Storage*;
- *Distributed Storage*.

De forma a recolher informação mais específica sobre o armazenamento de dados é formulada a questão de qual a base de dados, em que as opções retratam-se como se segue:

- *MySQL*;
- *PostgreSQL*;
- *SQLite*;
- *OracleDB*;
- *MariaDB*;
- *MongoDB*;
- *CosmosDB*;
- *DynamoDB*;
- *Cassandra*;
- *Other*.

Para finalizar as perguntas relacionadas com o tema de armazenamento de dados, é colocada uma questão que aborda qual a sensibilidade dos dados a armazenar e analisar:

- *Personal Information*;
- *Confidential Data*;
- *Critical Data*.

A autenticação é um fator fundamental na implementação de um sistema de modo a assegurar a segurança deste. A próxima questão aborda esse mesmo tema, questionando o utilizador sobre o tipo de autenticação do sistema. As opções associadas a esta pergunta demonstram-se a seguir:

- *No Authentication*;
- *Username and Password*;
- *Social Networks / Google*;
- *SmartCard*;
- *Biometrics*;
- *Two Factor Authentication*;
- *Multi Factor Authentication*.

A maior parte dos sistemas envolve interação direta de utilizadores com o sistema. A questão que se segue tem como objetivo retirar informação sobre o registo de utilizadores. Primeiro, questiona-se se existe registo de utilizadores no sistema a implementar e, em caso afirmativo, como é o modo de registo, se são os próprios utilizadores a efetuar o registo ou se é algum administrador que tem essa função:

- *The users will register themselves*;
- *Will be a administrator that will register the users*.

Outra área essencial no desenvolvimento de um software é a parte de implementação e, neste contexto, as linguagens de programação que são usadas para desenvolver o software ou sistema. De forma a saber qual ou quais as linguagens de programação que são utilizadas na implementação, propõem-se ao utilizador as seguintes opções:

- *C#*;
- *C / C++*;
- *Java*;
- *Javascript*;
- *PHP*;
- *Python*;
- *Ruby*;
- *Other / Property Language*.

As próximas três questões desta ferramenta a serem exibidas ao utilizador focam-se em perguntas dicotómicas, ou seja, perguntas que só apresentam duas opções de resposta, geralmente sim/não. Estas perguntas relacionam-se com o facto de o sistema permitir a inserção de dados, o envio de ficheiros e se apresenta um registo de *logs*.

A fim de finalizar o questionário, é mostrada uma pergunta ao utilizador que o interroga se deseja especificar detalhes de implementação do hardware. Caso responda afirmativamente são exibidas duas novas questões. A primeira refere-se ao tipo de autenticação integrada ou a integrar, em que as respetivas opções se identificam a seguir:

- *No Authentication*;
- *Symmetric Key*;
- *Basic Authentication (user/pass)*;
- *Certificates (X.509)*;
- *TPM (Trusted Platform Module)*.

A segunda questão relaciona-se com o apurar da existência de tecnologias de comunicação presentes no hardware. As opções a esta questão são as seguintes:

- *4G / LTE*;
- *3G*;
- *GSM (2G)*;
- *Radio Frequency*;
- *Bluetooth*;
- *Wi-Fi*;
- *GPS*;
- *RFID*;
- *NFC*.

De modo a finalizar esta secção, a tabela 3.2 apresenta as questões juntamente com as respetivas opções que são mostradas ao utilizador. Após o utilizador da ferramenta responder a todas as questões, é efetuada uma análise às opções escolhidas. De acordo com as respostas selecionadas, o relatório final produzido pela ferramenta irá conter práticas seguras relacionadas com a especificação do sistema a desenvolver.

3.4 Estruturação das Questões no Fluxo da Ferramenta

A ferramenta implementada no contexto deste projeto de dissertação engloba várias questões que são exibidas ao utilizador. Inicialmente são apresentadas duas questões fundamentais que têm como objetivo de determinar o fluxo do resto das perguntas durante a execução da ferramenta. A primeira questão essencial, como já foi referido anteriormente, diz respeito ao facto de como a ferramenta irá proceder à recolha e processamento dos dados. A escolha da resposta a esta questão representa as duas maiores ramificações da execução da ferramenta, o que significa que, o utilizador pode escolher responder o questionário pergunta a pergunta ou pode introduzir um ficheiro com as respostas já escritas, servindo de *input* à ferramenta de modo a processar os dados e produzir um relatório com boas práticas de segurança a ter em conta no desenvolvimento do sistema. A segunda questão incide sobre o estado de desenvolvimento do sistema, e conforme se o sistema está desenvolvido ou não, as restantes perguntas sobre detalhes da implementação do sistema irão ser afetadas na medida em que como são expostas ao utilizador, ou seja, a gramática da pergunta apresentada difere consoante a resposta escolhida a esta questão inicial.

As questões alusivas ao levantamento de detalhes de implementação do sistema abordam temas como, por exemplo, a arquitetura do sistema, armazenamento de dados, autenticação, registo de utilizadores, linguagens de programação, ou detalhes de implementação relativos ao hardware, entre outros. Algumas destas perguntas retratam particularidades mais específicas do sistema e só são expostas ao utilizador caso este responda afirmativamente a perguntas principais sobre esses temas relevantes. Exemplificando, se o utilizador responder positivamente à pergunta se o sistema utiliza uma base de dados, um conjunto de perguntas relacionadas com

o tema de armazenamento de dados — qual o tipo de armazenamento, qual a base de dados em específico e qual a sensibilidade do dados a armazenar — são mostradas ao utilizador. O mesmo acontece para quando o utilizador responde que existe um registo de utilizadores, onde é apresentada uma pergunta adicional que tem como objetivo saber qual o método de registo de utilizadores no sistema. Nas perguntas finais, também ocorre o caso de haver mais questões adicionais se o utilizador desejar especificar mais detalhes sobre o hardware a ser utilizado na implementação do sistema.

Todas as perguntas que têm como intenção a recolha de informação relativamente a detalhes de implementação do sistema dividem-se em duas categorias: perguntas de escolha múltipla e perguntas que contêm duas respostas possíveis (perguntas de sim / não). Nas perguntas de escolha múltipla existe uma opção que possibilita ao utilizador inserir manualmente a sua resposta. Assim, a identificação das respostas denota uma maior simplicidade e flexibilidade na recolha de informação.

A figura 3.1, apresentada a seguir, contém uma versão abreviada do fluxograma exposto no anexo A.1. Este anexo refere-se a todo o fluxo das perguntas no decorrer da execução da ferramenta.

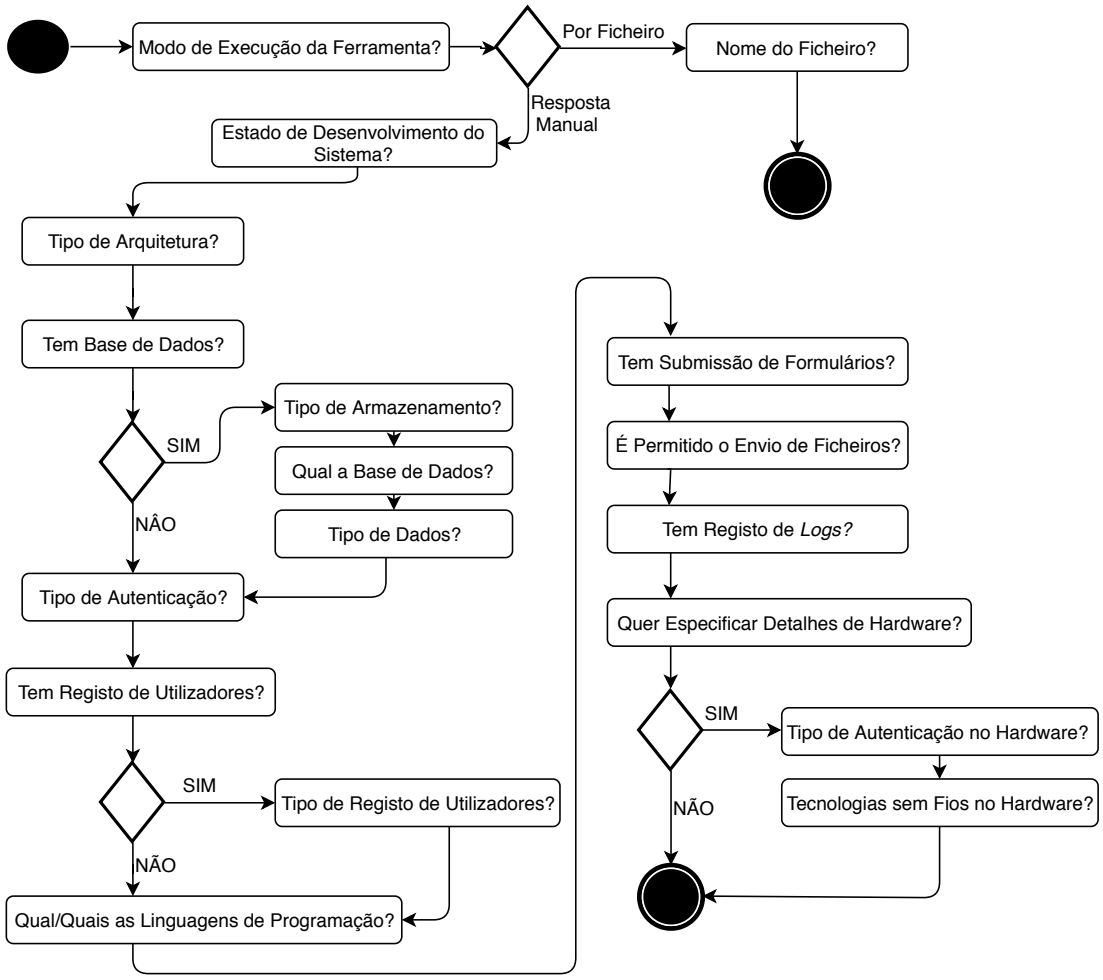


Figura 3.1: Fluxo das perguntas que a ferramenta apresenta, identificando os caminhos e bifurcações possíveis.

3.5 Conclusões

De forma a desenhar o fluxo da ferramenta, foi necessário analisar componentes, propriedades e interações que fazem parte de um sistema de software a desenvolver. A partir desta análise foram formuladas questões com determinadas opções, concluindo assim o planeamento da primeira parte da ferramenta. Um fluxo das perguntas foi proposto para a execução da ferramenta, permitindo assim traduzir o conjunto de questões que integram a mesma. A sua implementação será discutida no próximo capítulo, bem como a segunda parte da ferramenta: a produção de um relatório contendo boas práticas de segurança.

Perguntas	**Respostas**
Which way do you want to run this tool?	Answer the questions one by one. Use a text file with the answers
What is the status of development of the system?	The system is yet to be developed The system is already developed.
Which will be the architecture of the system?	Web Application; Web Service; Desktop Application; Mobile Application; Client Component; Server Component; API Service; Embedded System; Others
The system will use a Database?	Yes / No
Which will be type of data storage?	SQL; NoSQL; Local Storage; Distributed Storage
Which Database will be used?	MySQL; PostgreSQL; SQLite; OracleDB; MariaDB; MongoDB; CosmosDB; DynamoDB; Cassandra; Other.
Which type of data will be stored?	Personal Information; Confidential Data; Critical Data
Which type of authentication will be implemented?	No Authentication; Username and Password; Social Networks / Google; SmartCard; Biometrics; Two Factor Authentication; Multi Factor Authentication
There will be a user registration?	Yes / No
Which way of user registration will be used?	The users will register themselves; Will be a administrator that will register the users
Which programming languages will be used in the implementation of the system?	C#; C / C++; Java; Javascript; PHP; Python; Ruby; Other / Property Language
The system will allow user input forms?	Yes / No
The system will allow file uploads?	Yes / No
The system will produce logs?	Yes / No
Do you want to further specify hardware details concerning the system?	Yes / No
What will be the type of authentication implemented in hardware?	No Authentication; Symmetric Key; Basic Authentication (user/pass); Certificates (X.509); TPM (Trusted Platform Module)
What will be the wireless tecnologies presents in hardware?	4G/LTE; 3G; GSM(2G); Radio Frequency; Bluetooth; Wi-Fi; GPS; RFID; NFC.

Tabela 3.2: Tabela resumo de todas as questões e respetivas opções.

Capítulo 4

Implementação da Ferramenta

4.1 Introdução

No capítulo anterior foi proposto um conjunto de questões e respetivas respostas que permitem a identificação de propriedades e interações de um sistema de software. Este capítulo apresenta a transformação destas perguntas numa ferramenta, englobando assim todo o processo de implementação. A secção 4.2 revela as entradas e saídas da ferramenta, a secção 4.3 retrata os detalhes da implementação, como as dependências que são necessárias para o bom funcionamento da ferramenta e particularidades a nível da programação da ferramenta e, por fim, a secção 4.4 exemplifica a demonstração e validação da ferramenta de modo a comprovar se os resultados obtidos correspondiam ao esperado.

4.2 Formatos de Dados de Entrada e Saída

A ferramenta desenvolvida no contexto desta dissertação, cujo nome foi dado de SECURiot-PRACTICES, foca-se num conjunto de questões colocadas ao utilizador. Estas questões têm como principal finalidade obter informações específicas sobre um sistema de software a desenvolver na área de IoT. Após a recolha e análise das especificações do sistema através das respostas escolhidas pelo utilizador, a ferramenta irá processar a informação obtida de modo a produzir um relatório em que o seu conteúdo se identifica com práticas seguras a ter em conta no desenvolvimento.

Como dados de entrada, a ferramenta desenvolvida aceita respostas inseridas manualmente pelo utilizador, isto é, o utilizador pode responder a cada pergunta individualmente inserindo a opção que deseja, e aceita também um ficheiro de texto contendo as opções das respostas às perguntas. A funcionalidade de usar um ficheiro de texto com as opções como um método de inserção de dados na ferramenta surgiu da possibilidade do utilizador querer executar outra vez a ferramenta com os mesmos dados em vez de voltar a inserir manualmente as respostas. Este ficheiro em específico é gerado automaticamente no final de cada execução da ferramenta. Se o utilizador o desejar também o pode criar seguindo instruções de como escrever o seu conteúdo de forma a que o ficheiro seja aceite pela ferramenta. Pode-se observar um exemplo deste ficheiro com as opções seleccionadas pelo utilizador no anexo A.9 bem como as instruções para o elaborar caso o utilizador o desejar. Como já foi referido anteriormente, esta ferramenta gera um ficheiro com as opções escolhidas para as questões colocadas ao utilizador, mas não é só o que a ferramenta produz no final.

Com uma maior importância, esta ferramenta produz um relatório contendo práticas seguras a ter em conta durante o desenvolvimento do sistema especificado pelas opções escolhidas pelo utilizador às questões colocadas. Este relatório é redigido no formato *Markdown*. Esta decisão de escolha deve-se ao facto da linguagem *Markdown* permitir escrever texto usando um formato simples, de fácil leitura e escrita, em que depois é possível o texto ser convertido no formato HyperText Markup Language (HTML) estruturalmente válido [Joh]. Para uma melhor explicação, *Markdown* é visto de duas maneiras diferentes: uma sintaxe de formatação de texto simples e uma ferramenta de software que converte formatação de texto simples em *HTML*. No caso da construção do relatório final produzido pela ferramenta, foi usada a linguagem de *Markdown*, ou seja, a sintaxe de formatação. O objetivo principal do desenho desta sintaxe é tornar o documento mais legível possível, isto é, um documento formatado nesta linguagem pode ser publicado como está, em texto simples, sem apresentar *tags* ou instruções de formatação. De modo a fornecer uma maior flexibilidade e através do uso de bibliotecas auxiliares, a ferramenta já converte e devolve o relatório para outros formatos, como HTML e *Portable Document Format (PDF)*. Após a execução desta ferramenta, os resultados finais obtidos traduzem-se num ficheiro com as opções escolhidas pelo utilizador a cada questão e um relatório contendo práticas seguras referentes a vários temas em diferentes formatos. Na figura 4.1 pode-se verificar as opções de entrada de dados na ferramenta bem como todos os resultados que são obtidos.

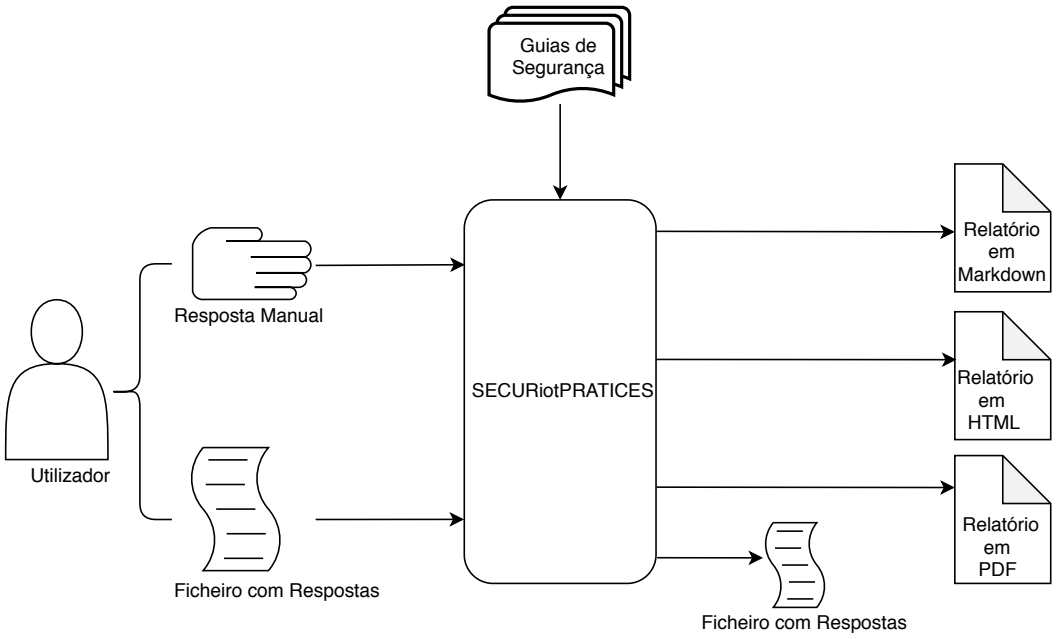


Figura 4.1: Entradas e saídas da ferramenta protótipo implementada no contexto deste projeto de dissertação.

4.3 Detalhes de Implementação

Após a identificação do formato das entradas e saídas da ferramenta desenvolvida no contexto desta dissertação, segue-se a descrição dos detalhes da transformação do conjunto de questões e respetivas opções num software. Esta secção irá abordar as dependências da ferramenta bem como particularidades da sua própria construção a um nível mais técnico.

4.3.1 Dependências

Na implementação da ferramenta, algumas bibliotecas adicionais foram utilizadas de modo a construir um relatório final com práticas seguras a adotar durante o desenvolvimento do sistema. De modo a fornecer mais informação sobre estas bibliotecas, alguns detalhes destas dependências são descritos a seguir:

- *Graph-Easy* [Shl] traduz-se num módulo construído em *Perl* que permite, através de dados de entrada com uma sintaxe específica, a conversão para um esquema em vários formatos: arte ASCII, HTML e Scalable Vector Graphics (SVG). Esta dependência possibilita a elaboração de um mini esquema do modelo do sistema em geral;
- *Python-Markdown* [Way] é uma implementação em *Python* da ferramenta *Markdown* que possibilita a conversão de um documento em formato *Markdown* para formato HTML. De maneira a que o relatório final apresente uma formatação simples e elegível, recorre-se ao formato *markdown* para esse efeito e utiliza-se esta biblioteca para a conversão para o formato HTML;
- A biblioteca *xhtml2pdf* [Lui] proporciona a conversão do documento em formato HTML para o formato PDF. Deste modo, o relatório apresenta uma maior flexibilidade no que diz respeito à sua apresentação, pois o utilizador tem mais um formato em que pode ler o conteúdo do relatório;
- Durante o desenvolvimento da ferramenta, devido ao esquema / figura do modelo do sistema se encontrar em formato SVG após a conversão do relatório para formato PDF, o esquema não era incorporado no mesmo. Para colmatar essa falha e ser possível converter uma imagem em formato SVG para Portable Network Graphics (PNG), recorreu-se a uma biblioteca auxiliar, *svglib* [Din], e aos seus módulos subjacentes (*reportlab*), que também foram instalados automaticamente.

4.3.2 Construção da Ferramenta

A ferramenta desenvolvida no contexto desta dissertação degenera, quando executada, num questionário proposto ao utilizador de forma a adquirir informação sobre detalhes de implementação de um sistema. Optou-se por usar a linguagem de programação *Python* para a prototipagem desta ferramenta, visto que esta linguagem é acessível e fornece a possibilidade de transpor esta ferramenta para a *Web*, tendo em vista o futuro desta ferramenta, como é de resto mencionado na secção 6.2 do último capítulo deste documento.

De forma a armazenar a informação respetiva sobre as questões e as repostas escolhidas decidiu-se utilizar uma estrutura de dados em *Python* denominada de dicionários (*Dictionaries*). Esta estrutura de dados implementa um mapeamento, ou seja, uma coleção de associações entre pares de valores. O primeiro elemento do par é a chave e outro é o conteúdo. Assim, o uso de dicionários em *Python* possibilita guardar quais as opções escolhidas pelo utilizador associadas a cada pergunta, como se demonstra no excerto de código 4.1. Para cada questão também foi construído um dicionário de maneira a associar o número da opção escolhida com a respetiva

resposta que é apresentada ao utilizador, como se pode verificar em 4.2. Através destes dicionários elaborados para cada pergunta facilita-se a construção de uma tabela com as todas as questões e respetivas opções escolhidas pelo utilizador, que é inserida no relatório final produzido pela ferramenta. Esta estrutura facilitará também a migração deste protótipo para uma ferramenta mais dinâmica, em que as questões e opções estarão numa base de dados.

```
# create a dictionary to store the
    answers to the questions
questions_and_answers = {
  "Q1": "",
  "Q2": "",
  "Q3": "",
  "Q4": "",
  "Q5": "",
  "Q6": "",
  "Q7": "",
  "Q8": "",
  "Q9": "",
  "Q10": "",
  "Q11": "",
  "Q12": "",
  "Q13": "",
  "Q14": "",
  "Q15": ""
}
```

Listing 4.1: Estrutura de dados (*Dictionary*) que armazena todas as questões e respostas.

```
question_1 = {
  "1" : "Web Application",
  "2" : "Web Service",
  "3" : "Desktop Application",
  "4" : "Mobile Application",
  "5" : "ClientServer > Client
      Component",
  "6" : "ClientServer > Server
      Component ",
  "7" : "API Service",
  "8" : "Embedded System",
  "9" : ""
}

question_2 = {
  "1" : "Yes",
  "2" : "No"
}
```

Listing 4.2: Estruturas de dados que armazenam cada questão e as respetivas opções.

O desenvolvimento desta ferramenta endereça duas funções principais, a captura de informação e o processamento da mesma. A primeira função engloba todas as questões que são colocadas ao utilizador, sendo que para cada questão foi criada uma função, e as respostas escolhidas são armazenadas na respetiva estrutura de dados criada. A segunda função diz respeito ao processamento da informação; ou seja, através das respostas selecionadas é construído um relatório contendo boas práticas de segurança. De modo a produzir este relatório são usadas três funções adicionais que envolvem a conceção e exibição de uma tabela com todas as questões e opções escolhidas, a geração de um ficheiro contendo as respostas selecionadas, a criação de um pequeno esquema do modelo do sistema e a conversão do relatório para vários formatos. Durante a execução do programa são utilizadas duas funções auxiliares que englobam o código de ler dados de um ficheiro e a validação da entrada de dados introduzidas pelo utilizador. A figura 4.2 representa o diagrama de todas as funções que permitem a execução da ferramenta.

Como foi referido anteriormente, a execução da ferramenta faz uso de duas funções auxiliares. Relativamente à primeira função, esta diz respeito à leitura de dados através de um ficheiro de entrada. Inicialmente é efetuada uma verificação de existência do ficheiro, e caso exista, procede-se à leitura do seu conteúdo. A informação presente em cada linha do ficheiro, exceto

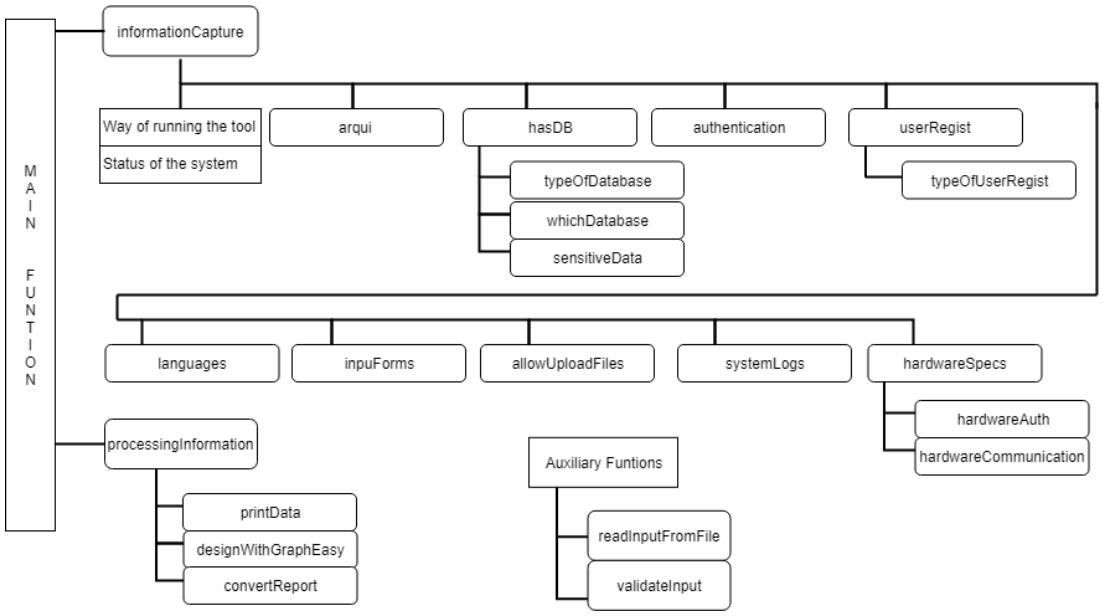


Figura 4.2: Diagrama de funções implementadas no protótipo.

comentários delimitados pelo carácter "#", corresponde à resposta a cada questão, e estas são armazenadas numa variável auxiliar. Esta função é invocada caso o utilizador decida escolher um ficheiro contendo as respostas como modo de entrada de dados na ferramenta. A segunda função auxiliar equivale à validação de dados de entrada — opções selecionadas às questões e respostas introduzidas manualmente — por parte do utilizador na utilização da ferramenta. Esta função implementa argumentos dinâmicos, isto é, dependendo da quantidade dos argumentos passados, irá ser efetuada a validação a um número inteiro ou a um conjunto de letras, uma *string*, como se demonstra no anexo A.2. De forma a validar um número, a função é invocada com dois argumentos, sendo o primeiro o valor `1` e o segundo o número de opções válidas da pergunta respetiva. Algumas questões possibilitam a introdução manual da resposta, e nesses casos é necessário validar o que foi inserido. Para isso, é passado um único argumento à função de validação com o valor `2`, significando que irá ser efetuada uma validação a uma *string*.

Para cada questão que é exibida ao utilizador foi uma criada uma função adicional. A questão em si e as opções respetivas são apresentadas, e depois é pedido ao utilizador que introduza o número da opção que deseja escolher, sendo esta depois validada. Em algumas questões em que exista a opção de escolher outra resposta, é pedido ao utilizador que escreva manualmente a sua própria resposta entre aspas. As opções escolhidas são guardadas na variável criada ao início do programa, o dicionário que armazena as perguntas e as opções selecionadas. As respostas que contêm mais do que uma opção são armazenadas juntas, sendo separadas por um ";". Tomou-se esta decisão visto esta simplificar o processamento das respostas selecionadas por pergunta.

Após a conclusão do questionário, prossegue-se ao processamento das opções selecionadas pelo utilizador de modo a construir um relatório de boas práticas de segurança durante o desenvolvimento de um sistema. Nesta fase do tratamento da informação recolhida, através das questões apresentadas, são utilizadas três funções adicionais: uma função para desenhar um pequeno esquema do modelo do sistema, uma função para converter o relatório em formato *Markdown* para HTML e PDF e, uma terceira função para associar quais as opções que foram selecionadas. Esta última função permite a elaboração de uma tabela com todas as questões e respetivas res-

postas, e a geração de um ficheiro com as opções escolhidas que pode ser usado futuramente como entrada de dados na ferramenta.

Para se construir o esquema do modelo do sistema é utilizada uma ferramenta complementar denominada *Graph-Easy* que necessita de ser instalada à parte. Esta dependência aceita, como entrada, um ficheiro de texto contendo dados específicos sobre como desenhar o esquema. Efetuando o uso de uma chamada de uma função do sistema operativo, executa-se o comando respetivo e obtém-se como resultado uma imagem contendo o esquema do sistema, como se pode verificar no excerto de código 4.3. Para finalizar, a imagem resultante é convertida para formato PNG, a fim de ser integrada no relatório em PDF.

```
def designWithGraphEasy():
  sInput = ("design_schemes.txt")
  cmd = ("graph-easy --as=svg " + sInput +" > design_schemes.svg")
  os.system(cmd)

  # convert image SVG to PNG
  drawing = svg2rlg("design_schemes.svg")
  renderPM.drawToFile(drawing, "design_schemes.png", fmt="PNG")
```

Listing 4.3: Função que permite desenhar um esquema através de uma ferramenta adicional.

De modo a que o relatório seja convertido para os formatos HTML e PDF, é necessário recorrer ao uso de bibliotecas auxiliares. Com o relatório já construído em formato *Markdown*, aplicou-se uma função da biblioteca *Python-Markdown* com o objetivo de converter o conteúdo para HTML, criando depois um ficheiro neste mesmo formato. Através do resultado da conversão do conteúdo em HTML, utilizando um método presente na biblioteca *xhtml2pdf*, foi criado o relatório em formato PDF. O excerto de código 4.4 traduz como foi executada a conversão do relatório de boas práticas de segurança para diferentes formatos.

```
# function to convert the markdown report to html and pdf format
def convertReport():
  input_filename = ("FINAL_REPORT.md")
  output_filename = ("FINAL_REPORT.html")
  with open(input_filename,"r" ) as f:
  html_text= markdown(f.read(), extensions=[ 'markdown.extensions.tables','
      markdown.extensions.sane_lists'])
  out= open(output_filename,"w")
  out.write(html_text)

  # writing the html content in the pdf file
  resultFile = open("FINAL_REPORT.pdf", "w+b")
  pisa.CreatePDF(html_text, dest=resultFile)
```

Listing 4.4: Função que permite a conversão do relatório em diferentes formatos.

No processamento da informação recolhida através do questionário proposto ao utilizador, foi criada uma função adicional, `printData`, que apresenta como objetivo efetuar a associação entre as opções escolhidas com as respetivas respostas, de forma a ser possível elaborar uma tabela

com todas as questões e respostas selecionadas. Esta função também implementa a funcionalidade da geração de um ficheiro com as opções selecionadas pelo utilizador, ficheiro este que pode ser utilizado como entrada de dados na ferramenta. Para que cada linha da tabela contenha a correta informação, é realizada uma associação entre os números das opções introduzidas pelo utilizador com as respetivas respostas. Percorre-se assim a estrutura de dados pertencente à questão em causa (*question_x*, em que x representa o número da questão), que contém os números das opções e as respostas, e verifica-se qual a opção que se encontra no dicionário que armazena as respostas que o utilizador introduziu (`questions_and_answers`). Se ocorrer uma correspondência, a questão e a(s) resposta(s) escolhida(s) são guardadas em variáveis. Estas variáveis permitem a elaboração da tabela e a criação de um ficheiro com as opções selecionadas. No exemplo 4.5 pode-se averiguar que, para retirar a resposta completa, é efetuada uma verificação se alguma opção do dicionário que contém as opções da pergunta se o sistema tem base de dados ("`question_2`") corresponde a alguma opção que esteja presente na parte da estrutura de dados que armazena todas as opções escolhidas ("`questions_and_answers["Q2"]`"). Para efetuar a correspondência às respostas completas em questões de escolha múltipla utiliza-se um delimitador para fazer a separação das opções. Compara-se cada opção a cada item presente no dicionário respetivo da questão em causa, e caso ocorra uma correspondência, a resposta à pergunta é redigida na tabela e também no ficheiro que contém as opções selecionadas.

```
1 for n in question_2:
2   item = questions_and_answers["Q2"]
3   if  item == n:
4     print("{:22} {:3} {:40} ".format("Has DB", ":" , question_2[n]) )
5     table_for_report.append( [ "Has DB" , question_2[n] ])
6     answers_list.append(questions_and_answers["Q2"])
7     comments_list.append(question_2[n])
```

Listing 4.5: Código que exemplifica a criação de uma linha da tabela elaborada com as opções selecionadas pelo utilzador às questões apresentadas.

Explicada a implementação das funções que fazem parte do processamento da informação, segue-se a descrição de como o relatório contendo guias seguros é concebido. Através de uma variável auxiliar que já contém todas as perguntas e respostas, é escrito no relatório uma tabela com esses dados. Após isso, é integrado no relatório um pequeno esquema do modelo do sistema. Para que tal aconteça, dependendo das opções selecionadas à questão referente ao tipo de arquitetura do sistema, é estruturado um ficheiro contendo dados que permitem elaborar um esquema. Este mesmo ficheiro, descrito adiante neste capítulo, serve como entrada da ferramenta *Graph-Easy* de modo a obter uma imagem com o esquema, em que depois este resultado obtido é inserido no relatório.

O conteúdo do relatório, à exceção da tabela e do esquema, é constituído por práticas seguras no desenvolvimento de um sistema abordando diversos temas, desde a autenticação, segurança em sistemas embutidos, armazenamento de dados, criptografia, validação de dados, registo de *logs*, entre outros. Cada guia relacionado com o seu tema foi redigido à parte com a intenção de haver mais facilidade na alteração ou na adição de mais informação relevante no futuro. Estes guias são escritos no relatório dependendo das opções selecionadas pelo utilizador, provocando assim que as respostas escolhidas apresentem um impacto na elaboração do relatório. Por exemplo, se o utilizador escolher que o sistema tem uma base de dados, será criada uma

secção no relatório que contém informação sobre a prevenção de ataques de *SQL Injection*. O mesmo acontece para quando o utilizador responde afirmativamente às questões sobre a utilização de formulários e sobre registo de *logs*. O relatório irá conter, respetivamente, secções sobre a validação de dados e sobre que tipo de informação convém colocar nos *logs* do sistema de modo a apresentar uma melhor segurança. O excerto de código 4.6 demonstra como algumas das secções do relatório são integradas de acordo com as opções selecionadas às questões apresentadas ao utilizador. A informação presente nestes guias foi retirada maioritariamente de *OWASP Cheat Series* [OWAc]. No entanto, contém também conteúdo retirado da documentação redigida pela *Mozilla*, *Secure Coding Guidelines* [moz], e da documentação da Microsoft sobre segurança em aplicações *Web*, *Cheat Sheet: Web Application Security Frame* [Mica]. Relacionado mais com sistemas embutidos, como é o caso de muitos sistemas em IoT, a informação sobre este assunto em específico é retirada de *IoT Security Guidance* [OWAa] pela OWASP e *Best Practice Guidelines* [IoT] por *IoT Security Foundation*.

```
# check if embedded systems are chosen
if questions_and_answers["Q1"].find("8") != -1:
  report.write("\n")
  report.write("\n")
  report.write( open("guides/IOT_Security_guide.md","r").read() )

# check if the database is chosen
if questions_and_answers["Q2"].find("1") != -1:
  report.write("\n")
  report.write("\n")
  # write SQL injection guide
  report.write( open("guides/SQL_Injection_guide.md","r").read() )
...
...
# check if input forms are used
if questions_and_answers["Q10"].find("1") != -1:
  report.write("\n")
  report.write("\n")
  # write input validation guide
  report.write( open("guides/Input_Validation_guide.md","r").read() )

# check if the system is for web
if questions_and_answers["Q1"].find("1") != -1 or
   questions_and_answers["Q1"].find("2") != -1 :
  report.write("\n")
  report.write("\n")
  # write session management guide
  report.write( open("guides/Session_Management_guide.md","r").read() )
  report.write("\n")
  report.write("\n")
  # write XSS guide
  report.write( open("guides/Cross_Site_Scripting_guide.md","r").read() )
```

Listing 4.6: Parte do código que permite construir um relatório com base nas opções escolhidas.

4.4 Demonstração e Validação da Ferramenta

Após o desenvolvimento da ferramenta, a próxima fase foca-se em testes à ferramenta de modo a verificar se os resultados obtidos correspondem às expetativas. Como esta ferramenta aceita duas formas distintas de entrada de dados, esta secção irá apresentar testes que representam a resposta ao questionário manualmente e à introdução de um ficheiro com as opções já escritas. Posteriormente, observam-se os resultados obtidos após a execução da ferramenta. Um exemplo de como as questões são apresentadas no ecrã caso o utilizador decida percorrer o caminho de responder a todas as perguntas da ferramenta inclui-se na figura 4.3. Neste exemplo, observa-se que o utilizador pode responder a mais do que uma opção em questões de escolha múltipla, e até mesmo introduzir uma resposta que não esteja nas opções. Por outro lado, um exemplo da interação com o utilizador para o caso em que este escolhe utilizar um ficheiro auxiliar como método de inserção de opções às perguntas é ilustrado na figura 4.4. Este exemplo mostra que, quando o utilizador escolhe esta ramificação da ferramenta, é-lhe pedido que insira o nome do ficheiro a usar, incluindo a extensão e entre aspas. Após a inserção do nome do ficheiro, a ferramenta irá proceder à leitura e processamento do conteúdo do ficheiro, produzindo assim um relatório de práticas seguras.

```
**Which way do you want to run this tool?**
 1 - Answer the questions one by one.
 2 - Use a text file with the answers already written.

 > 1
 ---

 **What is the status of development of the system?**

 1 - The system is yet to be developed.
 2 - The system is already developed.

 > 1
 ---

 **Which will be the architecture of the system?**
 (This is a multiple choice question. Enter several options and end with 0.)

 1 - Web Application
 2 - Web Service
 3 - Desktop Application
 4 - Mobile Application
 5 - Client-Server > Client Component
 6 - Client-Server > Server Component
 7 - API Service
 8 - Embedded System
 9 - Others

 > 1
 > 7
 > 9
 Please specify the architeture /component: (name between quotes)
 > "raspberry"
 > 0
```

Figura 4.3: Escolha de responder a todas as perguntas.

```
  **Which way do you want to run this tool?**

  1 - Answer the questions one by one.
  2 - Use a text file with the answers already written.

  > 2
---
  **What is the name of the input file ?**

  > 'ans.txt'

  Processing information.....

  Architeture            :   Web Application ; Embedded System
  Has DB                 :   Yes
  Type of data storage   :   SQL
  Which DB               :   MySQL
  Type of data stored    :   Confidential Data
  Authentication         :   No Authentication
  User Registration      :   No
  Type of Registration   :   N/A
  Programming Languages  :   C# ; C / C++
  Input Forms            :   No
  Upload Files           :   Yes
  The system has logs    :   Yes
  Hardware Specification :   Yes
  HW Authentication      :   No Authentication
  HW Wireless Tech       :   4G / LTE ; GPS
```

Figura 4.4: Escolha de usar um ficheiro como entrada de dados.

Através do processamento das respostas escolhidas, a ferramenta tem o objetivo de produzir um relatório contendo boas práticas de segurança a ter em conta no desenvolvimento de um sistema. Este relatório é apresentado nos formatos *Markdown*, HTML e PDF. O conjunto de ficheiros produzidos por uma execução da ferramenta é ilustrado na figura 4.5. Esta figura também mostra os ficheiros contendo os esquemas gerados para o sistema. O ficheiro `design_schemes.txt` contém uma representação em texto do esquema produzido para o sistema, conforme se pode observar na figura 4.6. Este formato é compatível com a biblioteca usada para gerar a figura em gráficos vetoriais que é depois integrado no relatório. Por fim, o ficheiro denominado `ans.txt` corresponde ao ficheiro que é gerado automaticamente após a execução da ferramenta e que contém as opções selecionadas pelo utilizador, podendo ser utilizado como entrada da ferramenta. Um exemplo deste ficheiro, como já foi mencionado anteriormente, encontra-se no anexo A.9.

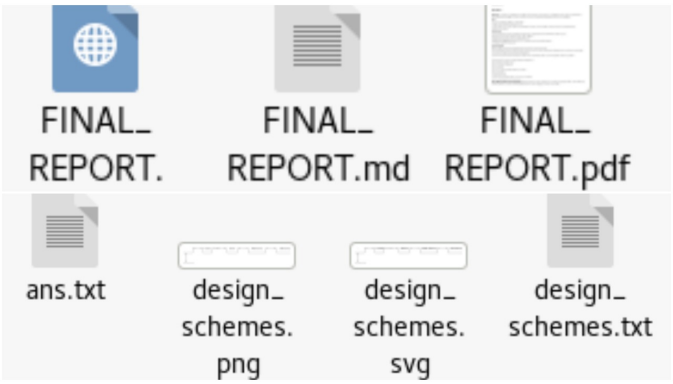


Figura 4.5: Relatório produzido pelo protótipo da ferramenta em diferentes formatos.

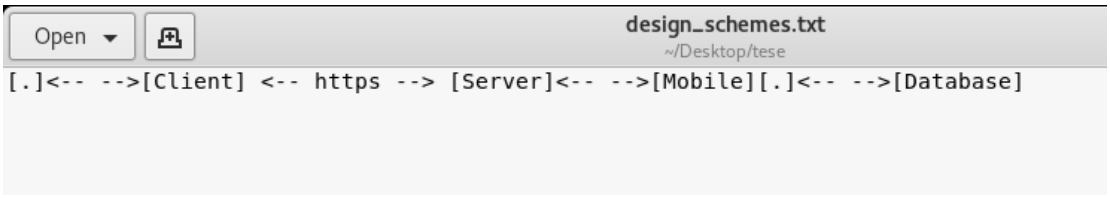


Figura 4.6: Dados que permitem elaborar um esquema do sistema em formato texto.

Como foi referido anteriormente, o relatório contém uma tabela com todas as questões e respostas escolhidas, um pequeno esquema do modelo do sistema, bem como secções que abordam práticas de segurança. Na figura 4.7 observa-se um excerto do relatório em formato *Markdown* e na figura a seguir, 4.8, em formato PDF. Nos anexos A.4, A.5, A.6, A.7 e A.8, são apresentados excertos das boas práticas de segurança que constam no relatório produzido pela ferramenta.

```
# Final Report

|                              |
|  :--------                   |  :---------
|  Architeture                 |  Web Application ; Mobile Application
|  Has DB                      |  Yes
|  Type of data storage        |  SQL
|  Which DB                    |  MySQL
|  Type of data stored         |  Confidential Data ; Critical Data
|  Authentication              |  Two Factor Authentication
|  User Registration           |  No
|  Type of Registration        |  N/A
|  Programming Languages       |  Python
|  Input Forms                 |  Yes
|  Upload Files                |  Yes
|  The system has logs         |  Yes
|  Hardware Specification      |  No
|  HW Authentication           |  N/A
|  HW Wireless Tech            |  N/A

![alt text](design_schemes.png)
```

Figura 4.7: Início do relatório em formato *markdown*.

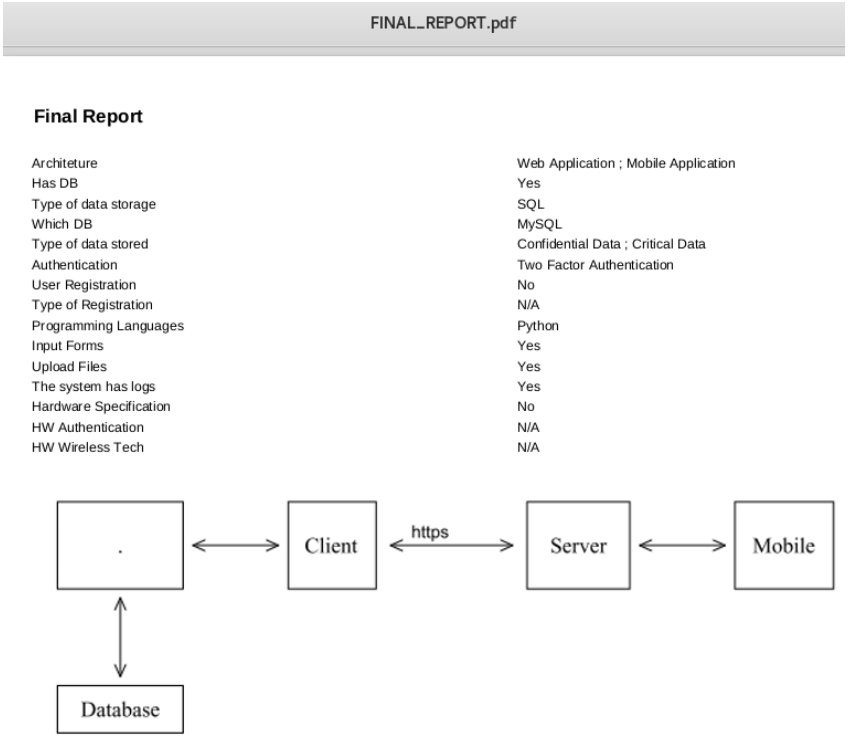


Figura 4.8: Início do relatório em formato PDF.

Relativamente à validação da ferramenta, foi proposto um questionário a um conjunto de doze pessoas, de diferentes idades e qualificações, com o objetivo de avaliar a ferramenta desenvolvida no contexto desta dissertação. Os resultados obtidos, através da ferramenta de criação de formulários da Google, encontram-se na figura 4.9 apresentada a seguir, e resumem as estatísticas das respostas fornecidas. Destas respostas conclui-se que 91,7% deram pontuação máxima à facilidade do uso da ferramenta. 58,3% atribuíram a pontuação máxima e 33,3% uma pontuação muito boa . Sobre o que a ferramenta produzia no final da sua execução, 66,7% escolheram a opção máxima sobre os resultados da ferramenta. No que diz respeito à compreensão das boas práticas de segurança que se encontram na documentação produzida, 91,6% tem opinião claramente positiva (58,3% para pontuação máxima e 33,3% para pontuação boa). Por fim, 83,3% atribuíram a máxima pontuação à questão que se refere à utilidade da ferramenta na criação de sistemas de IoT seguros. De um modo geral, os resultados obtidos deste questionário foram bastante positivos.

4.5 Conclusões

O conteúdo deste capítulo focou-se na apresentação de detalhes de implementação da ferramenta descrita neste documento. Os métodos de entrada de dados e os resultados produzidos pela ferramenta foram descritos, bem como as dependências necessárias e alguns detalhes mais específicos a nível da programação. Foram efetuados testes à ferramenta de modo a verificar os resultados produzidos, assim como para obter uma validação preliminar. Para realizar esta validação recorreu-se a um questionário de forma a recolher e a analisar a opinião de potenciais utilizadores finais da ferramenta, embora a população não fosse totalmente representativa (o ideal seria ter pessoas da indústria).

A implementação deste protótipo comprova que foi possível transformar um conjunto de questões e respostas numa ferramenta que produzisse algo que auxiliasse arquitetos de sistema a desenvolver sistemas com a segurança em mente. Neste caso, esta ferramenta produz, com base nas respostas escolhidas às questões apresentadas, documentação sobre práticas seguras no desenvolvimento de sistemas.

Foi proposta também uma série de questões a um conjunto de pessoas com o objetivo de validar a ferramenta desenvolvida e, através das resultados obtidos, comprovou-se que esta foi avaliada positivamente, tanto na sua facilidade de utilização, qualidade da documentação produzida e utilidade geral. O próximo capítulo irá retratar alguns exemplos de cenários de utilização em que esta ferramenta será aplicada, também como forma de validação, e os devidos resultados obtidos serão discutidos.

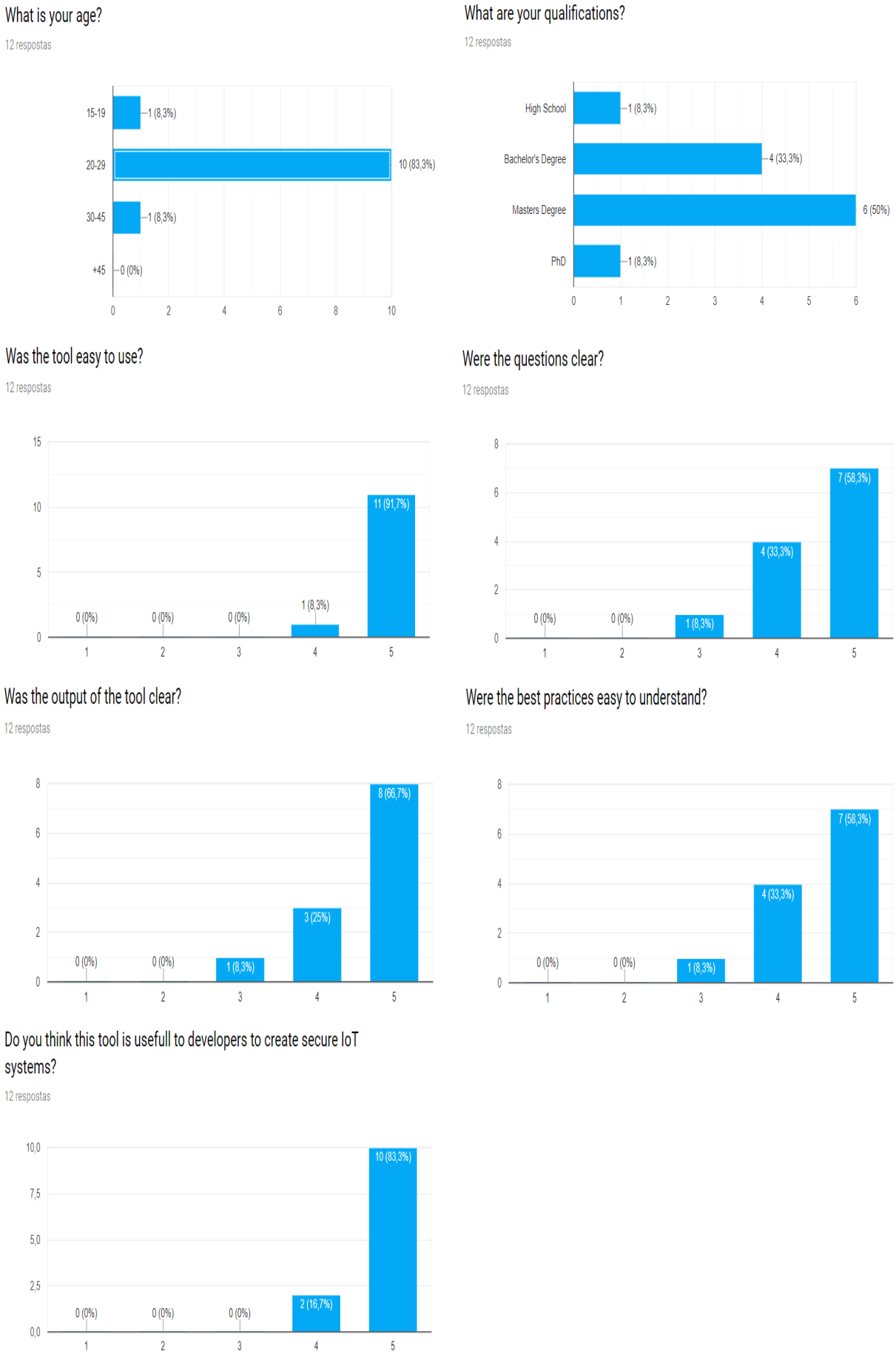


Figura 4.9: Gráficos dos resultados sobre a usabilidade da ferramenta.

Capítulo 5

Cenários de Utilização da Ferramenta

5.1 Introdução

Neste capítulo, alguns cenários de utilização são abordados como casos de teste à ferramenta desenvolvida. Após a execução da ferramenta verificam-se os resultados obtidos. Estes resultados identificam-se com a documentação produzida pela ferramenta que contém boas práticas de segurança a ter em conta na implementação do sistema. As secções 5.2 e 5.3 demonstram, respetivamente, a exemplificação dos cenários e a aplicação da ferramenta aos mesmos e, por fim, a secção 5.4, foca-se na discussão dos resultados.

5.2 Identificação dos Cenários

Esta secção apresenta alguns casos usados no contexto deste trabalho para análise da implementação de segurança. Estes casos pertencem a três grandes áreas no mundo da IoT: Transportes, Saúde e Casa. Os cenários dizem respeito a um carro elétrico e inteligente conectado à *Internet*, um equipamento médico que permite administrar medicação e monitorizar o paciente, e uma *webcam* que serve como um método de vigilância a recém nascidos.

5.2.1 Cenário I – Transportes

Este cenário refere-se a um carro elétrico e inteligente, na medida em que este se encontra ligado à *Internet*, tem a funcionalidade de piloto automático, apresenta ao utilizador informações sobre o trânsito, contém sensores que fornecem a localização e informações técnicas sobre o veículo, entre outras funcionalidades. Ligando-se à maior rede global de computadores, a *Internet*, o carro dispõe, por exemplo, de informações sobre o trânsito, acidentes na via rodoviária, notícias e, até mesmo a possibilidade de enviar um sinal de emergência caso haja uma. Este carro apresenta uma rede interna que controla funcionalidades do veículo como, por exemplo, a condução, o ar condicionado, funcionalidades multimédia, entre outras, e permite também que os passageiros possam ligar o seu *smartphone* ao veículo, possibilitando a transmissão de música e chamadas telefónicas. Sendo um veículo autónomo, possui uma inteligência artificial que possibilita a funcionalidade de piloto automático. Através de sensores adicionados ao veículo, estes transmitem dados sobre a localização e sobre informações mais mecânicas para a entidade a que pertencem e permitem avisar o condutor sobre possíveis avarias. Além

destes sensores, o carro também tem câmaras ao seu redor que auxiliam no estacionamento. As portas do veículo apresentam uma funcionalidade que através de uma ligação *bluetooth* ao carro e através de uma aplicação, podem ser fechadas ou abertas. Um esquema simplificado deste carro inteligente, em termos de componentes e comunicações IoT, encontra-se na figura 5.1.

Após a descrição um exemplo de um sistema IoT na área dos transportes, é necessário efetuar uma análise à segurança a implementar de modo a que potenciais vulnerabilidades sejam abordadas. A seguir é especificada uma breve lista de possíveis ataques ao carro:

- Caso os dados dos sensores sejam transmitidos sem estarem cifrados, o atacante pode ter acesso a esses dados, obtendo assim o conhecimento da localização do veículo e de outros detalhes mais técnicos;
- Uma intrusão efetuada com sucesso ao sistema pode levar os atacantes a controlarem o veículo, podendo direcioná-lo para fora da estrada, provocando um acidente. Também pode acontecer o caso de os atacantes terem acesso às câmaras do carro, alterar valores do ar condicionado, ou até mesmo escutar e gravar as chamadas efetuadas;
- Se as ligações não forem seguras, bem como a autenticação, o atacante pode obter dados confidencias dos passageiros quando estes se conectam ao computador de bordo do carro;
- Caso a aplicação do carro ou a ligação *bluetooth* não seja devidamente segura, o atacante pode conseguir abrir a porta do veículo, possibilitando assim o roubo do mesmo.

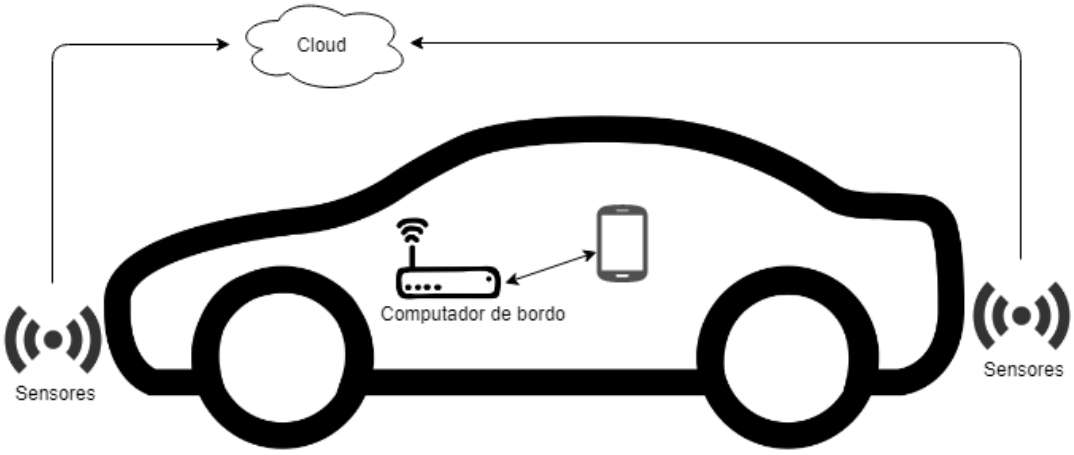


Figura 5.1: Representação de um *smart car*.

5.2.2 Cenário II – Equipamentos Médicos

A descrição deste caso retrata um equipamento médico que permite monitorizar os sinais vitais do paciente, bem com administrar medicação através de valores introduzidos pelo médico responsável. Também possibilita guardar todo histórico médico do paciente. O médico autorizado, através de um *tablet* ou *smartphone*, e após ter sido autenticado com sucesso, pode introduzir

manualmente os valores da medicação que o utente deve necessitar, ou programar automaticamente a toma do medicamento. O médico pode também consultar o histórico médico do paciente em causa. A figura 5.2 representa o equipamento supra descrito.

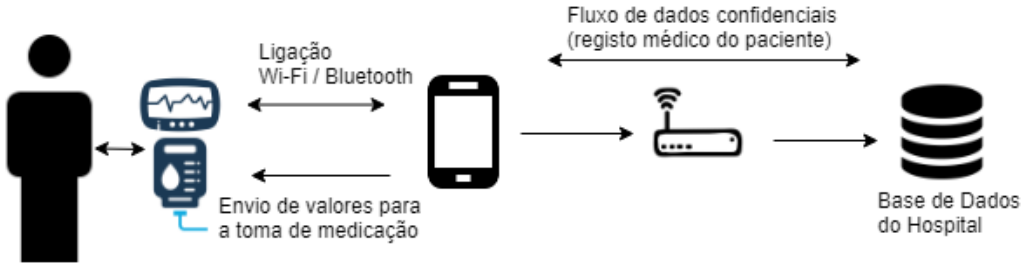


Figura 5.2: Representação das comunicações e componentes IoT no cenário de equipamento médico.

Tratando-se de um equipamento de tecnologia inovadora e moderna que atua na área da saúde, é de extrema importância que o nível de segurança seja significativo. Alguns exemplos de possíveis vulnerabilidades, e consequentemente ataques, transcrevem-se de seguida: se as comunicações e ligações não forem protegidas, o atacante pode conseguir alterar os valores da medicação fornecidos; o atacante, ao conseguir acesso ao equipamento ou à conta do médico responsável, pode obter todas as informações médicas do paciente, e também administrar valores incorretos da medicação, podendo levar a resultados catastróficos para a saúde do paciente.

5.2.3 Cenário III – Vigilância Eletrónica

Este caso diz respeito a uma câmara de vigilância para uso doméstico que permite monitorizar recém-nascidos. Este sistema é composto, como o próprio nome diz, por uma câmara e um microfone, e integra um conjunto de pequenos computadores embutidos e baterias. Para o bom funcionamento desta *webcam*, esta tem de estar ligada a uma rede *Wireless Fidelity (Wi-Fi)* da casa, e após a conexão ter sido efetuada, através de um computador, *tablet* ou *smartphone*, é possível ter acesso à imagem e som da câmara. A partir do momento que a câmara é ligada, o microfone também é ligado, e a imagem e som começam a ser gravados. Estes dados são transmitidos para uma *cloud* em que depois, através do IP da *webcam*, a imagem e o som a serem transmitidos podem ser acedidos através de outro dispositivo, como foi referido anteriormente. Um diagrama deste sistema simplificado e que engloba esta *webcam* é fornecido na figura 5.3.

À primeira vista, este tipo de sistemas concebidos para uso doméstico podem parecer inofensivos mas, se não apresentarem uma segurança adequada, podem permitir, a uma entidade maliciosa, realizar uma intrusão ao sistema e obter conhecimento de dados confidenciais ou privados. A seguir são demonstrados alguns exemplos de vulnerabilidades do sistema:

- Os dados transmitidos pela câmara só deviam ser acedidos por entidades autorizadas. Se à partida o sistema não implementa algum método de autenticação e autorização, um atacante pode visualizar e ouvir o que a *webcam* está a transmitir, invadindo assim a privacidade das pessoas a que a câmara pertence;

- Caso a ligação da câmara à rede interna na casa for insegura, o atacante pode conseguir obter dados pessoais dos utilizadores da rede, ou distribuir *malware* para outros dispositivos ligados à mesma rede;
- Ao ocorrer um acesso à *webcam* por uma entidade não autorizada, esta pode estudar o comportamento dos habitantes da casa, podendo ser capaz de, por exemplo, realizar um furto à casa quando estes não se encontram presentes.

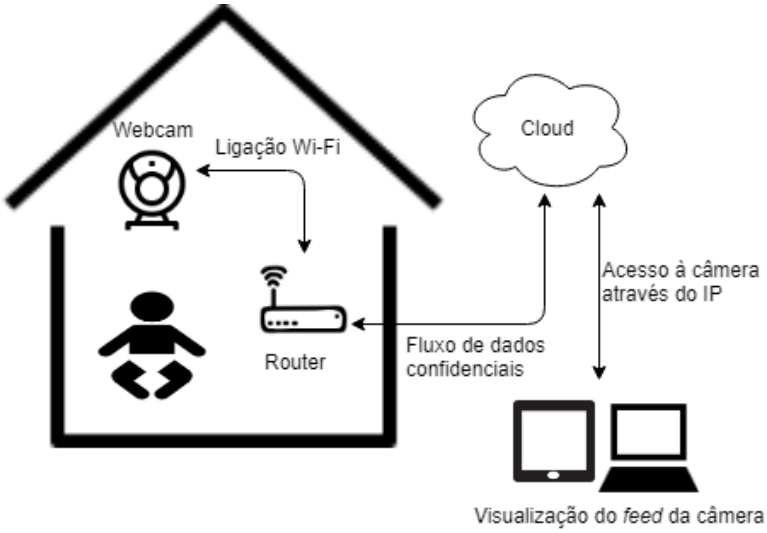


Figura 5.3: Representação do sistema IoT com uma *webcam* para monitorização de recém-nascidos.

5.3 Aplicação da Ferramenta nos Cenários

Esta secção aborda a aplicação da ferramenta desenvolvida no contexto desta dissertação aos cenários de utilização mencionados na secção anterior. Através de uma análise aos componentes e interações que fazem parte de cada sistema, são escolhidas as opções adequadas a cada questão que a ferramenta apresenta, e no final verifica-se o conteúdo presente no relatório produzido.

5.3.1 Cenário I – Transportes

Efetuada uma análise humana sobre a implementação de um sistema que corresponde a um carro com tecnologia moderna integrada, verifica-se que este sistema é composto por sensores que transmitem informação sobre detalhes técnicos do veículo para uma *cloud*, e contém um computador de bordo que é ligado à Internet. Este computador de bordo do carro controla algumas funcionalidades, como por exemplo a condução automática, e permite que dispositivos externos de passageiros se liguem ao carro. Através desta análise, as opções que são escolhidas nas questões apresentadas pela ferramenta demonstram-se nos seguintes items:

- À primeira pergunta sobre detalhes de implementação do sistema, nomeadamente sobre a

arquitetura deste, as opções selecionadas correspondem a sistemas embutidos (sensores e computador de bordo) que transmitem informações através de uma *Application Programming Interface (API)* para um servidor e, como também existe a possibilidade de se ligarem dispositivos externos, *smartphones* ou outros dispositivos semelhantes, estes também fazem parte do sistema;

- Relativamente a dados do sistema, estes são guardados numa base de dados e correspondem a informações do veículo, como por exemplo a sua localização. Como há uma ligação à Internet, todo o fluxo de informação que corresponde a essa ligação também faz parte do sistema. Todos estes dados são referenciados como confidenciais e também críticos;

- O tipo de autenticação implementado neste sistema corresponde a um nome de utilizador e palavra-passe padrão, visto ser necessário para que dispositivos externos como *smartphones* se possam ligar ao carro, de forma a terem uma ligação à Internet. Existindo só uma única conta, não há qualquer necessidade de haver um registo de utilizadores neste sistema;

- Este sistema não possui formulários em que o utilizador introduza manualmente dados, mas permite o envio de ficheiros, pois informações provenientes dos sensores do carro são enviadas para uma *cloud* através de ficheiros, e dispõe de um registo de *logs*;

- A nível de detalhes mais específicos sobre o hardware implementado, neste caso detalhes sobre os sensores do carro, estes não apresentam nenhum tipo de autenticação e utilizam como tecnologias de comunicação sem fios, o GPS e a rede 4G para transmissão de dados.

Após a execução da ferramenta com as respostas adequadas sobre um exemplo de um *smart car*, como demonstrado na figura 5.4, procedeu-se à observação dos resultados obtidos que irão ser abordados na próxima secção.

```
Architeture            :  Embedded System ; API Service ; Server Component ;
                          Mobile Application
Has DB                 :  Yes
Type of data storag    :  SQL
Which DB               :  SQL Server
Type of data stored    :  Confidential Data ; Critical Data
Authentication         :  Username and Password
User Registration      :  No
Type of Registration   :  N/A
Programming Languages  :  C# ; C / C++
Input Forms            :  No
Upload Files           :  Yes
The system has logs    :  Yes
Hardware Specification :  Yes
HW Authentication      :  No Authentication
HW Wireless Tech       :  4G / LTE ; GPS ; Bluetooth
```

Figura 5.4: Opções selecionadas para o cenário relativo a um *smart car*.

5.3.2 Cenário II – Equipamentos Médicos

Como foi referido na secção anterior, este sistema consiste num equipamento médico capaz de administrar medicação através de valores recebidos por um *tablet* ou *smartphone* pertencente ao médico responsável. O médico, além de indicar valores para administração de medicação ao paciente, pode também consultar o registo médico do mesmo. Analisando este sistema, as respostas escolhidas às questões da ferramenta apresentam-se como se segue:

- A arquitetura deste sistema corresponde a uma aplicação *mobile* que permite a administração de medicação e um servidor que armazena os dados médicos dos pacientes do hospital;
- Os dados correspondem a registos e históricos médicos dos pacientes e aos valores da medicação que são passados da aplicação para o equipamento médico. Estes dados são armazenados na base de dados do hospital e são considerados confidenciais e críticos;
- Relativamente ao tipo de autenticação, esta é implementada por nome de utilizador e palavra-passe. Para que cada médico possa ser autorizado a utilizar este equipamento existe um registo no sistema, efetuado por um administrador;
- Este sistema possui campos que permitem a inserção de dados por parte do utilizador, visto que os valores para administrar a medicação são introduzidos pelo médico. Não existe a funcionalidade de envio de ficheiros neste sistema, e há um registo de *logs*, em que informações como o historial de tomas de medicação e autenticações ao equipamento podem estar presentes.

Durante a execução da ferramenta, as opções selecionadas para este cenário sobre um equipamento médico refletem-se na figura 5.5.

```
Architeture            :  Mobile Application; Server Component;
Has DB                 :  Yes
Type of data storag    :  SQL
Which DB               :  MySQL
Type of data stored    :  Confidential Data ; Critical Data
Authentication         :  Username and Password
User Registration      :  Yes
Type of Registration   :  Will be a administrator that will register the users
Programming Languages  :  C# ; PHP ; Python
Input Forms            :  Yes
Upload Files           :  No
The system has logs    :  Yes
Hardware Specification :  Yes
HW Authentication      :  Basic Authentication(user/pass);
HW Wireless Tech       :  Bluetooth ; Wi-Fi
```

Figura 5.5: Opções selecionadas para o cenário relativo a um equipamento médico.

5.3.3 Cenário III – Vigilância Eletrónica

Por fim, o último cenário foca-se num exemplo de uma *webcam* que permite a monitorização de bebés. Esta câmara encontra-se ligada à rede sem fios da casa e a imagem e som que estão a ser gravados pela câmara são transmitidos para uma *cloud*. Através de um *tablet*, *smartphone*, ou computador, e do IP da *webcam*, é possível visualizar o que está a ser transmitido.

A respeito deste último cenário relativo a um equipamento para vigilância eletrónica, as opções selecionadas no decorrer da execução da ferramenta podem ser observadas na figura 5.6, e demonstram-se nas seguintes respostas:

- Relativamente à arquitetura deste sistema, este inclui um componente cliente, um componente servidor e um componente de sistema embutido;

- Os dados pertencentes a este sistema correspondem à imagem e ao som transmitidos pela *webcam*, mas não são armazenados. Quando o utilizador acede ao *feed* da câmara, é em tempo real. Estes dados são classificados como confidenciais, pois referem-se à privacidade dos utilizadores deste sistema;

- O utilizador acede à transmissão da câmara pelo IP correspondente, não havendo qualquer tipo de autenticação implementada nem registo de utilizadores;

- O sistema não implementa funcionalidades como inserção manual de dados, envio de ficheiros e registo de *logs*.

```
Architeture            :  Client Component; Server Component; Embedded System
Has DB                 :  No
Type of data storag    :  N/A
Which DB               :  N/A
Type of data stored    :  Confidential Data
Authentication         :  No Authentication
User Registration      :  No
Type of Registration   :  N/A
Programming Languages  :  Java ; Javascript
Input Forms            :  No
Upload Files           :  No
The system has logs    :  No
Hardware Specification :  No
HW Authentication      :  N/A
HW Wireless Tech       :  N/A
```

Figura 5.6: Opções selecionadas para o cenário relativo a uma *webcam* para monitorização de recém-nascidos.

5.4 Discussão dos Resultados

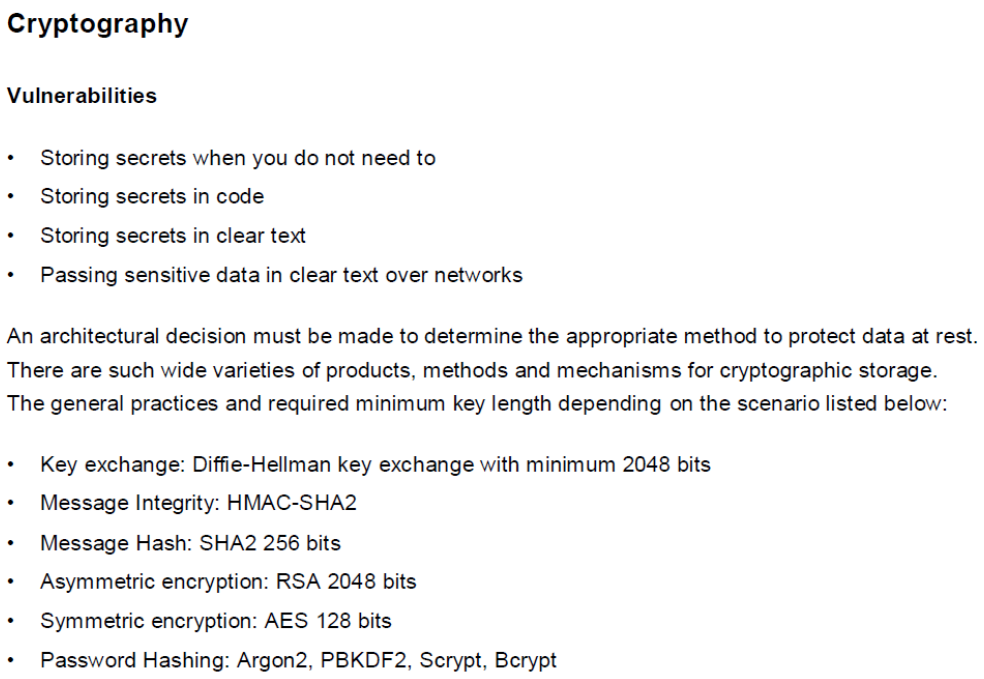
Esta secção apresenta uma discussão dos resultados obtidos pela ferramenta após a sua aplicação aos cenários de utilização identificados na secção 5.2. O primeiro exemplo dado refere-se a um *smart car*, que se encontra ligado à Internet e contém sensores que recolhem informações sobre o veículo. Efetuada uma análise humana à segurança deste sistema, esta indica que a segurança tem de ser aplicada em vários aspetos. O veículo tem de apresentar algum método de autenticação, visto que existe a possibilidade de *smartphones* se ligarem ao computador de bordo do carro. Devido à existência de sensores que transmitem informação sobre o veículo, estes têm de implementar alguma segurança, e como estes dados são armazenados numa base de dados, esta tem de estar devidamente protegida contra ataques de injeção. Durante a transmissão de dados do carro para o servidor, tanto os dados como a comunicação em si têm de ser cifrados. A comunicação é realizada através de uma API, e o respetivo desenvolvimento e aplicação desta deve seguir boas práticas de segurança. Algumas das informações do veículo são transmitidas em ficheiros, e deste modo é necessário aplicar medidas de segurança no que diz respeito à confidencialidade e correção dos ficheiros. Este sistema de software, bem como muitos outros, implementam um registo de *logs* e, como tal, é fundamental aplicar práticas seguras relativas à informação que é exposta nos *logs*.

Depois de analisar este sistema de um ponto vista humano, relativamente à segurança, procedeu-se à aplicação da ferramenta a este caso, e obtiveram-se resultados que consistem em boas práticas de segurança a adotar, relacionando-se diretamente com os aspetos abordados na análise humana. Conforme as opções escolhidas pelo utilizador às questões apresentadas pela ferramenta, foi produzido um relatório contendo secções de práticas seguras envolvendo vários temas. A documentação que foi produzida no caso do *smart car* contém práticas seguras sobre segurança em sistemas embutidos, pois foi selecionada a opção de sistemas embutidos à pergunta sobre componentes que fazem parte da arquitetura do sistema. Esta secção do relatório obtido aborda práticas que se relacionam com a segurança física, a inicialização segura do dispositivo, a segurança a nível do sistema operativo e da aplicação, gestão de credenciais, encriptação e sobre segurança na ligação em rede. Algumas destas práticas podem ser visualizadas no anexo A.8. Além destas práticas sobre segurança em sistemas embutidos, o relatório também aborda outros temas, como por exemplo práticas seguras sobre autenticação (anexo A.4), API e criptografia. A figura 5.7 demonstra a parte do relatório produzido e que aborda o tema de criptografia, onde são apresentadas vulnerabilidades, e algoritmos criptográficos que devem ser utilizados, demonstrando evidências da utilidade e alinhamento dos resultados com o pretendido.

Para completar esta documentação, existe conteúdo relacionado com práticas de prevenção de ataques de injeção à base de dados, práticas seguras sobre o envio de ficheiros e sobre gestão de erros e *logs*. Estas informações foram colocadas no relatório devido à resposta afirmativa às questões sobre se o sistema tinha uma base de dados, se permitia envio de ficheiros e se continha um registo de *logs*. Na figura 5.8 pode observar-se um excerto da secção sobre implementação segura de um registo de *logs*, e no anexo A.7 é mostrada a maior parte das práticas seguras relativas a este tema.

O segundo cenário estudado retrata um equipamento médico que permite, através de um dis-

positivo móvel pertencente a um médico, administrar doses de medicação a um paciente. Uma análise prévia à segurança deste sistema (i.e., anterior à aplicação da ferramenta) conclui que é necessário existir autenticação, visto que o médico precisa de se autenticar com sucesso, e também ser autorizado para usufruir do equipamento. Para além disso, tanto os dados como a comunicação precisam de ser devidamente cifrados, e a base de dados do hospital tem de estar protegida contra ataques de injeção. Como o médico introduz os valores da toma da medicação no dispositivo móvel, sendo que estes são depois passados ao equipamento médico (para que este administre o valor da toma do medicamento ao paciente), é necessário existir uma validação destes dados de entrada. Isto porque poderá haver um erro humano, ou até mesmo uma alteração não autorizada nestes valores, o que pode causar danos. Este sistema também implementa um registo de *logs* e, como tal, é preciso ter conta quais as informações a serem expostas, nomeadamente tentativas de acesso, autenticação de utilizadores, envio de informação relativa sobre a administração de medicamentos, ou consulta de dados da base de dados do hospital.



Cryptography

Vulnerabilities

- Storing secrets when you do not need to
- Storing secrets in code
- Storing secrets in clear text
- Passing sensitive data in clear text over networks

An architectural decision must be made to determine the appropriate method to protect data at rest. There are such wide varieties of products, methods and mechanisms for cryptographic storage. The general practices and required minimum key length depending on the scenario listed below:

- Key exchange: Diffie-Hellman key exchange with minimum 2048 bits
- Message Integrity: HMAC-SHA2
- Message Hash: SHA2 256 bits
- Asymmetric encryption: RSA 2048 bits
- Symmetric encryption: AES 128 bits
- Password Hashing: Argon2, PBKDF2, Scrypt, Bcrypt

Figura 5.7: Excerto da secção sobre práticas seguras de criptografia.

A aplicação da ferramenta a este exemplo produziu um relatório com boas práticas seguras que confirmam o que foi discutido numa análise humana à segurança ao sistema. Visto que o médico que necessite de usar o equipamento tem de ser autenticado com sucesso, e como a base de dados contém dados sensíveis, práticas seguras sobre autenticação, prevenção contra ataques de injeção à base de dados e criptografia estão presentes na documentação produzida. No questionário da ferramenta, a questão sobre se o sistema possui introdução de dados foi respondida afirmativamente e, como efeito, uma secção sobre segurança a nível de validação de dados irá ser apresentada no relatório obtido. Um exemplo destas práticas referidas pode ser visualizado na figura 5.9.

Como poderá haver mais do que um médico a usufruir do equipamento médico referido anteriormente, durante a execução da ferramenta foi respondido, às perguntas relacionadas com

utilizadores, que havia um registo de utilizadores no sistema e que esse registo era efetuado por um administrador. Dadas estas respostas, o relatório apresenta uma secção sobre práticas de controlo de acesso. Como este sistema também tem um registo de *logs*, a secção de boas práticas seguras sobre este tema também se encontra na documentação produzida pela ferramenta.

Which events to log

- Input validation failures e.g. protocol violations, unacceptable encodings, invalid parameter names and values
- Output validation failures e.g. database record set mismatch, invalid data encoding
- Authentication successes and failures
- Authorization (access control) failures
- Session management failures e.g. cookie session identification value modification
- Application errors and system events e.g. syntax and runtime errors, connectivity problems, performance issues, third party service error messages, file system errors, file upload virus detection, configuration changes
- Application and related systems start-ups and shut-downs, and logging initialization (starting, stopping or pausing)
- Use of higher-risk functionality e.g. network connections, addition or deletion of users, changes to privileges, assigning users to tokens, adding or deleting tokens, use of systems administrative privileges, access by application administrators, access to payment cardholder data, key changes, submission of user-generated content - especially file uploads.

Figura 5.8: Excerto da secção sobre práticas seguras de gestão de *Logs*.

Input Validation

This section appears because you answer "Yes" in the question about the system using a input forms.

Input validation is performed to ensure only properly formed data is entering the workflow in an information system, preventing malformed data from persisting in the database and triggering malfunction of various downstream components.

Vulnerabilities

- Using non-validated input in the Hypertext Markup Language (HTML) output stream
- Using non-validated input used to generate SQL queries
- Relying on client-side validation
- Looking for known bad patterns of input
- Failing to validate input from all sources including cookies, query string parameters, HTTP headers, databases, and network resources

Implementing input validation

- Data type validators available natively in web application frameworks
- Type conversion (e.g. Integer.parseInt() in Java, int() in Python) with strict exception handling
- Minimum and maximum value range check for numerical parameters and dates
- Minimum and maximum length check for strings.
- Array of allowed values for small sets of string parameters (e.g. days of week).
- Regular expressions for any other structured data covering the whole input string (^...$) and not using "any character" wildcard (such as . or \S)

Figura 5.9: Excerto da secção sobre práticas seguras de validação de entradas.

O último caso de estudo apresentado neste capítulo engloba uma *webcam* para monitorização de recém-nascidos em que a transmissão da imagem e som que são gravados podem ser acedidos através de um dispositivo móvel pela *Web*. Uma análise à implementação deste sistema enuncia que os dados transmitidos pela câmara só devem ser acedidos por entidades responsáveis, indicando assim que é necessário haver algum tipo de autenticação. Como esta câmara, de forma a transmitir dados, se liga à rede interna da casa, esta ligação tem de ser segura, e também os dados que são enviados devem ser cifrados. Este sistema engloba também a implementação de um registo de *logs*, sendo importante adotar medidas de segurança sobre este tema.

Após a aplicação da ferramenta a este exemplo, obteve-se documentação de boas práticas seguras sobre autenticação, criptografia e gestão de *logs* e erros. Devido ao facto de a transmissão dos dados da câmara serem visualizados através da *Web*, parte deste sistema pode ser considerado orientado à *Web*. Como tal, práticas seguras sobre *cross site scripting* e gestão de sessões são acrescentadas ao relatório obtido. Um excerto de boas práticas de segurança relativas a prevenção de ataques de *cross site scripting* pode ser visualizado no anexo A.6.

5.5 Conclusões

Neste capítulo foram abordados três cenários de utilização pertencentes a três grandes áreas: transportes, saúde e casa. A aplicação da ferramenta desenvolvida nestes exemplos produziu documentação necessária sobre boas práticas de segurança a adotar no desenvolvimento destes sistemas, de modo a apresentarem segurança desde o início da implementação, evitando assim potenciais vulnerabilidades. Os temas das boas práticas seguras que foram fornecidas pela ferramenta nestes casos entram em concordância com a análise humana que se fez previamente, o que indica que a ferramenta é útil no auxílio de integração de segurança em sistemas. Esta análise humana à segurança dos sistemas apresentados foi efectuada com auxílio de alunos de doutoramento que têm como especialidade a segurança informática. Com o intuito de auxiliar a integração de segurança, esta ferramenta foi testada também num caso real a pedido de uma empresa relacionada com dispositivos IoT, mas devido a um acordo de confidencialidade, informações sobre este caso não podem ser divulgadas.

Capítulo 6

Conclusões e Trabalho Futuro

Esta dissertação descreveu um projeto de investigação e desenvolvimento na área de segurança de sistema de softwares, nomeadamente, sistemas IoT, com o objetivo principal de criar uma ferramenta que servisse como guia a *developers* na adoção de boas práticas de segurança durante a implementação de um sistema. Este último capítulo apresenta as conclusões principais do trabalho realizado no contexto desta dissertação na secção 6.1, e aponta, na secção 6.2, potenciais linhas de investigação e desenvolvimento que podem ser usadas para melhorar este trabalho no futuro.

6.1 Conclusões Principais

O objetivo principal do trabalho apresentado neste documento consistia no desenho e elaboração de uma ferramenta que tem como função o auxílio a arquitetos e programadores de sistemas, mesmo aqueles que não têm muito conhecimento na área da segurança, a adotar medidas de segurança em fases iniciais do desenvolvimento de um sistema a implementar. A ideia que levou à criação desta ferramenta, e que também vai de encontro à motivação da sua construção, assenta na necessidade de auxiliar uma integração mais significativa de segurança na elaboração de sistemas modernos, visto que, atualmente, sistemas pertencentes a áreas de inovação, como por exemplo a área de IoT, não são elaborados com a segurança em mente desde os estágios iniciais de desenvolvimento. A implementação destes sistemas apresentam alguma complexidade significativa, fazendo com que os *developers* se foquem no desenvolvimento de funcionalidades, ignorando a aplicação de medidas de segurança. Havendo também uma menor percentagem na existência ferramentas que auxiliam a integração de segurança desde o início do desenvolvimento, e de falta de experiência em segurança por parte dos arquitetos do sistema, *security-by-design* tornou-se um grande desafio. Devido ao facto de existir alguma ignorância no que diz respeito à integração de segurança desde o início do desenvolvimento de sistemas, vulnerabilidades não são corrigidas durante a implementação, levando assim à exploração destas, podendo ocorrer violações de dados e incidentes. De forma a ajudar a colmatar estes problemas, procedeu-se ao planeamento e desenvolvimento de uma ferramenta simples e de fácil uso que produzisse documentação para auxiliar os *developers* a adotar medidas de segurança logo quando iniciam o desenvolvimento de um sistema. O planeamento e desenvolvimento da ferramenta discutida no contexto desta dissertação englobou várias etapas, nomeadamente:

- Estudo dos processos de engenharia de segurança do software, ferramentas existentes que permitissem a análise de um sistema auxiliando a integração de segurança e a pesquisa de documentação relacionada com boas práticas de segurança;

- Formulação de questões simples e respetivas respostas que permitissem a identificação de componentes, propriedades e interações que fazem parte do sistema a desenvolver;
- Transformação das questões num *software* protótipo, que a partir das opções selecionadas produzisse um relatório sobre boas práticas de segurança a ter em conta no desenvolvimento;
- Criação e estudo de alguns cenários de utilização para aplicação da ferramenta protótipo desenvolvida;
- Aplicação de melhorias à ferramenta e respetiva documentação.

Durante a investigação do trabalho relacionado, nomeadamente ferramentas existentes, notou-se que a maior parte das ferramentas que fazem parte da etapa de *design* no desenvolvimento de um sistema referem-se à modelação de ameaças, em que o método de utilização é o desenho do sistema através de diagramas, o que por vezes pode ser um pouco complexo de usar. Uma das etapas antes de se proceder ao desenvolvimento da ferramenta consistiu na recolha de boas práticas de segurança, no entanto, houve alguma dificuldade na execução desta tarefa visto que não existe muita documentação relacionada com este assunto. Concluindo a etapa de investigação, notou-se que, apesar de uma intensa pesquisa, o esforço atual direcionado à elaboração de documentação de práticas de segurança a adotar na implementação de sistemas de software, nomeadamente sistemas de IoT, bem como ferramentas que permitam a integração de segurança em estágios iniciais do desenvolvimento, é quase inexistente. Relativamente ao planeamento da ferramenta, o aspeto mais desafiante e que acarretou algumas dificuldades foi a identificação de um conjunto de perguntas que permitisse estimar a constituição de um sistema, como por exemplo, componentes, características e funcionalidades de forma a abordar aspetos relevantes de segurança.

A ferramenta desenvolvida no âmbito desta dissertação permite que os seus utilizadores, rapidamente e de uma forma simplista, tenham documentação sobre boas práticas seguras a adotar, consoante o sistema a desenvolver, servindo como um guia na integração de segurança desde os estágios iniciais no desenvolvimento do sistema, evitando assim falhas de segurança. Isto é alcançado através de uma série de questões simples, em que o utilizador introduz informações sobre o sistema e, em troca, é-lhe fornecido documentação sobre práticas de segurança que deve ter em conta durante a implementação do sistema. Esta abordagem é nova na área, e esta ferramenta pode ser utilizada por aqueles que pouco ou nenhum conhecimento têm em segurança de sistemas. Deste modo, a aplicação desta ferramenta fornece uma contribuição importante para melhorar a segurança em sistemas de software, como por exemplo sistema pertencentes à área de IoT. A ferramenta descrita neste documento faz parte de um conjunto de ferramentas que estão a ser desenvolvidas no âmbito de um projeto de investigação e desenvolvimento denominado SECUR IoT ESIGN (ver `https://lx.it.pt/securIoTesign/`) no qual esta ferramenta é denominada por SECURiotPRACTICES. O desenvolvimento deste projeto pode ser seguido em `https://github.com/SECURIoTESIGN/SECURIoTESIGN`. Além da ferramenta desenvolvida no contexto desta dissertação encontrar-se no *github* do projeto referido anteriormente também se encontra no *github* pessoal do autor (`https://github.com/EdiAires96/SECURiotPRACTICES`). Algumas ferramentas deste projeto, inclusive a ferramenta desenvolvida relacionada com boas práticas de segurança que foi descrita ao longo deste documento, foram testadas num caso real na área de *smart transportation*, a pedido de uma empresa sob um acordo de confidencialidade, e de acordo com o *feedback* recebido, alguns melhoramentos foram efetuados à ferramenta.

6.2 Trabalho Futuro

Tratando-se de um protótipo, a ferramenta desenvolvida pode ser melhorada em diversos aspetos de forma a aprimorar a sua utilidade e valor. Exemplos de várias funcionalidades que poderiam ser adicionadas constam a seguir:

- Adicionar mais questões e respetiva documentação sobre boas práticas de segurança;
- Implementação de uma base de dados para que, de uma forma dinâmica, seja possível a introdução das questões e respetivas opções bem como documentação de boas práticas de segurança na ferramenta;
- Inserção de exemplos de código nas diversos temas de práticas seguras de forma a dar alguns *insights* sobre como aplicar na prática;
- Migração desta ferramenta para a *Web*, isto é, disponibilizar esta ferramenta *online* que através de um *browser* se possa executar esta ferramenta e obter os respetivos resultados;
- Adicionar práticas de segurança relativamente à implementação de hardware;
- Integração de ameaças e vulnerabilidades presentes no repositório CVE;
- Adicionar documentação sobre leis e regulamentos acerca de proteção de dados;
- Nomeação de exemplos de ferramentas para serem aplicadas na fase de testes ao sistema;
- Transformação das boas práticas de segurança numa lista de tarefas.

Bibliografia

[ATB03] Auer, Tschurtschenthaler, and Biffl. A Flyweight UML Modelling Tool for Software Development in Heterogeneous Environments. In *2003 Proceedings 29th Euromicro Conference*, pages 267–272, Sep. 2003. 12

[Bay11] J. L. Bayuk. Systems Security Engineering. *IEEE Security Privacy*, 9(2):72–74, March 2011. 1

[Boe88] Barry W. Boehm. A Spiral Model of Software Development and Enhancement. *Computer*, 21(5):61–72, May 1988. Available from: http://dx.doi.org/10.1109/2.59. xiii, 8, 9

[Che] Cheatography. Cheatography [online]. Available from: https://www.cheatography.com/explore/ [cited 24 Junho 2019]. 17

[CNN17] CNN by Selena Larson. FDA Confirms That St. Jude's Cardiac Devices Can Be Hacked [online]. 2017. Available from: https://money.cnn.com/2017/01/09/technology/fda-st-jude-cardiac-hack/ [cited 24 de Junho 2019]. 2

[Con] Continuum Security. Irius Risk [online]. Available from: https://continuumsecurity.net/threat-modeling-tool/ [cited 24 Junho 2019]. 15

[Din] Dinu Gherman. Svglib [online]. Available from: https://pypi.org/project/svglib/ [cited 24 Junho 2019]. 29

[EAS+11] Golnaz Elahi, Tom Aratyn, Ramanan Sivaranjan, Rohit Sethi, and SK Eric. SD Elements: A Tool for Secure Application Development Management. In *CAiSE Forum*, pages 81–88, 2011. 14

[Fir] First. Common Vulnerability Scoring System [online]. Available from: https://www.first.org/cvss/ [cited 24 Junho 2019]. 15

[FMS07] Ivan Flechais, Cecilia Mascolo, and M. Angela Sasse. Integrating Security and Usability into the Requirements and Design Process. *Int. J. Electron. Secur. Digit. Forensic*, 1(1):12–26, May 2007. Available from: http://dx.doi.org/10.1504/IJESDF.2007.013589. 10

[GRMNT10] Sigrid Gürgens, Carsten Rudolph, Antonio Maña, and Simin Nadjm-Tehrani. Security Engineering for Embedded Systems - The SecFutur Vision. *ACM International Conference Proceeding Series*, 09 2010. 1

[HKZ18] O. Hachinyan, A. Khorina, and S. Zapechnikov. A Game-heoretic Technique for Securing IoT Devices against Mirai Botnet. In *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, pages 1500–1503, Jan 2018. 2

[IoT] IoT Security Foundation. Secure Design Best Pratices [online]. Available from: https://www.iotsecurityfoundation.org/wp-content/uploads/2019/03/Best-Practice-Guides-Release-1.2.1.pdf [cited 24 Junho 2019]. 16, 34

[Joh] John Gruber. Markdown [online]. Available from: https://daringfireball.net/projects/markdown/ [cited 24 Junho 2019]. 28

[Jü02] Jan Jürjens. UMLsec: Extending UML for Secure Systems Development. In *UMLsec: Extending UML for Secure Systems Development*, volume 2460, pages 412–425, 01 2002. 12

[Kas15] Kaspersky. Black Hat USA 2015: The Full Story of How That Jeep Was Hacked [online]. 2015. Available from: https://www.kaspersky.com/blog/blackhat-jeep-cherokee-hack-explained/9493/ [cited 24 de Junho 2019]. 2

[KKS17] G. Kambourakis, C. Kolias, and A. Stavrou. The Mirai Botnet and the IoT Zombie Armies. In *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, pages 267–272, Oct 2017. 2

[KKSV17] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. DDoS in the IoT: Mirai and Other Botnets. *Computer*, 50(7):80–84, 2017. 2

[LBD02] Torsten Lodderstedt, David A. Basin, and Jürgen Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *Proceedings of the 5th International Conference on The Unified Modeling Language*, UML '02, pages 426–441, London, UK, UK, 2002. Springer-Verlag. Available from: http://dl.acm.org/citation.cfm?id=647246.719477. 12

[LL17] W. Lee and P. Law. A Case Study in Applying Security Design Patterns for IoT Software System. In *2017 International Conference on Applied System Innovation (ICASI)*, pages 1162–1165, May 2017. 2

[Lui] Luis Zarate. XHTML2PDF [online]. Available from: https://pypi.org/project/xhtml2pdf/ [cited 24 Junho 2019]. 29

[McG06] Gary McGraw. *Software Security: Building Security In*. Addison-Wesley Professional, 2006. xiii, 10, 11

[Mica] Microsoft. Cheat Sheet: Web Application Security Frame [online]. Available from: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649461%28v%3dpandp.10%29 [cited 24 Junho 2019]. 16, 34

[Micb] Microsoft. Microsoft SDL [online]. Available from: https://www.microsoft.com/en-us/securityengineering/sdl/ [cited 10 Junho 2019]. 9

[Mic09] Microsoft. Announcing SDL for Agile Development Methodologies [online]. 2009. Available from: https://www.microsoft.com/security/blog/2009/11/10/announcing-sdl-for-agile-development-methodologies/ [cited 10 Junho 2019]. 8

[moz] mozilla wiki. WebAppSec/Secure Coding Guidelines [online]. Available from: https://wiki.mozilla.org/WebAppSec/Secure_Coding_Guidelines [cited 24 Junho 2019]. 16, 34

[MRRR02] Nenad Medvidovic, David S. Rosenblum, David F. Redmiles, and Jason E. Robbins. Modeling Software Architectures in the Unified Modeling Language. *ACM Trans. Softw. Eng. Methodol.*, 11(1):2–57, January 2002. Available from: http://doi.acm.org/10.1145/504087.504088. 12

[Nata] National Vulnerability Database. Common Platform Enumeration [online]. Available from: https://nvd.nist.gov/products/cpe [cited 24 Junho 2019]. 15

[Natb] National Vulnerability Database. Common Vulnerabilities and Exposures [online]. Available from: https://cve.mitre.org/ [cited 24 Junho 2019]. 15

[OWAa] OWASP. IoT Security Guidance [online]. Available from: https://www.owasp.org/index.php/IoT_Security_Guidance [cited 24 Junho 2019]. 34

[OWAb] OWASP. OWASP CheatSheet Series [online]. Available from: https://github.com/OWASP/CheatSheetSeries/tree/master/cheatsheets [cited 24 Junho 2019]. 16

[OWAc] OWASP. OWASP Secure Coding Practices Quick Reference Guide [online]. Available from: https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf [cited 24 Junho 2019]. 16, 34

[OWAd] OWASP. OWASP Threat Dragon [online]. Available from: https://www.owasp.org/index.php/OWASP_Threat_Dragon [cited 24 Junho 2019]. 15

[Pot09] Bruce Potter. Microsoft SDL Threat Modelling Tool. *Network Security*, 2009(1):15–18, 2009. 13

[Res18] Tobias Reski. Conception and Development of a Threat Modeling Tool, 2018. Available from: https://opus.hs-offenburg.de/frontdoor/deliver/index/docId/3339/file/Bachelorthesis_Tobias_Reski_18.02.2019.pdf. 15

[Roy87] W. W. Royce. Managing the Development of Large Software Systems: Concepts and Techniques. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press. Available from: http://dl.acm.org/citation.cfm?id=41765.41801. xiii, 7, 8

[SAN] SANS. Securing Web Application Technologies [SWAT] Checklist [online]. Available from: https://software-security.sans.org/resources/swat [cited 24 Junho 2019]. 17

[SFHB05] Markus Schumacher, Eduardo Fernandez, Duane Hybertson, and Frank Buschmann. *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, Inc., USA, 2005. 12

[Shl] Shlomi Fish. Graph-Easy [online]. Available from: https://metacpan.org/release/Graph-Easy [cited 24 Junho 2019]. 29

[SR+19] Phd Student, Diego Ray, Advisors , Antonio Maña, and Mariemma Valle. Integration of Security Patterns in Software Models based on Semantic Descriptions, 07 2019. 1

[TEA] TEAM academy eKnowledge. Team Mentor 4.5 [online]. Available from: https://teammentor.net/angular/user/index [cited 24 Junho 2019]. 15

[Vie05] John Viega. Building Security Requirements with CLASP. *SIGSOFT Softw. Eng. Notes*, 30(4):1–7, May 2005. Available from: http://doi.acm.org/10.1145/1082983.1083207. 10

[Vli08] Hans van Vliet. *Software Engineering: Principles and Practice*. Wiley Publishing, 3rd edition, 2008. 7

[Way] Waylan Limberg. Python-Markdown [online]. Available from: https://python-markdown.github.io/ [cited 24 Junho 2019]. 29

[Wir12] Wired. Flaw in Home Security Cameras Exposes Live Feeds to Hackers [online]. 2012. Available from: https://www.wired.com/2012/02/home-cameras-exposed/ [cited 24 de Junho 2019]. 2

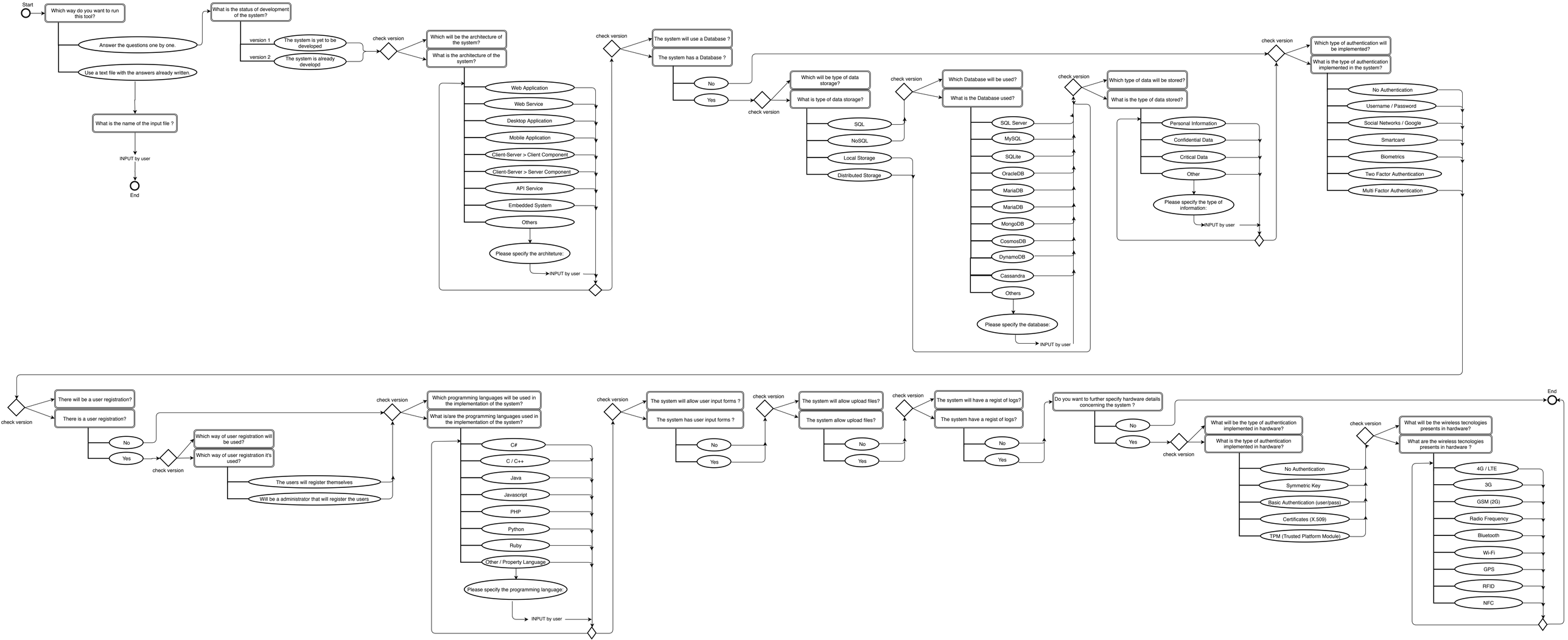
[WY15] Imano Williams and Xiaohong Yuan. Evaluating the Effectiveness of Microsoft Threat Modeling Tool. In *Proceedings of the 2015 Information Security Curriculum Development Conference*, InfoSec '15, pages 9:1–9:6, New York, NY, USA, 2015. ACM. Available from: http://doi.acm.org/10.1145/2885990.2885999. 13

Apêndice A

Anexos

A.1 Fluxo da Ferramenta

Este anexo contém todas as perguntas e respetivas respostas apresentadas pela ferramenta protótipo desenvolvida no contexto desta dissertação (página seguinte).


Start
Which way do you want to run this tool?
Answer the questions one by one.
Use a text file with the answers already written.
What is the name of the input file ?
INPUT by user
End
What is the status of development of the system?
version 1
The system is yet to be developed
version 2
The system is already developd
check version
Which will be the architecture of the system?
What is the architecture of the system?
Web Application
Web Service
Desktop Application
Mobile Application
Client-Server > Client Component
Client-Server > Server Component
API Service
Embedded System
Others
Please specify the architeture:
INPUT by user
check version
The system will use a Database ?
The system has a Database ?
No
Yes
check version
Which will be type of data storage?
What is type of data storage?
SQL
NoSQL
Local Storage
Distributed Storage
check version
Which Database will be used?
What is the Database used?
SQL Server
MySQL
SQLite
OracleDB
MariaDB
MariaDB
MongoDB
CosmosDB
DynamoDB
Cassandra
Others
Please specify the database:
INPUT by user
check version
Which type of data will be stored?
What is the type of data stored?
Personal Information
Confidential Data
Critical Data
Other
Please specify the type of information:
INPUT by user
check version
Which type of authentication will be implemented?
What is the type of authentication implemented in the system?
No Authentication
Username / Password
Social Networks / Google
Smartcard
Biometrics
Two Factor Authentication
Multi Factor Authentication
check version
There will be a user registration?
There is a user registration?
No
Yes
check version
Which way of user registration will be used?
Which way of user registration it's used?
The users will register themselves
Will be a administrator that will register the users
check version
Which programming languages will be used in the implementation of the system?
What is/are the programming languages used in the implementation of the system?
C#
C / C++
Java
Javascript
PHP
Python
Ruby
Other / Property Language
Please specify the programming language:
INPUT by user
check version
The system will allow user input forms ?
The system has user input forms ?
No
Yes
check version
The system will allow upload files?
The system allow upload files?
No
Yes
check version
The system will have a regist of logs?
The system have a regist of logs?
No
Yes
Do you want to further specify hardware details concerning the system ?
No
Yes
check version
What will be the type of authentication implemented in hardware?
What is the type of authentication implemented in hardware?
No Authentication
Symmetric Key
Basic Authentication (user/pass)
Certificates (X.509)
TPM (Trusted Platform Module)
check version
What will be the wireless tecnologies presents in hardware?
What are the wireless tecnologies presents in hardware ?
4G / LTE
3G
GSM (2G)
Radio Frequency
Bluetooth
Wi-Fi
GPS
RFID
NFC
End

A.2 Função que Permite a Validação de *Input*

Este anexo apresenta o excerto de código que permite efetuar a validação de entrada de dados (*input*) na ferramenta por parte do utilizador (excerto de código A.1).

```
def validateInput(*arg):
  while True:
  # validate a integer
    if arg[0] == 1:
      try:
        user_input = input("  > ")
      # syntax error, name error
      except (SyntaxError, NameError, TypeError):
        print("  Not a valid answer!  ")
        print("")
        continue
    else:
      if (type(user_input) is int) and (user_input in range(0, arg[1])):
        break
      else:
        print("  Not a valid answer!  ")
        print("")

  # validate a string
  if arg[0] == 2:
    try:
      user_input = input("  > ")
    # syntax error, name error
    except (SyntaxError, NameError, TypeError):
      print("  Not a valid answer!  ")
      print("")
      continue
  else:
    if type(user_input) is str:
      break
    else:
      print("  Not a valid answer!  ")
      print("")

 return user_input
```

Listing A.1: Função de validação de *input*.

A.3 Algoritmo para Associar as Respetivas Respostas às Opções Escolhidas

Este anexo demonstra um excerto de código do algoritmo que permite associar as respetivas respostas das questões às opções selecionadas (números introduzidos) pelo utilizador (excerto de código A.2).

```
list_aux = []
# find if the answer corresponds to option "others" (means that is user input
    text) - verify if it the answer has only letters
# find if the first character is a letter and if the answer has no more options
if questions_and_answers["Q1"][0].isdigit() == False and
   questions_and_answers["Q1"].find(";") == -1 :

   list_aux.append( questions_and_answers["Q1"])

else:
  # variable aux is a list that contains the answers chosen by the user to the
      question in cause
  # cut the string in the delimitator ";"
  aux = questions_and_answers["Q1"].split(";")

  # delete last item (= None)
  aux = aux[:-1]

  # iterate the answers chosen by the user
    for item in aux:
      # iterate the options of the question and check what the chosen answers
          match
      for option in question_1:
        if item == option:
          list_aux.append( question_1[option] )

        # case of user input text
        if item.isdigit() == False:
          list_aux.append (item)

print("{:22} {:3} {:40} ".format("Architeture", ":", ' ; '.join(list_aux) ) )
table_for_report.append([ "Architeture" , ' ; '.join(list_aux) ])
answers_list.append(questions_and_answers["Q1"])
comments_list.append(' ; '.join(list_aux))
```

Listing A.2: Código que demonstra a associação das opções selecionadas pelo utilizador à resposta em si.

A.4 Excerto da Secção sobre Práticas Seguras na Autenticação

Este anexo contém um excerto do relatório gerado após utilização da ferramenta protótipo (figura A.1). Neste caso, o excerto exibido diz respeito Práticas Seguras sobre Autenticação.

```
Authentication

Authentication is the process of verification that an individual, entity
or website is who it claims to be. Authentication is commonly performed
by submitting a user name or ID and one or more items of private information
that only a given user should know.

Vulnerabilities
• Using weak passwords
• Storing clear text credentials in configuration files
• Passing clear text credentials over the network
• Permitting over-privileged accounts
• Permitting prolonged session lifetime
• Mixing personalization with authentication

User ID's
Make sure your usernames/userids are case insensitive.
User 'smith' and user 'Smith' should be the same user.
User names should also be unique.
Email address as a USER ID.

Password Complexity
Applications should enforce password complexity rules to discourage easy to
guess passwords. Password mechanisms should allow virtually any character
the user can type to be part of  their password, including the space
character. Passwords should, obviously, be case sensitive in order to increase
their complexity. The password change mechanism should require a minimum
level of complexity that makes sense for the application and its user
population.

• Password must meet at least 3 out of the following 4 complexity rules
• at least 1 uppercase character (A-Z)
• at least 1 lowercase character (a-z)
• at least 1 digit (0-9)
• at least 1 special character (punctuation) space included
• at least 10 characters
• at most 128 characters
```

Figura A.1: Excerto da secção sobre práticas seguras sobre autenticação.

A.5 Excerto da Secção sobre Práticas Seguras na Validação de *Input*

Este anexo contém um excerto do relatório gerado após utilização da ferramenta desenvolvida no contexto do projeto de dissertação (figura A.2). Neste caso, o excerto apresentado diz respeito a Práticas Seguras sobre Validação de *Input*.

```
Input Validation

This section appears because you answer "Yes" in the question about the
system using a input forms.

Input validation is performed to ensure only properly formed data is entering
the workflow in an information system, preventing malformed data from
persisting in the database and triggering malfunction of various downstream
components.

Vulnerabilities
• Using non-validated input in the Hypertext Markup Language (HTML)
output stream
• Using non-validated input used to generate SQL queries
• Relying on client-side validation
• Using input file names, URLs, or user names for security decisions
• Using application-only filters for malicious input
• Looking for known bad patterns of input
• Trusting data read from databases, file shares, and other network resources
• Failing to validate input from all sources including cookies, query
string parameters, HTTP headers, databases, and network resources

Implementing input validation
• Data type validators available natively in web application frameworks
• Validation against JSON Schema and XML Schema (XSD) for input in these
formats.
• Type conversion (e.g. Integer.parseInt() in Java, int() in Python)
with strict exception handling
• Minimum and maximum value range check for numerical parameters and dates
• Minimum and maximum length check for strings.
• Array of allowed values for small sets of string parameters
(e.g. days of week).
• Regular expressions for any other structured data covering the whole input
string (^...$) and not using "any character" wildcard (such as . or \S)
```

Figura A.2: Excerto da secção sobre práticas seguras sobre validação de *input*.

A.6 Excerto da Secção sobre Práticas Seguras Relativas a *Cross Site Scripting*

Este anexo contém um excerto do relatório gerado após utilização da ferramenta protótipo descrita neste documento (figura A.3). Neste caso, o excerto retratado a seguir refere-se a Práticas Seguras sobre *Cross Site Scripting*.

```
Cross Site Scripting (XSS)

This section appears because you choose the option "Web Application" or
"Web Service" in the question about the architecture of the system.

Given the way browsers parse HTML, each of the different types of slots has
slightly different security rules. When you put untrusted data into these
slots, you need to take certain steps to make sure that the data does not
break out of that slot into context that allows code execution.

XSS Prevention Rules
• Never Insert Untrusted Data Except in Allowed Locations - The first rule
is to deny all
• HTML Escape Before Inserting Untrusted Data into HTML Element Content
• Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes
• JavaScript Escape Before Inserting Untrusted Data into JavaScript Data Values
• HTML escape JSON values in an HTML context and read the data with JSON.parse
• Ensure returned Content-Type header is application/json and not text/html.
• CSS Escape And Strictly Validate Before Inserting Untrusted Data
into HTML Style Property Values
• URL Escape Before Inserting Untrusted Data into HTML URL Parameter Values
• Sanitize HTML Markup with a Library Designed for the Job
• Prevent DOM-based XSS
• Use HTTPOnly cookie flag
• Implement Content Security Policy
• Use an Auto-Escaping Template System
• Use the X-XSS-Protection Response Header
• Properly use modern JS frameworks like Angular (2+) or ReactJS

References
Part of the text in this section was adapted or taken directly from: OWASP
CheatSheet Series, Secure Coding Guidelines by Mozilla wiki
https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets
/Cross_Site_Scripting_Prevention_Cheat_Sheet.md
https://wiki.mozilla.org/WebAppSec/Secure_Coding_Guidelines
```

Figura A.3: Excerto da secção sobre práticas seguras sobre *cross site scripting*.

A.7 Excerto da Secção sobre Práticas Seguras em *Logging* e Gestão de Erros

Este anexo contém um excerto do relatório produzido referente a Práticas Seguras em *Logging* e Gestão de Erros (figura A.4).

```
Logging and Error Handling

This section appears because you answer "Yes" in the question about the system using
a logging regist.

Vulnerabilities
• Failing to audit failed logons
• Failing to secure audit files
• Failing to use structured exception handling
• Revealing too much information to the client

Purpose of logging
• Identifying security incidents
• Monitoring policy violations
• Providing information about problems and unusual conditions
• Helping defend against vulnerability identification and exploitation through attack
detection

The application logs must record "when, where, who and what" for each event.

Which events to log
• Input validation failures e.g. protocol violations, unacceptable encodings, invalid
parameter names and values
• Output validation failures e.g. database record set mismatch, invalid data encoding
• Authentication successes and failures
• Authorization (access control) failures
• Session management failures e.g. cookie session identification value modification
• Application errors and system events e.g. syntax and runtime errors, connectivity
problems, third party service error messages, file upload virus detection
• Application and related systems start-ups and shut-downs, and logging initialization
(starting, stopping or pausing)
• Use of higher-risk functionality e.g. network connections, addition or deletion of
users, changes to privileges, assigning users to tokens, access to payment
cardholder data, use of data encrypting keys, submission of user-generated
content - especially file uploads.

Data to exclude
• Application source code
• Session identification values
• Access tokens
• Sensitive personal data and some forms of personally identifiable information (PII)
e.g. health, government identifiers, vulnerable people
• Authentication passwords
• Database connection strings
• Encryption keys and other master secrets
• Bank account or payment card holder data

Error Handling
Error messages displayed to the user should not contain system, diagnostic or
debug information.
```

Figura A.4: Excerto da secção sobre práticas seguras sobre *logging* e gestão de erros.

A.8 Excerto da Secção sobre Práticas Seguras em IoT (*Embbebed Systems*)

Este anexo contém um excerto do relatório gerado após utilização da ferramenta desenvolvida no contexto do projeto de dissertação (figura A.5). Neste caso, o excerto exibido diz respeito a Práticas Seguras em IoT.

```
IoT Security

This section appears because you choose the option "Embedded System" in the question
about the architecture of the system.

Internet of Things (IoT) devices fall into three main categories:
• Sensors, which gather data
• Actuators, which effect actions
• Gateways, which act as communication hubs and may also implement
some automation logic.

All these device types may stand alone or be embedded in a larger product. They may
also be complemented by a web application or mobile device app and cloud based service.
IoT devices, services and software, and the communication channels that connect them,
are at risk of attack by a variety of malicious parties.

Physical Security
• Any interface used for administrationor test purposes during development should be
removed from a production device, disabled or made physically
inaccessible.
• All test access points on production units must be disabled or locked, for example by
blowing on-chip fuses to disable JTAG.
• If a production device must have an administration port, ensure it has effective
access controls, e.g. strong credential management, restricted ports, secure protocols.
• Make the device circuitry physically inaccessible to tampering, e.g. epoxy chips to
circuit board, resin encapsulation, hiding data and address lines under
these components etc.
• Provide secure protective casing and mounting options for deployment of devices
in exposed locations.
• For high-security deployments, consider design measures such as active masking
or shielding to protect against side-channel attacks

Network Connections
• Activate only those network interfaces that are required (wired, wireless -
including Bluetooth etc.).
• Run only those services on the network that are required.
• Open up only those network ports that are required.
• Run a correctly configured software firewall on the device if possible.
• Always use secure protocols, e.g. HTTPS, SFTP.
• Never exchange credentials in clear text or over weak solutions such as HTTP
Basic Authentication.
• Authenticate every incoming connection to ensure it comes from a legitimate source.
• Authenticate the destination before sending sensitive data.
```

Figura A.5: Excerto da secção sobre práticas seguras sobre sistemas embutidos.

A.9 Exemplo de um Ficheiro com as Respostas e Instruções de como Elaborar este Ficheiro

Este anexo contém o exemplo do conteúdo de um ficheiro que contém as respostas dadas por um utilizador após usar a ferramenta (figura A.6). Contém também o conteúdo do ficheiro (em *markdown*) que descreve a forma como aquele ficheiro deve ser construído ou está formatado (figura A.7).

```
8;7;6;4;    # Embedded System ; API Service ; Server Component ; Mobile Application
1           # BD - Yes
1           # SQL
2           # MySQL
2;3;        # Confidential Data ; Critical Data
1           # No Authentication
2           # User Registration - No
N/A         # Type of Regist - N/A
1;2;6;      # C# ; C++ ; Python
2           # Input Forms - No
1           # File Upload - Yes
1           # Logs - Yes
1           # HW Specs - Yes
2           # HW Auth - No Authentication
1;7;        # HW Wireless Tech - 4G/LTE ; GPS ; Bluetooth
```

Figura A.6: Exemplo de um ficheiro gerado com as respostas.

```
# Instructions

* Each line is a question

* Single choice answer must be a digit that corresponds to the respective option
in the question.
    ** Example : 1

* Multiple choice questions must be answered on the respective line with the
delimitation character " ; " and END with it.
    ** Example : 1;2;3;

( If one answer is chosen in this type of question, the line must end it ";" )
    ** Example : 2;

( If multiple answers are chosen and one of the answer correspond to a user input text
answer (option which corresponds to "others"), the line is written has the same behavior
as only digits, with the user input text answer written in letters)
    ** Example : 1;2;user input text;5;

If the answer is only the user input text ( option "others") just write the answer
(user input text). (no need to end with ";" )
    ** Example : user input text

* It is possible to add comments to the answer. Just add the character '#' and
write the comment in front of it.
    ** Example : 1;4;   # web app , mobile

> Full example of a file with the answers written:
1;2;4;raspbery;
1
1
2
3; # critical data
7
1
2
another language
2
1
```

Figura A.7: Instruções de como elaborar um ficheiro com as respostas, se o utilizador assim o desejar.