Universidade Eduardo Mondlane

Faculdade de Engenharia

Curso de Engenharia Informática

Proposta de um modelo de ciclo de vida seguro de desenvolvimento de software.

Caso de estudo: CIUEM

Autor

Mondlane, Paulo Titos

Supervisor:

Eng. Délcio Chadreca

Co-supervisor:

Dr. Issufo Vali

Supervisor da Instituição:

Eng. Momade Marcos Henrique Abdul

Maputo, Março de 2023

Universidade Eduardo Mondlane

Faculdade de Engenharia

Curso de Engenharia Informática

Proposta de um modelo de ciclo de vida seguro de desenvolvimento de software.
Caso de estudo: CIUEM

Autor

Mondlane, Paulo Titos

Supervisor:

Eng. Délcio Chadreca

Co-supervisor:

Dr. Issufo Vali

Supervisor da Instituição:

Eng. Momade Marcos Henrique Abdul

Maputo, Março de 2023

UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

TERMO DE ENTREGA DE RELATÓRIO DO TRABALHO DE ESTÁGIO PROFISSIONAL

Declaro que o estudante Paulo Titos Mondlane entregou no dia ____/____/2023 as ____ cópias do relatório do seu Estágio Profissional com a referência: **2022EIEPD231** intitulado: Proposta de um modelo de ciclo de vida seguro de desenvolvimento de software. Caso de estudo: CIUEM.

Maputo, ____ de _____ de 2023

Chefe de Secretaria

_____

UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

DECLARAÇÃO DE HONRA

Declaro sob compromisso de honra que o presente trabalho é resultado da minha investigação e que foi concebido para ser submetido apenas para a obtenção do grau de Licenciatura em Engenharia Informática na Faculdade de Engenharia da Universidade Eduardo Mondlane.

Maputo, ____ de _____ de 2023

O autor

_____

(Paulo Titos Mondlane)

Dedicatórias

Dedico este trabalho aos meus pais: Titos Paulo Mondlane

e Hortência Moisés Machava que tanto se sacrificaram

para que eu chegasse aqui

Agradecimentos

Em primeiro lugar, agradecer ao Deus Altíssimo, que pela sua Omnipresença, Omnisciência e Omnipotência me guiou até que pudesse chegar a este momento da minha vida.

Em segundo lugar, agradecer imensamente aos meus supervisores: Eng. Délcio Chadreca, dr. Issufo Vali e Eng. Momade Abdul e a todos os intervenientes do DDSA do CIUEM, pelas orientações dadas ao longo da concepção deste trabalho. O meu muito obrigado.

Em terceiro lugar, endereçar os meus profundos agradecimentos aos meus pais: Titos Paulo Mondlane e Hortência Moisés Machava, que sempre deram tudo de si para que nunca me faltassem condições para que pudesse ter escolaridade.

Em quarto lugar, agradecer aos meus irmãos (Cláudia, Hamilton e Titos Júnior), à minha namorada (Síria Charmila), familiares e amigos que me apoiaram em todos os momentos de desânimo e fraqueza, estando sempre do meu lado.

Em quinto lugar, agradecer aos meus colegas da faculdade de engenharia (em particular aos que ao longo do tempo acabaram tendo uma aproximação especial para comigo), pelo suporte e pelas batalhas vencidas em conjunto.

Em último lugar, mas não menos importante, agradecer a todos que comigo partilharam do seu conhecimento ao longo da minha formação, em especial aos docentes da Faculdade de Engenharia da UEM e ao Director Executivo da Empire Cybersecurity Lda., Vanildo Pedro.

A todos o meu muito obrigado!

“A sabedoria torna o sábio mais poderoso

que uma cidade guardada por dez valentes.”

(Eclesiastes 7:19)

Resumo

Desde o seu surgimento, os *softwares* tornaram-se cada vez mais presente em nossas vidas, tornando-se um elemento essencial em diversas áreas, desde a comunicação até a automação de processos empresariais. Para o desenvolvimento destes *softwares*, foram criadas várias metodologias com vista a tornar este processo de desenvolvimento mais estruturado e flexível. O grande défice das metodologias tradicionais é o facto de se focarem mais na entrega de funcionalidades do produto final não priorizando, muitas vezes, o factor segurança.

O presente estudo teve como objectivo principal conceber um modelo de ciclo de vida de desenvolvimento de *software*, considerando aspectos de segurança no Centro de Informática da Universidade Eduardo Mondlane (CIUEM), ou seja, um modelo que introduza tarefas de segurança em todas as fases do ciclo de vida do *software*. Este estudo surge numa época em que os ataques cibernéticos têm ganhado espaço em todo o mundo, sendo que organizações são lesadas a todo o momento.

Para a materialização do objectivo retromencionado recorreu-se a várias técnicas, com destaque a pesquisas de material científico e a técnicas de observação directa. Uma vez que o autor esteve inserido no local de estudo, foi possível realizar tarefas práticas (por exemplo, a análise de vulnerabilidades) que permitiram ao autor ter uma visão clara do ambiente tecnológico do local de estudo, de modo a propor uma melhor solução para o problema identificado.

Depois de compilada toda a informação relevante, foi proposto um modelo de ciclo de vida seguro de desenvolvimento de *software* de acordo com a realidade da instituição, que promete não só incorporar segurança em todas as fases do ciclo de vida do *software* como também fazer dela uma cultura no seio da equipa de desenvolvimento.

Palavras-chave: Ciclo de Vida de Desenvolvimento de *Software*, Segurança de *Software*, Ataques cibernéticos, Vulnerabilidade.

Abstract

Since its genesis, software has become increasingly present in our lives, becoming an essential element in several areas, from communication to the automation of business processes. For the development of these software, several methodologies were created in order to make this development process more structured and flexible. The great deficit of traditional methodologies is the fact that they focus more on the delivery of features of the final product, often not prioritizing the security factor.

The main objective of this study was to design a software development life cycle model, considering security aspects at the Centro de Informática da Universidade Eduardo Mondlane (CIUEM), that is, a model that introduces security tasks in all stages of the software lifecycle. This study comes at a time when cyber-attacks are gaining ground around the world, with organizations being harmed all the time.

In order to achieve the aforementioned objective, various techniques were used, with emphasis on research of scientific material and direct observation techniques. Once the author was inserted in the place of study, it was possible to carry out practical tasks (for example, the analysis of vulnerabilities) that allowed the author to have a clear vision of the technological environment of the place of study, in order to propose a better solution for the identified problem.

After compiling all the relevant information, a secure software development lifecycle model was proposed in accordance with the institution's reality, which promises not only to incorporate security into all phases of the software lifecycle, but also to make it a culture within the development team.

Keywords: Software Development Lifecycle, Software Security, Cyber Attacks, Vulnerability.

Índice

Lista de Abreviaturas e Acrónimos

LAM	Linhas Aéreas de Moçambique
CIUEM	Centro de Informática da Universidade Eduardo Mondlane
UEM	Universidade Eduardo Mondlane
RAD	*Rapid Application Development*
CIA	*Confidentiality, Integrity, Availability*
NIST	*National Institute of Standards and Technology*
SSDLC	*Secure Software Development Life Cycle*
SDL	*Security Development Lifecycle*
SD3+C	*Secure by Design, Secure by Default, Secure in Deployment, and Communications*
OWASP	*Open Web Application Security Project*
SAMM	*Software Assurance Maturity Model*
TIC's	Tecnologias de Informação e Comunicação
DDSA	Departamento de Desenvolvimento de Sistemas e Aplicações
TLS	*Transport Layer Security*
SSL	*Secure Sockets Layer*
AAA	*Authentication, Authorization, Accounting*

Glossário de termos e expressões

***Internet*:** interconexão global de redes de computadores, que se comunicam utilizando protocolos, permitindo a troca de mensagens entre diferentes entidades de forma distribuída e descentralizada.

***Website*:** conjunto de páginas web hospedadas em um servidor, podendo ser acedido através de um navegador de internet.

***Software*:** conjunto de instruções projectadas para realizar determinada tarefa em um dispositivo digital.

***Deployment*:** processo que visa deixar um software pronto para ser disponibilizado ao público.

***Hardware*:** parte tangível dos dispositivos electrónicos.

***Cloud*:** metodologia que consiste no fornecimento de recursos computacionais através da *internet*, geralmente de forma paga.

***Fuzzing*:** técnica de teste de *software* que consiste no envio de dados aleatórios em campos de entrada de um *software*, com o objectivo de descobrir vulnerabilidades.

***Framework*:** qualquer abordagem estruturada ou directriz que visa oferecer uma estrutura e funcionalidade pré-construída que simplifica e acelera o processo de alcançar os resultados desejados.

***Build*:** processo de preparação de um conjunto de código fonte de *software* para ser executado.

***Trigger*:** evento ou condição que inicia um conjunto de acções em sistemas computacionais.

Log: registo de eventos ocorridos em um sistema computacional.

***Patch*:** actualização que visa resolver erros, vulnerabilidades de segurança ou outro tipo de inconformidades num *software*.

***Backup*:** processo que consiste na criação de uma cópia segura de dados a fim de protegê-los contra perda, dano ou destruição acidental ou intencional.

DevSecOps: metodologia de desenvolvimento de *software* que através da associação entre os desenvolvedores de *software*, a equipe de infraestructura e a equipe de segurança, integra a segurança em todo o ciclo de desenvolvimento do *software*.

***Crawler*:** agente de *software* que realiza varreduras na *internet* visando colectar informações.

Lista de figuras

Lista de tabelas

1. Capítulo I - Introdução

1.1. Contexto

As tecnologias de informação e comunicação surgiram para ajudar o ser humano a realizar suas actividades de forma mais flexível e optimizada. Com estas tecnologias é possível aceder a um conjunto de informações, independentemente da localização geográfica, necessitando apenas de um dispositivo com acesso à internet.

Moçambique ainda tem muito que fazer no que diz respeito à inclusão tecnológica da população:

> Em dezembro de 2019, o país contava com 6.523.613 usuários de internet. Cerca de 20,9% de uma população total de mais de 30 milhões. A maioria dos usuários no país acede à internet por meio de telefonia celular, como ocorre em outros países do Sul Global. (Cepik e Marcelino, 2021)

Apesar de Moçambique ainda apresentar muitos desafios no que concerne ao acesso à internet, esforços têm vindo a ser feitos no intuito de alterar este cenário. Actualmente, nas principais cidades, a grande maioria dos indivíduos a partir da adolescência tem acesso a um celular com ligação a *internet*. Tem se verificado também a proliferação de instituições de prestação de serviços de tecnologia, o que leva a crer que o país está em crescente avanço neste domínio.

A Universidade Eduardo Mondlane, sendo a mais antiga e prestigiada do país, possui um dos primeiros centros de informática do país. Neste centro são prestados vários serviços, entre eles alojamento de *sites*, montagem de infra-estrutura de redes, desenvolvimento de aplicações, entre outros.

Apesar de as tecnologias ajudarem imenso, elas vêm com o perigo de serem abusadas por criminosos para realizar qualquer tipo de actividade maligna. Durante o período da pandemia, escolas, universidades e organizações de todo o mundo foram severamente atingidas por ataques cibernéticos, comprometendo informação sensível causando, assim, danos financeiros avultados.

Em Moçambique, tem havido incremento de ataques cibernéticos nos últimos tempos, tendo se destacado há algum tempo o ataque à empresa LAM, e mais recentemente o ataque em

massa aos *websites* do governo. Há muito mais ataques que as empresas e instituições simplesmente não divulgam ao público por diversos motivos, sendo que um deles pode dever-se ao desejo de preservação da reputação das mesmas. Apesar da tentativa de ocultação de eventos relacionados a ataques cibernéticos, na maioria das instituições tem vazado informações destes eventos. Isso deixa claro que os perigos cibernéticos são um mal comum, independentemente da localização geográfica ou condição económica e ou social.

1.2. Formulação do problema

Os *softwares*, ao longo dos anos, passaram de ferramentas desconhecidas e complexas para ferramentas totalmente indispensáveis para a vida do ser humano, pois, através da interconexão destes, é possível realizar um conjunto de actividades de forma fácil e flexível. A despeito dos grandes benefícios que os *softwares* proporcionam, o que se observa é que, à medida que mais *softwares* vão sendo lançados, mais dispositivos se conectam e, consequentemente, mais vulnerabilidades são criadas e relatadas, o que ocasiona ataques cibernéticos.

Os ataques bem-sucedidos a sistemas têm tido várias causas, porém, falhas na concepção destes sistemas têm contribuído para a proliferação deste mal. Na maior parte das vezes, os desenvolvedores destas aplicações não têm noções de práticas de desenvolvimento seguro de *software*, deixando assim um conjunto de vulnerabilidades expostas aos criminosos digitais.

> Hoje, o sistema de ensino superior não forma especialistas que sejam bem versados no campo de desenvolvimento de software e no campo de sua segurança. Isso cria uma escassez de pessoal profissional nesta área. Muitas vezes, as próprias empresas de desenvolvimento de software precisam treinar programadores adicionalmente em questões de desenvolvimento seguro. A situação actual leva a custos adicionais de desenvolvimento e também dificulta a transmissão a todos os programadores de informações sobre as ameaças actuais e a importância da segurança. (Gorbatov e Meshcheriakov, 2017)

Em Moçambique, este défice na formação de especialistas em segurança de *software* também tem tido um impacto negativo, pois deixa *softwares* desenvolvidos por diferentes entidades em situação de vulnerabilidade.

O CIUEM possui e desenvolve diversos *softwares*, para uso interno na instituição, assim como para o uso público (externo). Estes *softwares* possuem diversas finalidades e utilizadores finais, sendo que no meio deles se encontram alguns de carácter muito crítico, cuja garantia de segurança é inegociável. O que se tem verificado nesta instituição (e em tantas outras nacionais e internacionais) é que se consideram alguns aspectos de segurança apenas no momento de *deploy* dos *softwares*, ignorando estes (aspectos) ao longo das outras fases do ciclo de desenvolvimento, o que do ponto de vista de boa prática de segurança não é recomendado. Falta, nesta instituição, procedimentos claros e consistentes visando a salvaguarda da segurança dos sistemas, o que se deve (entre outros factores) a pouco conhecimento da matéria por parte dos desenvolvedores destes sistemas.

Com base no que foi referido acima, surge a necessidade de definir um modelo claro de desenvolvimento de forma a maximizar a segurança dos *softwares* durante todo o seu ciclo de vida, visando assim diminuir o surgimento de vulnerabilidades que possam levar a um ataque bem-sucedido. Um ataque bem-sucedido a estes *softwares*, pode causar a violação de informação sensível dos utilizadores ou em último caso deitá-los abaixo, deixando pendentes várias actividades que dependam directa ou indirectamente destes.

A questão que surge é: Que modelo de ciclo de vida seguro de desenvolvimento de *software* melhor se adequaria à realidade do CIUEM?

1.3. Motivação

A informação é um elemento crucial para as organizações, pois é nela que reside o verdadeiro significado de todas as operações que se realizam. Não garantir o manuseamento e armazenamento seguro da informação é colocar em risco todos os intervenientes das operações da organização.

O CIUEM é uma das instituições mais antigas do país no ramo da tecnologia, produzindo actualmente *softwares* para a UEM e para instituições externas. Com a proliferação dos

ataques cibernéticos nos últimos tempos, é de extrema importância a preocupação com a segurança dos *softwares*, pois um ataque informático, em particular que culminasse com o roubo de dados mancharia e muito a imagem desta instituição, tanto a nível nacional como internacional.

A motivação para realizar este trabalho proveio da preocupação do autor em garantir a qualidade dos *softwares* produzidos no CIUEM, olhando para a vertente segurança dos mesmos, de modo que se garanta a confidencialidade, a integridade e a disponibilidade.

Além de se preocupar com a segurança dos sistemas do CIUEM, a motivação para realizar este trabalho prende-se ao facto de o autor se identificar com a área de estudo e pretender, por meio deste trabalho, aprimorar os seus conhecimentos.

1.4. Objectivos

1.4.1. Geral

- Conceber um modelo de desenvolvimento seguro de *software* que possa ser adequado ao processo de desenvolvimento de *softwares* do Centro de Informática da UEM.

1.4.2. Específicos

- Apresentar princípios e metodologias usados no desenvolvimento seguro de *softwares*;
- Descrever as metodologias de desenvolvimento de *software* aplicadas actualmente no CIUEM, apontando os seus constrangimentos;
- Propor um modelo de ciclo de vida seguro de desenvolvimento de *software*.

1.5. Metodologia

Segundo Lakatos e Marconi (2003), método (ou metodologia) “é o conjunto das actividades sistemáticas e racionais que, com maior segurança e economia, permite alcançar o objectivo”. Com este capítulo pretende-se, basicamente, descrever os principais mecanismos que serão adoptados para se realizar o trabalho.

1.5.1. Classificação da metodologia

No âmbito da realização de um trabalho de pesquisa, é necessário adoptar uma determinada metodologia, cujas classificações variam grandemente. Para a realização deste trabalho, foram adoptadas as seguintes metodologias, cada uma delas situada na sua respectiva classificação.

1.5.1.1. Quanto a abordagem

No que diz respeito à abordagem, o seguinte trabalho classifica-se como qualitativo.

Segundo Coelho (2019), a pesquisa qualitativa “considera que existe uma relação entre o mundo e o sujeito além daquela traduzida em números. Nessa abordagem, o objectivo central da pesquisa é entender a explicação de algum fenómeno.”

Este método, ao contrário do quantitativo, não se preocupa em trabalhar com modelos matemáticos sobre um conjunto de dados, mas sim em ir a fundo num determinado problema.

1.5.1.3. Quanto à natureza

No que diz respeito à natureza, o seguinte trabalho classifica-se como pesquisa aplicada.

Para Gil (2009), as pesquisas aplicadas “decorrem do desejo de conhecer com vista a fazer algo de maneira mais eficiente ou eficaz”. Neste tipo de pesquisa, o objectivo é aplicar conhecimentos (provenientes da pesquisa básica/pura) para resolver algum problema da vida prática.

1.5.1.4. Quanto aos objectivos

No que diz respeito aos objectivos, o seguinte trabalho classifica-se como exploratório.

Segundo Gil (2009), a pesquisa exploratória "tem como objectivo proporcionar maior familiaridade com o problema, com vista a torná-lo mais explícito ou a constituir hipóteses". Este tipo de metodologia é muito aplicado em situações em que o tema ainda não foi antes estudado em determinado ambiente.

1.5.1.5. Quanto aos procedimentos

No que diz respeito aos procedimentos, o seguinte trabalho classifica-se como:

- **Pesquisa bibliográfica:** Neste tipo de pesquisa, segundo Gil (2009) faz-se uso de material já produzido (ou elaborado), principalmente livros e artigos científicos.
- **Pesquisa documental:** Este tipo de pesquisa, segundo Gil (2009) "vale-se de materiais que não recebem ainda um tratamento analítico, ou que ainda podem ser reelaborados de acordo com os objectos da pesquisa". Neste grupo podem se achar relatórios de pesquisa, tabelas estatísticas, fotografias, gravações, regulamentos, entre outros.
- **Estudo de caso:** Este tipo de pesquisa, segundo Gil (2009), "consiste no estudo profundo e exaustivo de um ou poucos objectos, de maneira que permita seu amplo e detalhado conhecimento". Neste âmbito, o autor centrou-se no estudo de um local previamente seleccionado.
- **Levantamento:** Este tipo de pesquisa, segundo (Gil, 2009), consiste na busca de informações em forma de interrogação, nos indivíduos envolvidos no ambiente problemático de modo a se obter conclusões com os dados obtidos.

1.5.2. Técnica de colecta de dados

De modo a operacionalizar a presente pesquisa, fez-se uso das técnicas a seguir:

- **Pesquisa bibliográfica:** tendo-se recorrido a livros e artigos científicos (físicos e digitais), que abordam acerca de assuntos pertinentes ao trabalho em questão.
- **Pesquisa documental:** tendo-se recorrido a documentações, procedimentos de desenvolvimento, regulamentos, entre outras informações pertinentes fornecidas pela equipe de desenvolvimento do CIUEM.
- **Entrevistas:** tendo sido feitas entrevistas ao responsável do departamento de desenvolvimento de sistemas do CIUEM, de modo a ter informações cruciais para a realização do trabalho, não achadas na pesquisa documental.

1.6. Organização do Trabalho

- **Capítulo I - Introdução**

Neste capítulo são apresentados aspectos introdutórios da pesquisa, que servem de base para as próximas fases. Neste capítulo se destacam a contextualização, a descrição do problema, a motivação, os objectivos da pesquisa e a metodologia usada.

- **Capítulo II - Revisão de Literatura**

Neste capítulo são abordados aspectos teóricos fundamentais acerca da pesquisa em questão, obtidos de diferentes fontes. Com a revisão da literatura pretende-se cruzar os pensamentos de diferentes autores com vista a sustentar a pesquisa.

- **Capítulo III – Caso de estudo**

Neste capítulo é dada uma descrição detalhada do local de estudo, apresentando a sua estrutura, as suas metodologias de operação, entre outros aspectos pertinentes.

- **Capítulo IV – Proposta de solução**

Neste capítulo é dada uma descrição profunda da proposta de solução, com vista a resolver o problema previamente identificado no local de estudo.

- **Capítulo V – Conclusões e Recomendações**

Neste capítulo, são descritas as conclusões obtidas aquando da realização da pesquisa e são dadas recomendações para trabalhos futuros.

- **Capítulo VI – Bibliografia**

Neste capítulo, são listadas todas obras, que foram consultadas no âmbito da realização deste estudo.

2. Capítulo II - Revisão de literatura

2.1 Ciclo de vida de desenvolvimento de software

Os *softwares* podem ser entendidos como um conjunto de instruções concebidas para executar determinadas operações sobre computadores. Desenvolver um *software* é um processo complexo, devido aos vários componentes que dele fazem parte, tendo por isso sido elaborados padrões de desenvolvimento, também chamados de ciclos de vida de desenvolvimento de *software*.

O ciclo de vida de desenvolvimento de *software* pode ser entendido como uma "estrutura contendo processos, actividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de *software*, abrangendo a vida do sistema, desde a definição de seus requisitos até o término de seu uso." (De Macêdo e Spínola, s.d.). O principal objectivo destes padrões de desenvolvimento é criar um esquema bem definido com todas as actividades a executar durante o projecto, juntamente com mecanismos de verificação de progresso. Para *International Journal of Advance Research in Computer Science and Management Studies* (2013), "o emprego de ciclo de vida de desenvolvimento de *softwares* adequado permite que os gerentes de projecto regulem toda a estratégia de desenvolvimento do *software*". Ele torna o processo de desenvolvimento mais organizado, facilitando o trabalho de todas as partes intervenientes do processo de desenvolvimento.

2.1.1. Modelos

A escolha do modelo de ciclo de vida de desenvolvimento de *software* é uma das decisões mais importantes e precoces a serem feitas quando se pretende iniciar um projecto de criação de um produto de *software*, pois uma escolha errada deste modelo pode culminar num projecto falhado ou problemático. Não existe nenhum modelo padrão tampouco perfeito, por isso, De Macêdo e Spínola (s.d.) afirmam que alguns factores determinam a escolha de um modelo: "O perfil e complexidade do negócio do cliente, o tempo disponível, o custo, a equipa, o ambiente operacional são factores que influenciarão directamente na escolha do ciclo de vida de *software* a ser adoptado". É crucial que o responsável da equipa

de desenvolvimento avalie todos esses factores e escolha o modelo que melhor se adéque ao projecto de *software* em questão.

Um ponto importante que é necessário frisar é que os modelos de ciclo de vida não são estáticos, ou seja, eles podem ser moldados de acordo com a necessidade, de modo que possa haver melhor compatibilidade com a realidade da organização. Outro ponto importante é a possibilidade de uma organização utilizar mais de um modelo de ciclo de vida: “Da mesma forma, também é difícil uma empresa adoptar um único ciclo de vida. Na maior parte dos casos, vê-se a presença de mais de um ciclo de vida no processo” (De Macêdo e Spínola, s.d.). Dito isso, fica evidente que cada organização deve procurar adequar o(s) modelo(s) de ciclo de vida à sua realidade de modo a produzir *softwares* de boa qualidade.

Existem vários modelos de ciclo de vida de desenvolvimento de *software*, desde os mais tradicionais até aos mais modernos. A maioria destes modelos comportam as seguintes fases:

- **Requisitos:** O levantamento e análise de requisitos é geralmente a primeira etapa quando se pretende desenvolver um produto de *software*. “Na primeira etapa é feito o levantamento de requisitos com o cliente, para entender suas expectativas e definir quais funcionalidades devem ser implementadas no sistema” (*Modelo Cascata: O Que é e Por Que Está Ultrapassado?* | *Trybe*, s.d.). É onde se procura identificar as necessidades reais da organização.
- **Design:** Esta fase, também chamada de projecto, é onde se faz a **“**tradução dos requisitos do *software* para um conjunto de representações que podem ser avaliadas quanto à qualidade, antes que a codificação se inicie**”** (Nakagawa, 2016). É onde se realiza a especificação detalhada de aspectos do *software* e *hardware* como a base de dados, interfaces, estrutura de dados, entre outros. Também se realiza a organização da equipa de desenvolvimento indicando a tarefa de cada um e procura ter-se uma estimativa da duração do projecto.
- **Codificação:** Esta é a fase derradeira do projecto. É onde se codifica o *software* com base nas entradas das fases anteriores.

- **Teste:** Esta fase consiste em descobrir se o *software* codificado vai de acordo com os resultados esperados. Segundo Nakagawa (2016)**,** esta fase concentra-se em dois aspectos, nomeadamente: nos aspectos lógicos internos do *software* (com vista a garantir que todas as instruções tenham sido testadas) e nos aspectos funcionais externos (com vista a identificar erros e garantir que a entrada definida produza resultados esperados).
- **Implementação e Manutenção** Esta fase consiste em colocar o *software* em ambiente de produção, para que seja utilizado pelo usuário final. "Caso seja necessária alguma mudança, o *software* deve passar por uma manutenção" (*Modelo Cascata: O Que é e Por Que Está Ultrapassado?* | *Trybe*, s.d.). Nesta fase procura-se garantir que o *software* funcione sem nenhum tipo de erros nem constrangimentos.

Para a pesquisa em questão, será destacada apenas uma pequena parte deste universo de modelos, nomeadamente: Cascata, Incremental, RAD e Prototipagem.

2.1.1.1 Cascata

O modelo em cascata é um dos mais antigos, proposto por Winston Walker Royce no ano de 1970. Segundo De Macêdo e Spínola (s.d.), "o nome cascata foi atribuído em razão da sequência das fases, onde cada fase só começa quando a anterior termina; e da transmissão do resultado da fase anterior como entrada para a fase actual". A ideia deste modelo é executar todas as actividades de uma fase profundamente e só depois de terminadas, passar para a próxima fase do modelo.

Embora existam algumas variações na nomenclatura e organização, é possível visualizar a seguir um esquema ilustrativo do modelo em cascata:

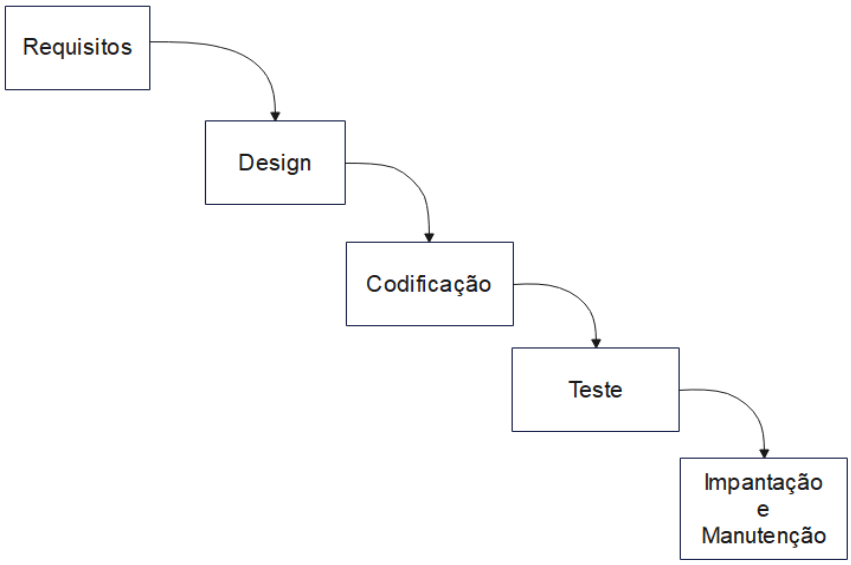

Figura 1: Modelo em cascata – Fonte: Autor

O modelo em cascata deve ser executado em tipos específicos de projectos. Segundo Bhuvaneswari e Prabaharan (2013), “sua intensa documentação e planejamento fazem com que funcione bem para projectos nos quais o controle de qualidade é uma grande preocupação”. Devido ao trabalho árduo que é feito nas fases iniciais deste modelo, há mais possibilidades de se eliminar erros, portanto, é um modelo apropriado para projectos altamente exigentes no que concerne a qualidade de resultados.

2.1.1.2 Incremental

O modelo incremental é um modelo que surgiu como uma melhoria do modelo em cascata. Segundo De Macêdo e Spínola (s.d.), “neste modelo, de Mills em 1980, os requisitos do cliente são obtidos, e, de acordo com a funcionalidade, são agrupados em módulos”. Uma vez que os desenvolvedores repartem os requisitos em módulos, o progresso do desenvolvimento do software vai incrementando à medida que um novo módulo é finalizado e se parte para outro.

Para Bhuvaneswari e Prabaharan (2013), a ideia básica deste método é desenvolver um sistema por meio de ciclos repetidos e em porções menores de cada vez, permitindo o reaproveitamento por parte dos desenvolvedores, do aprendizado obtido durante o desenvolvimento de partes/versões anteriores do sistema.

Uma especial atenção deve ser dada à integração dos módulos de modo a evitar erros, o que pode se conseguir com um bom agrupamento de requisitos, de modo que as funções comuns de todo o sistema sejam entregues ainda no primeiro incremento.

A seguir pode-se visualizar um esquema ilustrativo do modelo incremental:

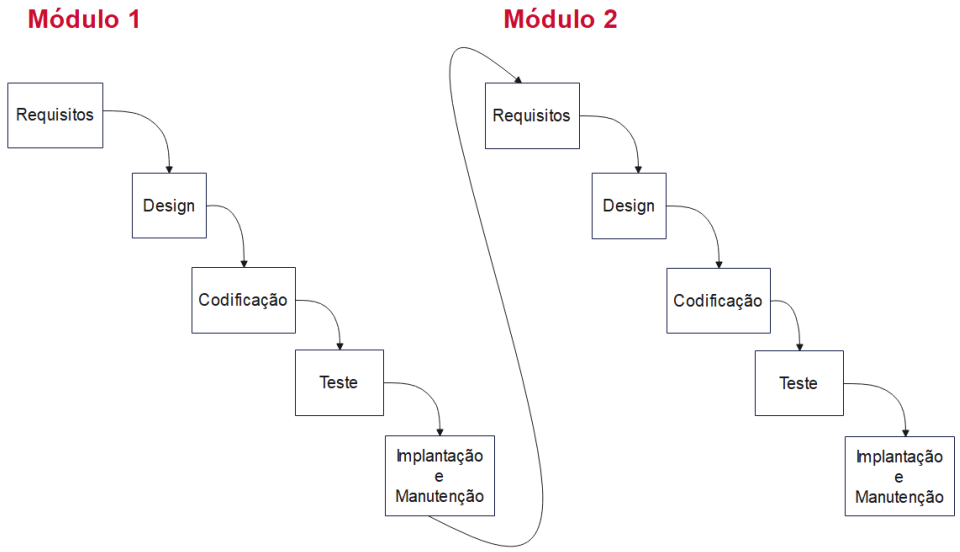

Figura 2: Figura 2: Modelo incremental – Fonte: Autor

O modelo incremental pode ser aplicado em casos em que o cliente precisa urgentemente do *software* em ambiente de produção. Através do desenvolvimento incremental, o cliente pode usar o *software* de forma gradual, ajudando também a equipe de desenvolvimento a perceber o nível de satisfação a cada entrega feita, permitido ao cliente a requisição de uma mudança se esta for necessária. Outro caso em que pode se usar este modelo é quando se possui uma equipe de desenvolvimento pequena e não há muito rigor no que concerne a documentação.

2.1.1.3 RAD

RAD, do inglês *Rapid Application Development,* é um modelo criado em 1991 pelo britânico James Martin, que visa essencialmente flexibilizar o desenvolvimento de *softwares*. Segundo De Macêdo e Spínola (s.d.), o modelo RAD "é um ciclo de vida incremental, iterativo, onde é preferível que os requisitos tenham escopo restrito". Neste modelo, os requisitos são divididos em módulos, sendo que cada módulo é produzido por uma equipe diferente em paralelo, porém, para isso acontecer os módulos devem ser bastante independentes.

Para Nakagawa (2016), para se usar este modelo "os requisitos devem ser bem entendidos e o alcance do projecto restrito". De modo a garantir esta boa percepção dos requisitos, são utilizados métodos mais dinâmicos e flexíveis para a recolha de requisitos, como workshops em detrimento de métodos mais formais como reuniões/entrevistas e também são utilizadas técnicas de prototipagem.

Segundo De Macêdo e Spínola (s.d.), para uma organização aplicar este modelo deve possuir antes alguns requisitos a nível de recursos humanos, de gestão e de ferramentas:

- **Recursos humanos:** a organização deve possuir uma boa quantidade de profissionais, e estes por sua vez devem ter boas capacidades de trabalho em equipe. É crucial também que estes profissionais tenham uma boa experiência de trabalho e tenham boas capacidades de adaptação.
- **Gestão:** a organização deve ser pragmática, sem muitas burocracias.
- **Ferramentas:** a organização deve estar munida de ferramentas de desenvolvimento estáveis (para evitar perda de tempo com resolução de erros de ferramentas), ferramentas reutilizáveis entre outros aspectos que não embaracem o rápido desenvolvimento.

A seguir é possível visualizar um esquema ilustrativo do modelo RAD:

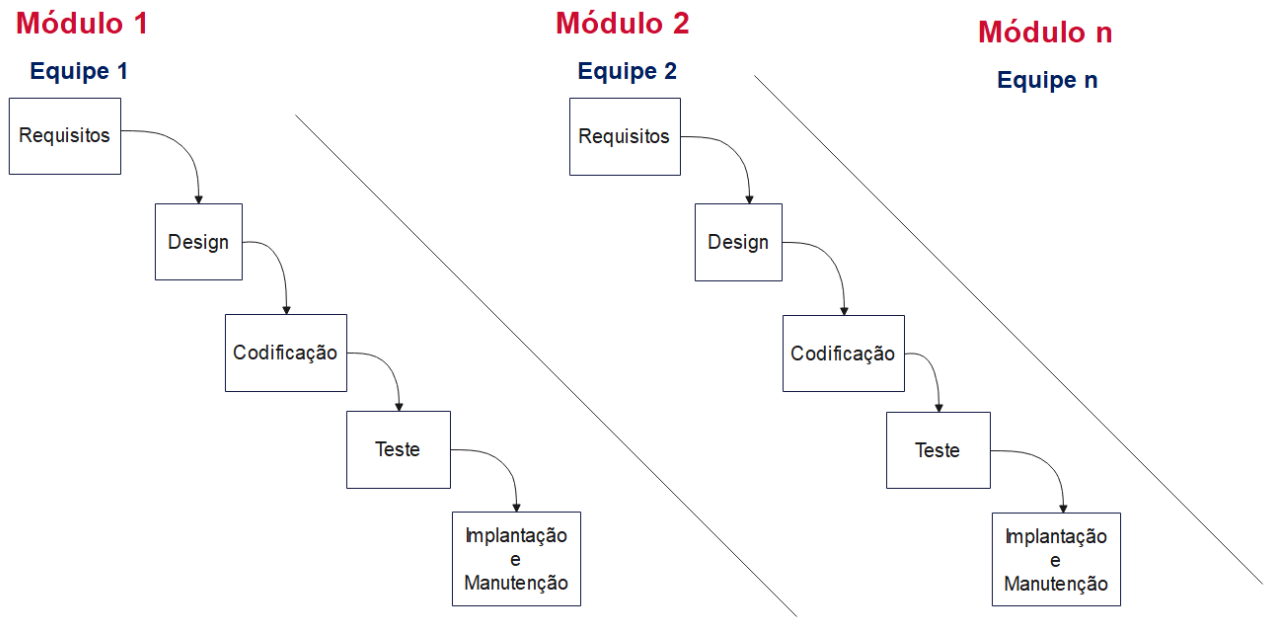

Figura 3: Modelo RAD – Fonte: Autor

2.1.1.4 Prototipagem

Este modelo é um dos mais interactivos, que pode ser usado tanto como modelo de ciclo de vida ou como ferramenta para outros modelos.

Antes de abordar sobre o modelo em si, é importante esclarecer o termo protótipo. Segundo *Significado de Protótipo (O Que é, Conceito e Definição) - Significados* (s.d.), “Protótipo é o termo usado para se referir ao que foi criado pela primeira vez, servindo de modelo ou molde para futuras produções”. Portanto, protótipo é um “rascunho” para a visualização rápida de um futuro produto. No contexto da engenharia de *software*, o protótipo pode ser um desenho de interface de um *software*, geralmente com algumas funcionalidades.

Neste modelo, é desenvolvido um pequeno protótipo com base em algumas informações dadas pelo cliente, sendo que, posteriormente o cliente avalia o protótipo e avalia se corresponde ao seu desejo. Correspondendo ao seu desejo, a equipe de desenvolvimento avança para o desenvolvimento do sistema real, caso contrário, faz-se modificações no protótipo e o cliente avalia novamente.

Segundo De Macêdo e Spínola (s.d.), “no ciclo de vida de prototipagem, não é exigido um conhecimento aprofundado dos requisitos num primeiro momento. Isso é bastante útil

quando os requisitos não são totalmente conhecidos, são muitos complexos ou confusos". A prototipagem serve exactamente para que o cliente possa experimentar na prática aquilo que seria o seu *software* pronto. A partir disso, ele informa à equipe de desenvolvimento sobre aspectos mal interpretados (aprofundando-os) e até sobre aspectos previamente não abordados, deixando assim os requisitos mais claros.

Um ponto muito importante a destacar é a relação entre os protótipos e a versão final do *software*. Segundo De Macêdo e Spínola (s.d.), é necessário que se informe ao cliente que a versão que se está a demonstrar é apenas de teste para evitar frustrações e também que se defina bem que aspectos do protótipo serão usados no sistema final para que se trabalhe segundo as boas práticas de engenharia.

A seguir é possível visualizar um esquema ilustrativo do modelo de prototipagem:

Figura 4: Modelo de Prototipagem – Fonte: *MICHELEVM (2013)*

O modelo de prototipagem é indicado para situações em que o cliente "não sabe muito bem o que quer". Através da visualização rápida de interfaces o cliente pode situar-se quanto às suas vontades. Este modelo não é recomendado para casos em que se deseja uma documentação rigorosa do *software*, pois devido ao desconhecimento inicial de todos requisitos e devido às constantes mudanças, a documentação torna-se difícil de manter.

2.1.2. Comparação entre modelos

Enquanto se abordava sobre alguns modelos de ciclo de vida de desenvolvimento de software, foi visto que cada um deles pode ser utilizado em diferentes ambientes devido às suas especificidades de funcionamento. A seguir é possível visualizar uma tabela com uma comparação resumida sobre os 4 modelos abordados:

Tabela 1: Comparação entre modelos de Ciclo de Vida de Desenvolvimento de Software – Fonte: Autor

Funcionalidade	**Modelo de Ciclo de Vida de Desenvolvimento de Software**			
	Cascata	**Incremental**	**RAD**	**Prototipagem**
Documentação	Alta	Intermédia	Intermédia	Baixa
Flexibilidade a mudanças	Difícil	Fácil	Fácil	Fácil
Complexidade	Simples	Intermédia	Muito simples	Simples
Envolvimento do cliente	No início	Intermédia	No início	Alto
Conhecimento técnico	Alto	Alto	Intermédio	Intermédio
Duração	Longa	Muito longa	Curta	Intermédia
Custo	Baixo	Baixo	Baixo	Baixo

Como já foi referido antes, a concepção de modelos de ciclo de vida permitiu que o desenvolvimento de *softwares* passasse a ser um processo organizado, formal, elevando assim a possibilidade de se ter um resultado com um alto nível de qualidade. Olhando para os modelos de ciclo de vida, é possível ter-se uma visão de todas as actividades a realizar e de todos os intervenientes do projecto.

O grande constrangimento destes modelos de ciclo de vida é o facto de eles darem pouca atenção ao factor segurança. Estes modelos se preocupam mais em garantir que o *software* seja produzido dentro do escopo de tempo definido e que todas funcionalidades estejam

operacionais (seguindo todos os passos pré-definidos), porém sem priorizar muito actividades de modo a elevar a segurança.

O desafio que surge é o de incorporar actividades de segurança desde a fase inicial de recolha de requisitos até a fase de manutenção destes *softwares*, de modo a diminuir potenciais vulnerabilidades a ser aproveitadas por indivíduos mal-intencionados. Antes de ilustrar essa fusão entre a segurança e os modelos de ciclo de vida, é necessário entender o que é a segurança de informação (pois os softwares lidam com informação) de modo que estes conceitos fiquem mais claros.

2.2. Segurança da Informação

Para iniciar o estudo é crucial que o termo segurança esteja bem claro. Segundo Rao e Nayak (2014) “segurança, em termos simples, é proteger o que você ou outros têm.”. Esta ideia deriva do facto de todos animais, e em particular o ser humano terem o instinto básico de garantir a própria segurança. *Introduction to Security Cyberspace, Cybercrime and Cybersecurity* (s.d.) não há uma definição clara de segurança, pois ela é um processo e não um estado final, ou seja, “segurança é o processo de manter um nível aceitável de risco percebido”: Rao e Nayak (2014) afirmam que “mesmo com a máxima sinceridade e tremendos esforços, não é possível ter uma segurança de informação 100% infalível, porque embora possa haver muitos problemas conhecidos, também pode haver um número igual de ocultos.”. A abordagem anterior remete a ideia de que um sistema não pode de facto ser considerado como seguro, mas que este estado de segurança é um processo contínuo de redução de riscos, até que se atinja um nível considerado aceitável. A busca desse nível aceitável é feita através da implementação de políticas, directrizes, procedimentos, governança e outras funções de software. Outro termo importante de compreender é o de informação. Segundo Lemos II (2011), “Informação é todo o conjunto de dados devidamente ordenados e organizados de forma a terem significado”. Como referido anteriormente, um distintivo importante da informação é o facto de ela possuir sentido e não ser apenas um conjunto de dados brutos. Conclui-se então que a informação representa um determinado significado, possuindo desse jeito valor.

Dito tudo isso, nota-se que já existem ferramentas suficientes para conceituar o termo segurança de informação. A segurança de informação pode ser entendida, desse jeito, como um conjunto de práticas ou processos que visam reduzir ao máximo o risco actuante sobre a informação. Para Rao e Nayak (2014) “a segurança da informação destina-se a proteger a informação e os sistemas de informação de utilizadores não autorizados aceder, usar, modificar ou destruir as informações.”. Nota-se então que o objectivo final da segurança da informação é garantir que a informação esteja segura, protegendo-a de perigos potenciais durante o seu processamento e armazenamento.

2.2.1. O trio CIA

O objectivo final da segurança de informação é manter a informação e os sistemas que a processam o mais segura possível, reduzindo os riscos ao máximo. Ao longo do tempo, foram desenvolvidos alguns modelos com o objectivo de especificar as propriedades ou características importantes dos activos de informação, isto para servir como base na criação de sistemas seguros e políticas. Estes modelos servem como uma espécie de requisitos que devem ser cumpridos para que a informação seja considerada num estado minimamente seguro.

Para o presente estudo, foi considerado o trio CIA, pois é o modelo mais utilizado nos dias actuais. Para Rao e Nayak (2014), “a importância do trio CIA aumentou nos últimos anos devido à forma como inserimos, transferimos ou armazenamos as informações.”. Ainda segundo Rao e Nayak (2014), é necessário que todas as três propriedades ou aspectos sejam assegurados, pois, enfatizar apenas um em detrimento de outros pode levar à redução da eficiência e eficácia de qualquer organização.

Este modelo é criação do *National Institute of Standards and Technology* (NIST)/ *U.S.Code* e preconiza que no âmbito da segurança da informação deve-se garantir 3 requisitos: a Confidencialidade, Integridade e Disponibilidade.

A seguir é possível visualizar uma ilustração do trio CIA:

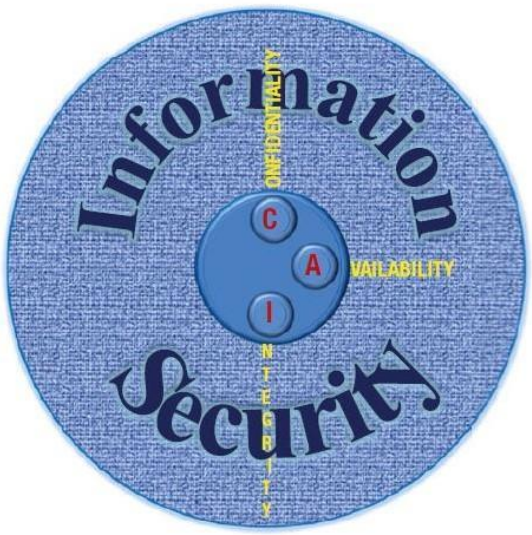

Figura 5: O trio CIA – Fonte: Rao e Nayak (2014)

2.2.1.1. Confidencialidade (*Confidentiality*)

A confidencialidade é um dos aspectos mais importantes na segurança da informação, porém é necessário entender o seu significado. Segundo *Introduction to Security Cyberspace, Cybercrime and Cybersecurity* (s.d.), confidencialidade é a "garantia de que as informações são compartilhadas apenas entre pessoas ou organizações autorizadas.", isto é, deve-se garantir que nenhuma terceira entidade não autorizada tenha acesso a determinada informação. Já Rao e Nayak (2014) consideram que confidencialidade é "preservar as restrições autorizadas de acesso e divulgação de informações, incluindo meios para proteger a privacidade pessoal e informações proprietárias". Esta segunda definição é mais explanatória, incluindo a componente da privacidade pessoal, que é altamente comprometida com a falta da confidencialidade.

Um exemplo do impacto da falha na garantia da confidencialidade seria no caso de uma terceira entidade ter acesso a informações sensíveis de uma nação inteira e vendê-la a uma outra, criando assim uma situação de risco nacional.

2.2.1.2. Integridade (*Integrity*)

A integridade é o segundo aspecto a ter conta na segurança da informação. Segundo *Introduction to Security Cyberspace, Cybercrime and Cybersecurity* (s.d.), ela é uma "garantia de que as informações são autênticas e completas". De facto, a informação pode ser alterada por um intruso durante a comunicação entre entidades legítimas ou mesmo quando ela se encontra armazenada num determinado local. É de crucial importância garantir que a informação nunca seja alterada por uma entidade não autorizada, de forma que a informação seja sempre íntegra. Para Rao e Nayak (2014), a integridade é a "protecção contra a modificação ou destruição imprópria de informações e inclui a garantia de não repúdio e autenticidade das informações". Da definição de Rao e Nayak (2014) surgiram 2 conceitos que podem ser conceituados a seguir:

- **Não repúdio:** segundo Kostopoulos (2017), o não repúdio é quando "os dados transmitidos ou armazenados são de autenticidade indiscutível, especialmente quando suportados por certificados digitais aceitáveis, assinaturas digitais ou outros identificadores explícitos". Dito de outra forma, o não repúdio proporciona mecanismos para se provar que um indivíduo realizou uma determinada acção e num determinado momento. Esta propriedade é muito importante, pois garante o controle dos autores de actividades realizadas no sistema, permitindo desse jeito a identificação de actividades externas não autorizadas.
- **Autenticidade:** segundo Whitman e Mattord (2011), "autenticidade da informação é a qualidade ou estado de ser genuíno ou original, ao invés de uma reprodução ou fabricação". Em sistemas de informação, é crucial que as informações trocadas entre remetente e receptor sejam originais, pois se isso não for garantido, indivíduos podem se intrometer no sistema manipulando-o de variadas maneiras.

2.2.1.3. Disponibilidade (*Availability*)

O terceiro aspecto a se levar em conta na segurança da informação é a disponibilidade. Segundo Whitman e Mattord (2011) "a disponibilidade permite que utilizadores autorizados

– pessoas ou sistemas de computador – acedam a informações sem interferência ou obstrução e as recebam no formato necessário". Nos dias actuais, as informações são armazenadas em vários dispositivos, como bancos de dados, dispositivos de armazenamento e mais recentemente na nuvem (*cloud*). O princípio da disponibilidade visa garantir que o usuário possa ter acesso à sua informação a qualquer altura em que ele a solicita, independentemente da localização da mesma informação.

2.2.2. Camadas

Para Rao e Nayak (2014), a segurança da informação é constituída por 4 grandes camadas, nomeadamente: camada de segurança pessoal, camada de segurança de rede, camada de segurança física e camada de segurança de software.

A seguir é possível visualizar uma ilustração da distribuição das camadas:

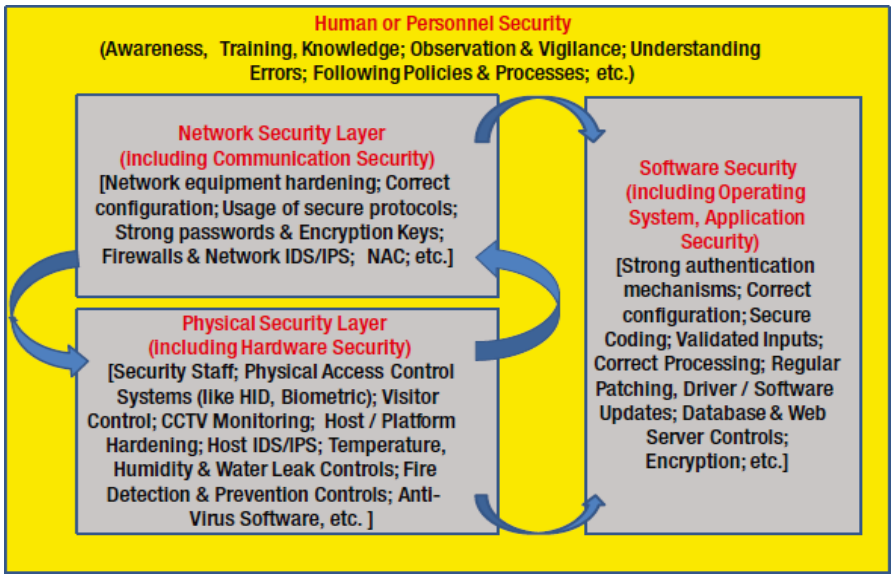

Figura 6: Camadas da Segurança de Informação – Fonte: Rao e Nayak (2014)

2.2.2.1. Camada de segurança pessoal

Esta camada, apesar de ainda não ser levada muito à sério, desempenha um papel muito importante na segurança de informação, pois muitas vulnerabilidades partem dela. O motivo pelo qual muitas vulnerabilidades partem das pessoas é a grande susceptibilidade que elas

têm de cometer erros, muitas vezes causados por negligência ou mesmo falta de conhecimento sobre potenciais riscos de algumas acções. Para Whitman e Mattord (2011), "... a menos que políticas, educação e treinamento, conscientização e tecnologia sejam empregues adequadamente para evitar que as pessoas danifiquem ou percam informações acidentalmente ou intencionalmente, elas continuarão sendo o elo mais fraco". É urgente que as organizações comecem a prestar mais atenção nesta camada, pois muitos prejuízos podem ser evitados através da simples conscientização dos utilizadores dos sistemas de informação.

2.2.2.2. Camada de segurança física

A segurança física trata os equipamentos de hardware como um activo físico e se esforça em proteger estes activos contra qualquer tipo de dano ou usurpação. Esta camada se preocupa com a aplicação de um conjunto de medidas para o controle de acesso aos equipamentos de hardware, pois são eles que armazenam e processam a informação. Para Whitman e Mattord (2011), "proteger a localização física dos computadores e os próprios computadores é importante porque uma violação da segurança física pode resultar na perda de informações". As medidas aplicadas para proteger os equipamentos físicos são variadas, podendo partir de algo simples como a montagem de fechaduras nas portas, a algo mais complexo como permitir acesso com base numa leitura de impressão digital ou mesmo da íris, contudo, todas medidas podem cooperar com o intuito de restringir acessos.

2.2.2.3. Camada de segurança de rede

O mundo actual está todo conectado através de redes de computador. Uma vez que existe todo esse emaranhado de comunicação entre dispositivos diferentes, situados ou não no mesmo local, mostra-se necessário proteger a forma como são realizadas as comunicações, isto para impedir que durante esta comunicação não se abra uma porta para uma actividade não desejada de terceiros. A camada de segurança de rede está intimamente relacionada com a camada de segurança física, isto porque o equipamento utilizado nas redes de computador é físico, porém apenas adoptar medidas da camada de

segurança física não se mostra suficiente:

> Ainda é importante aplicar as ferramentas tradicionais de segurança física, como fechaduras e chaves, para restringir o acesso e a interacção com os componentes de hardware de um sistema de informação; mas quando os sistemas de computador estão em rede, essa abordagem não é mais suficiente. (Whitman e Mattord, 2011)

O grande diferencial desta camada com a camada física situa-se no facto desta camada não se preocupar apenas com ameaças de forma física, mas também com aquelas provenientes de diferentes locais do mundo, através da comunicação entre redes de computadores. Medidas de segurança de redes podem incluir a filtragem de conteúdo, métodos de criptografia, entre outros.

2.2.2.4. Segurança de *software*

A quarta e última camada é a de segurança de *software*. Esta é muitas vezes considerada a camada mais difícil de proteger, devido à facilidade em se cometer erros nela. A segurança de *software*, lida directamente com os seguintes aspectos:

- Segurança do sistema operacional;
- Segurança de aplicativos;
- Segurança de ferramentas de *software* (incluindo a segurança de ferramentas usadas para proporcionar a segurança de informações).

2.2.3. Termos importantes da Segurança de Informação

Quando se fala de segurança de informação, é inevitável falar de alguns termos usados no dia-a-dia para quem trabalha nesta área. São termos que giram em torno de todas as actividades realizadas por estes profissionais da segurança de informação.

Tabela 2: Termos importantes da Segurança de Informação – Fonte: Autor

Termo	Descrição
Activo	Tudo o que é possuído por uma entidade e que possui valor.
Vulnerabilidade	Fraqueza em um sistema, passível de exploração.
Violação	Qualquer evento em que um indivíduo não autorizado tem acesso a informação confidencial.
Autenticação	Processo de verificação da identidade de uma entidade que solicita um recurso, com vista a confirmar se o que esta afirma ser, realmente é.
Autorização	Processo que visa conceder ou negar acesso a um recurso reivindicado por uma entidade, com base nas permissões atribuídas a esta.
Privacidade	Direito que um indivíduo tem sobre a informação, que lhe permite controlar a distribuição da mesma.
Risco	Um possível evento com um impacto negativo para determinada entidade.
Ameaça	Circunstância ou evento com capacidade de causar um impacto negativo numa entidade.
Auditoria	Registo de actividades de utilizadores e sistemas, a fim de monitoria, revisão e conformidade.

Com o termo “Segurança de informação” já devidamente explicado, parte-se agora para o derradeiro estudo da segurança em ciclos de vida de desenvolvimento de *software*.

2.3. Ciclo de vida seguro de desenvolvimento de *software*

Como já foi abordado no presente estudo, os modelos tradicionais de ciclo de vida desenvolvimento de *software*, apesar de definir todos processos, envolvidos e entre outros aspectos do desenvolvimento do produto de *software*, eles pecam no que concerne ao domínio segurança. Estes ciclos tradicionais, muitas vezes se preocupam apenas com a segurança na fase de *deployment* do produto de *software* abrindo espaço para vários perigos de segurança.

Os modelos de Ciclo de vida seguro de desenvolvimento de *software* (em inglês “*Secure Software Development Life Cycle* – SSDLC”) são modelos de desenvolvimento de *software* melhorados, ou seja, eles consideram aspectos de segurança durante todo o ciclo de desenvolvimento de *software*.

> A aplicação das tarefas de segurança no ciclo de vida do desenvolvimento torna-se vital e necessária para esclarecer vários problemas. Os elevados custos da remediação sempre que as vulnerabilidades forem identificadas após a implementação do software tornam-se o maior problema para a organização. Como consequências, estarão relacionados com uma falha e depois darão efeito a uma organização. Por conseguinte, a organização precisa de assegurar os controlos de segurança adequados com tarefas de segurança ao longo de todo o ciclo de vida do desenvolvimento. (Malaysia, 2020)

É um facto que as organizações actuais estão implementando cada vez mais a tecnologia para agilizar os seus processos, de forma a garantir ganhos financeiros maiores. Neste âmbito, é crucial que a equipe de desenvolvimento assegure-se que os *softwares* entregues tenham condições ideais de segurança, para evitar situações de vulnerabilidades que podem comprometer a operação normal das organizações, ou em pior caso, permitir o vazamento de informações altamente confidenciais.

Com o Ciclo de Vida de Desenvolvimento de *Software* Seguro procura-se adicionar actividades, ferramentas e acções concretas de segurança aos processos tradicionais de desenvolvimento de *software*, para que existindo alguma falha de segurança, seja resolvida

precocemente, reduzindo/eliminando o impacto no usuário final. Falando em prejuízos financeiros, o Ciclo de vida de desenvolvimento de *software* seguro surge como uma “porta de alívio”, pois quanto mais tarde se descobre uma falha, maior dispendiosa tende a ser a sua correcção.

Organizações que já possuem um Ciclo de Vida de Desenvolvimento de *Software*, podem, com base em alguns padrões (como por exemplo, o NIST SP 800-218), incorporar práticas de segurança ao seu ciclo já existente. Para aquelas que ainda não adoptam um modelo específico de desenvolvimento, podem já iniciar com um modelo de Ciclo de Vida Seguro de Desenvolvimento de Software já pré-fabricado por algumas entidades da área.

A seguir, se dará uma breve descrição de 2 modelos de Ciclo de Vida Seguro de Desenvolvimento de *Software*, muito usados no passado e nos dias actuais.

2.3.1. Microsoft SDL

O Microsoft SDL, do inglês *Security Development Lifecycle* (proposto inicialmente no ano de 2004 e com a última versão lançada em 2012), é um processo que visa garantir a segurança de *software* através da inserção de várias actividades de segurança ao longo do desenvolvimento do *software*.

Segundo Microsoft (2012), o desenvolvimento seguro de *software* possui basicamente 3 elementos, nomeadamente: melhores práticas, melhorias de processos e métricas, porém, neste caso, focar-se-á nos dois primeiros elementos. A ideia de base é reduzir as vulnerabilidades dos sistemas ao longo de cada fase: “O objectivo é minimizar as vulnerabilidades relacionadas com a segurança na concepção, código e documentação e detectar e eliminar as vulnerabilidades o mais cedo possível no ciclo de vida do desenvolvimento” (Microsoft, 2012). Como já foi dito antes, quanto mais cedo se identifica uma vulnerabilidade, mais fácil e barato é a sua resolução, em relação ao caso contrário.

Para ajudar a determinar onde são necessários esforços de segurança e privacidade, a Microsoft definiu um princípio chamado de SD3+C (*Secure by Design, Secure by Default, Secure in Deployment, and Communications).* Este princípio ajuda a equipe de

desenvolvimento a definir e priorizar a aplicação de segurança em sectores específicos, com vista a capitalizar recursos.

Compilando o SD3+C com as fases tradicionais de um ciclo de desenvolvimento de *software*, a Microsoft propõe o Ciclo de Vida de Desenvolvimento de *Software* Seguro a seguir:

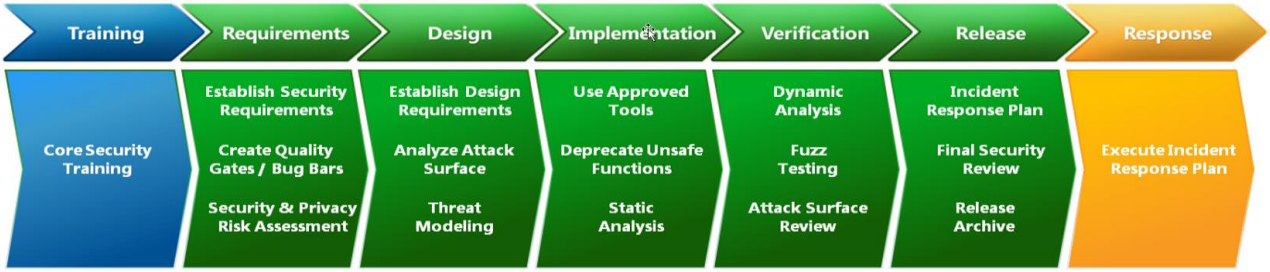

Figura 7: Microsoft SDL – Fonte: Microsoft (2012)

2.3.1.1. Treino *(Training)*

Esta fase, considerada a fase pré-requisitos, é onde é dado treinamento à equipe de desenvolvimento acerca de conceitos importantes de segurança. Segundo Microsoft (2012), "A formação em segurança pode ajudar a garantir que o *software* é criado tendo em mente a segurança e a privacidade e pode também ajudar as equipas de desenvolvimento a manterem-se actualizadas sobre questões de segurança". Basicamente, esta é uma fase de chamada à consciência sobre aspectos de segurança.

Esta fase não necessita de ser executada toda vez que vai se iniciar um projecto de *software*, no entanto, a Microsoft recomenda que cada um dos intervenientes da equipe de desenvolvimento participe em pelo menos um treino de segurança por ano. Estes treinos de segurança podem ser dados em forma de reuniões, workshops, cursos, entre outras formas.

Nesta fase, procura-se abordar questões como: codificação segura, modelagem de ameaças, testes de *software*, entre outros.

2.3.1.2. Requisitos (*Requirements*)

Nesta fase procura-se basicamente identificar as necessidades reais de segurança do projecto do *software* a ser criado. Identificadas as necessidades, procura-se balançar os custos de implementação de segurança com os seus potenciais benefícios, de modo que se perceba a viabilidade dos esforços a empreender.

Esta fase possui 3 actividades:

- **Requisitos de segurança:** nesta actividade procura-se especificar os requisitos mínimos de segurança e privacidade para a aplicação.
- **Portões de qualidade/Barras de protecção:** "os portões de qualidade e as barras de segurança são utilizados para estabelecer níveis mínimos aceitáveis de segurança e qualidade de privacidade" (Microsoft, 2010). Estes critérios servem para definir, por exemplo, até que nível de vulnerabilidade da aplicação vai se tolerar para a passagem de uma outra etapa de desenvolvimento.
- **Avaliação dos riscos de segurança e privacidade:** segundo Microsoft (2010), nesta fase procura-se identificar e rever mais a fundo aspectos funcionais do *software*, no que concerne a segurança. Dentre estes aspectos se encontram: identificar escopos dos testes de *fuzzing*, identificar quais partes do projecto vão requerer modelagem de ameaças, entre outros.

2.3.1.3. Design

Segundo Microsoft (2012), "Durante a fase de Design tu estabeleces as melhores práticas a seguir para esta fase através de especificações funcionais e de design, e efectuas análises de risco para identificar ameaças e vulnerabilidades no seu *software*". Nesta fase procura-se definir procedimentos que vão nortear o resto do processo de desenvolvimento.

Esta fase possui 3 actividades:

- **Requisitos de design:** segundo Microsoft (2010)**,** os requisitos de design "devem descrever características de segurança ou privacidade que serão directamente expostas aos utilizadores". Além de descrever estas características expostas aos utilizadores, esta actividade deve fazer a descrição do processo de implementação de aspectos de segurança nas diversas funcionalidades fornecidas pelo *software*.

- **Redução da Superfície de Ataque:** segundo Microsoft (2010)**,** esta actividade **"**engloba o encerramento ou restrição do acesso aos serviços do sistema, aplicando o princípio do privilégio mínimo, e empregando defesas em camadas sempre que possível". A ideia é reduzir ao máximo a superfície de ataque proveniente de uma possível vulnerabilidade.
- **Modelagem de Ameaças:** segundo Microsoft (2010), esta actividade "permite às equipas de desenvolvimento considerar, documentar e discutir as implicações de segurança dos desenhos no contexto do seu ambiente operacional planeado e de uma forma estruturada". Esta actividade é indicada em ambientes com grande risco de segurança, e deve ser feita não só pelos programadores e testadores, mas também pelos gestores do projecto.

2.3.1.4. Implementação (*Implementation*)

Segundo Microsoft (2012"a fase de Implementação é quando se estabelecem as melhores práticas de desenvolvimento para detectar e remover questões de segurança e privacidade no início do ciclo de desenvolvimento". Nesta fase também se realiza a criação de documentação e ferramentas de modo a permitir o *deploy* seguro do *software*.

Esta fase possui 3 actividades:

- **Utilização de ferramentas aprovadas:** esta actividade consiste em certificar que o projecto e desenvolvido utilizado ferramentas devidamente autorizadas e seguras, de forma a utilizar ferramentas vulneráveis.
- **Depreciar Funções Inseguras:** esta actividade está directamente ligada com o conceito de codificação segura, em que o objectivo é utilizar mecanismos seguros de criação de código de software.
- **Análise estática:** esta actividade consiste em usar ferramentas de analise do código em repouso. Para garantir maior eficiência, é importante complementar à esta actividade a análise manual de código.

2.3.1.5. Verificação (*Verification*)

Segundo Microsoft (2012), “durante a fase de Verificação, assegura-se de que o seu código cumpre os princípios de segurança e privacidade que estabeleceu nas fases anteriores”. Para isso, são feitos testes específicos utilizando ferramentas de detecção de vulnerabilidades.

Esta fase possui 3 actividades:

- **Análise Dinâmica do Programa:** esta actividade consiste em observar o comportamento do *software* em tempo de execução. Segundo Microsoft (2010), “Esta tarefa de verificação deve especificar ferramentas que monitoram o comportamento da aplicação para corrupção da memória, problemas de privilégios do utilizador, e outros problemas críticos de segurança”. A ideia e seguir todo o fluxo de execução da aplicação de modo a identificar vulnerabilidades.

- **Teste de *Fuzzing*:** a ideia do teste de *Fuzzing* é introduzir um conjunto de entradas aleatórias e/ou malformadas de modo a induzir erros no *software*, de modo a verificar se os mecanismos de tratamento de erros foram devidamente implementados.

- **Modelo de Ameaça e Revisão da Superfície de Ataque:** esta actividade consiste na revisão de todos os modelos de ameaça para assegurar que eles vão de acordo com o *software* produzido, de forma a evitar desactualizações que possam ter surgido com a alteração de requisitos durante o desenvolvimento.

2.3.1.6. Lançamento *(Release)*

Nesta fase, o *software* é entregue ao utilizador final. Segundo Microsoft (2012), nesta fase deve ser elaborado um plano de acção, para o caso de serem detectadas vulnerabilidades de segurança ou privacidade. Deve-se realizar também uma Revisão Final de Segurança e uma revisão de privacidade antes que o lançamento do *software* ao público ocorra.

- **Plano de Resposta a Incidentes:** esta actividade basicamente visa criar um plano de resposta para o caso de ocorrência de um incidente de segurança no *software*.

- **Revisão final de segurança:** esta actividade consiste na verificação de todas actividades de segurança realizadas ao longo do desenvolvimento do software, antes que o *software* seja lançado. Segundo Microsoft (2010), a Revisão final de segurança "inclui normalmente um exame de modelos de ameaça, pedidos de excepção, saída de ferramentas, e desempenho contra os portões de qualidade previamente determinada". Com base nas saídas obtidas, serão tomadas decisões.

- **Lançamento/Arquivamento:** com base nas fases anteriores, se o software obedecer aos requisitos de segurança definidos no início do projecto, é lançado para uso final. Nesta actividade também são armazenados todos recursos do projecto (documentação, código, etc.) para efeitos de manutenção futura (ou pós-lançamento).

Para além das actividades descritas anteriormente, pode-se encontrar outras actividades consideradas opcionais, como é o caso da revisão manual de código, do teste de penetração e da análise de vulnerabilidade de aplicações semelhantes.

2.3.2. OWASP SAMM

SAMM (*Software Assurance Maturity Model*) é um modelo aberto criado pela OWASP (*Open Web Application Security Project* – organização sem fins lucrativos que se dedica ao estudo de segurança de *softwares*). Mantido por voluntários especialistas em segurança dos ramos da educação e empresarial, o OWASP SAMM actualmente se encontra na versão 2.0.3 de 2022, sendo que a primeira versão (1.0) foi lançada no ano de 2009.

Segundo OWASP (2022), o OWASP SAMM proporciona um mecanismo efectivo e mensurável de organizações avaliarem e aperfeiçoarem a sua postura de segurança. Este modelo suporta o ciclo de desenvolvimento de software completo, independentemente da tecnologia e processos aplicados.

O modelo SAMM é baseado em 15 práticas de segurança agrupadas em 5 categorias (chamadas funções empresariais), sendo que cada prática de segurança possui um conjunto de actividades agrupadas em 2 fluxos diferentes (que cobrem diferentes aspectos

de uma prática de segurança), estruturadas em níveis de maturidade (posicionamento de segurança da organização em relação aos riscos e tolerâncias do seu meio envolvente).

A seguir é possível visualizar o esquema ilustrativo deste modelo:

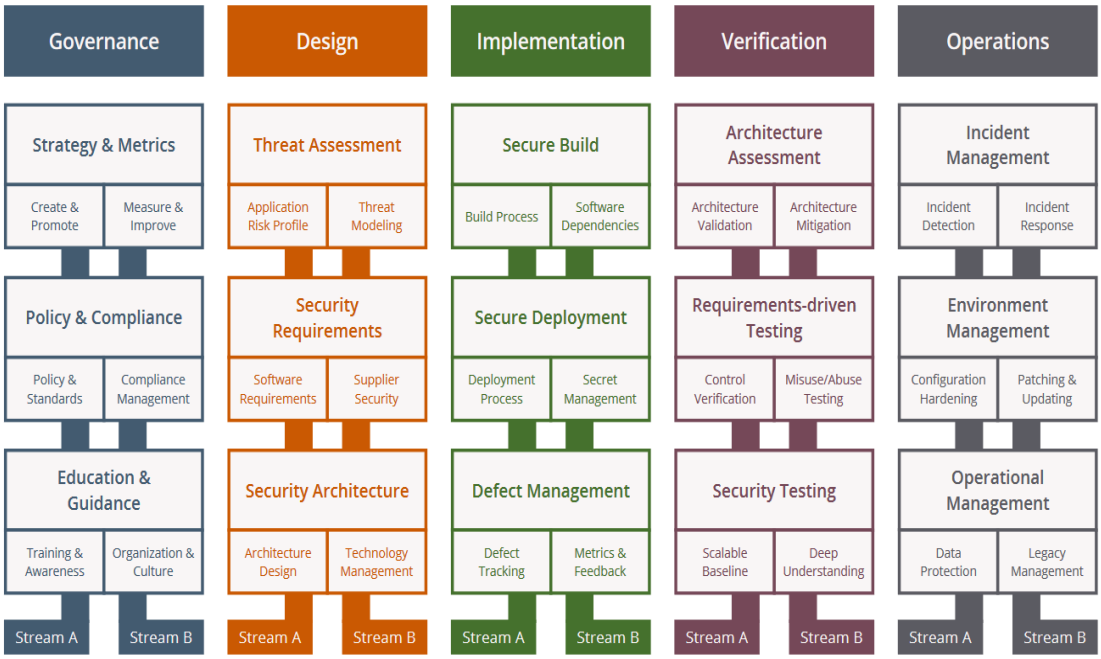

Figura 8: OWASP SAMM – Fonte: OWASP (2022)

2.3.2.1. Governança (*Governance*)

Segundo OWASP (2022), a governança "centra-se nos processos e actividades relacionados com a forma como uma organização gere as actividades globais de desenvolvimento de *software*". A governança visa exercer um papel de alto nível, realizando um plano com políticas especificas da organização e procurando elevar a consciência de segurança por parte dos colaboradores.

A Governança possui 3 práticas de segurança:

- **Estratégia e Métrica:** nesta prática faz-se um plano geral do que se pretende realizar no âmbito de actividades de segurança do *software*, de modo que se cumpram os objectivos da organização.

- **Política e Conformidade:** nesta prática, são criados mecanismos para a aderência às normas e regulamentos internos e externos, porém, assegurando o alinhamento com os objectivos comerciais da organização.
- **Educação e Orientação:** nesta prática ocorre a conscientização dos intervenientes do ciclo de desenvolvimento de *software* em aspectos de segurança de softwares.

Na tabela abaixo é dada uma breve descrição dos fluxos que compõem cada uma das práticas de segurança:

Tabela 3: Resumo da Governança – Fonte: Autor

Prática de segurança	Fluxo A	Fluxo B
Estratégia e Métrica	**Criar e promover:** neste fluxo procura-se elaborar e promover um guia para a segurança de *softwares*, baseando-se nos objectivos empresariais, aumentando desse jeito o envolvimento entre todos os intervenientes	**Medir e Melhorar:** este fluxo visa basicamente validar e melhorar o guia criado, através da realização de medições de desempenho dentro da organização.
Política e Conformidade	**Política e Padrões:** neste fluxo procura-se manter um conjunto de políticas e padrões, disponibilizando-os de modo a integrar com o Ciclo de	**Gestão de Conformidade:** este fluxo visa identificar requisitos de conformidade, que servirão de base para a integração no Ciclo de

	Desenvolvimento de *Software*.	Desenvolvimento de *Software*.
Educação e Orientação	**Formação e Sensibilização:** este fluxo visa dotar os intervenientes da organização de conhecimentos gerais sobre segurança de *software*.	**Organização e Cultura:** Neste fluxo procura-se promover práticas de segurança durante o Ciclo de Desenvolvimento de *Software*, realçando a sua importância.

2.3.2.2. Design

Segundo OWASP (2022), “O design diz respeito aos processos e actividades relacionados com a forma como uma organização define objectivos e cria *software* no âmbito de projectos de desenvolvimento”. É neste passo onde serão especificados detalhes técnicos sobre aspectos de segurança a implementar.

O Design possui 3 práticas de segurança:

- **Avaliação de Ameaças:** nesta prática basicamente se identifica potenciais riscos de segurança para o *software*, tendo como base as suas funcionalidades e características do ambiente em tempo de execução.
- **Requisitos de segurança:** nesta prática define-se os requisitos de segurança de acordo com o *software* a desenvolver e dos fornecedores de *software*.
- **Arquitectura de segurança:** esta prática visa basicamente realizar a gestão de potenciais riscos a nível arquitectónico do *software*. Esta prática lida directamente com a segurança dos componentes e tecnologias utilizadas durante a concepção da arquitectura do *software*.

Na tabela abaixo é dada uma breve descrição dos fluxos que compõem cada uma das práticas de segurança:

Tabela 4: Resumo do Design – Fonte: Autor

Prática de segurança	Fluxo A	Fluxo B
Avaliação de Ameaças	**Perfil de risco da aplicação:** neste fluxo, procura-se identificar aplicações que podem causar uma ameaça grave para a organização em caso de um ataque ou violação.	**Modelagem de Ameaças:** este fluxo visa conscientizar as equipes de desenvolvimento sobre os riscos existentes no produto a ser desenvolvido e como mitigar tais riscos.
Requisitos de segurança	**Requisitos de *software*:** este fluxo descreve os objectivos e expectativas que norteiam a protecção de serviços e dados da aplicação.	**Segurança do Fornecedor:** este fluxo se concentra nos requisitos de segurança de *software* em casos de desenvolvimento por uma entidade externa.
Arquitectura de segurança	**Design de Arquitectura:** neste fluxo basicamente aplicam-se boas práticas de segurança com vista a conceber uma arquitectura de *software* adequada.	**Gestão Tecnológica:** este fluxo visa garantir a segurança de todos os componentes usados no desenvolvimento de *software*, como

		tecnologias, *frameworks*, entre outros.

2.3.2.3. Implementação (*Implementation*)

Segundo OWASP (2022), a implementação "está centrada nos processos e actividades relacionados com a forma como uma organização constrói e implementa componentes de *software* e os seus defeitos relacionados. Esta é a derradeira fase de criação do *software* e deve seguir um controle rigoroso de segurança para que não ocorram falhas.

A implementação possui 3 práticas de segurança:

- **Construção segura:** esta prática visa garantir que os *softwares* sejam criados de forma padronizada e repetível, usando bibliotecas e ferramentas seguras.
- **Implementação segura:** esta prática foca-se na segurança no momento da Implementação do *software* no ambiente de produção.
- **Gestão de Defeitos:** esta prática visa basicamente realizar a gestão de defeitos de segurança do *software*, tendo em conta as suas métricas.

Na tabela abaixo é dada uma breve descrição dos fluxos que compõem cada uma das práticas de segurança:

Tabela 5: Resumo da implementação – Fonte: Autor

Prática de segurança	Fluxo A	Fluxo B
Construção segura	**Processo de *build*:** neste fluxo procura-se implementar actividades de segurança no processo de *build* do *software*, de modo a garantir uma aplicação mais consistente.	**Dependências de *software*:** este fluxo visa assegurar que as bibliotecas externas implementadas tenham um bom nível de segurança.
Implementação segura	**Processo de Implementação:** este fluxo procura garantir consistência através da utilização de artefactos de *software* correctos para a produção.	**Segurança do Fornecedor:** este fluxo visa garantir um tratamento adequado de todos os elementos de dados sensíveis dentro do ambiente da organização, como o caso de credenciais.
Gestão de Defeitos	**Detecção de defeitos:** este fluxo, “gere a recolha e o acompanhamento de todos os problemas potenciais num pedaço de *software*, desde falhas arquitectónicas a problemas de codificação e vulnerabilidades de	**Métricas e Feedback:** o objectivo deste fluxo é produzir métricas e feedbacks de modo a melhorar as actividades de segurança.

	tempo de execução” (OWASP, 2022). O objectivo é ter o conhecimento exacto da proveniência do problema.	

2.3.2.4. Verificação (*Verification*)

Segundo OWASP (2022), a verificação “centra-se nos processos e actividades relacionados com a forma como uma organização verifica e testa os artefactos produzidos ao longo do desenvolvimento de *software*”. Nesta etapa encontramos tipicamente actividades relacionadas com testes, revisão e avaliação.

A verificação possui 3 práticas de segurança:

- **Avaliação de Arquitectura:** esta prática visa garantir que a arquitectura do *software* e da infra-estrutura vão de acordo com todos os requisitos fundamentais de segurança e conformidade, atenuando as ameaças de segurança identificadas.
- **Testes orientados por requisitos:** esta prática visa garantir que os mecanismos de segurança implementados no projecto funcionem devidamente e cumpram os requisitos estabelecido no projecto.
- **Testes de Segurança:** nesta prática são usadas técnicas automáticas e manuais para a detecção e resolução de problemas de segurança.

Na tabela abaixo é dada uma breve descrição dos fluxos que compõem cada uma das práticas de segurança:

Tabela 6: Resumo da Verificação – Fonte: Autor

Prática de segurança	Fluxo A	Fluxo B
Avaliação de Arquitectura	**Validação de Arquitectura:** este fluxo visa basicamente realizar a confirmação da segurança do software e da arquitectura de suporte, verificando o fornecimento de objectivos dos componentes e seus requisitos de segurança	**Mitigação de Arquitectura:** neste fluxo, são mitigadas todas as ameaças identificadas na avaliação de ameaças e as arquitecturas de referência são actualizadas para lidar com quaisquer ameaças novas ou não controladas.
Testes orientados por requisitos	**Verificação de controlo:** este fluxo realiza a validação de controlos e requisitos de segurança, verificando se eles foram satisfeitos através de testes, evitando assim a introdução de bugs em futuros lançamentos.	**Teste de mau uso/abuso**: este fluxo faz uso de técnicas especificas com o intuito de identificar fraquezas nas características do software, que podem ser usadas num ataque.
Testes de Segurança	**Linha de base escalável:** este fluxo realiza um teste amplo nas aplicações sem demasiada profundidade, através do uso de ferramentas	**Compreensão profunda:** neste fluxo, são realizados testes profundos de forma manual, utilizando vectores de ataque

	automatizadas que validam a segurança no processo de construção e implementação.	complexos com o intuito de tornar os testes de segurança mais avançados.

2.3.2.5. Operações (*Operations*)

Segundo OWASP (2022), esta etapa “engloba as actividades necessárias para assegurar a confidencialidade, integridade e disponibilidade durante toda a vida operacional de uma aplicação e dos seus dados associados”. A ideia e garantir que o *software* e resiliente e responde a mudanças no panorama operacional.

Esta etapa possui 3 práticas de segurança:

- **Gestão de Incidentes:** esta prática visa basicamente descrever actividades realizadas para melhorar a detecção e resposta a incidentes de segurança na organização.
- **Gestão do Ambiente:** esta prática visa descrever actividades levadas a cabo com o intuito de proteger os ambientes em que os aplicativos da organização operam.
- **Gestão Operacional:** esta prática visa descrever actividades necessárias a nível operacional para manter a segurança durante todo o ciclo de vida do produto de *software*.

A seguir é possível visualizar uma breve descrição dos fluxos que compõem cada uma das práticas de segurança:

Tabela 7: Resumo da etapa de operações – Fonte: Autor

Prática de segurança	Fluxo A	Fluxo B
Gestão de Incidentes	**Detecção de incidente:** este fluxo consiste basicamente na identificação de incidentes de segurança quando estes ocorrem e no posterior início de procedimentos de resposta a estes.	**Resposta a Incidentes:** neste fluxo procura-se iniciar procedimentos de resposta a incidentes, ou seja, mecanismos a aplicar de forma a atenuar danos de um incidente de segurança.
Gestão do Ambiente	**Reforço da configuração:** este fluxo se foca na gestão de configurações de todos os elementos tecnológicos, com enfase nos obtidos de terceiros, porque a sua concepção se encontra fora do controle da organização.	**Correcção e Actualização:** este fluxo procura garantir que todos os elementos tecnológicos de terceiros estejam devidamente actualizados e que todos os utilizadores dos softwares desenvolvidos pela organização (clientes) recebam actualizações**.**
Gestão Operacional	**Protecção de dados:** este fluxo visa garantir a protecção dos dados, desde a sua criação, tratamento,	**Gestão do legado:** este fluxo preocupa-se com a identificação, gestão e monitoria de sistemas e outros componentes que

	armazenamento e processamento.	se encontram em desuso, de modo que seja removido tudo que for desnecessário com vista a diminuir a superfície de um potencial ataque.

Os 2 modelos ilustrados deixam claro que a preocupação com o estudo de segurança dos *softwares* em cada etapa do ciclo de desenvolvimento é uma realidade e que é de extrema importância que as organizações comecem a adoptar ou mesmo que concebam os seus próprios modelos. Com a adopção de modelos de ciclo de vida seguro de *software*, a preocupação não se prende apenas com a execução eficiente das várias acções do *software*, mas que estas acções a executar, sejam com o máximo de segurança de forma a salvaguardar os activos das organizações.

3. Capítulo III - Caso de estudo

3.1. Centro de Informática da UEM (CIUEM)

O CIUEM é o órgão que serve como base para o funcionamento dos sistemas da UEM.

> O CIUEM é um órgão académico especializado no ramo de informática, que se dedica ao ensino, investigação fundamental e aplicada e serviços na procura e implementação de soluções e metodologias que permitam expandir a utilização das tecnologias de informação e comunicação (TIC's) e trazer os benefícios da sua utilização para os processos de produção, disponibilização de serviços, melhoramento do ensino e aprendizagem assim como a investigação, na perspectiva de resolver de certo modo as necessidades da Universidade Eduardo Mondlane em particular e do país em geral. (*Quem Somos | CIUEM*, s.d.)

Esta instituição deu os seus primeiros passos no ano de 1969 quando foram oferecidos dois computadores Elliott 803B ao Laboratório de Cálculo Numérico e Máquinas Matemáticas, mas só em Janeiro de 1982 viria a ser oficializada. Segundo *Correios* (s.d.), "em 1986, o centro tornou-se uma unidade autónoma, fora da Faculdade de Matemática, mantendo um Departamento de Manutenção e a Unidade de Processamento de Salários.". Hoje, o CIUEM revela-se uma instituição de elevada importância para a UEM e o país em geral, participando activamente no processo de desenvolvimento tecnológico.

3.1.1. Visão

Ser um Centro de Excelência em Tecnologias de Informação e Comunicação (TIC's) no País, na Região e no Mundo.

3.1.2. Missão

Contribuir para o progresso da Universidade Eduardo Mondlane e para a implementação das políticas nacionais através de tecnologias de informação e comunicação, assumindo um papel de liderança no desenvolvimento de soluções tecnológicas inovadoras e prestação de serviços, com destaque para o ensino e investigação.

3.1.3. Estrutura

A seguir é possível visualizar a estrutura geral do CIUEM:

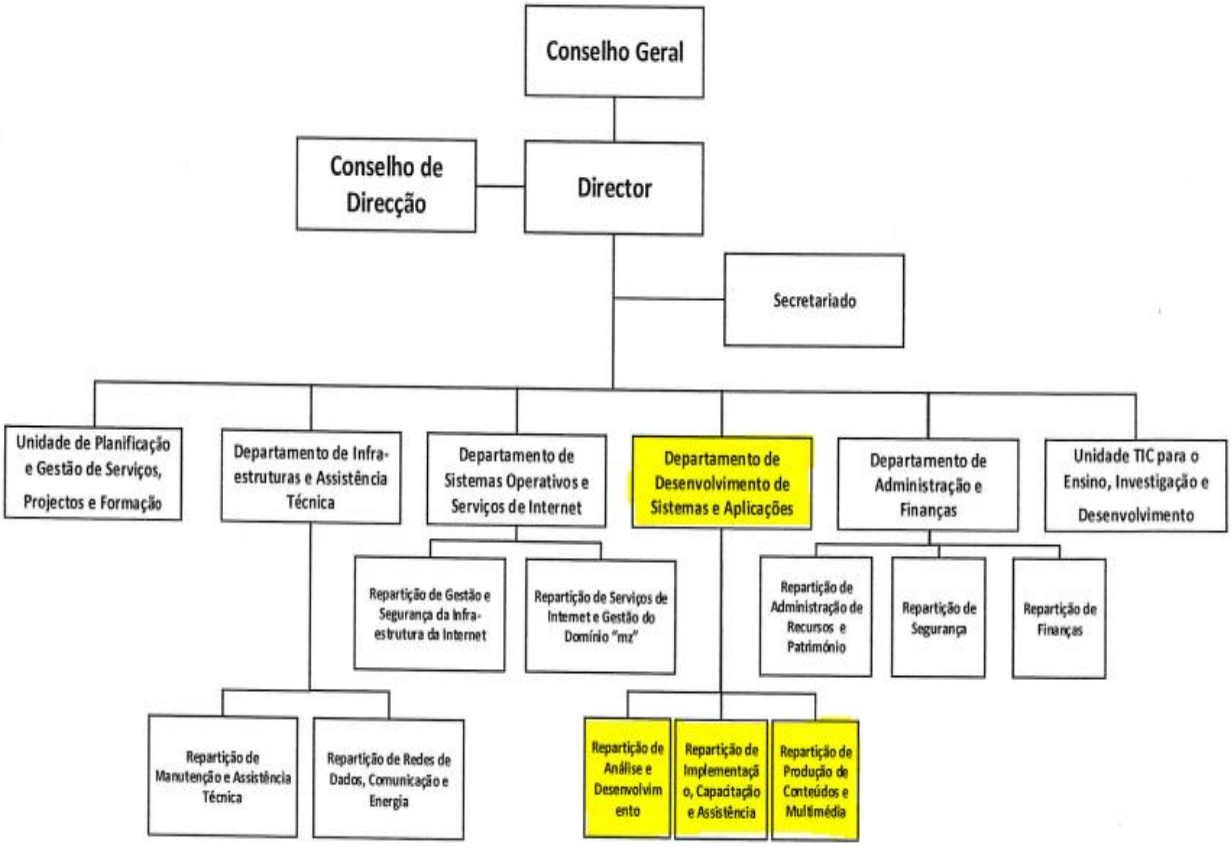

Figura 9: Estrutura CIUEM – Fonte: *CIUEM (2022)*

No âmbito da realização das suas actividades, o autor foi inserido no Departamento de Desenvolvimento de Sistemas e Aplicações (DDSA), que é o departamento que está directamente ligado com as actividades relacionadas com o tema de estudo. Segundo CIUEM (2022), dentre as principais responsabilidades deste departamento se enquadram:

- Realizar e gerir as actividades de análise e desenvolvimento de sistemas de informação, de acordo com a sua metodologia de desenvolvimento padronizada, incluindo as fases de concepção, elaboração, construção e implementação;
- Desenhar e produzir websites e materiais multimédia;
- Prestar assistência técnica na concepção, desenvolvimento e gestão técnica de plataformas de suporte ao ensino, investigação, gestão e administração na UEM;
- Prestar assistência técnica na concepção, desenvolvimento e gestão técnica do portal e dos websites da UEM e do público em geral;

- Prestar assistência técnica na concepção, desenvolvimento de sistemas para clientes externos.

Abaixo se encontra a estrutura da equipe que desenvolve os sistemas no DDSA:

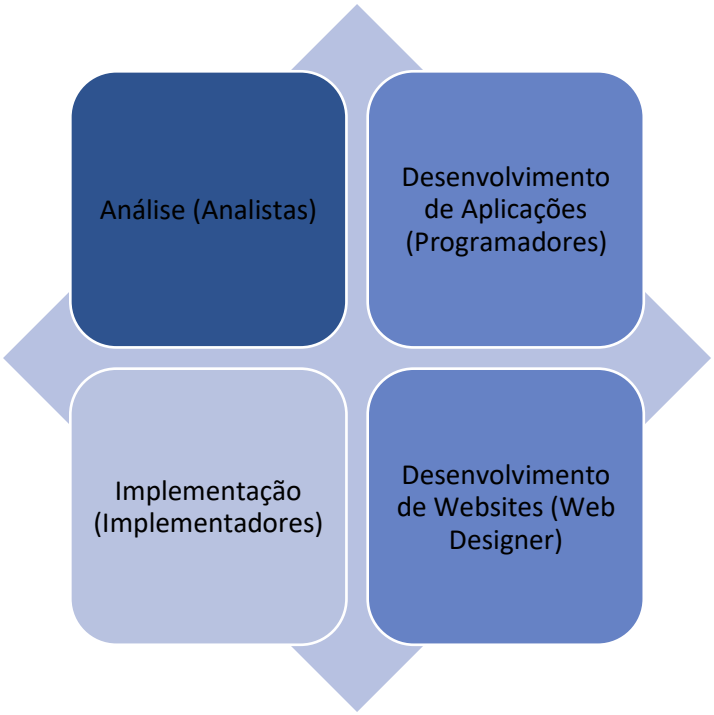

Figura 10: Estrutura DDSA – Fonte: Autor

3.1.4. Descrição da situação actual

Os membros do DDSA trabalham em conjunto para o desenvolvimento dos vários projectos a si confiados. Para o desenvolvimento dos sistemas, o DDSA aplica certos modelos de Ciclo de Vida de Desenvolvimento de Software, de acordo com as necessidades específicas de cada projecto. Para cada projecto, actualmente, é aplicado um dos modelos a seguir:

- Incremental;
- RAD;
- Prototipagem.

No que concerne a segurança, foi identificado que existe algum défice de aplicação de mecanismos durante e após a execução do projecto, sendo que apenas são aplicados (em alguns casos) mecanismos básicos já em fases finais do desenvolvimento como por exemplo o uso de certificados TLS/SSL.

No âmbito do estudo, foi realizada uma análise de vulnerabilidades (automática e manual) em 13 servidores que alojam aplicações desenvolvidas e mantidas pelo DDSA, sendo que a ilustração se encontra patente nos anexos do presente relatório. A seguir é possível visualizar um resumo dos resultados obtidos:

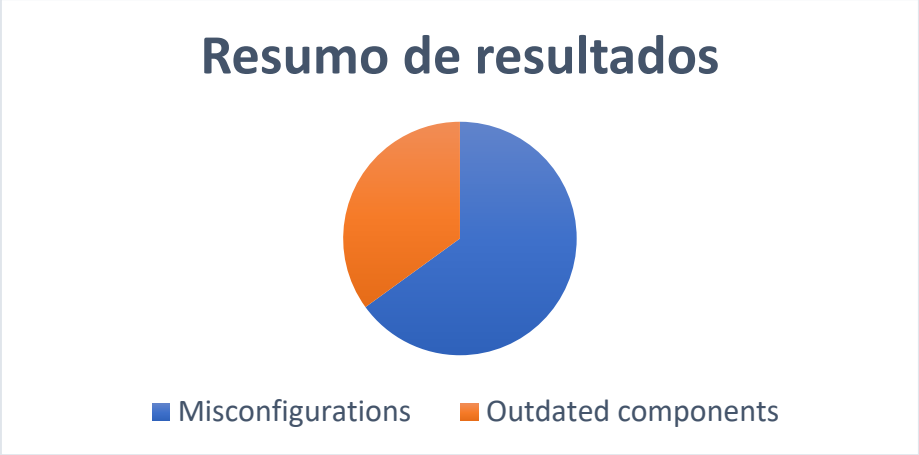

Figura 11: Resumo dos resultados da análise de vulnerabilidades dos servidores de aplicações da CIUEM – Fonte (Autor)

Da figura acima pode-se notar que as vulnerabilidades identificadas provém de componentes desactualizados e configurações mal efectuadas. Durante a análise de vulnerabilidades, o autor foi mais a fundo, tendo explorado uma das vulnerabilidades que culminou com o acesso a um dos sistemas.

Da análise realizada, ficou a ideia de que existe uma clara necessidade de implementação de um conjunto de práticas de modo a evitar situações de vulnerabilidades durante todo o ciclo de vida das aplicações.

4. Capítulo IV - Proposta de solução

Após a realização de pesquisas teóricas e constatações práticas no local de estudo, o autor entendeu que seria benéfico melhorar o processo de desenvolvimento de *software* aplicado actualmente no CIUEM, através da produção de um modelo de ciclo de vida seguro de desenvolvimento de software, com vista a incluir actividades de segurança.

4.1. Descrição da proposta de solução

A proposta de modelo é baseada nos padrões já citados anteriormente neste relatório, além de outro padrão também muito relevante, desenvolvido pelo governo da Malásia (*Guidelines for Secure Software Development Life Cycle* (SSDLC)). O autor pretende, através da experiência adquirida destes padrões, produzir um modelo que realmente se adeqúe às especificidades do local de estudo.

O autor propõe um modelo contendo as 5 fases tradicionais de desenvolvimento de software e 1 adicional, que além das actividades habituais, abarcará também actividades de segurança em cada uma delas. Visto que o presente estudo surge como uma melhoria de processos já existentes, serão ignorados detalhes sobre as actividades habituais de um ciclo de vida de desenvolvimento de software, concentrando-se somente nas actividades de segurança em cada fase.

É importante ressaltar que as fases utilizadas na proposta de modelo apresentada pelo autor, podem ser usadas para criar outros tipos de modelos de ciclo de vida seguro de desenvolvimento de software, sendo que a aplicação de um modelo em detrimento de outro estará ao critério da equipe de desenvolvimento. Para este caso específico, o autor propõe um modelo baseado no incremental, incluindo tarefas de segurança em cada uma das fases.

A proposta do autor é sustentada com os seguintes pontos:

- Actualmente, um dos modelos em uso é o incremental, o que facilita a adaptação do modelo proposto pelo autor;
- O modelo incremental facilita a incorporação de tarefas de segurança;

- Uma vez que o modelo incremental proporciona a entrega da aplicação em pequenos pedaços, é mais fácil identificar e corrigir falhas de segurança, do que seria numa aplicação completa.

A seguir é possível visualizar o esquema do modelo:

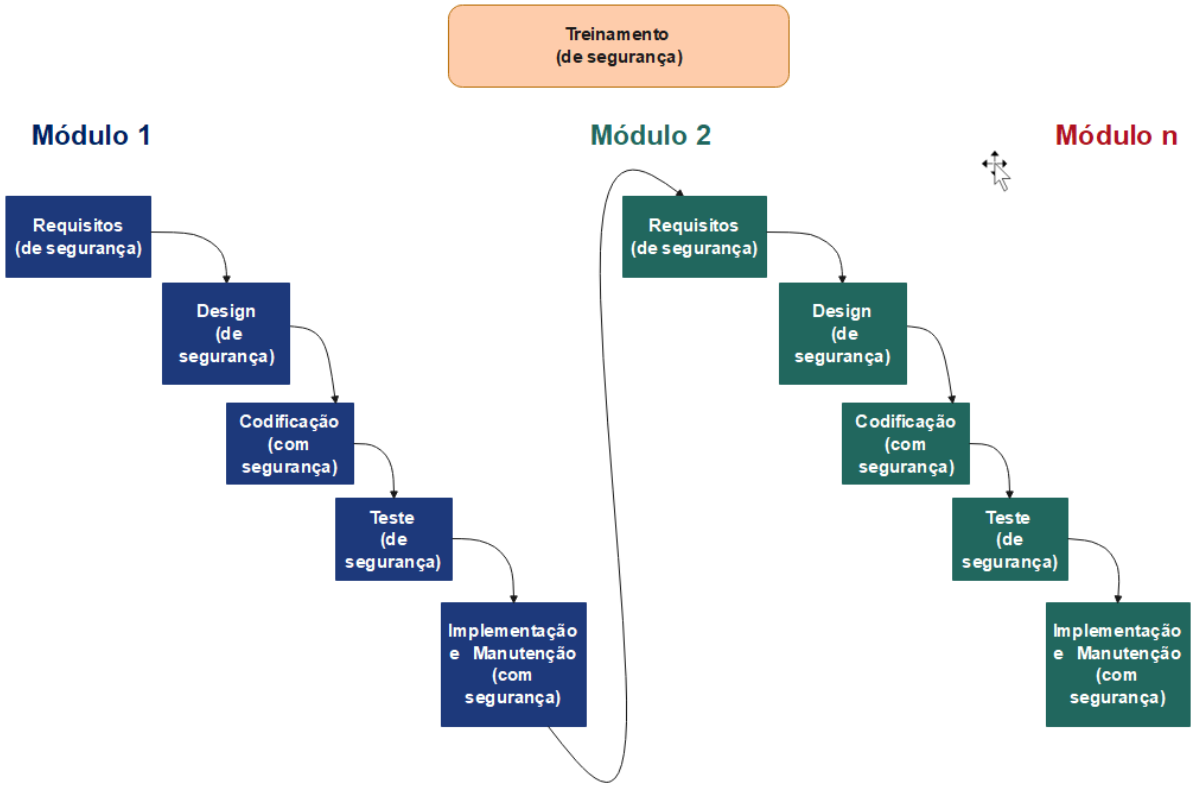

Figura 12: Proposta de modelo de Ciclo de Vida Seguro de Desenvolvimento de Software – Fonte: Autor

O esquema acima apresentado é complementado por uma lista de controle de actividades presente nos anexos do presente relatório, cujo objectivo é auxiliar a equipe de desenvolvimento no controle de actividades do presente modelo proposto.

4.1.1. Treinamento (de segurança)

Esta é a primeira fase sugerida pelo modelo. Nesta fase, haverá uma conscientização de todos os envolvidos na organização, especialmente da equipe de desenvolvimento, sobre aspectos de segurança através de *workshops* e treinamentos.

É importante destacar que esta fase não será executada a cada novo projecto/incremento, mas sim periodicamente com um intervalo de tempo fixo, seguindo um cronograma previamente criado pelas camadas de gestão em parceria com a equipe de desenvolvimento. A fase de treinamento sustentará as fases subsequentes através de transmissão de conhecimentos para a melhor incorporação de aspectos de segurança.

O autor propõe que os treinamentos abranjam todas as áreas de segurança para solidificar os conhecimentos gerais sobre a área, uma vez que a segurança é um processo que envolve muitas áreas e todas elas precisam estar seguras para que não haja falhas que afectem directamente as aplicações.

Um dos conhecimentos importantes a ser transmitido aos programadores durante os treinamentos é sobre a codificação segura, de modo a evitar as vulnerabilidades lançadas periodicamente pela Owasp Top 10:

> O OWASP Top 10 é um documento de conscientização padrão compartilhado publicamente para desenvolvedores das dez vulnerabilidades de segurança de aplicativos da Web mais críticas, de acordo com a Fundação. A lista OWASP Top 10 é desenvolvida por especialistas em segurança de aplicativos da Web em todo o mundo e é actualizada a cada dois anos. Ele visa educar empresas e desenvolvedores sobre como minimizar os riscos de segurança de aplicativos. (OWASP Top Ten | OWASP Foundation, s.d.)

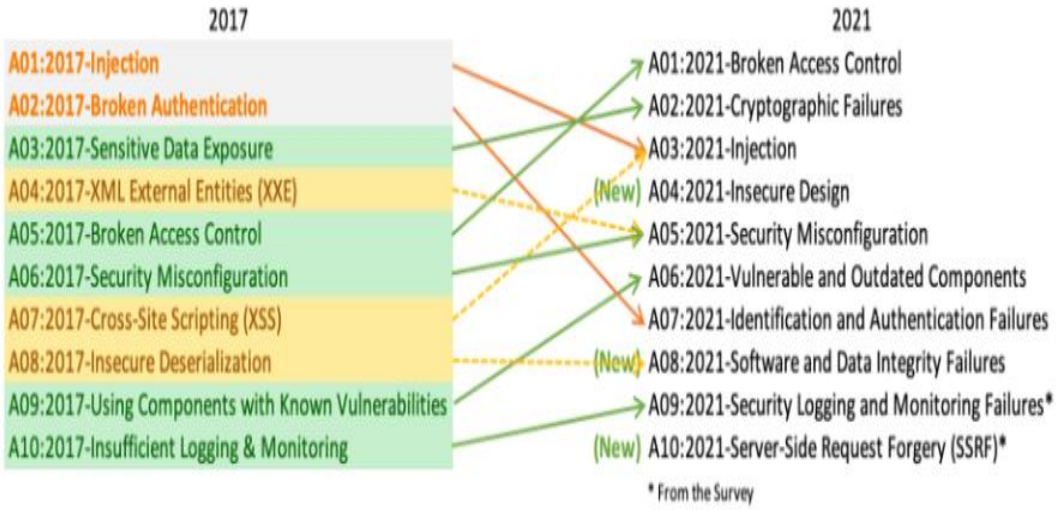

Figura 13: OWASP top 10 2017 VS OWASP top 10 2021 – Fonte: OWASP Top Ten | OWASP Foundation (n.d.)

4.1.2. Requisitos (de segurança)

A fase de requisitos é a segunda deste modelo e a primeira cuja execução é obrigatória em cada projecto de *software*. Do mesmo jeito que é feita a colecta de requisitos funcionais e não funcionais da aplicação, nesta fase são traçados todos os requisitos de segurança desejados para o projecto em questão (estes requisitos vão delinear todos os aspectos de segurança a aplicar na aplicação). A cada novo incremento (módulo), são revisados os requisitos de forma a perceber se existe necessidade de alguma alteração, com base na experiência adquirida de um módulo anterior.

É importante que os requisitos traçados respondam (entre outros) ao trio CIA (já abordado neste trabalho) e a mecanismos de autenticação e autorização de forma a maximizar a segurança da aplicação. Os requisitos de segurança variam de acordo com o projecto, pois cada um deles possui especificidades, havendo uns com necessidades de segurança mais altos que os outros.

De modo a facilitar a implementação dos requisitos de segurança nas fases subsequentes, organizam-se os requisitos em grau de prioridade:

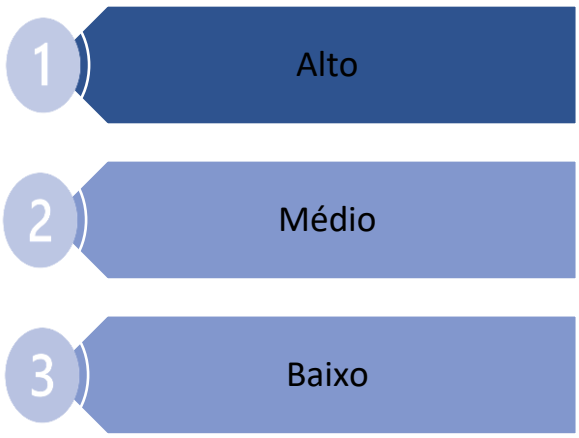

Figura 14: Grau de prioridade de requisitos – Fonte: Autor

Uma actividade a destacar nesta fase é de elaboração do diagrama de casos de uso indevido, pois ele ajuda a "descrever o comportamento que o proprietário do sistema/entidade não quer que ocorra". Tendo uma visão geral de cada funcionalidade do software, torna-se mais fácil prever situações de falhas de segurança e consequentemente elaborar requisitos que colmatem esta fraqueza. Este diagrama é de grande utilidade pois ele auxilia na prevenção de falhas de segurança através da previsão de possíveis comportamentos de utilizadores, que podem culminar numa violação de segurança da aplicação.

4.1.3. Design (de segurança)

Na fase de design, procura-se especificar a arquitectura e lógica da aplicação a ser criada. É durante esta fase em que se olharão para todos os requisitos de segurança especificados na fase anterior, e procurar-se-ão mecanismos para a satisfação de cada requisito.

Esta fase abrange vários aspectos, porém vale destacar alguns como:

- **Tipo de linguagem de programação:** na escolha da linguagem de programação, deve-se antes responder as seguintes questões:
  - A linguagem tem vulnerabilidades conhecidas?
  - Como a linguagem realiza a gestão de memória?
  - A linguagem tem suporte a ferramentas de segurança?
  - A linguagem tem uma boa comunidade de desenvolvedores?
  - A linguagem de programação é *unmanaged* ou *managed*?

  Respondendo a estas questões, já é possível escolher uma linguagem de programação adequada, de forma a aumentar o nível de segurança.

- **Segurança da base de dados:** é crucial que os dados da aplicação a armazenar estejam seguros, pois os dados são o coração de qualquer negócio. Dito isto, é importante primeiramente realizar uma boa escolha do sistema de gestão de base de dados de modo a fortificar a segurança. Aspectos como controle de acesso, suporte a criptografia, suporte a backups e restaurações, entre outros, devem ser tomados

em conta durante a escolha. Depois da escolha do sistema de gestão base de dados, é importante desenhar a base de dados seguindo as melhores práticas, aplicando aspectos como normalização, encriptação, desenho de *triggers* e *views*, entre outros.

- **Interface:** durante o design de interfaces, é importante que se garanta que nenhum tipo de informação sensível seja exibido (erros, *passwords*, etc). Durante o desenho de interfaces deve-se criar também telas de gestão de aplicação e de exibição de *logs* da aplicação.

É importante salientar que algumas das actividades desta fase são executadas somente no início do projecto, não havendo necessidade de repetição em cada fase, porém, outras, devido à sua natureza exigem uma constante revisão.

4.1.4. Codificação (com segurança)

Esta é a derradeira fase de criação da aplicação. Nesta fase, procura-se aplicar todos os requisitos e especificações de segurança definidas nas fases anteriores. A ideia é implementar na prática todos os conhecimentos de codificação segura de *software* aprendidos na fase de treinamento, de modo a evitar vulnerabilidades de segurança.

Deve-se, durante a fase de codificação, utilizar ferramentas que auxiliem no processo de análise do código (de forma a inspeccionar a qualidade do código), no controle de versões das bibliotecas (de forma a evitar vulnerabilidades), no controle de versões do código (de forma a manter o código íntegro), entre outros. É nesta fase que os desenvolvedores realizam pequenos testes de segurança no *software* (estáticos e dinâmicos), à medida que vão adicionando mais funcionalidades.

4.1.5. Teste (de segurança)

Durante a fase de teste, a preocupação é aferir se o *software* desenvolvido cumpre todos os requisitos de segurança estabelecidos no início do projecto. A ideia é “varrer” todas as

funcionalidades do *software* em busca de potenciais falhas de segurança, de modo a resolvê-las antes do *software* entrar em ambiente de produção.

Para realizar este teste de segurança, são utilizados vários métodos e ferramentas, a destacar as análises estática e dinâmica de código e a varredura de vulnerabilidades.

Segundo Malaysia (2020), eis alguns pontos relevantes a considerar durante a fase de teste:

- Testagem da validação de entradas;
- Testagem do tratamento de erros;
- Testagem do tratamento de testes de injecção;
- Testagem de controles contra a escalação de privilégios.

Quando finalizada a fase de testes, havendo-se achado vulnerabilidades na aplicação, se voltará à fase anterior de forma a corrigir o que estiver mal, e se realizará novamente o teste. Só se avançará à última fase do ciclo de vida, depois de se garantir que a aplicação possui boas condições de segurança.

4.1.6. Implementação e Manutenção (com segurança)

Na fase de Implementação, se coloca o *software* no ambiente de produção. A preocupação é de assegurar que o ambiente que vai correr a aplicação possui o mais alto nível de segurança de modo a evitar vulnerabilidades. Nesta fase procura-se também realizar os últimos testes de segurança e últimas verificações nas funcionalidades de forma a evitar falhas de segurança.

Após a Implementação, o próximo objectivo é a manutenção do *software*. Na manutenção da *software*, deve-se garantir a monitoria tanto do software como do ambiente de produção, através da varredura de vulnerabilidades e testes de penetração, com uma periodicidade previamente determinada. Deve-se garantir a instalação imediata de *patches* de segurança no ambiente de produção, assim como nas dependências da aplicação, porém, com a devida cautela de modo a evitar períodos longos de interrupção de actividades.

Outra actividade crucial a realizar nesta fase é a de *backups*. Deve-se realizar um plano de *backups* de todos os componentes da aplicação (incluindo a base de dados), sendo que

este deve ser em local seguro, isolado e desligado da rede (pode-se também, em simultâneo, realizar o *backup* na *cloud*). Desta forma, ainda que ocorra um ataque bem-sucedido, a instituição pode rapidamente retomar às suas actividades, depois de devidamente preparado o ambiente.

5. Capítulo V - Discussão de resultados

O presente trabalho visava conceber um modelo de ciclo de vida seguro de desenvolvimento de *software* no CIUEM. Este objectivo surgiu da necessidade identificada na instituição de adicionar práticas de segurança durante todo o processo de desenvolvimento dos *softwares*.

Para conceber a proposta, o autor primeiramente realizou uma pesquisa de várias fontes bibliográficas (livros, artigos, revistas, *internet*). Nestas pesquisas bibliográficas se falou dos ciclos tradicionais de desenvolvimento de software em que se realizou a caracterização e no final se fez uma análise comparativa entre estes. De seguida o autor fez uma breve descrição do conceito Segurança de Informação, tendo apontado os seus pilares, suas camadas e tendo apresentado alguns termos relevantes. Por fim, o autor realizou uma descrição de 2 modelos de ciclo de vida de desenvolvimento de *software* internacionalmente utilizados no âmbito do desenvolvimento de softwares.

Para adquirir informações fundamentais da instituição, o autor realizou uma entrevista ao responsável do DDSA, sendo que o guião da entrevista se encontra patente nos anexos. Como resultado desta análise, o autor conseguiu obter toda a informação pertinente para o presente estudo, como: a visão e missão da organização, a estrutura organizacional, a composição da equipe de desenvolvimento, as metodologias empregues para o desenvolvimento e a situação actual da infraestructura dos *softwares* no que concerne à segurança. Através de uma varredura de vulnerabilidades, o autor pôde identificar um conjunto de brechas nos servidores da instituição, tendo recomendado mecanismos de mitigação.

Na fase da proposta de solução, o autor compilou toda a informação recolhida nas fases anteriores para conceber um modelo que realmente se adequasse à realidade da instituição. O modelo proposto possui 6 fases e segue a lógica de um modelo famoso, nomeadamente o modelo incremental. De forma a complementar o modelo proposto, o autor apresentou um *template* contendo uma lista de controle de actividades que visa auxiliar a equipe de desenvolvimento a perceber o grau de evolução do projecto, no que diz respeito ao emprego de segurança do *software* em desenvolvimento.

6. Capítulo VI - Conclusões e recomendações

6.1. Conclusões

O desenvolvimento de *software* é um processo complexo e delicado, que exige a todos os intervenientes, além de habilidades técnicas, total comprometimento e responsabilidade. Ao longo do desenvolvimento do presente trabalho, ficou evidente que o desenvolvimento de software, além de se concentrar apenas na funcionalidade final, deve priorizar a questão de segurança, desde a fase inicial de colecta de requisitos até a fase derradeira de Implementação e manutenção do software.

Ao longo do trabalho, procurou-se descrever as principais metodologias tradicionais usadas no desenvolvimento de *software*, tanto a nível geral assim como na instituição em estudo. Constatou-se que estas metodologias se concentram mais na garantia de funcionalidade das aplicações e não tanto no factor segurança, deixando assim uma porta aberta para vulnerabilidades de segurança. Ao longo dos anos, à medida que os ataques cibernéticos ficavam mais frequentes, o factor segurança começou a ganhar força na área de desenvolvimento de *software*, tendo sido criados vários modelos que visam incorporar tarefas de segurança em todo o ciclo de desenvolvimento de *software.* Durante o trabalho foram abordados 2 modelos internacionalmente aceites, com vista a elucidar o leitor acerca da aplicabilidade deste conceito num ambiente realístico.

No local de estudo, o autor pôde ter uma visão realística dos procedimentos aplicados actualmente pela equipe de desenvolvimento ao longo da concepção dos softwares, da estrutura organizacional da equipe, da cultura de segurança da organização como um todo e da situação actual de segurança tanto dos softwares como dos servidores que as alojam. O que o autor constatou, é que a organização aplica alguns mecanismos básicos de segurança já nas fases finais de desenvolvimento, não existindo, porém, mecanismos ou mesmo tarefas claras de garantia de segurança desde as fases iniciais. Como resultado desta falta de tarefas claras de garantia de segurança, o autor identificou diversas vulnerabilidades nas aplicações e nos seus respectivos servidores, com mais destaque aos servidores, tendo posteriormente indicado mecanismos de correcção junto da equipe responsável.

Com base em todas as constatações feitas no local de estudo e em trabalhos teóricos pesquisados, o autor avançou com uma proposta de modelo de ciclo de vida seguro de desenvolvimento de *software*, com vista a colmatar o défice do factor segurança ao longo do ciclo de vida dos *softwares* desta instituição. A proposta dada pelo autor, além de abranger todo o ciclo de vida de desenvolvimento de *software*, se alinha com a estrutura organizacional e procedimentos aplicados actualmente pela equipe de desenvolvimento de *software*. Com a aplicação do modelo proposto pelo autor, a organização está dando os primeiros passos nos esforços com vista a garantir aplicações com o nível mais alto possível de segurança.

6.2. Recomendações

Ao longo da concepção da proposta de solução, o autor se deparou com alguns constrangimentos devido à falta de existência de uma equipe dedicada à segurança de *softwares*, que trabalhe de forma contínua com a equipe de desenvolvimento. O desafio do autor foi empreender esforços de modo a propor um modelo que pudesse ser operacionalizado com os recursos humanos existentes na instituição.

Para a instituição, o autor recomenda que se invista mais em formação e contratação de recursos humanos de forma a implementar a cultura de segurança a todos os níveis, pois a segurança se consegue aplicando um conjunto de esforços em diferentes camadas.

Para os próximos pesquisadores, o autor recomenda que se reveja o presente modelo, de modo a acrescentar mais tarefas de segurança em cada fase proposta pelo autor, indo de acordo com os recursos existentes na altura da pesquisa. Outra recomendação dada pelo autor, é que se pesquisem mecanismos de associação entre a equipe de desenvolvimento e a equipe de operações, porém considerando aspectos de segurança. Um conceito muito utilizado para se conseguir esse feito, é o de DevSecOps.

7. Capítulo VII - Bibliografia

7.1. Referências bibliográficas

1. Bhuvaneswari, T., e Prabaharan, S. (2013). A Survey on Software Development Life Cycle Models. *International Journal of Computer Science and Mobile Computing*, *2ª ed*.
2. Cepik, M. A. C., e Marcelino, H. M. (2021). Segurança cibernética em Moçambique: Conceitos, infraestrutura e desafios de implementação. *Carta Internacional*, *16*(3).
3. Chun Wei, S. (2005). Department of Computer Science, University of Auckland.
4. CIUEM. (2022). *Visita de Estudo UNITIVA ao Centro de Informática*.
5. Coelho, B. (2019, September 20). *Tipos de pesquisa: Aprenda como escolher a correta para seu trabalho*. Blog da Mettzer. https://blog.mettzer.com/tipos-de-pesquisa/
6. http://www.mcm.org.mz/mv/instituicoes/ciuem.html, 20 de Outubro de 2022
7. De Macêdo, A. B. L., e Spínola, R. (s.d.). *Ciclos de Vida do Software—Conhecendo os Bastidores*.
8. Gil, A. C. (2008). *Métodos e técnicas de Pesquisa Social*, 6ª ed., Editora Atlas, S.A.
9. Gil, A. C. (2009). *Como elaborar projetos de pesquisa*, 4ª ed., Atlas.
10. Gorbatov, V. S., e Meshcheriakov, A. A. (2017). Secure software development training course. *Bezopasnost informacionnyh tehnology*, n.2, p.35-41.
11. *International Journal of Advance Research in Computer Science and Management Studies*. (2013), vol *1*, n.5, p.64-69.
12. *Introduction to Security Cyberspace, Cybercrime and Cybersecurity*. (s.d.).
13. Kostopoulos, G. K. (2017). *Cyberspace and cybersecurity*, 2ª ed., CRC Press, Taylor e Francis Group.

14. Lakatos, E. M., e Marconi, M. de A. (2003). *Fundamentos de metodologia científica*, 5ª ed., Atlas.
15. Lemos II, D. L. (2011). *Tecnologia da Informação*, 2ª ed., IF-SC.
16. Malaysia, C. (2020). *Guidelines for Secure Software Development Life Cycle (SSDLC)*. 1ª ed..
17. MICHELEVM. (2013). Modelo Prototipagem, retirado a 01 de Dezembro de 2022 de https://julianakolb.wordpress.com/2013/12/15/prototipagem/
18. Microsoft. (2010). *Simplified Implementation of the Microsoft SDL*.
19. Microsoft. (2012). *SDL Process Guidance Version 5.2*. http://www.microsoft.com/sdl
20. *Modelo cascata: O que é e por que está ultrapassado?* | *Trybe*. (s.d.). Retirado a 30 de Novembro de 2022, de https://blog.betrybe.com/tecnologia/modelo-cascata/
21. Nakagawa, E. Y. (2016). *Modelos de Processo de Software*.
22. OWASP. (2022). *SAMM* (Vol. 2).
23. *OWASP Top Ten* | *OWASP Foundation*. (s.d.). Retirado a 12 de Março de 2023, de https://owasp.org/www-project-top-ten/
24. *Quem Somos* | *CIUEM*. (s.d.). Retirado a 12 de Março de 2023, de https://www.ciuem.mz/quem-somos/
25. Rao, U. H., e Nayak, U. (2014). *The InfoSec Handbook: An Introduction to Information Security*. apress.
26. *Significado de Protótipo (O que é, Conceito e Definição)-Significados*. (s.d.). Retirado a 01 de Dezembro de 2022, de https://www.significados.com.br/prototipo/
27. Whitman, M. E., e Mattord, H. J. (2011). *Principles of Information Security*, 4ª ed., Cengage Learning.

Anexos

Anexo 1 - Lista de controle de actividades do modelo de ciclo de vida seguro de desenvolvimento de software proposto

Tabela A1-1: Lista de controle de actividades do modelo de ciclo de vida seguro de desenvolvimento de software proposto – Fonte: Autor

Actividade	**Tarefas**	**Responsável (eis)**	**Estado (✓/✗)**
Fase: Requisitos de segurança			
1. Identificar requisitos do trio CIA (*Confidentiality, Integrity, Availability*)	1.1. Identificar requisitos de Confidencialidade		
	1.2. Identificar requisitos de Integridade		
	1.3. Identificar requisitos de Disponibilidade		
2. Identificar requisitos do trio AAA (Autenticação, Autorização, Auditoria)	2.1. Identificar requisitos de Autenticação		
	2.2. Identificar requisitos de Autorização		
	2.3. Identificar requisitos de Auditoria		
3. Identificar requisitos gerais de segurança	3.1. Identificar requisitos de tratamento de erros e excepções		

	3.2. Identificar requisitos do ambiente de produção		
	3.3. Identificar requisitos de controle de sessões		
4. Criar um diagrama de casos de uso indevido	4.1. Identificar actores		
	4.2. Identificar casos de uso		
	4.3. Identificar ameaças		
	4.4. Avaliar ameaças e desenvolver contramedidas		
Fase: Design (de segurança)			
1. Realizar design do trio CIA (*Confidentiality, Integrity, Availability*)	1.1. Realizar design de Confidencialidade		
	1.1. Realizar design de Integridade		
	1.1. Realizar design de Disponibilidade		
2. Realizar design do trio AAA (Autenticação, Autorização, Auditoria)	2.1. Realizar design de Autenticação		
	2.2. Realizar design de Autorização		

	2.3. Realizar design de Auditoria		
3. Realizar design de segurança de aspectos gerais	3.1. Identificar linguagem de programação segura		
	3.2. Realizar desenho de interface segura		
	3.3. Identificar Sistema de Gestão de Base de Dados seguro		
	3.4. Identificar outras tecnologias seguras a utilizar durante o projecto		
Fase: Codificação (com segurança)			
1. Rever principais bases de dados de vulnerabilidades de aplicações	1.1. Rever principais conceitos e pesquisas sobre vulnerabilidades de aplicações		
2. Implementar codificação segura	2.1. Validação de entrada		
	2.2. Tratamento e registo de erros		
	2.3. Tratamento de excepções		

	2.4. Codificação de saída		
	2.5. Autenticação e gerenciamento de senhas		
	2.6. Gerenciamento de sessão		
	2.7. Controle de acesso		
	2.8. Práticas de criptografia		
	2.9. Protecção de dados		
	2.10. Segurança da comunicação		
	2.11. Configuração do sistema		
	2.12. Segurança do banco de dados		
	2.13. Gestão de arquivos		
	2.14. Gerenciamento de memória		
	2.15. Sanitização		
	2.16. Outras		

3. Realizar análise do código fonte	3.1. Realizar análise estática		
	3.2. Realizar análise dinâmica		
4. Proteger o código	4.1. Restringir acesso ao código		
	4.2. Implementar mecanismos de versionamento do código		
Fase: Teste (de segurança)			
1. Realizar análise de vulnerabilidades de segurança	1.1. Realizar varredura de vulnerabilidades na aplicação		
2. Realizar análise do código fonte	2.1. Realizar análise estática		
	2.2. Realizar análise dinâmica		
Fase: Implementação e Manutenção (com segurança)			
1. Realizar Implementação segura	1.1. Realizar configuração segura do servidor de produção		
	1.2. Realizar última revisão		

	1.3. Realizar última varredura de vulnerabilidades		
2. Realizar manutenção segura	2.1. Realizar backups		
	2.2. Realizar varredura de vulnerabilidades na aplicação		
	2.3. Realizar varredura de vulnerabilidades no servidor de produção		
	2.4. Instalar *patches*		
	2.3. Contratar serviço de teste de penetração		

Anexo 2 – Proposta de Ferramentas/Plataformas de segurança

Tabela A2-1: Proposta de Ferramentas/Plataformas de segurança – Fonte: Autor

Ferramenta/Plataforma	Descrição	Acesso
OWASP Academy	Plataforma de treinamento online que fornece recursos relacionados à segurança de aplicações web, como artigos, metodologias, documentação, ferramentas e tecnologias.	Gratuito
PortSwigger Academy	Plataforma de treinamento online que fornece recursos relacionados à segurança de aplicações web, como artigos, metodologias, documentação, ferramentas e tecnologias.	Gratuito
Synopsys	Plataforma que oferece diversos serviços de tecnologia entre eles treinamento em segurança de aplicações, com a possibilidade de ter aulas em directo ou mesmo gravadas	Pago
Snyk *Code*	Ferramenta de análise estática de código. Dentre muitas vantagens, esta ferramenta oferece integração com vários ambientes de desenvolvimento integrado, permite análise de repositórios na nuvem e possui um bom tempo de execução.	Gratuito/Pago

OWASP Top 10	Base de dados de vulnerabilidades de aplicações, desenvolvido pela OWASP	Gratuito
NVD	Base de dados de vulnerabilidades mantido pela NIST	Gratuito
CWE Top 25 Most Dangerous Software Weaknesses	Base de dados de vulnerabilidades de aplicações, mantida pela MITRE Corporation	Gratuito
OWASP Zap	Ferramenta utilizada para executar uma variedade de tarefas de teste de segurança, tais como interceptar e modificar o tráfego HTTP, verificar vulnerabilidades e atacar automaticamente aplicações web.	Gratuito
Burp Suite	Ferramenta que fornece um conjunto de funcionalidades para testar aplicações web, incluindo um servidor proxy, um scanner de aplicações web, um repetidor para manipular e reenviar pedidos, e muitas outras.	Gratuito/Pago
Nessus *Vulnerability Scanner*	Ferramenta de análise de vulnerabilidades que utiliza uma base de dados de vulnerabilidades conhecidas e sistemas de verificação destas vulnerabilidades, e também realiza uma série de testes para detectar erros de configuração, senhas fracas, e outras questões de segurança.	Gratuito/Pago

Acunetix	Ferramenta que fornece uma gama de características para digitalização e teste de aplicações web, incluindo um scanner para detecção de vulnerabilidades, um *crawler* para exploração automática de páginas web e um sistema de gestão de vulnerabilidades para rastrear e priorizar questões.	Pago

Anexo 3 – Exemplo de estrutura de organização de requisitos de segurança

Tabela A3-1: Exemplo de estrutura de organização de requisitos de segurança – Fonte: Malaysia (2020)

Tipo de segurança	Descrição do requisito	Comentários	Prioridade
Disponibilidade	O sistema de backup deve armazenar as fontes de recuperação num sistema de rede.	Para ajudar em caso de falha ou acção de intrusão	1
	O sistema deve fazer o espelhamento para permitir que os dados e o software estejam disponíveis em locais fisicamente separados (site separado quando a aplicação está na *Web*)	Ajuda a minimizar o risco de um único ponto de falha.	3
	O sistema deve aplicar uma solução de alta disponibilidade, tal como *clustering*.	Ajuda a minimizar o impacto de potenciais falhas do sistema.	3
Integridade	O sistema deve assegurar a coerência de todos os dados fornecidos pelo software (ou criar um novo e válido estado de dados, ou, devolver todos os dados ao seu estado antes de uma transacção ser iniciada).	Para evitar que uma pessoa ou sistema autorizado altere os dados inadvertidamente ou intencionalmente.	1
Auditoria	O sistema deve manter registos históricos (*logging*) de eventos e processos executados em ou por uma aplicação.	Definir *loggings* de segurança mais específicos para permitir recriar uma imagem clara dos eventos de segurança.	1

Anexo 4 – Exemplo de diagrama de casos de uso indevido

No diagrama de casos de uso indevido, procura-se ilustrar maneiras como o software pode ser utilizado de forma maliciosa ou indevida. O objectivo e de a partir destes casos de uso indevidos, se produzir requisitos que colmatem todas as ameaças.

Segundo Sindre (2000) citado por Chun Wei (2005), um caso de uso pode mitigar um caso de uso indevido (ele reduz as chances de ele acontecer), como também um caso de uso indevido pode ameaçar um caso de uso (ele pode explorar um caso de uso).

A seguir é possível visualizar uma ilustração de um diagrama de casos de uso indevido:

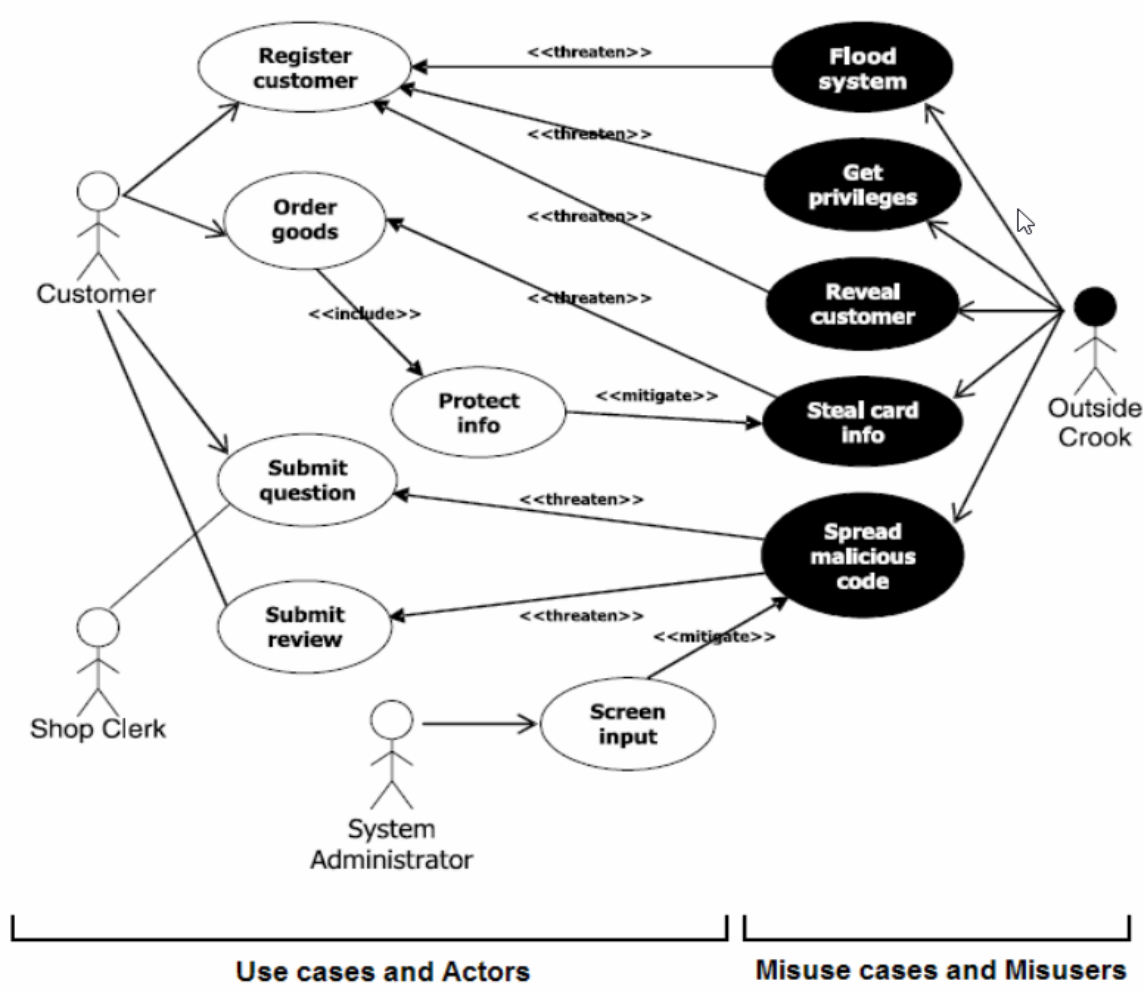

Figura A4-1: Exemplo de diagrama de casos de uso indevido – Fonte: Chun Wei (2005)

Anexo 5 – Resultado da varredura de vulnerabilidades nos servidores de aplicações do CIUEM

Por razões de protecção da imagem da instituição, foi recomendado ao autor omitir a identidade de cada servidor. Sendo assim, o autor nomeou todos servidores de forma uniforme, indicando, porém, em cada um deles um número inteiro designando a sua posição na contagem de representação. Os resultados a seguir foram tirados da ferramenta "*Nessus Vulnerability Scanner*", utilizada para realizar as varreduras.

Tabela A5-1: Resultado da varredura de vulnerabilidades nos servidores de aplicações do CIUEM – Fonte: Autor

Identificador	**Grau de criticidade**				
	Crítico	**Alto**	**Médio**	**Baixo**	**Nulo/Informativo**
Servidor 1	3	2	4	2	32
Servidor 2	1	9	6	1	22
Servidor 3	2	13	9	4	24
Servidor 4	2	13	8	4	23
Servidor 5	0	1	7	0	41
Servidor 6	2	13	8	4	25
Servidor 7	0	1	8	0	42
Servidor 8	0	1	4	0	23
Servidor 9	0	1	7	0	35
Servidor 10	0	1	4	0	25
Servidor 11	0	7	7	0	31
Servidor 12	0	0	2	0	28
Servidor 13	0	1	2	0	25

Anexo 6 – Guião de entrevista

Entrevista ao responsável do DDSA do CIUEM

1. Quantos sistemas estão sobre a tutela do CIUEM?
2. Existem sistemas críticos?
3. Se tiver respondido de forma positiva à questão anterior, quais são os sistemas críticos?
4. Como é gerida a questão da segurança dos sistemas já desenvolvidos e em desenvolvimento neste recinto?
5. Há alguma actividade/política envolvida com segurança dos sistemas?
6. Já houve registo de algum ataque cibernético?
7. Se tiver respondido de forma positiva a questão anterior, pode dar uma descrição sucinta do evento?
8. Há políticas de backups? Quais são?
9. Se tiver respondido de forma positiva a questão anterior, pode descrever as políticas
10. Aplica-se algum mecanismo de teste de segurança durante o ciclo de desenvolvimento dos softwares?