Departamento de Ciências e Tecnologias da Informação

Análise de Aplicações no Sector Financeiro
Vulnerabilidades e Mitigações

Tiago Miguel Paulino Vieira

Dissertação submetida como requisito parcial para a obtenção do Grau de

Mestre em Informática e Gestão

Orientador:
Prof. Doutor Carlos José Corredoura Serrão,
Escola de Tecnologia e Arquitetura ISCTE-IUL/ISTAR-IUL

Coorientador:
Mestre David Burgo Baptista

Setembro, 2016

Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

Análise de Aplicações no Sector Financeiro
Vulnerabilidades e Mitigações

Tiago Miguel Paulino Vieira

Dissertação submetida como requisito parcial para a obtenção do Grau de

Mestre em Informática e Gestão

Orientador:
Prof. Doutor Carlos José Corredoura Serrão,
Escola de Tecnologia e Arquitetura ISCTE-IUL/ISTAR-IUL

Coorientador:
Mestre David Burgo Baptista

Setembro, 2016

“Onde houver soberba, aí haverá também ignomínia;

Onde houver humildade haverá sabedoria.”

Provérbio

Autor: desconhecido

Resumo

A segurança de tecnologias de informação é actualmente uma preocupação crescente para indivíduos e organizações. Esta preocupação é ainda maior no sector financeiro, não só pelas quantias monetárias envolvidas, mas também pela informação sensível e privada de clientes e organizações. Como meio de aferir segurança de infraestruturas, redes, aplicações *web* e muitos outros activos tangíveis ou intangíveis, organizações estão a investir em testes que simulam o comportamento de um atacante malicioso mas, num ambiente controlado para identificar as suas próprias vulnerabilidades.

Esta dissertação foca-se na análise dos resultados de auditorias de segurança conduzidos em diversas aplicações financeiras de uma entidade reguladora do sector com auxílio de ferramentas automáticas para avaliar o seu nível de maturidade. Para alcançar esta análise e classificação de risco com testes de intrusão, o processo foi idealizado de raiz desde a definição das "*rules of engagement"*, escolha da abordagem de testes, seleção de *web scanners*, realização de testes de intrusão, confirmação de vulnerabilidades e classificação das mesmas.

Para registo e apresentação de resultados da auditoria de segurança e priorização de intervenções, foi criado um painel de instrumentos para acompanhamento das vulnerabilidades encontradas em qualquer auditoria de segurança categorizadas com a classificação de risco CVSS v3.0 e capaz de gerar relatórios – o resultado formal de qualquer auditoria baseada em testes de intrusão.

Com base nesta auditoria, pode ser estabelecida uma relação com outras entidades reguladores no sector acerca das vulnerabilidades emergentes cujas aplicações *web* se baseiam nas mesmas tecnologias. Para concluir, as vulnerabilidades encontradas foram mapeadas contra os sistemas de vulnerabilidades mais comuns e severas como o OWASP Top 10 e identificado o impacto no regulador e entidades reguladas.

Palavras-chave: testes intrusão, segurança web, vulnerabilidade, setor financeiro, análise de risco, painel de instrumentos, CVSS v3.0.

Abstract

Information security is today an increasing concern for individuals and organizations. This concern is even greater in the finance sector, not only because the financial amount involved but also clients and organization's private and sensitive information. As a way to test security in infrastructures, networks, deployed web applications and many other important tangible and intangible assets, organizations have been performing penetration testing which simulates an attacker's behavior in a controlled environment in order to identify its own vulnerabilities.

This work focus on the analysis of the results of security audits conducted on several financial web applications from one market regulator institution with aid of automatic tools in order to assess their web applications maturity security level. In order to accomplish this security analysis and risk classification with penetration testing, the full process was built from scratch - from the defining the *rules of engagement*, choosing the approach, selecting the web scanners, conducting the penetration testing, confirming vulnerabilities and identifying mitigations, and classifying vulnerabilities.

In order to register and show results of the security audit and for mitigation prioritize, a web dashboard was developed to keep track of vulnerabilities found in any security audit categorized with CVSS v3.0 and able to build reports - the formal result of any pentest-based security audit.

Based on this security audit, a correlation with other regulator entities can be made and what vulnerabilities emerge in the finance sector web applications that are based on the same development technologies. On top of that, the vulnerabilities discovered were mapped against well-known and well-adopted security risks ranking methods, such as the OWASP Top 10 and measure the impact in the regulator and regulated entities.

Keywords: web security; finance sector; pentesting; penetration testing; vulnerability; risk analysis; *dashboard*, CVSS v3.0.

Agradecimentos

Ao meu professor Carlos Serrão, pela dedicação, apoio e orientação durante esta viagem. Agradeço o seu espírito crítico, o seu acompanhamento e iniciativas no sentido de promover a partilha de conhecimento e *networking* entre alunos e profissionais da área. O que dá a este trabalho ainda mais sentido e relevância. Obrigado por este contributo e esforço para realização da dissertação.

Ao meu professor David Baptista, pelo tempo e esforço despendidos para que esta auditoria pudesse ser realizada. Pelo apoio na aprendizagem e acompanhamento durante todo processo. Pelo interesse e entusiasmo partilhados para que pudesse crescer como profissional na área.

A todas as entidades que participaram e autorizaram a realização desta auditoria e logística necessária. Pela confiança depositada para que ambos pudéssemos crescer, como aluno e como instituição.

Ao Nuno Sequeira. pela receptividade e colaboração na realização da auditoria independentemente da gestão adicional e constrangimentos inerentes. Sem o teu interesse, seria um pouco mais complicado atingir esta abrangência.

A Teresa Alves pela força e apoio inicial, alguém muito especial que me ajudou a manter as forças até ao final.

À família e amigos pela paciência durante esta pequena ausência em corpo. Pelo apoio e motivação constantes.

Glossário

OWASP: *Open Web Application Security Project*. Entidade sem fins lucrativos formada nos Estados Unidos da América dedicada a ajudar organizações a conceber, desenvolver, utilizar e manter aplicações *web* seguras.

Hacker: especialista informático que usa o seu conhecimento aliado a ferramentas automáticas para comprometer um sistema informático ou aceder a informação sensível.

Ethical Hacker: profissional de segurança formado em técnicas de exploração de sistemas com o mesmo conhecimento que um *hacker* malicioso mas, que usa essas técnicas para identificar vulnerabilidades de acordo com a lei e sempre autorizado pela entidade auditada.

HTTP: protocolo de comunicação entre sistemas distribuídos. Um dos protocolos mais utilizados na internet e aplicações *web*. Definido por métodos que caracterizam a acção que se pretende realizar como obtenção de recursos alteração ou alteração dos mesmos.

GET: método do protocolo HTTP para obtenção de recursos, por exemplo, uma página *web*.

POST: método do protocolo HTTP para submissão de informação. Em aplicações web, está tipicamente associado à submissão de valores num formulário de preenchimento de dados.

***Fuzzing*:** técnica de testes de software, frequentemente automatizada ou semi-automatizada, que consiste em enviar dados inválidos ou aleatórios como input de uma aplicação ou sistema para identificação de falhas.

Phishing: tentativa de fraude onde um individuo mal-intencionado tenta obter informação sensível como credenciais ou informações bancárias apresentando-se como uma entidade confiável através de email ou outros canais de comunicação.

Risco: uma medida do grau em que uma entidade está ameaçada por uma potencial circunstância ou evento e, normalmente, com impacto adverso e uma probabilidade de ocorrência.

Avaliação de Risco: processo de identificação de riscos operacionais para uma organização, activos, indivíduos ou outras organizações provenientes da utilização de sistemas de informação.

Vulnerabilidade: fraqueza num sistema de informação, procedimento de segurança, controlo ou implementação que pode ser explorada por uma ameaça a fim de comprometer o sistema.

Índice

Índice de Tabelas

Índice de Figuras

Introdução

O tema da segurança informática preocupa cada vez mais as pessoas e as organizações que estão ligadas e necessitam das tecnologias de informação e de comunicação (TIC) para poderem operar normalmente (Razzaq, Hur, Haider, & Ahmad, 2009). As instituições, cada vez mais, procuram testar as defesas dos seus sistemas simulando o comportamento do atacante e, através de um processo continuado e planeado, violar as premissas de segurança das aplicações, nomeadamente a confidencialidade e/ou integridade dos dados, e também a sua disponibilidade (Walker, 2012, p. 16). Assim sendo, várias empresas dedicam-se neste momento à segurança informática e vendem serviços de consultoria baseados em testes de penetração e até mesmo formação de profissionais na área, como por exemplo a *EC-Council* (EC Council, 2015) e a *Offensive Security* (Offensive Security Training, Certifications and Services, 2015). Este trabalho irá ter como base alguns dos guias e recomendações para a realização de testes de intrusão destas (e de outras) instituições para deteção e mitigação de vulnerabilidades em aplicações *Web* do sector financeiro, recorrendo a um caso de estudo específico.

Os testes de intrusão (ou penetração) são na realidade simulações de ataques informáticos por parte de entidades certificadas e/ou autorizadas pela própria instituição para validar as capacidades de resiliência a ataques de segurança que essa mesma instituição possui (Buchler, Oudinet, & Pretschner, 2012). Este tipo de testes utiliza o conhecimento e o engenho de um profissional de segurança para explorar diversos vetores de ataque sem fins maliciosos, na tentativa de aceder a aplicações e à sua informação, identificando a possibilidade de extravio ou danificação da mesma (Bozic & Wotawa, 2015).

1.1 Descrição do problema

O principal objetivo deste trabalho é realização de uma auditoria de segurança a diversas aplicações *web* representativas do sector financeiro, baseando-se numa instituição reguladora do sector, identificar e verificar que vulnerabilidades comprometem a confidencialidade, integridade e/ou disponibilidade e que mitigações podem ser aplicadas. O âmbito deste estudo estará circunscrito às aplicações *web* que estejam disponíveis para entidades externas (consequentemente mais expostas a ataques), recorrendo ao apoio de diversas ferramentas automáticas. Desta forma, torna-se necessário proceder posteriormente à avaliação manual de

quaisquer vulnerabilidades encontradas com o intuito de confirmar se representam ameaças reais, se podem ser exploradas, qual o risco que apresentam para a organização (reputacional, técnico e financeiro) e quais as mitigações possíveis (Dukes, Yuan, & Akowuah, 2013).

Este trabalho será realizado com recurso a múltiplas ferramentas automáticas de forma a garantir a melhor cobertura possível dos testes para que sejam identificadas o maior número possível de ameaças e vulnerabilidades. Este é um aspeto muito importante, pois oferece um maior grau de cobertura dos testes, uma vez que os resultados que diferentes ferramentas podem apresentar, podem ser igualmente distintos. Não obstante, as vulnerabilidades encontradas pelas aplicações devem ser sempre validadas manualmente de forma a confirmar até que ponto comprometem a confidencialidade, integridade e disponibilidade do sistema (Dukes, Yuan, & Akowuah, 2013).

Para resolver o problema apresentado, é importante considerar os seguintes objetivos:

1. Efetuar um levantamento sobre as principais ferramentas *open source* automáticas de testes de intrusão aplicacional, efetuar uma comparação das mesmas, e selecionar as mais apropriadas;
2. Definir a metodologia de testes de intrusão e definir a abordagem de utilização das ferramentas e os testes a realizar (definir as "*rules of engagement"*) (Walker, 2012, p. 27);
3. Identificar e selecionar aplicações web financeiras que vão ser alvo dos testes de intrusão, validar os resultados obtidos manualmente para confirmar as vulnerabilidades detetadas;
4. Realizar uma análise de risco de acordo com métricas padrão, e recomendar ações de mitigação para lidar com esse mesmo risco;
5. Criar um *dashboard web* que permita visualizar rapidamente as principais vulnerabilidades encontradas, o nível de risco e as mitigações (McGraw, Migues, & West, 2015), (OSSTMM v3.0 - The Open Source Security Testing Methodology Manual, 2010, p. 11).

1.2 Motivação

A motivação do autor para escolha deste tema é pessoal e profissional. A motivação pessoal deve-se à curiosidade, fascínio e preocupação pela temática de segurança informática. A nível profissional, a realização de ataques informáticos e testes de vulnerabilidades num cenário real segundo a perspetiva de um atacante permitirá compreender melhor as preocupações de segurança nos sistemas de informação e as competências profissionais adquiridas serão uma mais-valia na carreira profissional do autor permitindo no futuro desenvolver sistemas mais seguros e robustos.

Será também possível contribuir para uma melhoria efectiva da segurança por parte da instituição financeira alvo após realização dos testes de intrusão e até sensibilizar a instituição na questão de testes de intrusão para além das já existentes preocupações de segurança. Nomeadamente, influenciar o ciclo de desenvolvimento de software com a inclusão da preocupação de segurança a nível de desenvolvimento desde cedo na instituição (Teodoro & Serrão, 2011).

1.3 Contribuição

Relativamente a relevância social ou contribuição, este estudo tem como objetivo ajudar uma instituição financeira de renome, representativa do sector e com grande visibilidade em Portugal e na Europa a proteger os seus dados e sistemas ou mitigar esses problemas, no contexto da segurança aplicacional. As vulnerabilidades de segurança encontradas e as mitigações sugeridas podem ser analisadas por qualquer instituição do mesmo sector que recorra a sistemas e aplicações semelhantes.

Relativamente a contributos científicos, este trabalho permitirá reunir informação sobre iniciativas no domínio da segurança de software e boas práticas para atingir esse propósito através da análise de entidades promotoras de segurança de sistemas de informação e quais ferramentas e guias recomendam.

A análise e recolha de resultados provenientes das ferramentas utilizadas (*web application scanners*) possibilitará avaliar, dentro das escolhidas, a sua precisão e avaliar as que melhor potenciam identificação de vulnerabilidades.

Através da recolha de características de segurança de software em aplicações *web*, guias de testes de intrusão e ferramentas de testes automáticos, será composta uma metodologia baseada

em padrões internacionais, flexível e metódica que poderá ser reutilizada noutras aplicações ou contextos com as mesmas características tecnológicas.

Uma outra importante contribuição deste trabalho, é o desenvolvimento de um *dashboard web*, que com base nos testes realizados, permitindo à organização e aos seus decisores ter uma visão clara e sucinta sobre o nível de risco aplicacional num determinado momento. Sugerir acções de mitigação e criando um repositório para análise de vulnerabilidades passadas para desenvolvimentos futuros. Contribuindo de forma efectiva para gestão, controlo e priorização de acções de mitigação.

Capítulo 1 Estado da Arte

Esta secção apresenta o estado da arte atual no que concerne a testes de intrusão a aplicações *web*, metodologias existentes, ferramentas automáticas de testes e tratamento de resultados. Estes elementos, em cooperação, são uma forma eficaz de auditar o estado de segurança de uma aplicação *web*. Cada elemento tem disponível uma panóplia de soluções e nenhuma deve ser descartada.

Uma aplicação *web* é suportada por uma arquitetura cliente/servidor (Client Server Architecture, 2016), constituída por uma máquina central onde o processamento é realizado e clientes que realizam pedidos ao servidor. Numa aplicação *web*, os clientes apenas precisam de um *browser web* para realizar pedidos ao servidor, este processa a resposta e envia-a para o cliente. A lógica aplicacional é realizada no servidor e o cliente não necessita de *software* adicional além do browser (à excepção de extensões adicionais que possam ser usados para visualizar determinados tipos de conteúdo). Uma aplicação *web*, pode estar disponível na internet ou numa rede privada, por exemplo, uma intranet (Client Server Architecture, 2016).

É uma vantagem tanto para as organizações como para os utilizadores porque, a instalação é centralizada numa única máquina reduzindo o número de licenças de *software* e o utilizador não precisa da instalação de qualquer *software* para além do seu *browser*. Os testes de intrusão aplicam-se seja em que condições a aplicação estiver disponível – na intranet ou publicamente na internet.

1.1 Pen Testing / Testes de Intrusão

Pen(netration) testing ou testes de intrusão, é uma linha de defesa de aplicações e sistemas informáticos que consiste em realizar tentativas de exploração de vulnerabilidades numa aplicação, rede ou sistema informático (Hasan & Sajib, 2015). Nos testes de intrusão, simula-se o comportamento de um atacante (vulgarmente designado por *hacker*) num ambiente controlado de forma a identificar e mitigar possíveis vulnerabilidades existentes. Um *hacker* pode ser um especialista em segurança de informação, uma pessoa com conhecimentos na área das novas tecnologias ou alguém com acesso a ferramentas informáticas capazes de

comprometer outros sistemas com intenções maliciosas e que tentam de alguma forma, explorar as falhas e funcionalidades de um sistema informático (Buchler, Oudinet, & Pretschner, 2012). *Pen testing* é mais que apenas detetar vulnerabilidades, é o processo de verificar se as mesmas podem ser exploradas (Mullins, 2005).

Uma aplicação *web* disponível através da internet está exposta ao mundo e portanto, aberta a indivíduos mal intencionados com ferramentas automáticas que podem tentar identificar e tomar partido de vulnerabilidades. Em aplicações com informação confidencial, é ainda mais crítico do que para outro tipo de *software* que o seu nível de maturidade de segurança seja testado (OWASP Testing Guide v4, 2014).

À disposição do auditor de segurança estão vários guias e ferramentas como por exemplo OWASP *Testing Guide* (OWASP Testing Guide v4, 2014) e *Certified Ethical Hacking* (EC Council, 2015). Estas ferramentas abordam comportamentos de segurança, ferramentas de testes e boas práticas de testes de intrusão. Algumas estão focadas exclusivamente a aplicações *web* enquanto outras cobrem várias áreas dentro da segurança.

Existem três tipos de *pen testing*: *white box*, *grey box* e *black box*. As suas diferenças são em termos de acesso e conhecimento do sistema que estão a testar. Um teste *black box* simula por completo o comportamento de um atacante externo sem qualquer conhecimento sobre o sistema e rede. Um ataque *grey box*, dos mais comuns para *pen testing*, dão ao especialista conhecimento da rede, máquinas e aplicações para teste. Um ataque *white box* sugere que o especialista tenha conhecimento do sistema, rede e acesso ao código fonte (Tung, Tseng, Shih, & Shan, 2014). Em termos empresariais, *black box* faz menos sentido devido ao tempo que se perde em reconhecimento, isto é, reunir informação sobre a empresa. Outra situação que as instituições não têm interesse nos testes de intrusão, é que um auditor em cenário *black box*, teste máquinas ou aplicações que não têm significado para a instituição, seja por não conterem ativos valiosos (OWASP Top 10, 2013, p. 21), seja por opção da instituição. É comum a instituição limitar a gama de endereços IP a testar.

1.1.1 Técnicas existentes

Pen testing é apenas uma de várias técnicas de testes de intrusão existentes (OWASP Testing Guide v4, 2014), outras abordagens em diferentes momentos no ciclo de desenvolvimento de software podem ser adotadas entre as quais:

- Inspeções Manuais;
- Modelação de Ameaças;
- Revisão de Código.

As inspeções manuais são procedimentos que testam pessoas, políticas de segurança e processos (OWASP Testing Guide v4, 2014, p. 13). Podem ainda incluir análise tecnológica e de arquitetura do desenvolvimento de *software*. São realizadas através de análise de documentação e entrevistas com os analistas de sistemas. É um processo simples e flexível para identificar potenciais pontos vulneráveis. Este tipo de iniciativa permite controlar se as pessoas envolvidas no desenvolvimento de uma aplicação estão cientes das políticas de segurança e se adotaram alguma abordagem suspeita. Uma inspeção manual, aplica-se cedo no ciclo de vida de desenvolvimento de *software* (*Software Development Life Cycle*, SDLC) e não requer nenhuma tecnologia específica. As desvantagens desta técnica são o elevado tempo de execução e requerem suporte por parte da equipa de desenvolvimento (OWASP Testing Guide v4, 2014, p. 13).

A modelação de ameaças consiste no trabalho desenvolvido pelos arquitetos de sistema dum projeto, de analisar e identificar ameaças que os seus sistemas e aplicações enfrentam. É recomendado pela OWASP que seja aplicado desde cedo no SDLC indicando a referência à norma NIST 800-30 (NIST 800-30 Risk Management Guide for Information Technology Systems, 2002) que envolve decompor a aplicação em funcionalidades, classificar os ativos como tangíveis ou intangíveis e em termos da sua importância, desenvolver cenários de ataque de teste e criar estratégias de mitigação. A OWASP também tem disponível no *Code Review Guide* (OWASP Code Review V2 Project, 2008) uma metodologia para modelação de ameaças. Tal como os testes de intrusão, também a modelação de ameaças age como uma visão do atacante mais prática, mas um bom modelo de ameaças não significa necessariamente *software* seguro (OWASP Testing Guide v4, 2014, p. 13).

A revisão de código é um processo de verificação do código fonte das aplicações para questões de segurança (Stuttard & Pinto, 2011, p. 701). Esta verificação permite encontrar certos tipos

de vulnerabilidades que, por exemplo, os testes de intrusão não conseguem. Os tipos de problemas mais facilmente diagnosticados são lógicas de negócio mal implementadas, *backdoors* e configurações incorretas. Este processo pode ser muito rápido e eficaz, mas requer conhecimento da linguagem de programação utilizada e dos paradigmas assim como da lógica do negócio. Além do mais, a aplicação disponível aos utilizadores reais pode não ser a versão que é testada. Quando os profissionais de segurança não têm acesso ao código fonte da aplicação, devem então recorrer a testes de intrusão (Curphey & Arawo, 2006).

1.2 Ameaças e Ataques

De forma simplista, podemos classificar uma ameaça como interna ou externa. A ameaça responde à pergunta, "quem" realiza um ataque, e que pode variar entre um funcionário de uma empresa até uma avaria nos ares condicionados da sala dos servidores (Wheeler, 2011). Um ataque é a uma tentativa de exploração de uma vulnerabilidade, por exemplo, através do envio de *phishing*.

Um dos projetos mais reconhecidos para vulnerabilidades de software (em particular, para aplicações web) na indústria é o OWASP Top 10 (OWASP Top 10, 2013). Identifica e explica os riscos mais comuns a nível de aplicações *web.* Esta organização, aberta e independente, tem como objetivo aumentar a preocupação de segurança de instituições que recorrem a sistemas de informação para prestar, auxiliar ou realizar os seus serviços quanto aos riscos mais críticos e serve para educar os programadores, analistas, arquitectos e gestores quando às consequências de fraquezas existentes nas aplicações.

Este manual foi inicialmente lançado em 2003 com pequenas correções em 2004 e 2007. Uma segunda versão surgiu em 2010 com a premissa que, as vulnerabilidades passavam agora a ser priorizadas por risco. O mesmo acontece com a última versão de 2013. Algumas das vantagens mais importantes deste projeto são a sua independência de qualquer instituição e pressão comercial que a isenta de obtenção de lucro. Além disso, o facto da OWASP interligar os seus vários projetos de forma a criar valor com a experiência de cada um deles beneficia os profissionais de segurança que adotem estas práticas (OWASP Top 10, 2013).

Estes riscos identificados no OWASP Top 10, são os mais comuns, no entanto não devem ser a única preocupação de uma análise e teste de vulnerabilidades (Wheeler, 2011). Isto é especialmente relevante numa área complexa e em constante evolução em que novas falhas vão sendo descobertas assim como novos vetores de ataque. A OWASP recomenda o uso de outros

projetos como OWASP *Developers Guide* (OWASP Developers Guide, 2014), OWASP *Code Review Guide* (OWASP Code Review V2 Project, 2008), *Application Security Verification Standard* (OWASP Application Security Verification Standard Project, 2015) e *Testing Guide* (OWASP Testing Guide V3, 2008).

O Top 10 de 2013 identifica os seguintes riscos ordenados pela prevalência do risco e a probabilidade de ocorrência do mesmo:

A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object Reference
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using known Vulnerable Components
A10 – Unvalidated Redirects and Forwards

Este modelo teve como origem 8 conjuntos de fontes de dados fornecidos por 7 firmas especializadas em segurança de aplicações e faz parte dum esforço da OWASP de acompanhar as tendências atacantes de forma a desenvolver e orientar os seus projectos às preocupações de segurança emergentes na indústria.

Em termos de alterações face à versão anterior e justificando a evolução constante da área de segurança, a "A2 – Broken Authentication and Session Management" subiu para a posição 2. Segundo a OWASP, esta subida deve-se possivelmente ao investimento nesta vulnerabilidade por parte de atacantes. Isto causou a descida do risco A2 pelo "A3 – Cross-Site Scripting (XSS)" na versão de 2013. A OWASP crê que a descida do risco A3 deve-se ao facto de ser um tema bastante desenvolvido por várias *frameworks* e portanto, está mais maduro. O risco de 2010 "A8 – Failure to Restrict URL Access" evoluiu para "A7 – Missing Function Level Access Control" e cobre agora vários níveis de acessos de controlo, não apenas URL.

Os riscos "A7 – Insecure Cryptographic Storage" e "A9 – Insufficient Transport Layer Protection" fundiram-se numa só denominada "A6 – Sensitive Data Exposure", inclui riscos de informação sensível no cliente e cobre a segurança de informação na comunicação entre cliente

e aplicação. A última alteração face à versão anterior, é a inclusão de "A9 – Using known Vulnerable Components", já estava presente anteriormente, mas face ao crescimento de risco, tem agora uma categoria própria.

O OWASP Top 10 é um manual com a definição de cada risco onde é possível encontrar como mitigá-lo assim como referências a outros projetos e sua bibliografia (OWASP Top 10, 2013).

A *Web Application Security Consortium* (WASC) (Web Application Security Consortium, 2014), composta por um grupo de especialistas, praticantes da indústria dedicados ao desenvolvimento de *standards* de segurança *open source* apresenta também uma lista onde enumera possíveis ataques e fraquezas de uma aplicação *web* num documento que corresponde à sua segunda versão - o WASC *Threat Classification* (WASC Threat Classification, 2010). Apesar da sua última atualização datar de 2010, é uma referência na área e é constituído por uma descrição individual de cada um dos ataques. Na Figura 1.1 podemos observar a listagem das ameaças e fraquezas.

Um outro exemplo de classificação de ataques mas, numa perspetiva mais abrangente, é a CAPEC View (CAPEC View: Domain Attacks, 2015) que além de ataques *web*, acrescenta outros possíveis vetores organizados por domínio (Engenharia Social, Comunicações, Software, etc.). Em termos de aplicações *web*, não acrescenta outros ataques para além dos que os que foram mencionados pela WASC mas, é também um referência com vários exemplos. Esta informação está associada à MITRE (MITRE, 2015), uma organização sem fins lucrativos mas, com suporte do governo dos EUA. Uma das suas áreas nucleares é a segurança de tecnologias de informação.

Ataques	Fraquezas
Abuse of Functionality	*Application Misconfiguration*
Brute Force	*Directory Indexing*
Buffer Overflow	*Improper Filesystem Permissions*
Content Spoofing	*Improper Input Handling*
Credential/Session Prediction	*Improper Output Handling*
Cross-Site Scripting	*Information Leakage*
Cross-Site Request Forgery	*Insecure Indexing*
Denial of Service	*Insufficient Anti-automation*
Fingerprinting	*Insufficient Authentication*
Format String	*Insufficient Authorization*
HTTP Response Smuggling	*Insufficient Password Recovery*
HTTP Response Splitting	*Insufficient Process Validation*
HTTP Request Smuggling	*Insufficient Session Expiration*
HTTP Request Splitting	*Insufficient Transport Layer Protection*
Integer Overflows	*Server Misconfiguration*
LDAP Injection	
Mail Command Injection	
Null Byte Injection	
OS Commanding	
Path Traversal	
Predictable Resource Location	
Remote File Inclusion (RFI)	
Routing Detour	
Session Fixation	
SOAP Array Abuse	
SSI Injection	
SQL Injection	
URL Redirector Abuse	
XPath Injection	
XML Attribute Blowup	
XML External Entities	
XML Entity Expansion	
XML Injection	
XQuery Injection	

Figura 1.1 – Classificação WASC de ataques e fraquezas (WASC Threat Classification, 2010, p. 21)

1.3 *Frameworks* de Testes de intrusão

Existem à disposição *frameworks* de testes de intrusão que consistem num conjunto de boas práticas e ferramentas utilizadas de forma metódica para realização da auditoria de segurança sendo da competência da equipa de segurança a escolha do modelo mais adequado ao seu trabalho. O uso de uma *framework* ajuda a organização a definir os passos a realizar para gerir um programa de testes a aplicações web e ajuda as instituições a preparem-se para uma auditoria, se aplicável (OWASP Testing Guide v4, 2014, p. 7), (The CIS Critical Security Controlls for Effective Cyber Defense version 6.0, 2015), (Austin & Williams, 2011).

Estes guias, ferramentas estão disponíveis quer em projectos *open source*, quer em certificações de instituições dedicadas à segurança de informação. Eis as *frameworks* com maior destaque:

- Testing Guide (OWASP) (OWASP Testing Guide v4, 2014);
- Certified Ethical Hacking (EC Council, 2015);
- Offensive Security Certified Professional (Offensive Security Training, Certifications and Services, 2015);
- Technical Guide to Information Security Testing and Assessment (NIST SP 800-115 Technical Guide to Information Security Testing And Assessment, 2008);
- Open Source Testing Methodology Manual (OSSTMM) (OSSTMM, 2015);
- Penetration Testing Execution Standard (PTES, 2014);
- SANS (SANS, 2015).

1.3.1 OWASP Testing Guide

Um dos manuais de testes de intrusão *web* mais reconhecidos é o da OWASP, o Testing Guide versão 4 (OWASP Testing Guide v4, 2014), que substituiu a versão 3 (OWASP Testing Guide V3, 2008) no início de 2015. Este guia normaliza o processo de análise e teste de uma aplicação de forma a garantir um processo metódico e uma avaliação de segurança mais fiável. Este guia integra com duas *frameworks* da OWASP, o *Developers Guide* (OWASP Developers Guide, 2014) e o *Code Review Guide* (OWASP Code Review V2 Project, 2008) para padrões de desenvolvimento e revisão de código respetivamente. Além disso, a OWASP possui um portfólio de outros projetos, *web scanners* e documentação mantida por inúmeros profissionais da área de testes de intrusão *web*, factores que combinados, alavancam o potencial desta metodologia (OWASP Testing Guide v4, 2014). A última versão do *Testing Guide* inclui novos capítulos incluindo, por exemplo, o de criptografia e aumenta o número de cenários de exemplo.

Esta metodologia é *open source* e exclusivamente dedicada para a segurança *web* (OWASP Testing Guide v4, 2014).

1.3.2 Certified Ethical Hacking

A *Certified Ethical Hacking* (CEH) é uma certificação da *EC Council* (EC Council, 2015) que introduz um guia e ferramentas de teste de intrusão. A CEH cobre em exaustão diversas áreas da segurança desde criptografia, segurança *web*, DoS, entre muitos outros. A versão mais recente data a fim de 2015 e aumenta para 18 o número de domínios de conhecimento. A *EC Council* tem ao serviço esta e outras formações de segurança e garante aos certificados um elevado nível de conhecimento (EC Council, 2015).

Esta nova versão incrementa novos laboratórios de teste para um total de 140 assim como detalhe de vulnerabilidades recentes. Além disso, tem agora capítulos sobre segurança de computação na nuvem e plataformas móveis como *tablets*. O acesso a esta metodologia é pago, um profissional de segurança ou iniciado pode propor-se a formação ou realizar um exame de certificação mediante prova de exercício na área (EC Council, 2015).

1.3.3 Offensive Security Certified Professional

A par com a EC Council, está a Offensive Security (Offensive Security Training, Certifications and Services, 2015) com a formação Offensive Security Certified Professional (OSCP). Para além da certificação, a Offensive Security é responsável por várias ferramentas de testes de intrusão utilizadas e reconhecidas por outras instituições metodologias (Malhotra, 2014). A distribuição de Linux, Kali Linux (Kali Linux, 2015), configurada de raiz com um vasto leque de ferramentas e aplicações para testes de intrusão incluindo *web scanners* e é um projeto da *Offensive Security*. *Exploit Database* (Exploit Database, 2015), *Metasploit* (Metasploit, 2015) são outros exemplos de ferramentas geridas pela *Offensive Security*.

1.3.4 Technical Guide to Information Security Testing and Assessment

O *National Institute of Standards and Technology* (NIST) é desde há muito tempo responsável por desenvolver normas de indústria na área de segurança. Entre os vários projetos encontra-se o *Technical Guide to Information Security Testing and Assessment* (NIST SP 800-115 Technical Guide to Information Security Testing And Assessment, 2008) cujas diretrizes de segurança foram definidas para uso por parte das instituições federais dos EUA (NIST.org, 2015). Estas normas não estão sujeitas a direitos de autor (NIST.org, 2015).

O objetivo do documento é munir as instituições de guias para planear e conduzir avaliações e testes de segurança para que possam ser aplicadas mitigações. Contudo e apesar da NIST se manter ativamente em desenvolvimento, a sua metodologia não tem revisões desde 2008 (Prandini & Ramillo, 2010), (NIST.org, 2015), (Zitta, Marik, & Neradova, 2014). Foi um precursor nas questões de auditoria de segurança e continua a ser hoje uma referência em boas práticas graças a outras publicações na área (Prandini & Ramillo, 2010).

1.3.5 Open Source Testing Methodology Manual

Com início de produção em 2001 pelo *Institute for Security and Open Methodologies* (ISECOM), encontra-se o *Open Source Security Testing Methodology Manual* (OSSTMM) (OSSTMM, 2015). O ISECOM é uma instituição sem fins lucrativos com escritórios em Barcelona, Espanha e Nova Iorque, EUA.

A sua fonte de financiamento é alcançada através de formações, parcerias, subscrições, licenciamento, seminários e projetos de investigação privados (OSSTMM, 2015). Relativamente ao OSSTMM, é um contributo para o teste e implementação de segurança. Apesar do ISECOM ter diversas formações disponíveis, o seu manual de testes de intrusão é gratuito com a ressalva que, as últimas versões/atualizações estão disponíveis apenas para membros (OSSTMM, 2015).

A acrescentar aos vários conteúdos das ISECOM mas ainda em versão *draft* e para membros apenas, está a OSSTMM *Web App Draft* (OSSTMM, 2015). Este guia é exclusivo para testes de segurança web e que certamente aumentará o contributo desta instituição na vertente web (OSSTMM, 2015).

1.3.6 Penetration Testing Execution Standard

O *Penetration Testing Execution Standard* (PTES) é uma referência recente como metodologia de testes de intrusão. Um dos objetivos mais relevantes do PTES é fornecer aos potenciais clientes um termo de comparação para determinar a qualidade dos testes de intrusão (Diniz & Serrão, 2014). Esta metodologia cobre o vasto processo de testes em qualquer vertente desde *engagement*, recolha de informação, modelação de ameaças, análise de vulnerabilidades, ataques e criação de relatórios (PTES, 2014). Contudo, não tem especificações técnicas quanto à execução do teste *per si*, essa responsabilidade é complementada com o *PTES Technical Guidelines* (PTES Technical Guidelines, 2012). Ambos os documentos são livres e gratuitos.

1.3.7 SANS

A *System Administration, Networking and Security* (SANS) é outra organização dedicada à certificação e divulgação de segurança (SANS, 2015). Relativamente a segurança web e *pen testing*, disponibilizam a formação *Network Penetration Testing and Ethical Hacking* (SANS Network Penetration Testing Ethical Hacking, 2015) e o *Web Application Penetration Tester and Ethical Hacking* (SANS Web App Penetration Testing Ethical Hacking, 2015), cada curso com uma certificação associada.

Disponível para a comunidade em geral sem encargos, encontra-se o documento *Critical Security Controls* (CIS) (CIS Critical Security Controls, 2015) que reúne 20 ações ou controlos para defesa dos ataques atuais mais perigosos. Um dos controlos recomendados é a realização de testes de intrusão e exercícios com *Red Teams*. Estes controlos foram desenvolvidos pelo *Center for Internet Security* (Center for Internet Security, 2015), uma organização americana orientada para a preparação e resposta a questões de segurança de informação nos setores público e privado. Tem parcerias governamentais e com fortes entidades da indústria de TI. Estes controlos não são apenas métodos de defesa, incluem também uma metodologia para ataques já realizados e sistemas comprometidos.

Estes controlos encontram-se priorizados mas, não constituem uma solução ótima para qualquer caso, mesmo um número relativo de controlos não podem ser aplicados todos em simultâneo e cabe à organização planear, implementar e gerir o processo de forma a proteger os ativos que considera mais valiosos (The CIS Critical Security Controlls for Effective Cyber Defense version 6.0, 2015).

1.3.8 Conclusão

O que têm estas *frameworks* em comum e qual a que melhor se aplica? Os meios financeiros do profissional de segurança ou da instituição são um fator de decisão. Os guias e ferramentas disponíveis através de certificação profissional são de acesso limitado a apenas algumas instituições ou indivíduos.

Em termos de processo, quer para testes de intrusão no geral, quer para a vertente *web* apenas, é possível concluir ao analisar as várias *frameworks* que um teste de intrusão começa por uma fase de recolha de informação, tipicamente conhecida como reconhecimento. Esse

reconhecimento pode ser detetado pela instituição alvo ou não, dependendo se usarem técnicas ativas ou passivas. Em seguida, surge uma fase de enumeração onde já existe contacto também passivo ou ativo com o sistema ou rede do alvo onde se tentam identificar vulnerabilidades a explorar. Estas duas fases constituem a maior parte do processo de testes de intrusão. Quanto mais informação tiverem reunido e quanto melhor conhecerem o sistema ou aplicação a testar, mais hipóteses haverá de preparar um ataque bem-sucedido.

A fase intermédia de qualquer teste de intrusão, comum entre vários guias, é a fase de ataque e manutenção de privilégios. Trata-se da realização dum ataque com a exploração de uma vulnerabilidade e posteriormente, garantir que essa exploração possa ser realizada novamente sem deteção, pois esse é o cenário mais preocupante. A última fase e também muito importante, é a fase de relatórios. É baseado na descrição de resultados que a gestão pode tomar ação e priorizar os riscos. Uma *framework* deve fornecer informação para criação de relatórios técnicos para os profissionais de TI e relatórios executivos para avaliação por parte das chefias.

Mesmo num cenário exclusivamente *web* e independentemente do nível de conhecimento da estrutura que se está a explorar, deve haver recolha de informação sobre a organização, sobre as aplicações, sobre os sistemas que suportam as aplicações e utilizar essa informação em prol de um ataque premeditado.

1.4 Outras técnicas de pen testing

O objetivo de um profissional de testes de intrusão (*pentester)* é encontrar e explorar vulnerabilidades de forma a comprometer o sistema e aceder a informação privilegiada (Walker, 2012, p. 26). Independentemente de âmbito ou metodologia, o papel do *pentester* é uma arte em que o engenho e imaginação são ótimos aliados para encontrar e explorar vulnerabilidades (Austin & Williams, 2011).

Uma aplicação *web* está assente num computador, ligado a uma rede com inúmeras tecnologias, software e hardware e por esse motivo, reunir um vasto leque de possibilidades de ataque pode aumentar a percentagem de sucesso (Walker, 2012, p. 187). Faz parte de uma estratégia de segurança atualizar software com *patches* mas, instituições com negócios críticos não podem realizar a instalação dum *patch* sem primeiro avaliar o seu impacto no sistema ou na interoperabilidade com outros sistemas. Uma vez que os diversos fornecedores mantêm bases de dados de correções com descrição de correções ou até mesmo através da pesquisa de bases de dados online de vulnerabilidades, um atacante pode identificar para o sistema em teste

inúmeras oportunidades para explorar (Walker, 2012, p. 187). Para estes casos, bases de dados de vulnerabilidades são uma fonte muito útil em testes de intrusão, elas reúnem informação sobre o ataque para exploração de uma vulnerabilidade já conhecida. Existem ferramentas automáticas, como por exemplo, o *Nessus* (Nessus Vulnerability Scanner, 2015), que realizam testes e através de uma base de dados própria, procuram vulnerabilidades existentes face à aplicação em análise.

1.5 Web Scanners

Web scanners são aplicações automatizadas capazes de realizar testes de intrusão em aplicações *web* na sua total extensão com pouca ou nenhuma intervenção humana (Vulnerability Scanning Tools, 2015). De forma geral, um *web scanner* começa por mapear a estrutura completa do site num processo designado de *crawling* e em seguida realiza uma série de ataques a cada um desses recursos do site identificados no *crawling* (WASC Threat Classification, 2010, p. 151).

O processo de *crawling* é um dos momentos mais importantes de qualquer *web scanner*, se este processo falhar em identificar a estrutura da aplicação web em teste, os resultados não cobrirão a totalidade da aplicação e as conclusões de segurança serão incompletas sem que os auditores e a entidade responsável pela empresa tenham consciência induzindo uma falsa sensação de segurança (Acunetix, 2011).

Um *web scanner* tenta identificar vulnerabilidades de segurança sem aceder ao código fonte da aplicação. Existem diversos tipos de *web scanners*, aplicações comerciais com um custo de aquisição, aplicações *open source* e aplicações gratuitas. A diferença entre as aplicações *open source* e as gratuitas é a possibilidade de realizar alterações à aplicação.

Web scanners são um tipo de ferramenta ideal para qualquer teste em que não se tem acesso ou conhecimento da linguagem do código fonte da aplicação testada o que impede uma análise estática. Estes são dois motivos para a realização de testes de intrusão com ferramentas automáticas (OWASP Testing Guide v4, 2014). Outra questão que os auditores têm que lidar na análise de código duma aplicação, é o conhecimento da linguagem de programação utilizada no seu desenvolvimento, paradigmas de desenvolvimento, padrões de implementação e *frameworks* de suporte à aplicação. Os auditores de segurança estão mais à vontade com ferramentas de testes de intrusão do que com diversas linguagens de programação (Curphey & Arawo, 2006).

As principais vantagens de um *web scanner* em termos de resultados conclusivos para o auditor, são a rapidez de execução de testes mesmo para uma aplicação de dimensão considerável. Doutra forma, um profissional de segurança demoraria imenso tempo a testar cada recurso da aplicação podendo a aplicação ser explorada por um atacante durante esse tempo, portanto, um *web scanner* permite reduzir o tempo de realização dos testes (Austin & Williams, 2011, p. 2). Outra das principais vantagens, é a redução erros na configuração do ataque e leitura de resultados. O *web scanner* realiza inúmeras tentativas de ataque e identifica potenciais vulnerabilidades automaticamente.

Um *web scanner* baseia-se nas vulnerabilidades mais comuns para realização dos seus ataques através de técnicas de *proxy*, *fuzzing*, teste de *inputs*, entre outros. mas, um *web scanner* pode não ser suficiente e a análise de duas aplicações distintas revelar resultados diferentes, isto porque as aplicações *web* podem ser desenvolvidas em várias plataformas e um *web scanner* pode mostrar-se mais eficaz em determinadas situações (Austin & Williams, 2011, p. 1). Além da capacidade de identificação de vulnerabilidades, um *web scanner* deve ser fiável e o número de falsos positivos encontrados, reduzido.

Um falso positivo, é a designação data a uma situação encontrada por um *web scanner* incorretamente classificada como vulnerabilidade (Austin & Williams, 2011, p. 2). A confirmação de possibilidade de exploração e comprometimento dum sistema só pode ser confirmada manualmente por um auditor, caso a aplicação revele muitos falsos positivos, o auditor despenderá bastante tempo a analisar situações mal identificadas o que não acrescenta benefícios à auditoria (Acunetix, 2011).

Outro ponto fundamental para uma avaliação eficaz é a configuração do *web scanner* antes de iniciar o ataque (Acunetix, 2011). Por exemplo, se o acesso a um site requerer autenticação, um *web scanner* pode não conseguir avaliar mais que a página de *login*. Através da definição de credenciais válidas numa navegação manual, o acesso a todas as funcionalidades da aplicação fica disponível. Uma correta configuração pode igualmente reduzir o tempo de execução dos testes de intrusão automáticos, cabe ao auditor com o conhecimento da complexidade e dimensão da aplicação escolher o nível de exaustão do seu teste (Acunetix, 2011).

Na vanguarda da definição de características que definem um bom *web scanner* está a *Web Appplication Security Consortium* (WASC) (Web Application Security Consortium, 2014). São uma organização composta por um grupo de especialistas da indústria dedicados ao

desenvolvimento de standards de segurança *open source*. Publicam regularmente informação técnica e artigos sobre segurança de tecnologias de informação e vários profissionais e académicos recorrem para apoio sobre o tema. Um dos seus projetos é o *Web Application Security Scanner Evaluation Criteria* (WASSEC) (Web Application Security Scanner Evaluation Criteria, 2009). Este projeto é um guia para avaliação de um *web scanner* e identifica as características que este deve ter independente do fornecedor, no entanto, a escolha de um *web scanner* cabe ao auditor avaliar de acordo com as suas necessidades. O documento (Web Application Security Scanner Evaluation Criteria, 2009) define uma lista de requerimentos dum *web scanner* para que seja considerado completo. Os critérios de avaliação de um *web scanner* considerados imprescindíveis são os seguintes:

- Suporte multi protocolo
- Autenticação
- Gestão de Sessão
- *Crawling*
- *Parsing*
- Testes
- Comando e controlo
- Relatório

Suporte multi protocolo (*Protocol Suport*) especifica os protocolos de comunicação que um *web scanner* deve ser capaz de suportar, HTTP 1.1 e SSL/TLS são alguns exemplos. Uma vez que o *web scanner* pode não ter acesso direto ao sítio para teste, deve também ser capaz de suportar testes de intrusão através de um *proxy* porém, este tipo de ligação pode aumentar o número de falsos positivos (Web Application Security Scanner Evaluation Criteria, 2009).

Autenticação lógica é um dos mecanismos de defesa de inúmeros sites, é uma garantia de que a pessoa que está a tentar aceder a um recurso, tem permissão para o consultar, por essa razão, um *web scanner* tem que ser capaz de suportar diversos mecanismos de autenticação, entre os quais, *Basic*, *Digest* e *Client SSL Certificates* (Web Application Security Scanner Evaluation Criteria, 2009).

De forma a que o *web scanner* consiga manter a ligação e o acesso com a aplicação em teste, deve ser capaz de suportar gestão de sessões (*Session Management*). A manutenção de uma sessão válida permite ao processo de *crawling* navegar por toda a aplicação e na fase de testes,

evitar que os pedidos à aplicação não sejam ignorados pela mesma resultando em perda de informação (Web Application Security Scanner Evaluation Criteria, 2009).

Como indicado anteriormente, o *crawling* é uma peça fundamental na auditoria de segurança de uma aplicação *web*, sendo por isso, um requisito de qualquer *web scanner*. A capacidade de identificar e navegar entre páginas numa aplicação é o que permite ao *crawler* mapear todos os caminhos possíveis e possibilitar ao teste ativo cobertura de toda a aplicação de teste.

As aplicações *web* assentam sobre várias tecnologias e protocolos, o *parsing* é a capacidade que permite a um *web crawler* compreender os vários conteúdos presentes numa aplicação *web* desde HTML, Javascript, XML e os vários protocolos conhecidos pelo *browser* (Web Application Security Scanner Evaluation Criteria, 2009).

O ponto seguinte nos critérios de avaliação dum *web scanner* é a característica de reconhecer e suportar diversos tipos de vulnerabilidades. Quantas mais conhecer, maior probabilidade de encontrar alguma, mas, também mais moroso se torna o teste. Alguns *web scanners* funcionam com extensões o que lhes permite evoluir em termos de testes de intrusão ou testar um vetor de ataque em específico (Web Application Security Scanner Evaluation Criteria, 2009).

O "Command and control" é um tipo de funcionalidade que acrescenta valor através da possibilidade de parar e retomar um teste de vulnerabilidades, assim como agendar os mesmos. O controlo através de um *interface* gráfico ou por API são outros exemplos da diversidade de controlos que aumentam as possibilidades de um *web scanner*. Um interface apelativo e intuitivo é um critério de escolha de *web scanners* importante (Web Application Security Scanner Evaluation Criteria, 2009).

Por último, a capacidade de criar configurar relatórios é uma mais valia pois, esse é o resultado final de qualquer teste de intrusão, a demonstração de resultados para técnicos e chefias.

1.6 Comparação de ferramentas

Como vimos anteriormente, uma única ferramenta pode não ser suficiente mas, muitas ferramentas podem apresentar um custo temporal elevado, por isso é necessário proceder a uma avaliação de necessidades. Está especificado no WASSEC que deve ser o auditor a definir as necessidades e consoante as mesmas, proceder a uma comparação de resultados de vários *web scanners* atribuindo uma pontuação a cada rubrica considerada (Web Application Security Scanner Evaluation Criteria, 2009). As necessidades podem variar desde a capacidade de

explorar uma determinada vulnerabilidade ou simplesmente ter uma interface fácil de compreender e utilizar.

Existem diversas comparações na literatura entre várias ferramentas mas, especificamente para um tipo de vulnerabilidade – como é o caso de *SQL injection* ou *Cross Site Scripting*. Ainda no sentido de melhorar as técnicas de testes de intrusão existem aplicações disponíveis construídas especificamente com vulnerabilidades para realização de testes de intrusão. A OWASP mantém uma lista destas ferramentas que se encontram disponíveis em (OWASP Vulnerable Web Applications Directory Project, 2015). Aqui estão listados diversos projetos de aplicações com vulnerabilidades que podem ser utilizadas a partir da internet ou descarregadas e testadas num ambiente controlado como forma de desenvolvimento do conhecimento dos profissionais de análise de vulnerabilidades. A WASC não recomenda o uso destas aplicações para testar os *web scanners* por considerar que estes estão otimizados para essas aplicações mas, pelo menos é possível conhecer e treinar a utilização dum *web scanner* (Web Application Security Consortium, 2014). Uma listagem das aplicações mais conhecidas do género retirada de (Lyon, 2015) é apresentada na Figura 1.2 (assinaladas com o símbolo "€" estão as comerciais).

Figura 1.2 - Lista de Web Scanners

Num estudo de (Doupé, Cova, & Vigna, 2010) foi criada uma aplicação com vários tipos de vulnerabilidades e testada contra diversos *web scanners*, comerciais e *open source*. A recomendação da WASC foi seguida em não utilizar aplicações próprias para o efeito mas, o estudo em si, como o autor indica não é exaustivo pois apenas foi utiliza uma aplicação, contudo, é uma boa base para a prestação dos *web scanners*. Os resultados dos *web scanners*

comerciais como por exemplo os resultados do Burp foram superiores e mais fidedignos aos resultados de ferramentas não comerciais.

Outro estudo recente e do mesmo género que o anterior, realizado por (Makino & Klyuev, 2015), compara duas aplicações *open source*, o OWASP ZAP (OWASP ZAP, 2015) e o Skipfish (Skipfish, s.d.). Neste estudo, a ferramenta da OWASP foi superior e os testes foram realizado em três aplicações criadas para o efeito.

1.7 Metodologia em Laboratório

Uma técnica que pode ser utilizada com diversas vantagens no campo de testes de intrusão é a realização de testes em laboratório, isto é, num ambiente controlado e com aplicações fictícias próprias para o efeito. Em laboratório não há risco de comprometer a disponibilidade de uma aplicação e podemos antecipar comportamentos estranhos por parte de um *web scanner* (Zitta, Marik, & Neradova, 2014, p. 2).

Em laboratório é possível explorar um *web scanner* e uma metodologia de testes de intrusão para as aplicar eficaz e posteriormente nas aplicações alvo de teste, no mundo real. A necessidade de recorrer a laboratórios surge também pela diminuta existência de resultados de auditorias partilhados pelas instituições.

A WASSEC e estudos como o de (Doupé, Cova, & Vigna, 2010) não recomendam o uso de aplicações criadas com vulnerabilidades para testar *web scanners*, isto é, não devem ser realizados testes de comparação entre *web scanners* nestas aplicações de teste devido ao facto dos fornecedores as poderem otimizar para exceder as espectativas em termos de vulnerabilidades encontradas (Web Application Security Scanner Evaluation Criteria, 2009).

1.8 Risco e Avaliação de Vulnerabilidades

A segurança é um processo, não é um produto (Schneier, 2000). Diversas entidades como OWASP, BSIMM (McGraw, Migues, & West, 2015) advogam que o processo de segurança deve estar integrado com o SDLC de um projeto do início ao fim mas, não é possível controlar o que não se pode medir (OWASP Testing Guide v4, 2014), (Teodoro & Serrão, 2011). Existem várias medições a nível de desenvolvimento de software, processo, e de vulnerabilidades como BSIMM6 (McGraw, Migues, & West, 2015) ou CVSS (CVSS, 2015) com *frameworks* criadas

para avaliação e definição do patamar em que se encontra a empresa em termos de desenvolvimento de software para que possa priorizar as melhorias a realizar.

Uma aplicação está à mercê de ataques, vulnerabilidades, ameaças e isso constituí um risco. Nem todos os ataques são bem-sucedidos, nem todas as vulnerabilidades conduzem a uma violação de segurança. Se considerarmos esta afirmação verdadeira, como podemos definir um risco?

Uma possível definição de risco, poderá ser: a probabilidade de uma vulnerabilidade ser explorada por uma ameaça, que resulta num determinado grau de perda de confidencialidade, integridade e disponibilidade de um ativo, com um impacto negativo para o negócio (KPMG Risk Management, 2015).

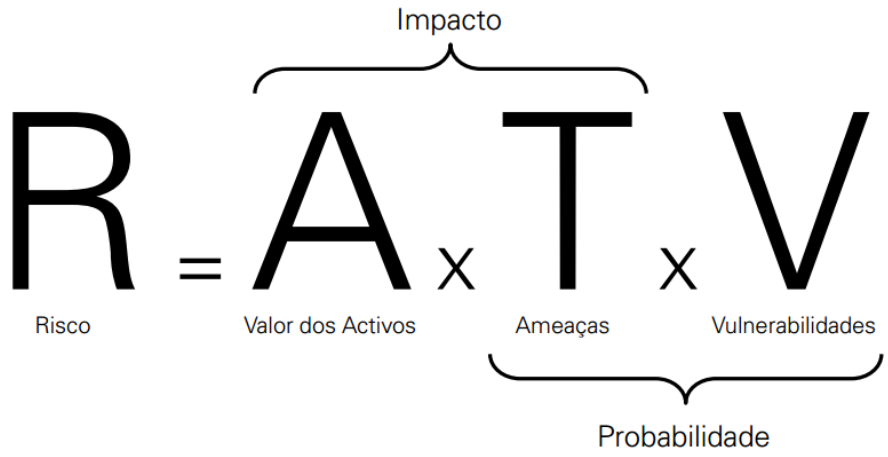


Figura 1.3 - Definição de Risco (KPMG Risk Management, 2015)

Esta caracterização de risco, traduz-se numa avaliação qualitativa com diversas falhas. Apesar de tudo, permite monitorizar os riscos e definir uma estratégia. Face a um risco, uma organização pode aceitar, evitar, mitigar ou transferir para terceiros (KPMG Risk Management, 2015).

Outra possível definição de risco pode ser "a frequência e magnitude prováveis de perda no futuro de confidencialidade, integridade, disponibilidade e responsabilização" (Wheeler, 2011). Wheeler também defende que uma instituição deve utilizar os seus recursos para priorizar riscos e definir uma estratégia de segurança, isto é, atacar a raiz do problema em vez de resolver os sintomas um a um.

De uma perspetiva organizacional, a segurança não é privilegiada face ao lucro mas, é reconhecido que uma estratégia de risco não se aplica em todos os casos e que, uma organização deve identificar as áreas críticas e o valor dos seus ativos que merecem maior investimento. Cada departamento, assim como um *software*, têm o seu nível de tolerância ao risco, ou seja, a aplicação de padrões de segurança de forma balanceada entre o custo de controlo e potencial redução de exposição ao risco (Wheeler, 2011, p. 8).

Independentemente do cálculo da probabilidade e impacto de um risco, estes podem ser combinados num grau de severidade e representados pela matriz da Figura 1.4.

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Figura 1.4 - Matriz Severidade de Risco - Impacto vs Probabilidade (OWASP Risk Rating Methodology, 2015)

É fácil compreender que quanto mais provável e maior impacto um risco comportar, maior prioridade deverá ser atribuída à sua resolução.

Threat Agents	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
App Specific	Easy	Widespread	Easy	Severe	**App / Business Specific**
	Average	Common	Average	Moderate	
	Difficult	Uncommon	Difficult	Minor	

Figura 1.5 - Classificação de Risco OWASP (OWASP Risk Rating Methodology, 2015)

Representada na Figura 1.5, estão as combinações possíveis para classificação de um risco retiradas do OWASP Risk Rating Methodology (OWASP Risk Rating Methodology, 2015). Através deste modelo simples, é possível compreender um risco e evoluir para uma escala probabilidade versus impacto. Quer um profissional de TI, quer um gestor, conseguem melhor

através de classificação em várias rúbricas, perceber uma vulnerabilidade e interpretar melhor onde poderão estar os pontos críticos de cada risco (OWASP Risk Rating Methodology, 2015).

Eis um modelo de avaliação de risco proposto pela NIST SP800-37 na Figura 1.6 (Guide for Applying the Risk Management Framework to Federal Information Systems, 2010). Composto por seis passos, a *Risk Management Framework* é um processo cíclico de Categorização, Seleção (controlos de segurança), Implementação (controlos de segurança), Avaliar, Autorização e Monitorização com tantas iterações quantas necessárias.

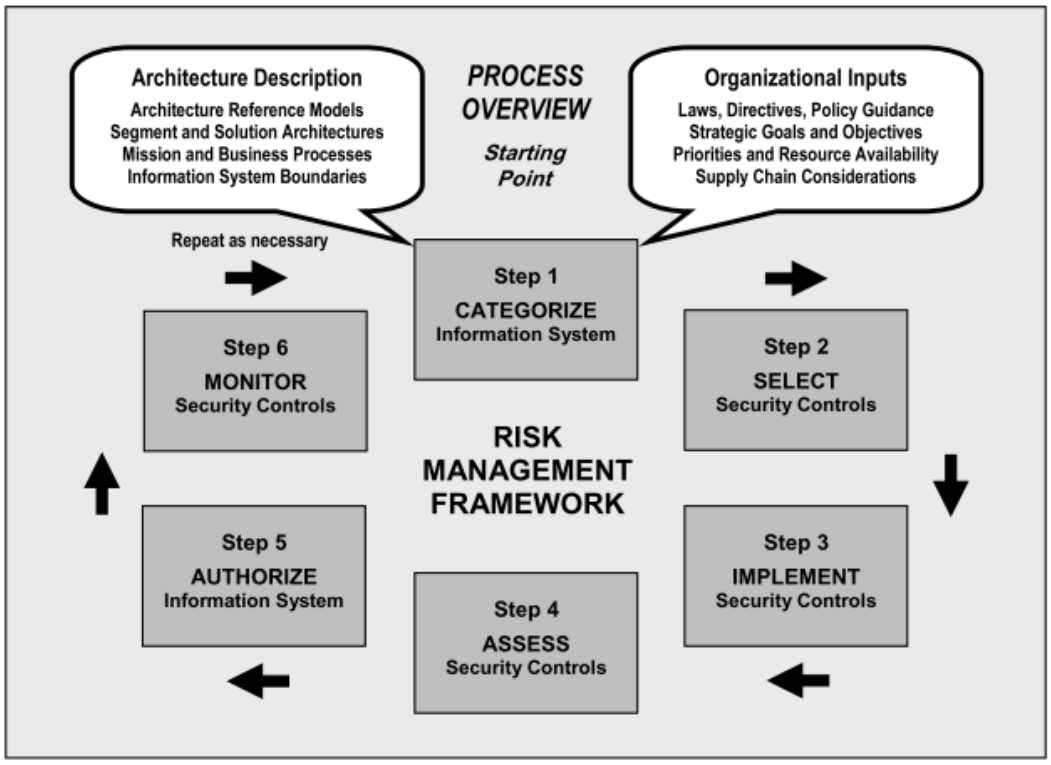


Figura 1.6 - Modelo de avaliação de Risco proposto pelo NIST (Guide for Applying the Risk Management Framework to Federal Information Systems, 2010)

O grupo de normas ISO 27000, é um conjunto de *standards* para medir e avaliar quão bem o funcionamento do Sistema de Gestão de Segurança de Informação (ISMS) numa organização é executado (ISO 27000, 2013). Deve também ser implementado e revisto ciclicamente. Um dos processos desta norma é a escolha de controlos a implementar. A norma específica para esse efeito é a ISO 27002 (ISO 27002, 2013). Este grupo de normas da 27001 à 27006 complementam-se na gestão do ISMS.

Outra *framework* exemplo deste processo de gestão de segurança de informação é o COBIT que além de fazer a ligação entre as TI e as áreas de negócio, introduz controlos de segurança numa organização (COBIT, 2014).

1.8.1 *Common Vulnerability Scoring System* - CVSS

O CVSS tem como objetivo capturar as principais características de uma vulnerabilidade e produzir uma pontuação que reflete a severidade da dita vulnerabilidade. Este valor pode ser traduzido para uma escala qualitativa (Baixo, Médio, Alto) para ajudar a gestão das organizações (Scarfone & Mell, 2009). É composto por três grupos que categorizam as métricas neles inseridas, são eles o grupo Base, Temporal e Ambiental. O primeiro representa as características de uma vulnerabilidade que não variam ao longo do tempo, o grupo Temporal categoriza vulnerabilidades que variam no tempo mas não considera ambientes de utilizadores e o grupo Ambiental caracteriza vulnerabilidade específicas do ambiente do utilizador.

O recurso ao CVSS tem a vantagem de, além de ser um standard de indústria, beneficia a análise de uma vulnerabilidade num sistema informático de forma a extrair outras métricas como o tempo de resolução. É um sistema com base em vários vetores relacionados com as vulnerabilidades que por si só, definem o tipo de vulnerabilidade. É assim possível para a empresa compreender melhor a vulnerabilidade e priorizar a sua resolução (Younis & Malaiya, 2015).

O CVSS apresenta uma proposta mais quantitativa, que além de constituir uma fonte de dados sobre vulnerabilidades de fabricantes reconhecidos, atribuí uma pontuação a uma vulnerabilidade segundo vários critérios:

1. Impacto na Confidencialidade;
2. Impacto na Integridade;
3. Impacto na Disponibilidade;
4. Complexidade de acesso;
5. Autenticação;
6. Acesso não autorizado;
7. Tipo de Vulnerabilidade.

A versão atual do CVSS é a 3, lançada no final de 2014 tendo substituído a versão 2 de 2007. Foi adotada em normas como PCI DSS (PCI Security Standards Council, 2015), norma de pagamentos com cartões de crédito e pelo NIST, associação de standards tecnológicos dos

EUA. A versão 2 é reconhecida formalmente como padrão internacional desde Abril de 2011. A última versão tem mudanças significativas ao nível análise de vulnerabilidades e de definição de pontuação.

Esta nova atualização deriva novas métricas a partir das existentes na anterior versão e permite definir melhor características das vulnerabilidades incluindo por exemplo a complexidade de exploração, âmbito que compromete e tempo de execução. Por exemplo, a métrica Âmbito, introduzida nesta versão colmata a incapacidade de explicar se uma vulnerabilidade compromete a aplicação ou o sistema onde está inserida por completo.

Em termos de diferenças significativas, a versão 3, por exemplo, categoriza as vulnerabilidades quanto ao impacto de um componente, isto é, software, rede, etc. Essa caracterização é definida pela métrica âmbito, criada nesta versão. Surge também uma outra métrica que identifica a necessidade de interação por parte do utilizador para que, uma vulnerabilidade possa ser explorada. Associado a esta vulnerabilidade estão ataques de engenharia social ou falta de conhecimento de políticas de segurança.

Outra novidade na versão 3, é a capacidade de pontuar várias vulnerabilidades num mesmo ataque. Esta necessidade surge para ataques que são realizados explorando várias vulnerabilidades em cadeia e, apesar de não ser uma métrica formal, está incluída no modelo. Este processo consiste em atribuir uma pontuação individual e uma pontuação face ao encadeamento. Este encadeamento permite definir melhor um ataque ou criar cenários alternativos com base numa vulnerabilidade encontrada (CVSS, 2015).

A Figura 1.7 representa a relação das várias métricas resultantes numa pontuação final e atribuindo uma pontuação de severidade.

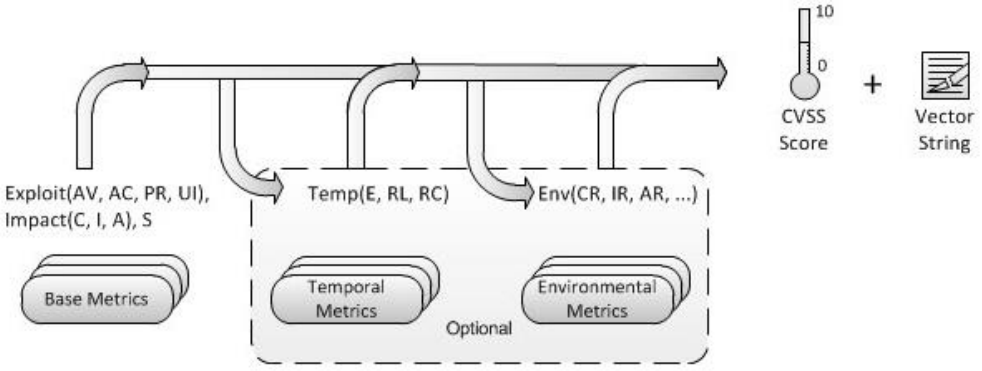


Figura 1.7 - Cálculo de Pontuação de Vulnarabilidade CVSS (CVSS, 2015)

Esta versão 3 do CVSS é ainda recente e por ter sido sujeita a alterações estruturais significativas, a classificação de vulnerabilidades anteriores pode sofrer alterações como aconteceu com a evolução da versão 1 para a versão 2. Apesar de tudo, também pode vir, graças aos seus novos vetores (como por exemplo, se requer interação humana ou não para exploração da vulnerabilidade) corrigir a questão apontada no estudo (Younis & Malaiya, 2015) quanto a elevado número de falsos positivos.

Os falsos positivos poderão ser explicados por vários motivos. Falta de conhecimento dos valores a indicar em cada vetor do CVSS, pela complexidade ou má interpretação dos vetores presentes ou por as aplicações e sistemas se apresentarem por camadas e como tal, aumentar os pontos de defesa impendido que uma vulnerabilidade seja explorada. A principal desvantagem de uma avaliação incorreta de severidade é a eventual priorização de resolução de situações improváveis ou remotas de acontecer ou mitigadas por outras defesas.

1.9 Segurança e o SDLC

Testes de intrusão são apenas uma de várias possíveis linhas de defesa no desenvolvimento de software e na segurança de uma aplicação e mesmo apesar da sua realização, identificação e mitigação de vulnerabilidades, a organização pode estar a adquirir uma falsa sensação de segurança. Por esse motivo, a adoção de uma metodologia de desenvolvimento com considerações de segurança desde cedo no processo é fundamental.

Entre as metodologias mais reconhecidas encontram-se:

- BSIMM6 (Creative Commons)
- Application Security Verification Standard (ASVS – OWASP)
- Software Assurance Maturity Model (SAMM - OWASP)

Estas metodologias apresentam-se como guias de boas práticas e são o resultado de estudos de instituições de referência e das práticas das mesmas. Apresentam assim, considerações de elementos de segurança de diversos domínios em várias áreas que empresas com a Cisco ou Microsoft adotam diariamente para continuarem a desenvolver software seguro (McGraw, Migues, & West, 2015).

A BSIMM6 é uma *framework* construída por análise do estado atual das instituições internacionais de referência e as suas práticas. Definem a sua metodologia como unidade de medida e não com um "guia" para segurança. Reflete o estado atual das boas práticas aplicadas

numa instituição por comparação. A existência da *framework* serve a necessidade de ter uma base reutilizável e comparável na indústria.

Em última análise, o resultado de uma avaliação BSIMM é um *scorecard* de características de segurança que a instituição utiliza face às suas necessidades. Em teoria, quantas mais atividades incorporar na instituição, mais madura estará a mesma em termos de segurança (McGraw, Migues, & West, 2015). Sucintamente, esta metodologia está dividida em quatro domínios desde Gestão, *Inteligence*, *SSDL Touchpoints* e Desenvolvimento. Cada domínio tem três práticas e somam um total de 112 atividades (ver Anexo 2). A Figura 1.8, representa o gráfico de uma comparação entre duas instituições em termos de processos BSIMM.

É um dos objetivos desta dissertação sensibilizar esta e outras instituições da importância de inclusão de técnicas de segurança no processo de desenvolvimento de software. Apesar de tudo, as instituições tomam algumas precauções quanto a segurança atualmente, mas nem sempre as testam por exemplo com testes de intrusão. Uma vez que é objetivo fornecer um meio digital de consulta e registo de vulnerabilidades (*dashboard*), as metodologias mais reconhecidas de desenvolvimento de software seguro apresentam algumas ideias pois algumas são construídas por comparação de práticas na indústria por vários fornecedores de serviços e software (McGraw, Migues, & West, 2015), (OSSTMM v3.0 - The Open Source Security Testing Methodology Manual, 2010, p. 11).

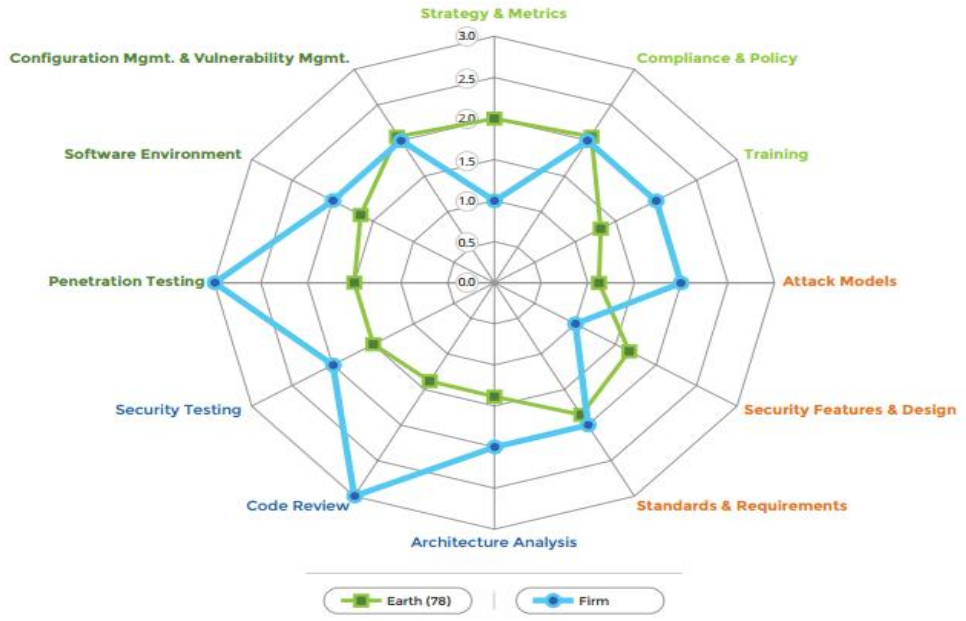


Figura 1.8- Análise comparativa de maturidade BSIMM de duas organizações no domínio de desenvolvimento (McGraw, Migues, & West, 2015)

As atividades desta *framework* mais aplicadas, isto é, comuns a mais instituições ao nível de *Deployment* são, por exemplo, requisição de testes de intrusão por parte de entidades externas, uma entidade externa pode adicionar um novo par de olhos para análise de um problema e contribuir com experiência. Uma outra atividade muito aplicada é garantir que a segurança da rede está atualizada, a infraestrutura de rede deve ser segura e monitorizada pois é aí que assentam todos os sistemas e aplicações web (Curphey & Arawo, 2006).

De realçar que, as dez instituições com maior pontuação BSIMM têm um grupo de interessados como programadores ativos, arquitetos, gestores de software, entre outros, com afinidade por segurança e organizados por um programa organizacional que visa instituir, medir e evoluir atividades de segurança de software de forma coordenada. A criação de *dashboard* informativos de vulnerabilidades e partilha de informação de segurança são outras atividades recomendadas de forma a promover o *feedback* e troca de conhecimento. Não quer isso dizer que toda a organização deve ter acesso a essa informação, mas entre grupos de desenvolvimento pode, promover a cultura de segurança e competição interna (McGraw, Migues, & West, 2015).

E já anteriormente referida OWASP, tem inúmeros projetos que suportam o desenvolvimento seguro, não propriamente uma *framework* de desenvolvimento e inclusão no SDLC, está a *Application Security Verification Standard* (OWASP Application Security Verification Standard Project, 2015) que introduz uma *checklist* de requisitos ou testes que os arquitectos, *developers* e *testers* devem utilizar para definir uma aplicação segura. Pela mesma razão que outras metodologias ou padrões, permite consumidores alinharem as suas necessidades pela mesma medida. É composta por três níveis de segurança: oportunista, *standard* e avançado. O primeiro nível, o oportunista, representa as ameaças mais comuns como por exemplo as OWASP Top 10. Deve ser aplicado em qualquer software. O nível intermédio ou *standard* é desenhado para aplicações com informação sensível e que requerem proteção. O terceiro nível e mais crítico, aplica-se quando existem transações de grande valor ou informação de elevado valor. Esta metodologia divide-se em vários domínios e, através de uma análise de necessidades, identificam-se as verificações que devem ser percorridas em cada *software*. Por essa razão, é fundamental a definição de valor destes ativos (Wheeler, 2011).

O primeiro nível de verificação da ASVS, indispensável a qualquer software pode ser alcançado com testes de intrusão e ferramentas automáticas sem acesso ao código fonte das aplicações. Uma avaliação mais crítica, requer maior investimento.

Outro projeto da OWASP é o projeto *Software Assurance Maturity Model* (SAMM) (SAMM, 2009). O propósito desta *Framework* é ajudar as organizações a formular uma estratégia para segurança de software (Teodoro & Serrão, 2011). SAMM tem duas camadas, uma primeira de Funções de Negócio (BF) uma segunda de Práticas de Segurança (PS). As Funções de Negócio incluem os domínios Governo, Construção, Verificação e *Deployment* que definem o SDLC desta *framework*. As práticas de segurança, inseridas nas Funções de Negócio, asseguram que, em cada fase do SDLC as validações de segurança são cumpridas. Também esta metodologia tem vários níveis de maturidade que, devem ser aplicados consoante as necessidades de proteção alvo. São três níveis de maturidade que podem ser aplicados em cada Prática de Segurança aumentando o nível de confiança na aplicação se o valor dos ativos assim justificar (SAMM, 2009, p. 34).

O caminho para aplicar segurança não é fácil e implementar qualquer uma destas metodologias é uma tarefa morosa, por essa razão estas metodologias, apesar de reconhecidas, são fortemente criticadas (Teodoro & Serrão, 2011) mas, é certo que a segurança deve ser aplicada em todos os momentos do SDLC (OWASP Testing Guide v4, 2014).

1.10 Painel de instrumentos / *Dashboard*

O objectivo desta secção é recolher informação sobre ferramentas disponíveis para controlo e apresentação de vulnerabilidades. Aplicações capazes de recolher características de vulnerabilidade através de relatórios de auditorias de segurança e resumir a informação em painéis de instrumentos ou *dashboards* através de gráficos. De forma que a análise de vulnerabilidades por técnicos e gestores possa ser simplificada e partilhada.

Pretende-se obter também, como aplicações do tipo *dashboard* se relacionam com *web scanners* e que análises de risco são compatíveis com as mesmas.

A pesquisa de estudos sobre utilização de *dashboards* e suas características revelou-se fraca pois, pouca ou nenhuma informação existe a nível académico sobre esta matéria, mas surgem diversa organizações com ferramentas desenvolvidas. Seguidamente discutiremos as suas características:

1.10.1 Serpico

O projecto *Simple Report and Colaborative tool* (Serpico) é uma aplicação direcionada para a criação de relatórios no âmbito de segurança de tecnologias de informação. Através de vários *templates* ou modelos disponíveis, permite escolher que vulnerabilidades deve ser incluída na geração de relatório. O Serpico permite exportar relatórios para o formato Word assim como complementar a informação do relatório com informação sobre a auditoria e a organização.

As funcionalidades reunidas desta ferramenta são as seguintes:

- Edição de modelos de relatórios;
- Base de dados de modelos;
- Permite anexar imagens ao relatório com evidências de vulnerabilidades;
- Exportação para formato .doc;
- Importação de relatórios dos *web scanners* Nessus e Burp;

É uma ferramenta simples, mas flexível na medida em que, possibilita a configuração ou utilização de vários modelos para apresentação de resultados. A importação de relatórios criados por *web scanners* é outra funcionalidade desta ferramenta que reduz o tempo de inserção de vulnerabilidades na base de dados e reduz o erro de inserção. É compatível com o Nessus e o Burp. A classificação de vulnerabilidades identificada para o Serpico é através de uma escala

qualitativa de acordo com a prioridade, alta, média, baixa e informação (Serpico, 2016). O Serpico está sujeito à licença de distribuição do GitHub.

1.10.2 Thread Fix

O Thread Fix é uma ferramenta de consolidação de resultados de auditorias de segurança. A primeira característica que se destaca é a capacidade de recolher informação de vários tipos de ferramentas de testes de intrusão, além de *web scanners*, suporta aplicações de testes com análise de código estático.

A capacidade de importação de relatórios de ferramentas externas, estende-se até cerca de 20 aplicações diferentes. Alcança uma maior compatibilidade e possibilita maior adaptabilidade à abordagem de uma organização devido a esta diversidade. A classificação de vulnerabilidade é segundo informação CWE[1].

Esta ferramenta é também capaz agendar tarefas como obter relatórios de auditorias realizadas e integrar os novos resultados na sua base de dados. Como certas aplicações possuem APIs, é possível conciliar e automatizar determinados processos. Os resultados obtidos podem também ser avaliados de forma a agrupa-los e eliminar redundância. Esta funcionalidade pode ser muito vantajosa em auditorias que utilizem vários *web scanners*.

As funcionalidades identificadas resumem-se a:

- Importação de vários tipos de ferramentas com diversos tipos de análise;
- Importação de relatórios de 24 *web scanners* diferentes;
- Agrupamento de vulnerabilidades;
- Agendamento de tarefas e integração;
- Classificação de vulnerabilidades através de CWE;

O Thread Fix é uma ferramenta capaz de ligar o trabalho de várias equipas e aumentar assim a eficiência, por exemplo, pela redução de documentação. Através da análise de auditorias e vulnerabilidades através num painel de instrumentos, simplifica-se a alocação de recursos (ThreadFix, 2016).

[1] Common Weakness Enumeration (CWE): conjunto de informação normalizada e quantificada sobre vulnerabilidades mantida pela instituição Mitre.

1.10.3 Faraday

O Faraday não é apenas uma ferramenta de análise e apresentação de vulnerabilidades, além de *dashboard*, é também uma ferramenta de integração automática de auditorias de segurança recolhendo informações dos *web scanners* em tempo real.

O Faraday está disponível em três versões:

Community: está limitado a perfis de utilizador e não inclui geração de relatórios nem importação de relatórios de *web scanners.*

Profissional: a versão intermédia do Faraday, tem limitação de utilizadores, mas, inclui a maioria das funcionalidades excepto importação de relatórios. Tem disponíveis os perfis de utilizador *pentester, manager* e cliente.

Corporate: esta versão inclui todas as funcionalidades e utilizadores ilimitados. A importação de relatórios de *web scanners* está presente.

As funcionalidades principais são:

- Linha de comandos de controlo;
- Painel de instrumentos com diversas opções;
- Perfis de utilizador;
- Três versões, entre as quais uma gratuita e com actualizações;
- Importação de relatórios de *web scanners*;
- Criação de relatórios;

Portanto, esta ferramenta divide-se numa linha de comandos para configuração e integração de resultados com complementos informativos visuais e num painel de instrumentos com a informação inserida (Faraday, 2016).

1.10.4 Tenable SecurityCenter

O SecurityCenter é uma ferramenta desenvolvida pela mesma empresa responsável pela aplicação de testes de intrusão Nessus. Ambos complementam-se sendo que, o SecurityCenter é a interface que reúne as funcionalidades de *dashboard* e criação de relatórios.

As principais características do SecurityCenter são:

- Integração com outros sistemas;

- Sistema de alertas;
- Criação de relatórios configuráveis em HTML 5;
- *Dashboard;*
- Compatível com classificação CVSS v2;
- Criação de relatórios.;

Esta simbiose entre o Nessus e o SecurityCenter aplicam-se de forma excelente em auditorias a aplicações pois, através da capacidade de actualizações sobre vulnerabilidades existentes em aplicações, é possível aplicar validações às mesmas de forma automática e manter auditorias regulares face a progressos na segurança de tecnologias de informação. Um exemplo é a actualização com informação sobre vulnerabilidades *zero day* (Tenable, 2016).

A análise das ferramentas disponíveis no mercado e suas funcionalidades demonstrou que as características mais pertinentes se foram em:

- Interfaces simples e intuitivos;
- Informação caracterizadora de vulnerabilidades;
- Desenvolvimento de painéis de instrumentos com informação quantitativa para suporte à decisão;
- Importação de relatórios das ferramentas mais comuns de testes de intrusão;
- Integração com outros sistemas empresariais (ex: email);
- Criação de relatórios com detalhes e evidências das descobertas;

Os relatórios gerados pelas ferramentas ZAP e Burp foram também utilizados para identificar que características seriam pertinentes registar e que informação estaria disponível para integrar com informação adicional de vulnerabilidades e controlos associados que serviriam para identificar mitigações.

Capítulo 2 Auditoria de segurança web

Este capítulo descreve a metodologia seguida na auditoria realizado e todo o processo e todo o processo envolvente passando por autorizações, confidencialidade de informação e resultados obtidos e testes de intrusão.

2.1 Enquadramento legal

A confidencialidade da informação obtida nesta auditoria, quer pela utilização das aplicações testadas, quer pelos resultados dos testes de intrusão, é de carácter crítico. As aplicações auditadas encontram-se disponíveis a entidades externas e qualquer informação divulgada, pode comprometer ou ser usada para comprometer quer a instituição, quer qualquer entidade relacionada.

Uma preocupação consiste em garantir que não são divulgados detalhes para identificação das aplicações alvo de teste. Outra razão para ofuscar ou ocultar dados das aplicações testas, é a possibilidade de algumas poderem conter dados provenientes de sistemas disponíveis a utilizadores reais, por exemplo testes de performance. Por último, fornecer dados sobre exploração de vulnerabilidade de um sistema seria armar um atacante com a capacidade de comprometer a instituição e não é esse o comportamento nem objectivo de um auditor de segurança.

Como os testes realizados foram em redes e servidores internos da instituição e, havendo autorização formal para realização dos testes, apenas foi necessário acordar as questões de confidencialidade. Um exemplo do acordo de confidencialidade acordado pela instituição pode ser consultado em Anexo 1.

2.2 Âmbito de testes

Num primeiro contacto com a instituição para realização de uma auditoria de segurança, surgiram várias aplicações possíveis, que ainda não haviam sido sujeitas a testes de intrusão mas, já tinham sido revistas em termos de segurança. No entanto, a disponibilidade para realização da auditoria das mesmas estava dependente dos desenvolvimentos e testes a que estes ambientes estão sujeitos.

O cenário de testes apresentava-se com os pressupostos de um teste *grey box*, isto é, cujo algum conhecimento da infraestrutura e aplicações existe. As aplicações a testar, fazem parte de uma amostra de conveniência mediante a disponibilidade e calendário de desenvolvimento. Dado que as aplicações se encontram em manutenção ou mesmo, evolução, nem sempre houve disponibilidade para aceder e testar as mesmas devido à possibilidade de comprometer os ambientes. No entanto, os testes foram sempre antecipadamente planeados e com o devido acompanhamento para qualquer eventualidade de constrangimento.

Em termos de tecnologias, apenas para contextualizar os testes de intrusão face às informações que possam ser tomadas públicas, as aplicações testadas baseiam-se em tecnologias Microsoft ASP.Net com versões de *framework* compreendidas entre 2.0 e 4.0. As aplicações seguem normas de desenvolvimento seguro como protecção de *sql injection* através de consultas à base de dados através de *stored procedures* sem recurso a sql dinâmico. A infraestrutura utilizada é baseada em padrões *model driven domain*, o sistema de gestão de bases de dados é o Microsoft SQL Server e os servidores HTTP são o Internet Information Services. Os servidores HTTP e bases de dados estão em redes distintas. O acesso às aplicações é através de uma linha dedicada cedida aos utilizadores e com largura de banda de comunicações limitada. A Figura 2.1 ilustra a arquitectura aplicacional no âmbito com os respectivos intervenientes.

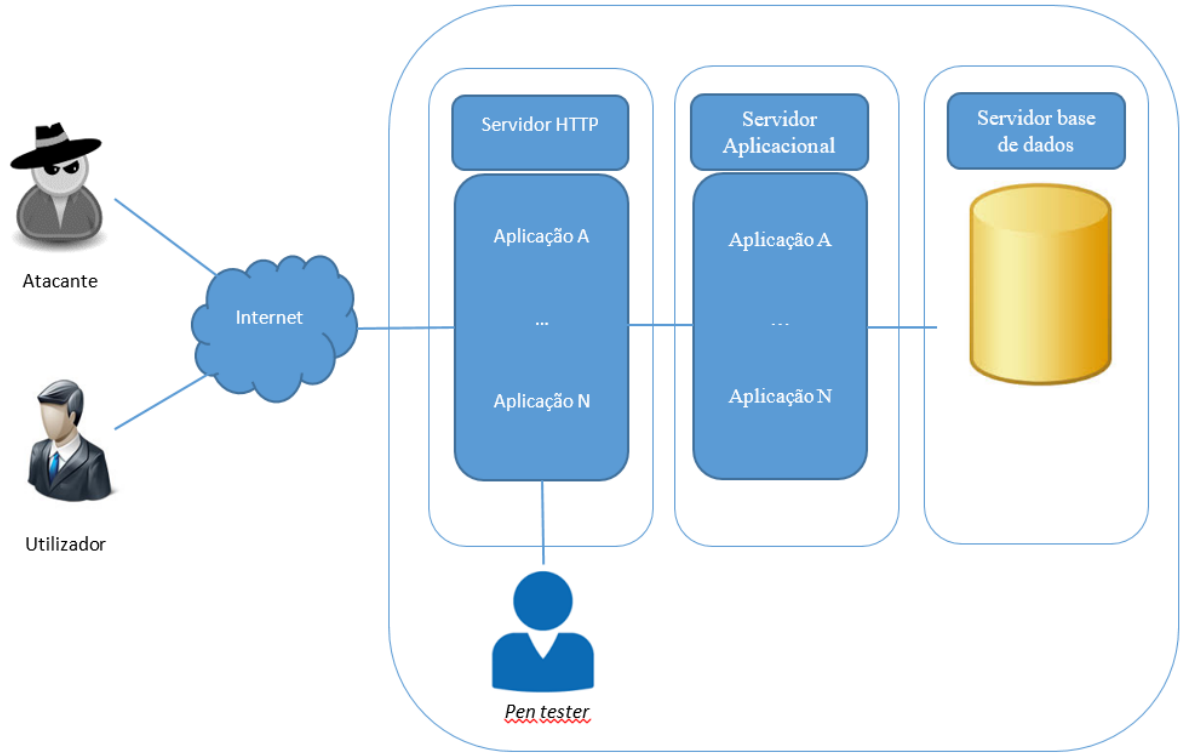


Figura 2.1 - Estrutura aplicacional e intervenientes

Nos ambientes disponibilizados ao público, o mecanismo de autenticação é baseado em autenticação federada. A autenticação é realizada fora do âmbito da aplicação à qual o utilizador tenta aceder, que é reencaminhado para a página de login. Um módulo HTTP responde a pedidos e de acordo com os dados fornecidos pelo utilizador, cria um *token* com permissões de aplicações e outras informações desse utilizador. Em caso de autenticação bem-sucedida, o utilizador é reencaminhado para o site inicial com o *token* gerado. As características dum utilizador passam a ser geridas por um fornecedor de identidade. A aplicação à qual este se autentica recebe o *token* com a informação necessária, ex: idade e perfil de utilizador. Desta forma, um utilizador necessita apenas de uma conta (Alrodham & Mitchell, 2011).

Devido à dualidade de mecanismo de autenticação nos ambientes de testes e nos ambientes disponíveis aos utilizadores, e face à segurança que o mecanismo de autenticação federada oferece, não foi contemplado no âmbito deste estudo. Nos ambientes disponíveis para testes, cada aplicação tem a sua página de autenticação por utilizador e password.

2.3 Metodologia

Esta secção descreve os passos realizados durante a auditoria, requisitos necessários, acordos com a instituição e opções tomadas.

2.3.1 Escolha de web scanners

A escolha dos *web scanners* utilizados, ZAP e Burp, foi baseada em estudos comparativos de utilização de ferramentas, apesar da fraca existência deste tipo de avaliações, foi possível obter *feedback* positivo face a estas duas ferramentas em estudos comparativos (Doupé, Cova, & Vigna, 2010), (Makino & Klyuev, 2015). O ZAP é uma ferramenta mantida pela OWASP e o Burp uma ferramenta comercial.

Numa tentativa de optimizar os resultados com a ferramenta Burp, pois esta é comercial, primeiro foram realizados testes com a versão gratuita para aprendizagem, seguindo de testes com a versão completa durante o período de experimentação. O ZAP não apresenta limitações e está disponível apenas na sua versão gratuita.

Houve tentativa de utilização de uma terceira ferramenta, o W3AF mas, esta demonstrou-se incapaz de realizar pedidos HTTP POST automáticos. A comunicação entre cliente e servidor numa aplicação web está assente num protocolo de comunicação HTTP constituído por vários métodos sendo o método GET e o método POST os mais comuns e utilizados. Geralmente, o GET está associado a obtenção de informação e o POST associado a envio de informação para o servidor nomeadamente, formulários de submissão de dados. Um pedido GET pode ser realizado de forma simples pelo *browser*, por exemplo, aceder a uma página introduzindo um URL na barra de endereço. Um pedido POST, requer conhecimentos de HTML ou utilização de ferramentas para o efeito. A incapacidade de gerar pedidos POST, inviabiliza todas as tentativas de comprometer uma aplicação via informação submetida do cliente. Não sendo capaz de realizar estes ataques de forma automática, por questões de tempo não se tornou viável utilizar esta ferramenta.

Poderia ser útil em testes bem delimitados, mas sempre configurados manualmente. Algo que as outras duas ferramentas eram capazes de fazer de forma simples e rápida. Não podemos concluir que o W3AF não é uma boa ferramenta de testes de intrusão, simplesmente não se adaptava a esta auditoria.

A utilização de *web scanners* permite testes de exploração de vectores de forma automática com inúmeros *plugins* já com ataques pré configurados assim como testes configurados manualmente e direcionados a um vector específico, portanto, permite um modo de ataque mais automático e outro mais inteligente em que se aliam as vantagens de rapidez dum web scanner e o conhecimento de uma metodologia de testes de intrusão.

Os testes foram realizados nas instalações da instituição e decorreram durante o período entre abril e julho fora do horário de expediente mediante disponibilidade do autor e autorização da instituição. A distribuição da utilização dos *web scanners* foi a seguinte apresentada na Figura 2.2.

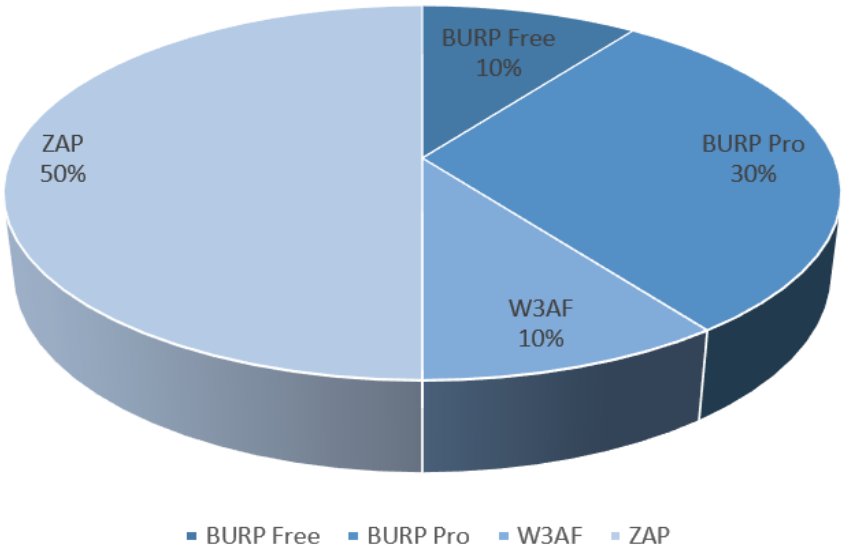


Figura 2.2 - Distribuição de utilização de web scanners

De forma a ganhar experiência na utilização destes *web scanners* foram experimentados contra aplicações criadas para a formação na área de testes de intrusão, não se trata de uma análise da precisão ou eficácia dos *web scanners* mas sim, aprendizagem da sua utilização. As aplicações utilizadas para este efeito foram o OWASP WebGoat.Net (Hoff, 2014) e o OWASP Mutillidae (OWASP Mutillidae, 2016).

Apesar deste processo de aprendizagem, não é possível prever o comportamento nas aplicações reais nem a duração dos testes. As dificuldades de utilização do *web scanner* W3AF mencionadas anteriormente são exemplo deste facto.

2.3.2 Auditoria

O processo de testes de intrusão, na fase de teste efectivo de sistemas ou aplicações consiste nas fases enumeradas no estado da arte, são as seguintes: reconhecimento, enumeração, ataque, aumento de privilégios, manutenção de privilégios e eliminação de *logs*.

Fazendo a ponte entre estre estes passos e a auditoria realizada, neste contexto de auditoria, não faz sentido eliminar quaisquer dados excepto artefactos utilizados nos testes (ex: dicionário com

utilizadores e palavras passe para testes de autenticação) ou quaisquer registos de vulnerabilidades que possam ser obtidos por terceiros. Aumento de privilégios e manutenção dos mesmos, segue o mesmo princípio. Foi definido como objectivo identificar e mitigar vulnerabilidades, não confirmar até que ponto poderiam comprometer o sistema.

A fase de recolha de informação, teve um âmbito mais reduzido dada a natureza da auditoria. Existe conhecimento da infraestrutura, da instituição e o acesso à aplicação é bastante restrito. Isto é, é necessária uma linha segura dedicada além dos mecanismos de autenticação.

Na fase de enumeração, houve tentativa de investigar e compreender como as vulnerabilidades se comportavam em aplicações com características semelhantes. Parte da enumeração, consistia também em analisar o guia OWASP Testing Guide. Esta ferramenta situa-se entre a enumeração e a realização de ataques ao sistema. Tratando-se de aplicações *web*, estava fora de âmbito testar portas ou serviços expostos pelos servidores, o principal meio de acesso era através do protocolo HTTP.

A auditoria às várias aplicações ocorreu de acordo com a disponibilidade dos ambientes das mesmas, uma vez que algumas ainda se encontram em manutenção ou em desenvolvimento. Sempre previamente planeadas e em contacto com os administradores de sistemas, foram realizadas durante um intervalo de 4 meses, várias auditorias que incluíam recolha de informação, enumeração de ataques, tentativas de ataque e confirmação de vulnerabilidades.

As principais acções de ataque executadas durante os testes de intrusão podem ser descritas pela Figura 2.3.

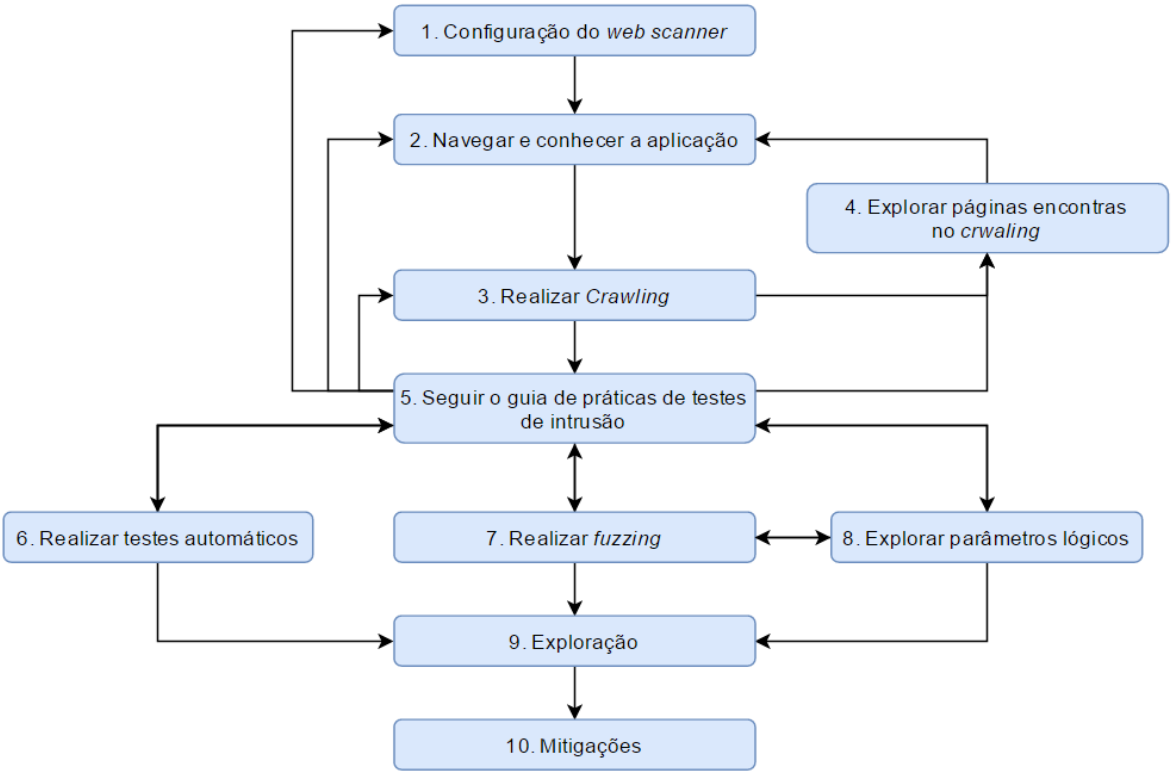


Figura 2.3 - Principais acções nos testes de intrusão

Estas acções realizadas durante a fase de ataque correspondem à seguinte descrição (Vieira & Serrão, 2016):

1. Configuração de *web scanner*: tratando-se dum teste com conhecimento da infraestrutura e tecnologias utilizadas, esta primeira permite que os testes sejam objectivos, permite definir o âmbito de captura pelo *web scanner* ignorando páginas externas ou outras que não se pretendam testar. O *web scanner* age como proxy e regista toda a informação transmitida entre cliente e servidor.
2. Navegar e conhecer aplicação: navegar pela aplicação, identificar como funciona e como está estruturada. Os *web scanners* utilizados constroem um modelo das aplicações através de vista de árvore que representam o mapa da aplicação web.
3. Realizar *crawling*: explorar a funcionalidade de *crawling* para que o *web scanner* possa mapear toda a aplicação. Esta funcionalidade é útil em percorrer links que não estão visíveis imediatamente, mas estão programados.
4. Explorar páginas encontradas no *crawling*: navegar pelas novas páginas encontradas, podendo repetir-se várias vezes juntamente com o passo anterior.
5. Seguir o guia de práticas de testes de intrusão: seguir o guia ou *checklist* de forma metódica para que nenhuma área seja esquecida.

6. Realizar testes automáticos: permitir ao *web scanner* ataques que reúnem exploração das vulnerabilidades mais frequentes. Não são ataques eficazes, mas são extremamente rápidos e em aplicações menos robustas podem produzir resultados imediatos.
7. Realizar *Fuzzing* nas aplicações: focar ataques a vulnerabilidades específicas através de configuração do *web scanner* alcançando um maior número de testes de forma rápida.
8. Explorar parâmetros lógicos: as aplicações lidam com variáveis passadas como parâmetros que podem ter um significado lógico para a aplicação. Os *web scanners* têm a capacidade de testar vários valores, mas não conseguem interpretar o seu significado lógico. Combinar as funcionalidades do *web scanner* com o conhecimento que o auditor reúne durante a análise da aplicação.
9. Exploração: fase de confirmação de exploração de vulnerabilidades encontradas e análise das implicações no sistema, com o pressuposto de não comprometer o sistema, reunir evidências físicas ou teóricas que uma vulnerabilidade pode ser realizada.
10. Mitigação e reporte: os *web scanners* têm como output sugestões de mitigação que devem ser entregues à instituição juntamente com informação adicional do auditor. O relatório será gerado através do *dashboard* criado onde as vulnerabilidades serão inseridas e classificadas.

Em qualquer momento durante a realização dos testes, seja constrangimento dos ambientes ou vulnerabilidades críticas detectadas, estas devem ser o mais rapidamente informadas aos responsáveis. Este fluxo é um processo flexível e cíclico podendo serem realizadas tantas iterações quantas o auditor entender serem necessárias.

Foi constatado que, apesar de ter sido realizada uma aprendizagem de utilização dos *web scanners* em aplicações para o efeito, sem conhecimento do modo de funcionamento das aplicações reais e como estão implementadas, não é possível prever dificuldades, a capacidade do *web scanner* testar a aplicação nem o tempo de execução de testes. A limitação da utilização do W3AF mencionada anteriormente só pôde ser detectada durante os testes e interacção com as aplicações.

Durante a realização da auditoria de segurança, houve sempre a preocupação de explorar as vulnerabilidades com cautela, não era objectivo verificar o impacto e danos que poderiam ser atingidos pela exploração mas sim, simplesmente confirmar a sua existência e verificar que poderia ser explorada. Verificação essa, caso houvesse confiança que não ocorreriam danos no sistema. Foi acordado com a instituição que a confirmação por prova de conceito seria

suficiente. Comprometer uma aplicação pode permitir identificar vulnerabilidades mas também causar constrangimentos na continuidade dos testes.

Em suma, podemos dividir o processo de testes de intrusão per si nas seguintes tarefas apresentadas na Figura 2.4.

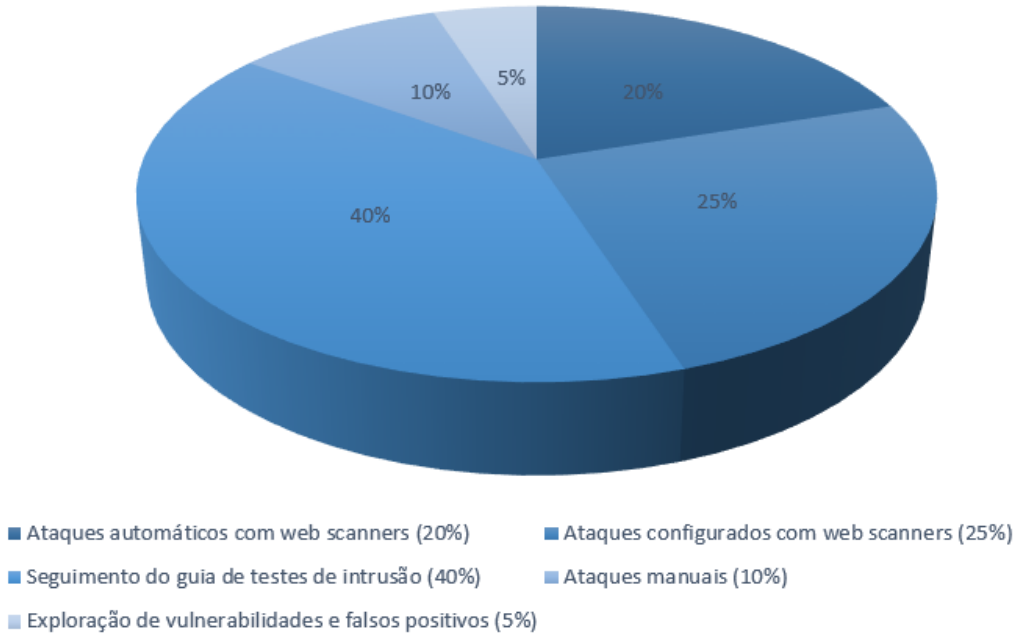


Figura 2.4 - Distribuição de tarefas de testes de intrusão per si

Ao tempo dedicado no seguimento do guia de testes, atribui-se o maior esforço. A confirmação de exploração de vulnerabilidades ou falsos positivos ocupou o menor esforço dos testes. A informação presente no guia de testes, a informação registada para análise pelos *web scanners* e a possibilidade de replicar cada evento permite resultados céleres.

Através da execução dos testes automáticos pelos *web scanners*, alcança-se uma diversidade de ataques que, ao percorrer o guia podem ser retomados, complementados ou excluídos consoante os resultados. O processo de auditoria com recurso a um guia de testes, torna-se assim um processo flexível e eficaz.

2.3.3 Confidencialidade de resultados

Dada a importância da informação tratada, quer dos resultados dos testes, quer dos dados das próprias aplicações, foram definidas medidas de segurança para lidar com a confidencialidade e protecção de informação.

Todas as evidências recolhidas de testes e documentos criados para análise das mesmas permaneceram em dois dispositivos de *hardware* externo, encriptados através do software BitLocker, algoritmo AES com chave de 256 bits. Sendo esta réplica uma questão de salvaguarda em caso de danos ou outra situação externa.

Após entrega de resultados, todos os dados, originais e cópias serão eliminados. Será assegurado que apenas a instituição fica com qualquer fonte de informação da auditoria realizada e seus resultados.

Capítulo 3 *Dashboard*

Este capítulo reúne as considerações de decisão para construção de uma ferramenta de apresentação de resultados e criação de relatórios de vulnerabilidades.

3.1 Objectivos

Um dos objectivos principais desta dissertação consiste em apresentar os resultados num *dashboard* ou painel de informações de forma a que os resultados pudessem ser analisados e mantidos nesta e em auditorias futuras. Algo que foi recebido pela instituição onde esta tese foi realizada com grande entusiasmo. Este objectivo aproxima-se também de um outro objectivo de realizar a análise de risco das vulnerabilidades encontradas. Através de informação compilada e organizada em gráficos e uma correcta classificação de risco face às vulnerabilidades encontradas, permitiria simultaneamente compreender a informação apresentada e decidir sobre a estratégia a tomar face ao estado de segurança das aplicações.

Sendo que a análise de risco pretendida consiste em utilizar a versão mais recente do CVSS v3.0 devido a ser um standard da indústria e também um dos métodos utilizados pela instituição, começou aqui a surgir a necessidade da criação dum *dashboard* de raiz. Entre as várias ferramentas disponíveis, nenhuma conjugava as seguintes características; classificação CVSS v3.0 (apenas a versão 2.0), sem custos e importação automática a partir dos relatórios gerados pelos *web scanners*.

A conjunção destes factores provou que seria melhor desenvolver um *dashboard* específico para o efeito. Reuniram-se então funcionalidades de ferramentas semelhantes existentes no mercado e concluiu-se que as características desta plataforma seriam:

- Interface apelativa;
- Interface baseada nos outputs dos *web scanners* e funcionalidades de *dashboards* existentes no mercado;
- Repositório de informação de vulnerabilidades focada no âmbito web;
- Apresentação de gráficos de compreensão simples;

- Expressasse o volume e estado de segurança das aplicações através de classificação CVSS v3.0;
- Permitisse gerar relatórios para técnicos e chefias.

Deste modo, a análise de risco e relatórios seriam gerados e apresentados pela aplicação. A vantagem final da construção dum *dashboard* de raiz, é a liberdade para a instituição instalar e utilizar sem limites proprietários além de puder adaptar a aplicação às suas necessidades.

O foco deste *dashboard*, é o âmbito *web* e vulnerabilidades inerentes. Trata-se da modelação da aplicação desenvolvida adaptada ás características das vulnerabilidades *web*. Vulnerabilidades essas, cujas informações não diferem assim tanto de uma vulnerabilidade doutro sistema e que poderiam ser encaixadas na estrutura construída. A classificação CVSS é compatível com qualquer tipo de vulnerabilidade seja ou não do âmbito *web*.

3.2 Arquitectura Dashboard

Foi decidido que a infraestrutura da aplicação desenvolvida seria em ambiente *web* precisamente para que pudesse ser disponibilizada numa intranet e para que o único requisito de utilização independentemente das tecnologias utilizadas no desenvolvimento, fosse um *web browser*.

As tecnologias utilizadas foram tecnologias Microsoft ASP.NET MVC, SQL Server para base de dados devido à familiaridade do autor com estas tecnologias. A designação atribuída a esta aplicação web foi de VDashboard, abreviatura para *Vulnerability Dashboard*.

Iniciando pelo modelo de dados de suporte para o *dashboard*, as entidades identificadas como chave são as seguintes presentes na Figura 3.1.

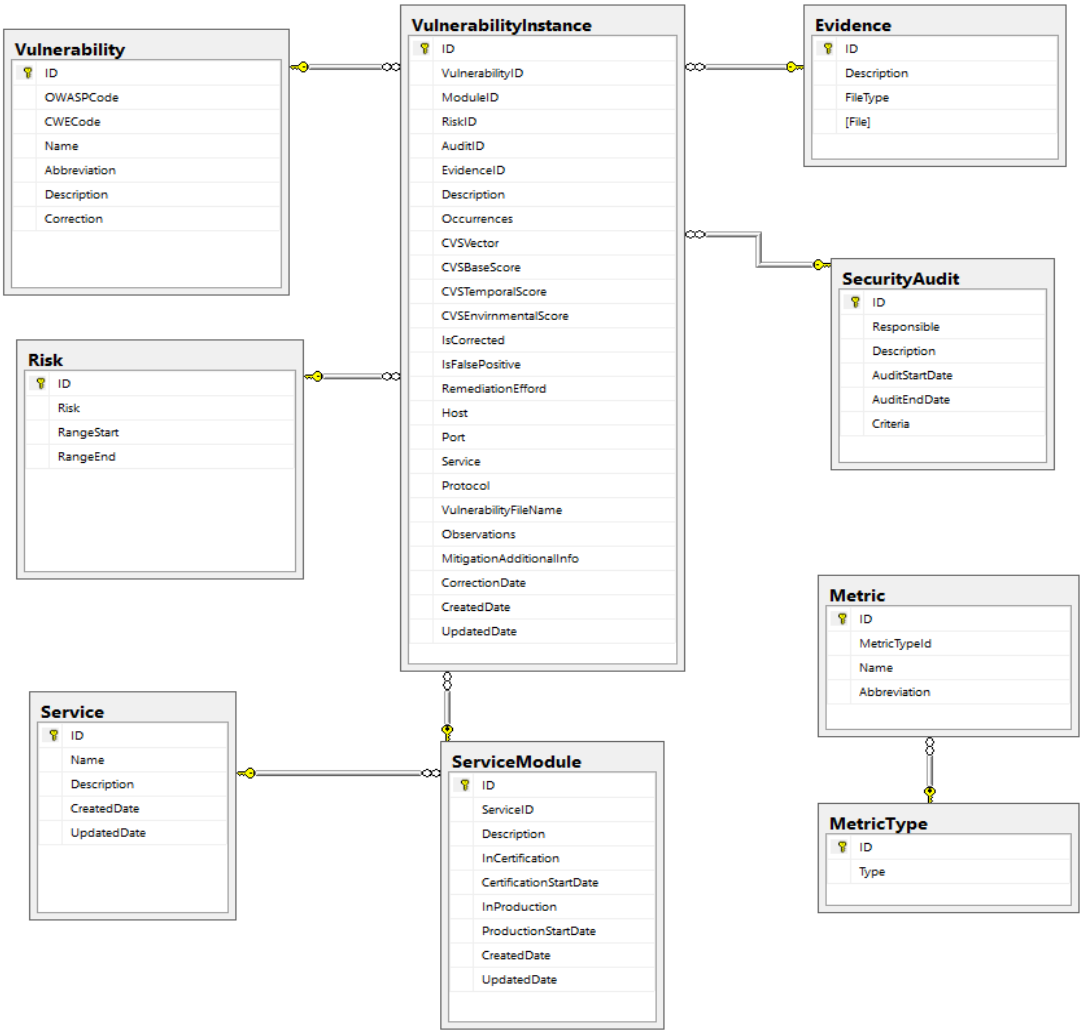


Figura 3.1 - Modelo Físico de VDashboard

VulnerabilityInstance: a raiz de agregação dos dados é a entidade VulnerabilityInstance que corresponde a uma instância de uma vulnerabilidade presente num módulo com uma classificação de risco e um determinado grau de severidade. Nesta entidade estão presentes atributos que definem uma vulnerabilidade desde a sua identificação até data de conclusão. Tem presente notas de mitigação, registo temporal de identificação, auditoria associada à identificação, ficheiro de evidências para análise posterior. Permite definir o número de ocorrências para evitar registar inúmeras instâncias com os mesmos dados. Permite também classificações como corrigida ou falso positivo.

Vulnerability: nesta entidade, são identificadas as vulnerabilidades existentes e classificadas pelas entidades de referência na segurança *web*. São na verdade dados estáticos que servem de suporte para uma instância de uma vulnerabilidade. Definem o contexto da variável e sugestões

de mitigação genéricas. Têm também categorização de códigos de vulnerabilidade para mapeamento no acto de importação de relatórios de *web scanners*.

Evidence: esta entidade consiste numa evidência dos testes realizados como por exemplo o ficheiro de relatório de uma auditoria ou um conjunto de imagens para exploração da vulnerabilidade. Permite associar a uma instância mais informação sobre uma vulnerabilidade noutro formato que não, texto.

Security Audit: ou auditoria de segurança permite criar auditorias e suas características como responsável e datas de realização. É posteriormente associada a uma instância e pode servir como ponto de partida para criação de inúmeras estatísticas.

Service: em conjunto com o Module permitem classificar o âmbito em que se encontram inseridas as vulnerabilidades. *Service* poderá definir um projecto numa instituição mas, para que não sejam criadas ambiguidades com projecto e fases do ciclo de vida optou-se pela designação *Service*. Trata-se de uma categoria e subcategoria.

Module: define uma subcategoria de *Service*, um projecto pode por exemplo ter vários componentes *web* desenvolvidos em diferentes momentos no tempo aos quais corresponde a entidade *Module*.

MetricType: consiste nos grupos de métricas da classificação CVSS v3.0, métricas base, temporais e ambiente.

Metric: consiste nas métricas existentes para cada grupo de métricas que classificam uma vulnerabilidade na versão CVSS v3.0.

Os resultados e gráficos apresentados no V*Dashboard*, foram construídos com o propósito de apresentar o momento presente, como tal, têm como base três características fundamentais das instâncias de vulnerabilidades, a pontuação de severidade, a identificação de correção e se é falso positivo ou não. As vulnerabilidades corrigidas ou falsos positivos são importantes mas, para efeitos de histórico. Para priorização de ações de mitigação e estado actual do nível de segurança das aplicações, as instâncias "abertas", isto é, por corrigir, são as determinantes.

3.3 Funcionalidades VDashboard

Seguidamente, apresentamos a aplicação e as considerações tomadas no desenvolvimento do VDashboard. A página inicial do *dashboard* foi definida como sendo a mais importante para

apresentar informação. As restantes páginas são fundamentais mas, a sua função é alimentar a aplicação com dados, na página inicial, é apresentada uma fotografia do estado de segurança até ao momento sob a forma de elementos de rápida leitura e compreensão, isto é, gráficos.

3.3.1 Painel de decisão

Caracteriza-se a página inicial como o painel de decisão, onde surgem as informações sobre as vulnerabilidades registadas no *dashboard*.

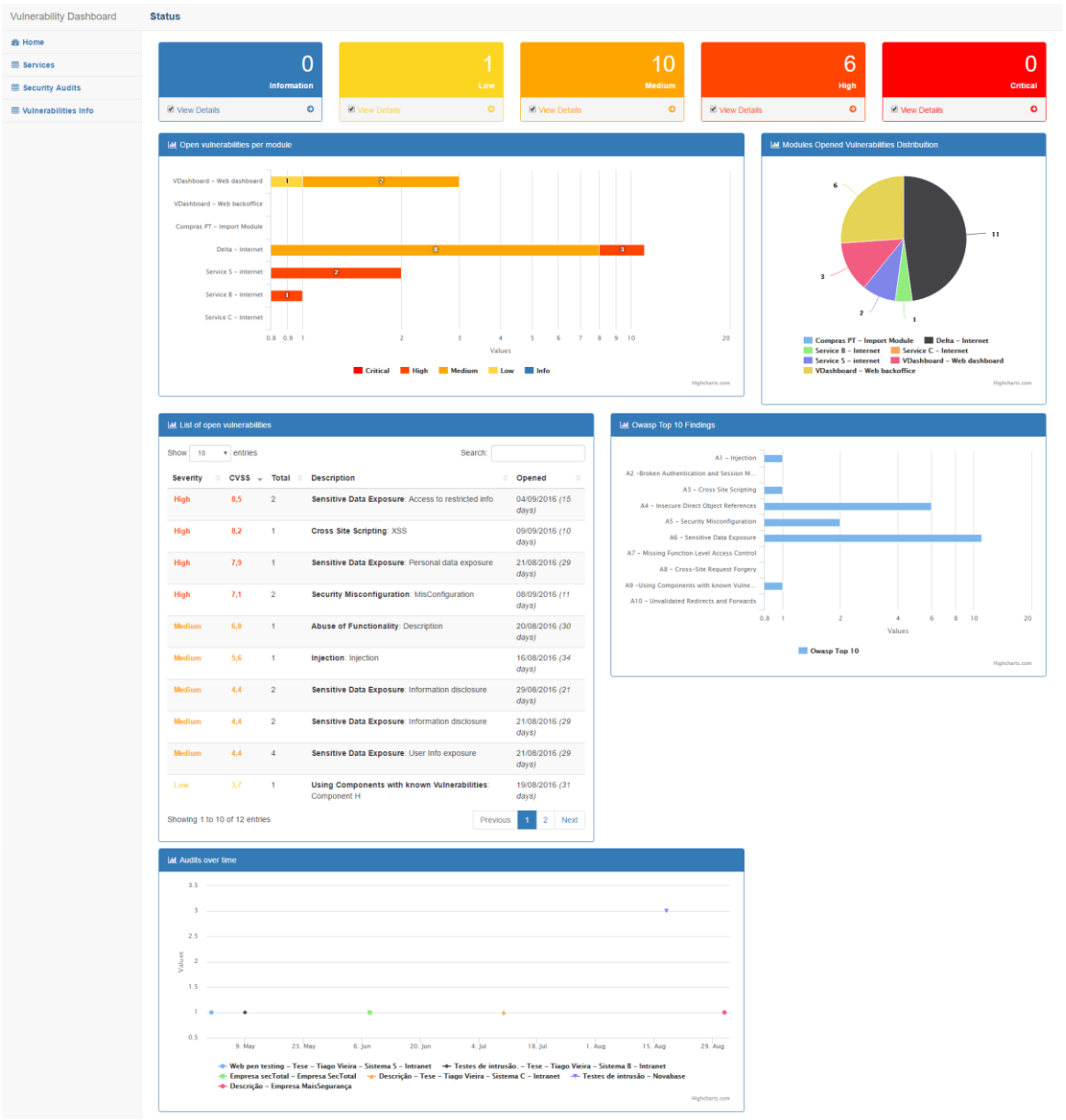


Figura 3.2 – VDashboard - Página inicial

A Figura 3.2 apresenta a o ecrã inicial da aplicação, consiste na informação apresentada ao utilizador ao aceder ao VDashboard. Foi objectivo que esta página apresentasse o estado de

segurança actual dos módulos e auditorias registados na aplicação e por corrigir. As informações disponíveis são, no topo, o total de instâncias, agrupadas por severidade desde Informação (0), risco baixo (4), médio (10), alto (0) e crítico (0). À esquerda, o menu com as opções de navegação da aplicação para alteração de contexto. Permite navegar para a lista de serviços (Services), lista de auditorias (Security Audits) e lista de vulnerabilidades (Vulnerabilities List), corresponde à informação estática sobre as mesmas, não corresponde às instâncias.

São criados quatro gráficos e uma tabela na página inicial. De cima para baixo, da esquerda para a direita, o primeiro gráfico apresenta a distribuição de instâncias de vulnerabilidade por módulo e severidade. O segundo apresenta o total de instâncias abertas para cada módulo registado na aplicação. A tabela lista as vulnerabilidades abertas e suas informações principais como o tempo desde foram identificadas. O gráfico à direita da tabela, apresenta o total de instâncias constituintes do OWASP Top 10. O último gráfico, em baixo, reúne as auditorias realizadas no tempo aos sistemas registados.

3.3.2 Consulta e inserção de Serviços e Módulos

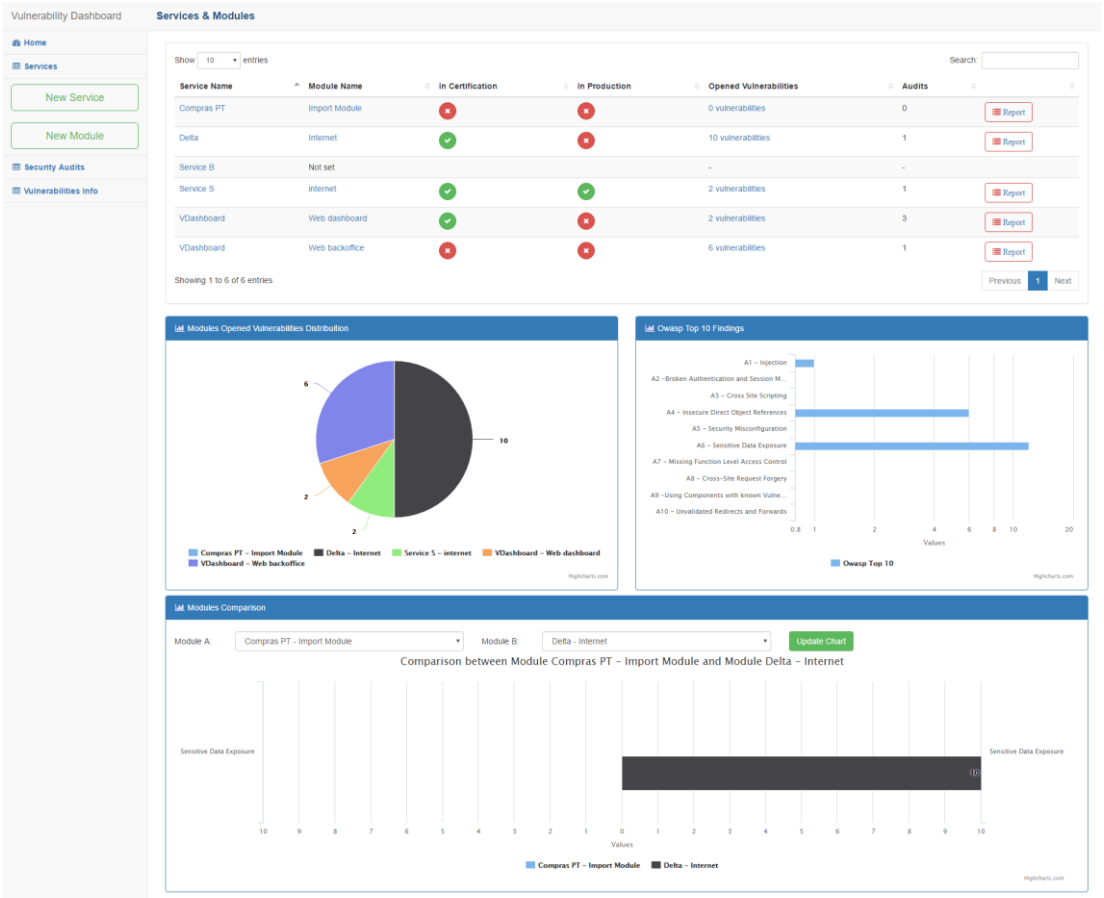


Figura 3.3 – VDashboard - Listagem serviços e módulos

A Figura 3.3, apresenta a lista de serviços e módulos associados assim como três gráficos. Em cima, a distribuição de instâncias por módulo e as vulnerabilidades constituintes do OWASP Top 10 encontradas e abertas. O terceiro gráfico permite realizar uma comparação de vulnerabilidades encontradas entre dois módulos. Esta opção poderá ser útil para comparar que sistemas merecem maior atenção.

No menu surgem duas novas opções, uma para adicionar serviços, outra para adicionar módulos. A listagem de serviços permite através de *link* e botão aceder a serviços e módulos para edição e gerar um relatório.

3.3.3 Consulta de vulnerabilidades

Selecionando o link de vulnerabilidades, podemos aceder à listagem de vulnerabilidades pertencentes a um módulo apresentado na Figura 3.4. Estes interfaces seguem a mesma estrutura, opções de contexto no menu lateral e links para o detalhe da vulnerabilidade na tabela.

A lista apresenta uma descrição da vulnerabilidade, as pontuações aos vários grupos de métricas e o número de ocorrências. É possível definir a instância da vulnerabilidade como corrigida ou como falso positivo a partir do controlo *checkbox* na lista. Esta página tem presente apenas um gráfico, as vulnerabilidades não corrigidas constituintes do OWASP Top 10 para o módulo selecionado.

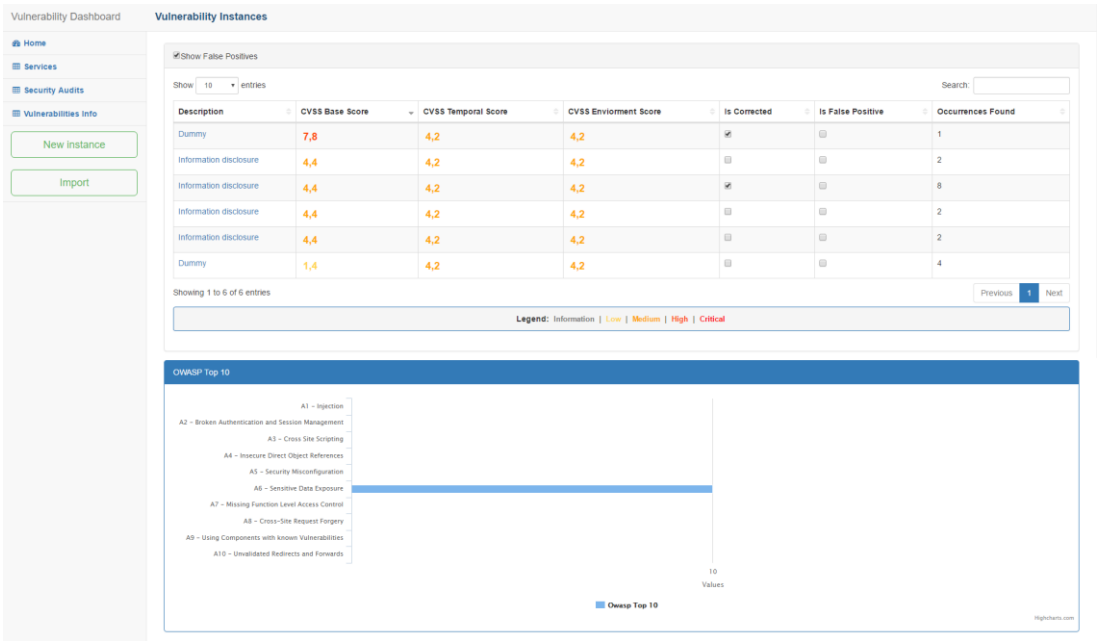


Figura 3.4 - VDashboard - Lista de Vulnerabilidades

As opções de contexto, "*New Instance*" e "*Import*" permitem criar uma nova instância e importar um relatório de uma das ferramentas utilizadas.

3.3.4 Registo de vulnerabilidades

A interface de criação de instância de vulnerabilidade é apresentada na Figura 3.5. É construído pelos campos descritivos como descrição, tipo de vulnerabilidade, ocorrências, notas de mitigação e a classificação CVSS v3.0 apresentada sob a forma de gráfico.

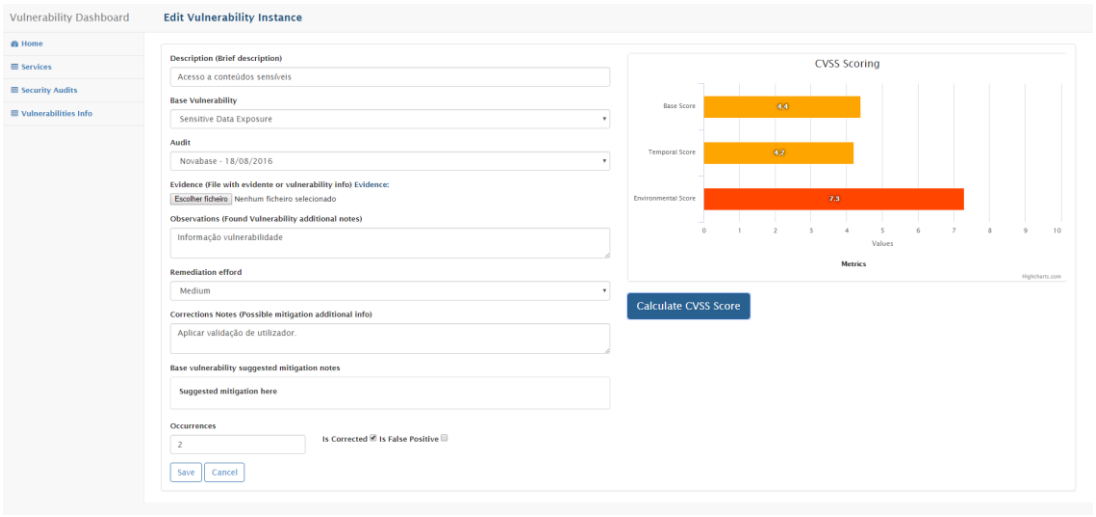


Figura 3.5 - VDashboard - Criar / Editar Instância de Vulnerabilidade

A edição das métricas da pontuação CVSS surge ao carregar no botão *Calculate CVSS Score* sob a forma de *popup* devido à limitação de espaço na página. Ao actualizar as métricas, o gráfico de pontuação é actualizado. Este ecrã é apresentado na Figura 3.6

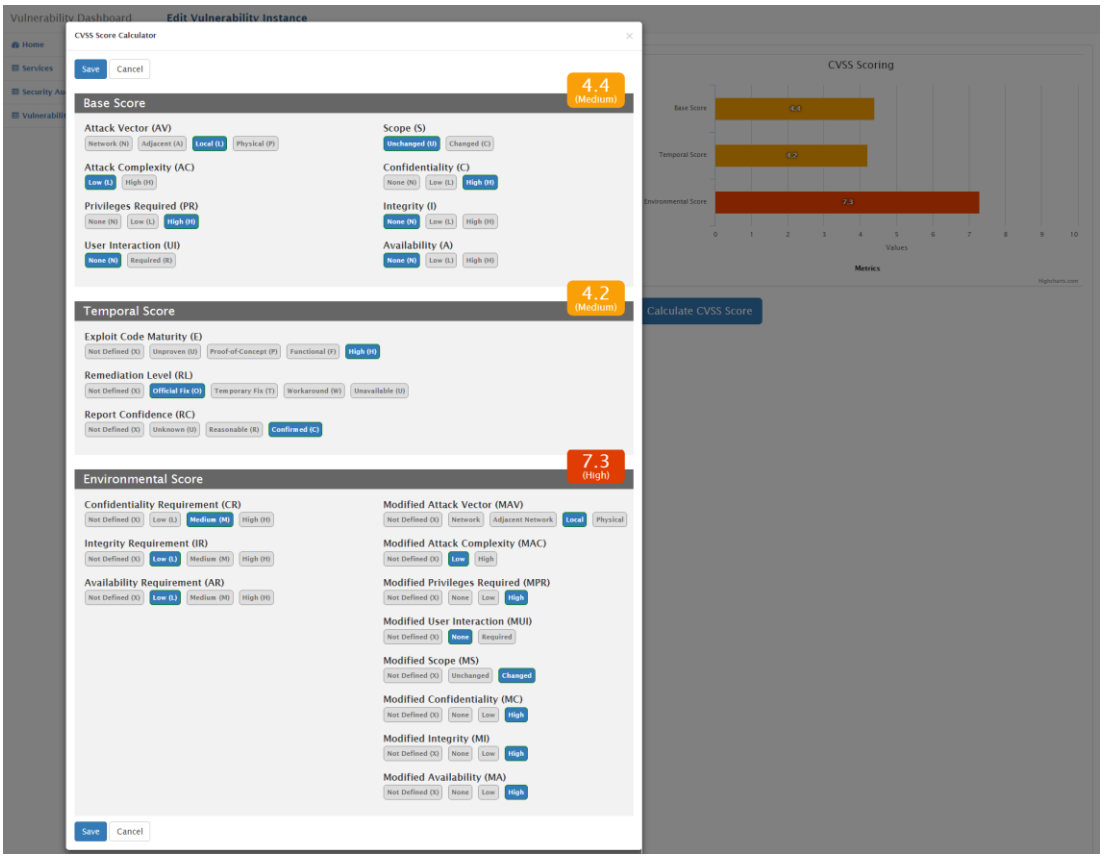


Figura 3.6 - VDashboard - Edição de classificação CVSS

3.3.5 Relatórios

Um elemento fundamental desta aplicação, é a criação de relatórios. O *VDashboard* permite a criação de relatórios a partir de uma auditoria ou a partir da selecção de um módulo. Ambas as opções apresentam a listagem de vulnerabilidades associadas para que o utilizador escolha as que pretende apresentar. A selecção de vulnerabilidades para geração de relatório é apresentada na Figura 3.7.

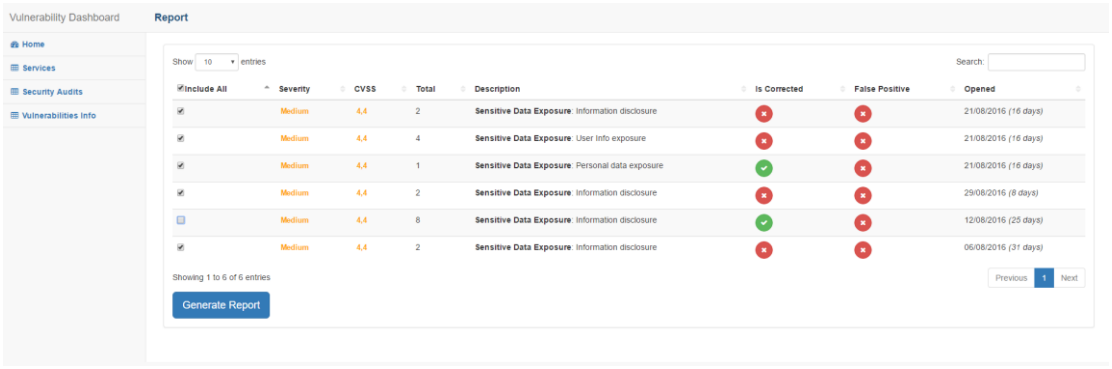


Figura 3.7 - VDashboard - Selecção de vulnerabilidades para relatório

O aspecto final do relatório é o seguinte apresentado na Figura 3.8. Este relatório é dividido em três partes. A primeira introduz o contexto do relatório e uma descrição sumária do mesmo. Descreve se se trata dum relatório baseado num módulo ou num relatório baseado numa auditoria, a data de geração e um gráfico com os totais das instâncias de vulnerabilidades presentes no relatório.

A segunda parte do relatório é uma descrição dos conteúdos presentes, isto é, quis as vulnerabilidades identificadas e presentes no relatório ordenadas por severidade. A parte final corresponde à listagem e detalhe de cada vulnerabilidade. Cada instância apresenta as pontuações para os grupos de métricas CVSS, a descrição da vulnerabilidade, a sugestão de mitigação, uma estimativa de esforço para resolução e a informação detalhada das métricas CVSS.

Este conjunto de informação, permite a um técnico e a um gestor com conhecimento técnico limitado obterem informações para a priorização ou mitigação das vulnerabilidades encontradas.

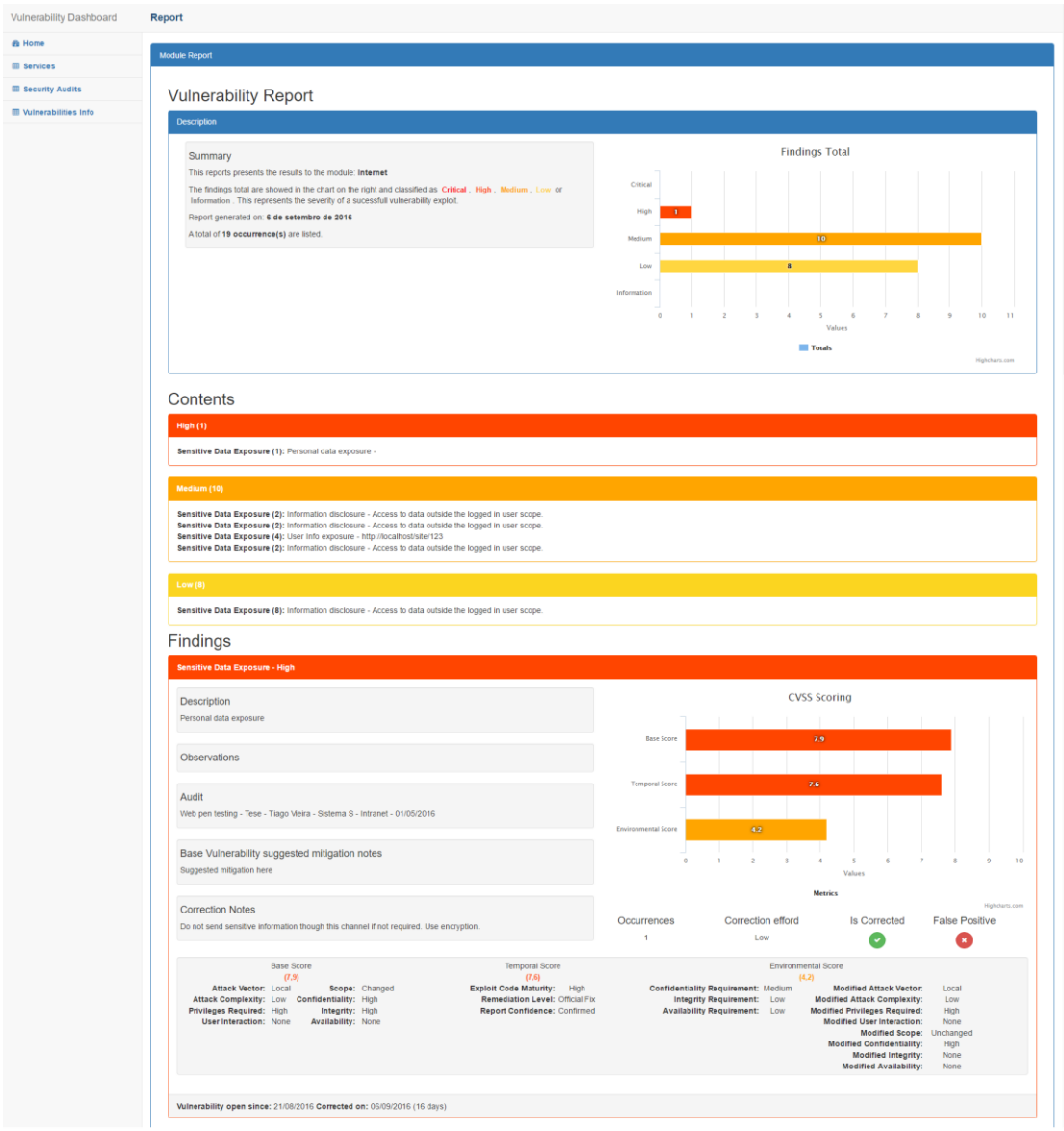


Figura 3.8 - VDashbord - Relatório de Vulnerabilidades (parcial)

3.3.6 Importação de relatórios de *web scanners*

A última funcionalidade presente no *VDashboard* é a possibilidade de importar relatórios das ferramentas utilizadas. A importação simplifica e reduz o erro de registo de vulnerabilidades. Uma vez que nenhum dos *web scanners* analisados atribui uma classificação CVSS, esta terá que ser preenchida posteriormente pelo utilizador. A Figura 3.9 apresenta o *popup* para selecção de relatório para importação posterior pela aplicação.

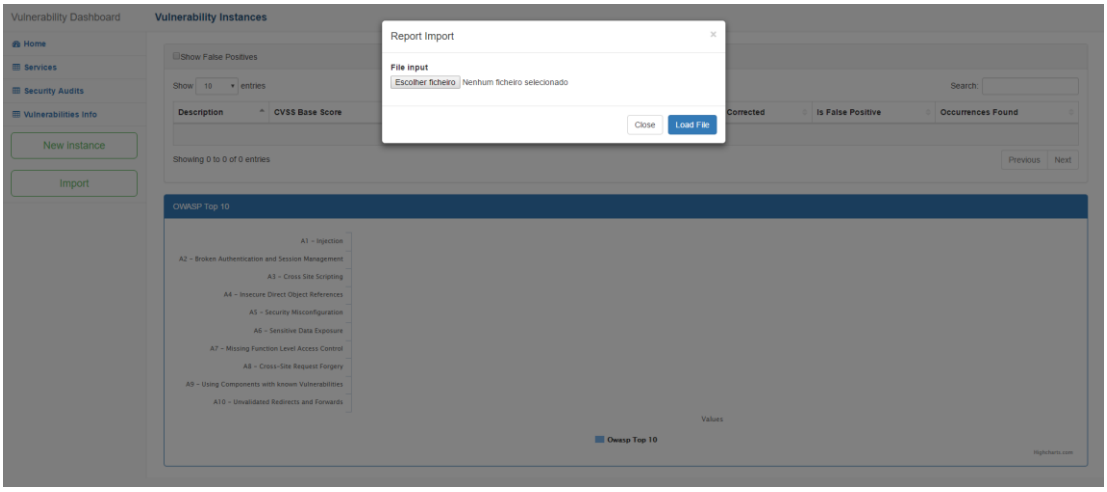


Figura 3.9 - VDashboard - Selecção relatório para importação

Após a selecção do ficheiro a importar, as instâncias são lidas e apresentadas ao utilizador para que este possa escolher quais as que quer importar. Por questões de limitação de ecrã, nem todos os conteúdos do relatório como sugestões de mitigação podem ser apresentados mas, no processo de importação essa informação é persistida de forma a fornecer o máximo de informação para a sua análise.

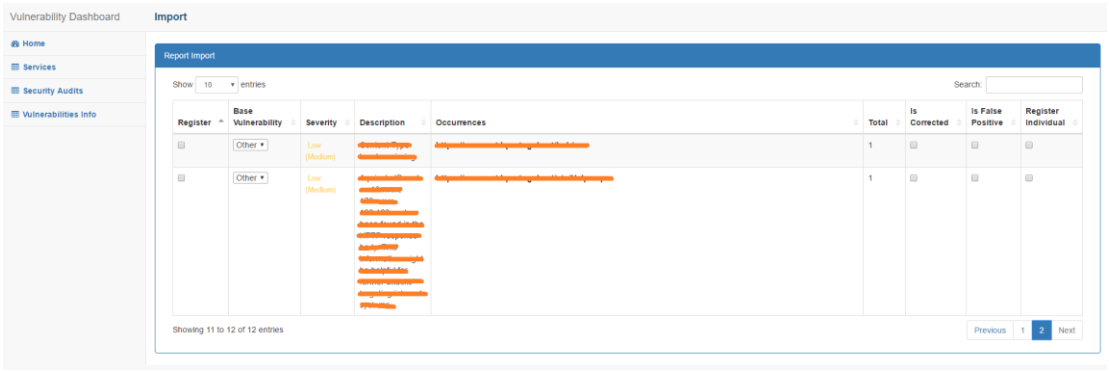


Figura 3.10 - VDashboard - Selecção de vulnerabilidades para importação

Os ecrãs apresentados anteriormente traduzem-se nas funcionalidades principais desenvolvidas para uma primeira versão dum *dashboard* de vulnerabilidades *web*.

Capítulo 4 Análise de Resultados

Neste capítulo, apresentamos os resultados obtidos durante os testes de intrusão às aplicações e seus módulos. Durante o processo de auditoria foram gerados mais de 2Gb de log em respostas e pedidos capturados pelos *scanners* ZAP e Burp. Este volume de informação evidencia as vantagens do uso destas ferramentas.

4.1 Resumo de Resultados Obtidos

Na Figura 4.1, são apresentadas as vulnerabilidades e informações nos testes realizados. Apesar da listagem de vulnerabilidades, não serão divulgados, por questões de confidencialidade, dados sobre os sistemas ou exploração das vulnerabilidades.

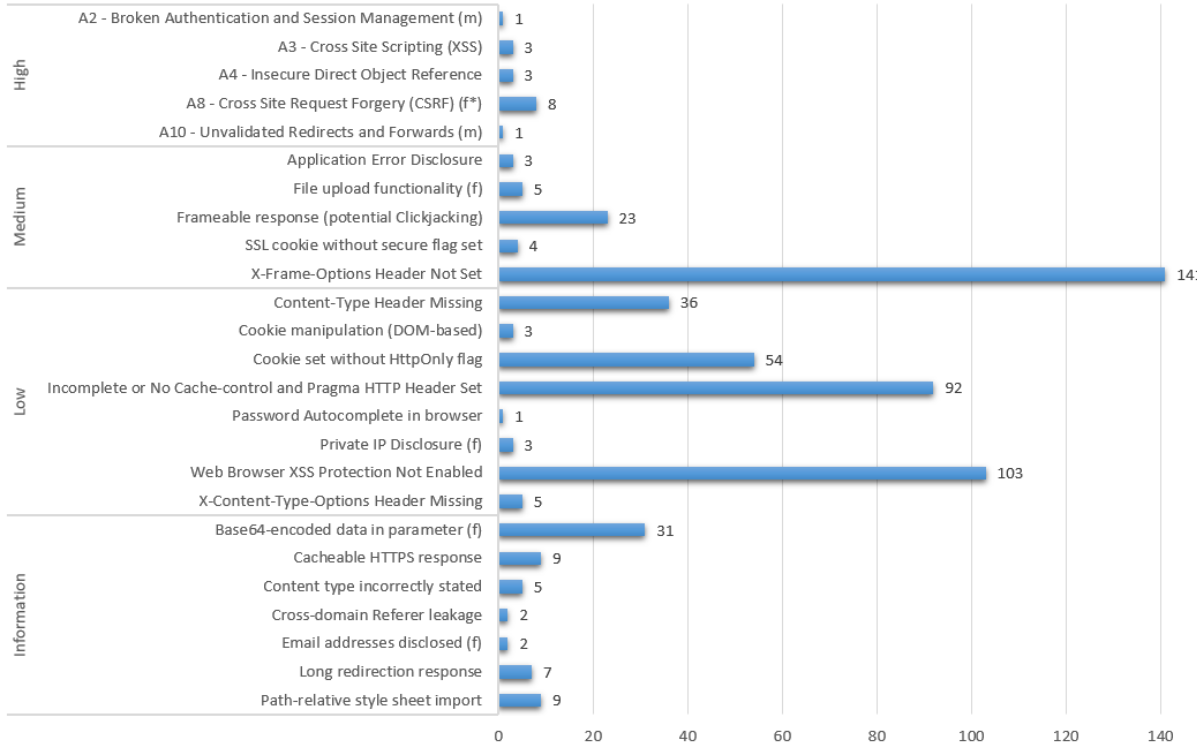


Figura 4.1 - Vulnerabilidades e informação identificadas na auditoria

Os resultados finais apresentam um total de 25 vulnerabilidades diferentes divididas por 4 aplicações com cerca de 554 instancias encontradas. No gráfico, as marcações na descrição das vulnerabilidades representam falsos positivos (f), falsos positivos parciais (f*) e detecções

manuais (m). O total de vulnerabilidades (excluindo os falsos positivos) com severidade alta foi de 14, 171 com severidade média e 294 de baixa severidade.

Os resultados da figura anterior representam a reunião das instâncias encontradas pelos testes automáticos, testes automáticos configurados e testes manuais, em seguida apresentamos a que *web scanners* estão associadas.

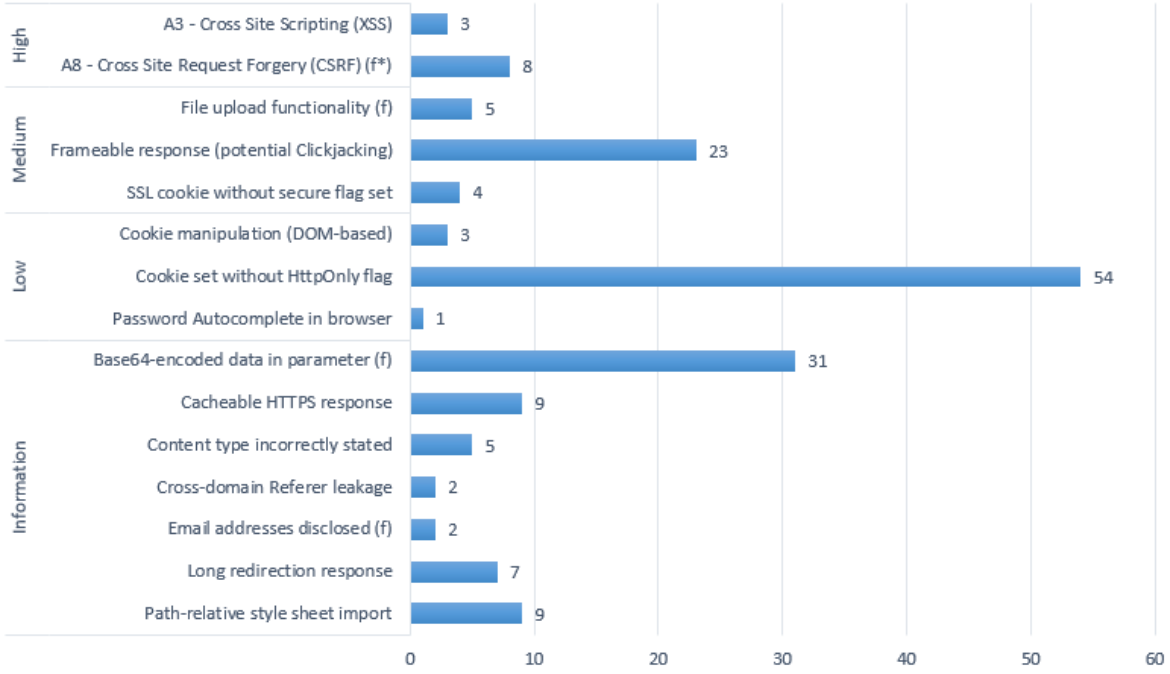


Figura 4.2 - Vulneraabilidades encontradas pelo web scanner Burp

A Figura 4.2 apresenta os resultados do Burp. de igual modo, na Figura 4.3 são apresentados os resultados do ZAP.

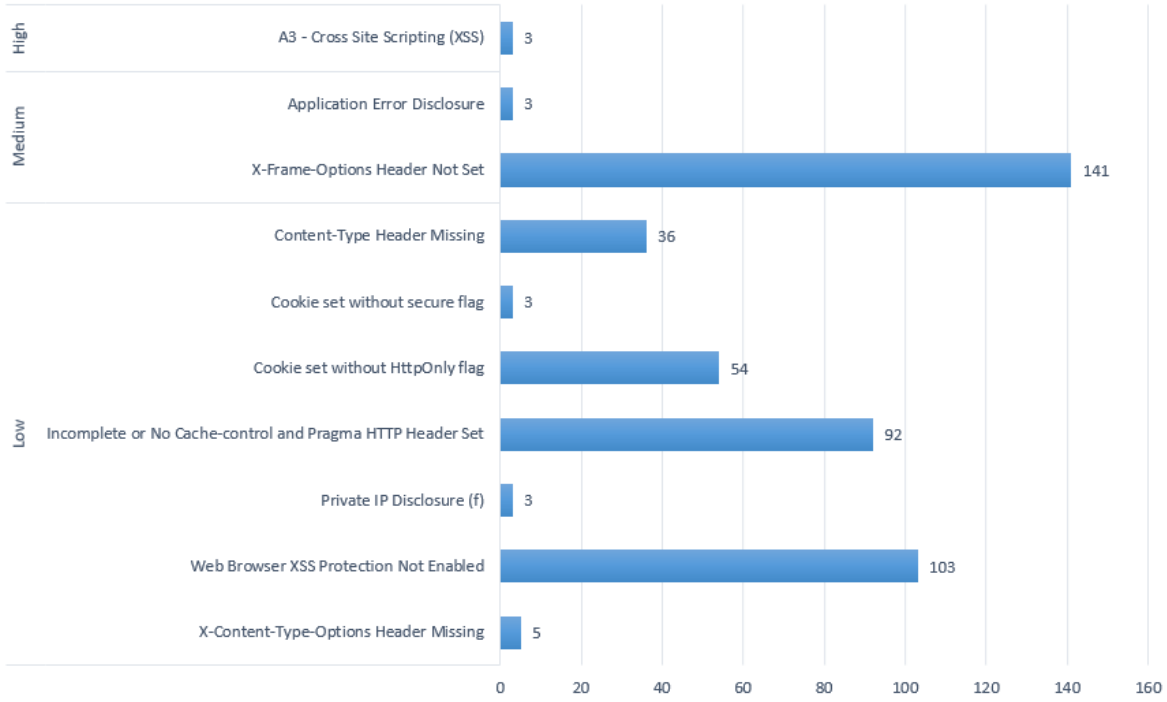


Figura 4.3 - Vulnerabilidades encontradas pelo web scanner ZAP

Na tabela seguinte estão reunidas as informações das diferenças mais evidentes entre os resultados encontrados:

	Burp	ZAP
Total de vulnerabilidades distintas	15	10
Severidade Alta	2 (11 instâncias)	1 (instância)
Severidade Média	3 (32 instâncias)	2 (144 instâncias)
Severidade Baixa	3 (58 instâncias)	7 (296 instâncias)
Informação	5 (65 instâncias)	0
Totais de instâncias	166	443

Tabela 1 - Totais de vulnerabilidades e instâncias encontradas

O número de vulnerabilidades identificadas pelo Burp é de quinze enquanto que o ZAP identificou dez distintas. Cinco das vulnerabilidades identificadas pelo Burp estão classificadas com o tipo informação.

Relativamente a vulnerabilidades mais severas, o Burp identifica duas distintas enquanto o ZAP apenas identificou uma. No entanto, para a vulnerabilidade *A8 – Cross Site Request Forgery*, foram identificados dois falsos positivos nos resultados do Burp.

O ZAP identificou um maior número de instâncias mas, estas diferenças também estão associadas aos tipos de vulnerabilidades. Podem ser explicadas pela reutilização de conteúdos no desenvolvimento das aplicações *web*.

Na Figura 4.4, apresentam-se as percentagens de vulnerabilidades associadas a um grau de severidade, grau esse, classificado pelo *web scanner*. As vulnerabilidades classificadas como informação não serão contempladas a partir deste ponto.

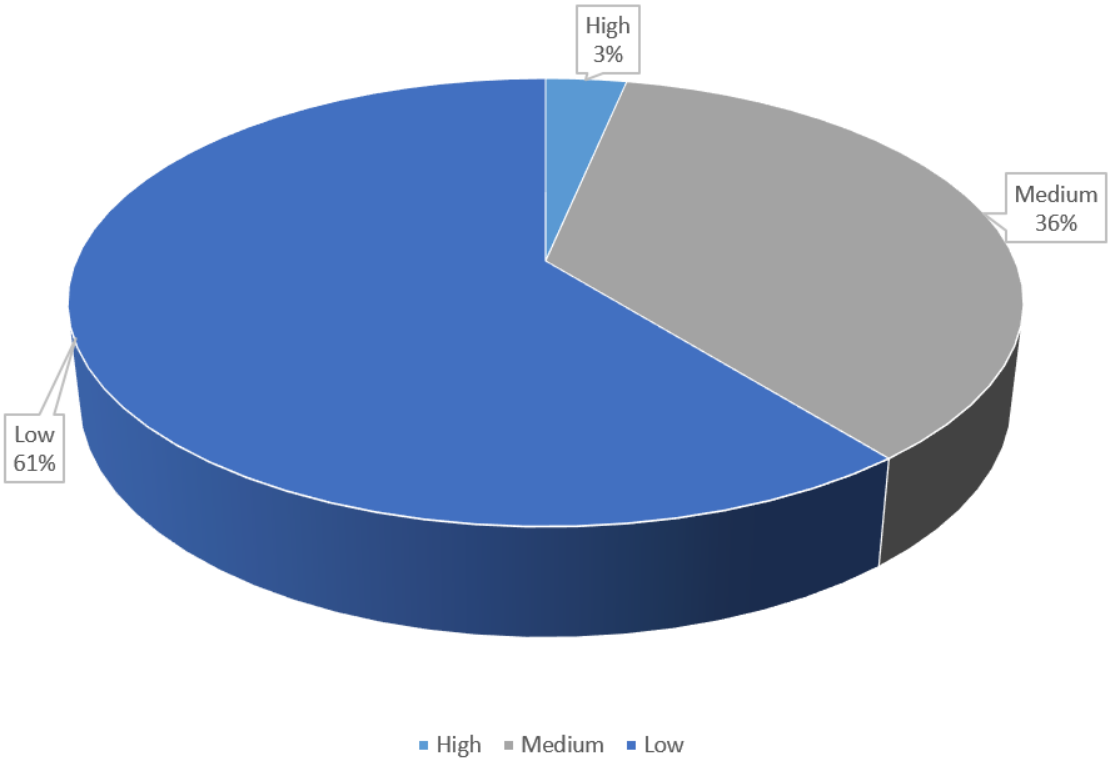


Figura 4.4 - Distribuição de severidade de vulnerabilidades encontradas

A distribuição de vulnerabilidades encontradas constituintes do grupo OWASP top 10 encontra-se na Figura 4.5. Estas são consideradas as vulnerabilidades com potenciais consequências mais críticas. Nestes resultados, não estão contabilizados os falsos positivos.

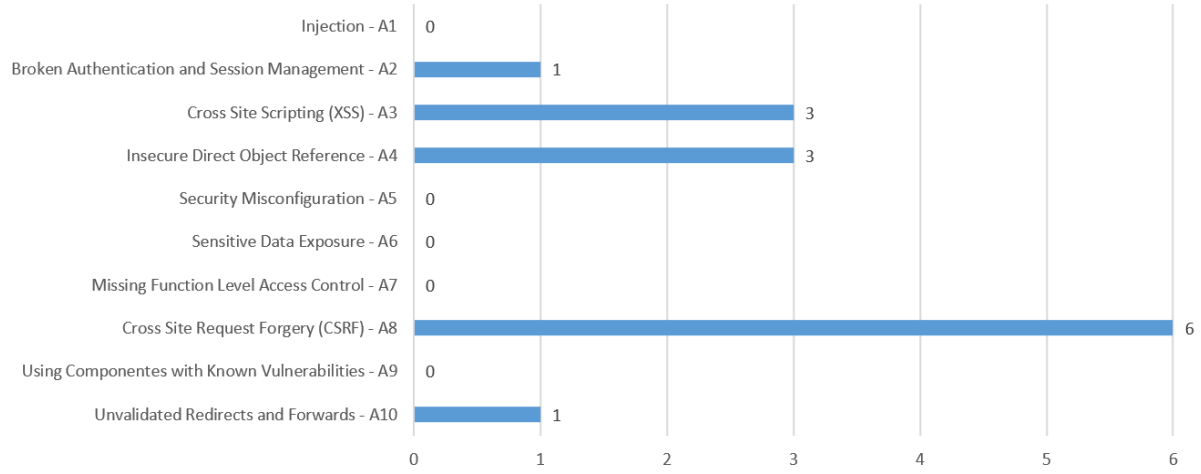


Figura 4.5 - Distribuição de vulnerabilidades no OWASP Top 10

Os resultados obtidos foram alavancados pelas funcionalidades dos *web scanners* utilizados e pelo seguimento da *framework OWASP Testing Guide v4* que permitiu uma célere e abrangente cobertura das aplicações. Em suma, estes resultados são resultado da combinação de testes automáticos com relativa configuração nos *web scanners*, testes automáticos orientados a temas de segurança, isto é testes orientados a um tipo de vulnerabilidade em específico e por último, testes manuais também orientados por vulnerabilidade específica.

Os resultados individuais das aplicações 1, 2 e 3 são os seguintes disponíveis nas figuras: Figura 4.6, Figura 4.7 e Figura 4.8.

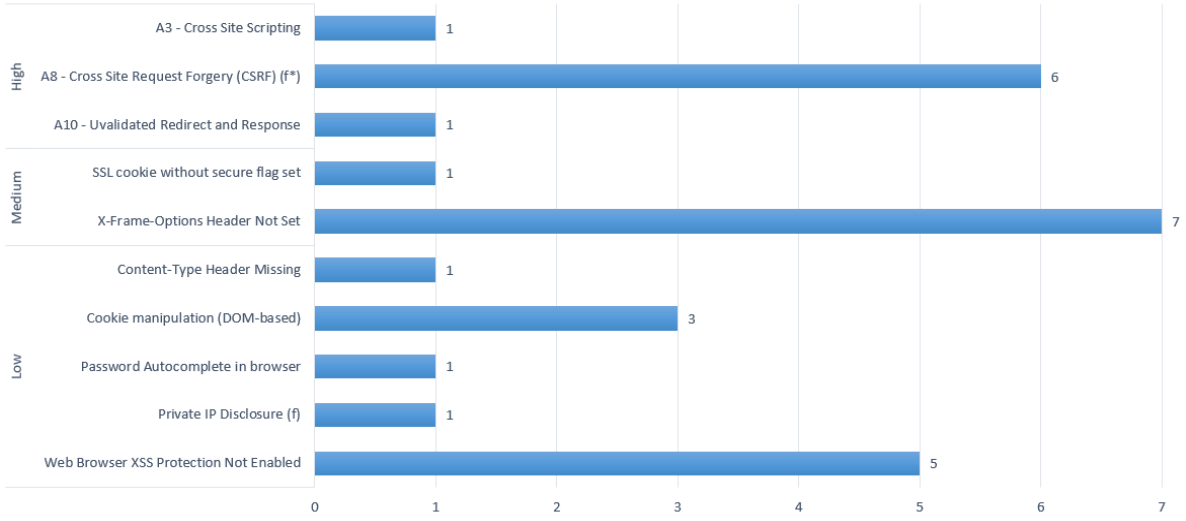


Figura 4.6 - Vulnerabilidades detectadas nos Sistemas 1A e 1B

Os sistemas 1A e 1B são apresentados em conjunto devido a terem uma base semelhante. A distribuição de vulnerabilidades críticas entre sistemas é considerada deminuta mas, os seus riscos não devem ser menosprezados.

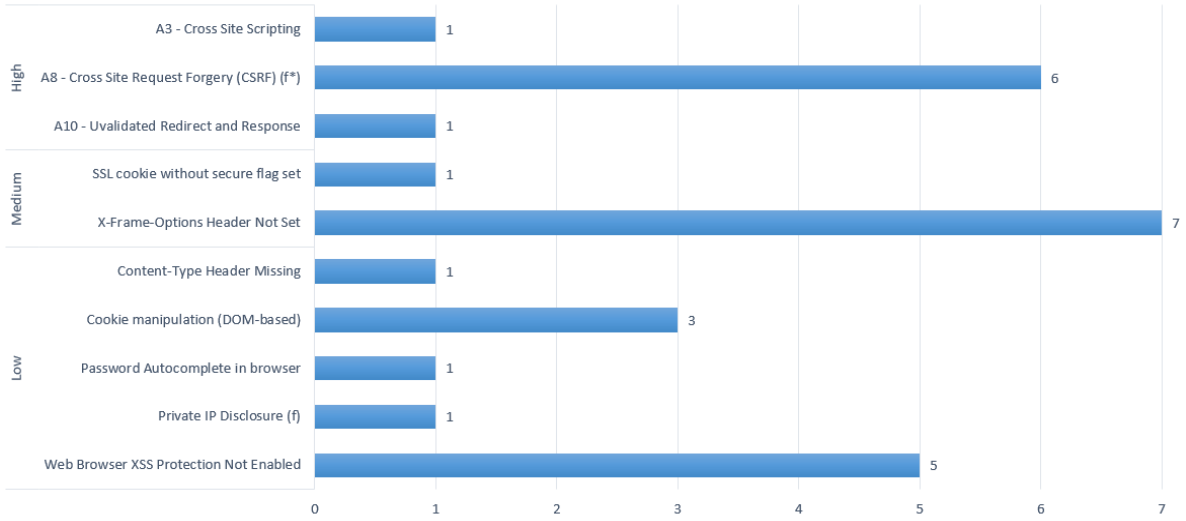


Figura 4.7 - Vulnerabilidades detectadas no Sistema 2

Há que considerar na análise e confirmação de vulnerabilidades, é provável que, em aplicações desenvolvidas por camadas e com considerações de segurança, existam vários níveis de validação. Não obstande, devem ser analisadas.

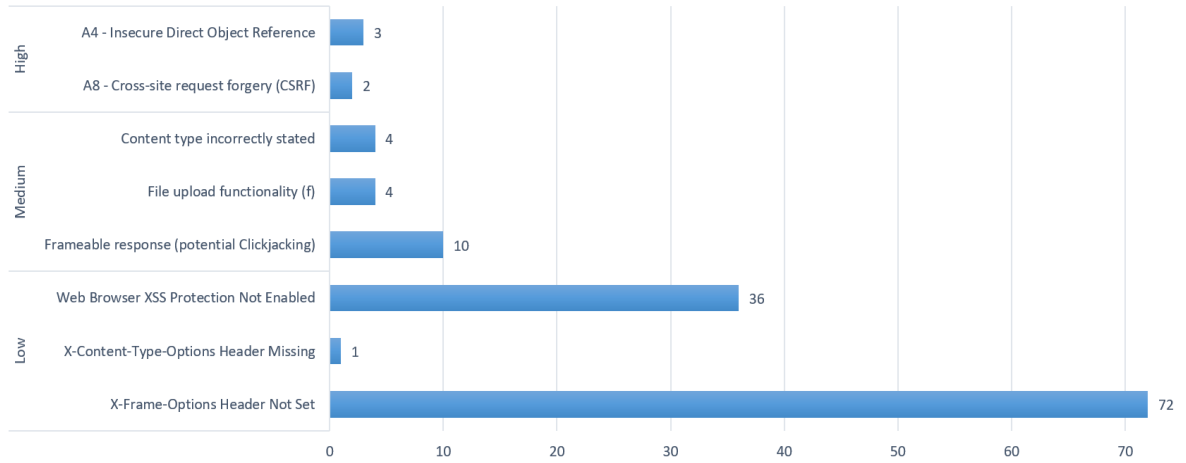


Figura 4.8 - Vulnerabilidades detectadas no Sistema 3

Analisando os gráficos, é possível encontrar bastantes avisos ou instâncias de severidade baixa, essas devem ser analisadas com o intuito de perceber se podem escalar ou serem aproveitadas por outras vulnerabilidades. Nomeadamente, a proteção contra XSS activa (*Web Browser XSS Protection Not Enabled*) pode ser explorada por uma vulnerabilidade como XSS. Esta vulnerabilidade pode ser mitigada através de configuração ou implementada manualmente a nível aplicacional.

Analisando o total de vulnerabilidades mais críticas em cada sistema testado, podemos concluir que o número é bastante reduzido, não pode, no entanto, concluir-se sem análise das vulnerabilidades que são aplicações seguras, pois o objectivo dum atacante é encontrar apenas uma vulnerabilidade. Por essa razão, só com uma análise de risco podem ser quantificados os riscos.

A2 - Broken Authentication and Session Management: autenticação é um mecanismo de segurança utilizados em inúmeros sistemas e também em aplicações *web*, é a forma que uma entidade tem de provar que tem acesso à aplicação. A sessão em aplicações *web* é o que permite ao servidor HTTP associar pedidos de um cliente aos seus dados e conta para que este não necessite realizar autenticação em cada acção. Um atacante que consiga credenciais dum utilizador ou sequestrar a sessão em utilização, pode conseguir acesso a informação sigilosa entre outros ataques. O principal impacto é na confidencialidade.

A3 – Cross Site Scripting (XSS): esta vulnerabilidade permite por exemplo, ataques em que código é executado no *browser* do cliente com execução de acções mal intencionadas contra o utilizador ou em nome do utilizador. Em conjugação por exemplo, com a vulnerabilidade acesso a *cookies* via *javascript*, um atacante poderia extraviar informação sensível. Esta vulnerabilidade está também associada por exemplo a ataques de engenharia social.

A4 – Insecure Direct Object Reference: acesso inseguro a referências de objectos traduz-se em acesso a informações ou activos aos quais não devia existir acesso. Pode ser causado por um erro de programação omitindo uma validação num determinado acesso.

A8- Cross Site Request Forgery (CSRF): este ataque é uma simulação de um pedido válido feito pelo utilizador no decorrer do uso de uma aplicação *web*. Consiste numa preparação por parte do atacante que envia de alguma forma esse ataque ao cliente. Caso o cliente tenha uma sessão para uma aplicação activa e a aplicação não esteja protegida, este pode executar acções às quais o atacante não tem acesso. Este cenário é de difícil preparação e exige que o atacante conheça a aplicação para construir o pedido falso. Tanto confidencialidade como integridade podem ser colocados em causa por um ataque desta natureza.

A10 – Unvalidated Redirects and Forwards: esta vulnerabilidade permite entre outras, a possibilidade de um utilizador ser reencaminhado do contexto aplicacional em que se encontra para um página mal-intencionada criada por um atacante. Página essa que pode ser uma réplica de um sistema conhecido e confiável.

Uma vulnerabilidade permite explorar vários vectores de ataque assim como um ataque pode ser executado pela exploração de várias vulnerabilidades. Para zelar pela segurança de uma instituição, deve ser dada merecida atenção a todas as instâncias e analisadas como um todo. Outra consideração interessante presente nos *web scanners* é uma análise de vulnerabilidades para vários *browsers*. Os *browsers* mais recentes são capazes de detectar e evitar determinados ataques mas, há determinadas aplicações *web* que apenas estão preparadas para *browsers* mais antigos e sem mecanismos de defesa.

4.2 Sistema de classificação

Esta secção, é responsável pela análise de risco e tratamento de vulnerabilidades encontradas durante a auditoria de segurança. A classificação de risco a utilizar foi definida como a *Common Vulnerability Scoring System* (CVSS v3.0). Como referido anteriormente, esta classificação tem como principal objectivo definir as características de uma vulnerabilidade de forma a descrevê-la desde o vector de ataque, passando por complexidade do mesmo e impacto na confidencialidade, disponibilidade e integridade.

Uma análise de risco permite identificar o nível de maturidade de segurança das aplicações testas e extrapolar sobre a segurança de outras aplicações desenvolvidas com as mesmas tecnologias e até mesmo noutras instituições da mesma área de negócio. Permite também compreender os riscos a que os activos estão sujeitos e priorizar mitigações ou outras medidas.

A classificação de risco divide-se em três grupos, o grupo base, com as características de uma vulnerabilidade, o grupo temporal, que define como as vulnerabilidades podem ser exploradas ao longo do tempo e o grupo de ambiente, que classifica a vulnerabilidade face aos activos comprometidos. O grupo de ambiente é o mais difícil de classificar devido ao desconhecimento dos activos e do seu valor. É a própria entidade que gere esse activos que os pode classificar melhor e identificar qual o impacto de uma exploração de vulnerabilidade que comprometa os mesmos.

A classificação de risco é na verdade a definição das métricas pertencentes à classificação de risco escolhida com a maior precisão possível. Uma breve descrição das métricas é descrita em seguida:

Métricas Base	
Vector de ataque	Define o meio de acesso à vulnerabilidade.
Complexidade de ataque	Define a complexidade de realização do ataque.
Privilégios necessários	Define os privilégios sobre a aplicação/sistema/rede necessários para realização do ataque.
Interação por parte de utilizador	Necessita de interação por parte dum utilizador.
Âmbito	Âmbito afectado pela exploração da vulnerabilidade.
Confidencialidade	Impacto sobre a confidencialidade.
Integridade	Impacto sobre a integridade.
Disponibilidade	Impacto sobre a disponibilidade,

Tabela 2 - Métricas base de classificação CVSS

As métricas base, são as características base da vulnerabilidade e devem ser preenchidas apenas com o conhecimento sobre as características da mesma, isto é, ignorando o valor dos activos comprometidos.

Métricas temporais	
Maturidade de exploração	Define a confiança na existência da vulnerabilidade.
Confiança de correção	Existência de forma de mitigação da vulnerabilidade.
Confiança de reprodução	Confiança na reprodução da vulnerabilidade.

Tabela 3 - Métricas temporais de classificação CVSS

As métricas temporais são importantes para compreensão da vulnerabilidade ao longo do tempo, permite classificar uma vulnerabilidade como possível de se explorar em situações em que não se consiga confirmar a existência. Todas estas métricas servem qualquer processo de auditoria de segurança, não só testes de intrusão. Por essa razão, as métricas permitem descrever uma vulnerabilidade apenas como hipótese. Outros métodos de análise de segurança podem não conseguir confirmar a exploração da vulnerabilidade, mas provar que pode acontecer, por exemplo, através de uma análise estática de código.

Há também que considerar que, os ambientes de testes muitas vezes têm limitações que os ambientes reais não têm, por exemplo, o volume de dados é inferior, a diversidade de informação também e nem todas as funcionalidades podem ser corretamente testadas porque contêm dados fictícios para testes específicos.

Métricas de ambiente	
Confidencialidade	Efeito na organização sob perda de confidencialidade.
Integridade	Efeito na organização sob perda de integridade.
Disponibilidade	Efeito na organização sob perda de disponibilidade.
Vector de ataque modificado	As seguintes métricas avaliam, tal como as métricas base, as características da vulnerabilidade mas, orientada ao valor dos activos comprometidos. Por essa razão, torna-se complicado por vezes a sua classificação por entidades externas à instituição ou em situações de testes *black box*. Estas métricas servem para identificar por exemplo se uma vulnerabilidade encontrada e classificada pelas métricas base como severa, é mitigada por outro mecanismo de defesa ou cujos activos não são tão valiosos mas, o auditor nem sempre tem esse conhecimento.
Complexidade de ataque modificado	
Privilégios necessários modificados	
Interação por parte do utilizador modificada	
Âmbito modificado	
Confidencialidade modificada	
Integridade modificada	
Disponibilidade modificada	

Tabela 4 - Métricas de ambiente de classificação CVSS

As métricas de ambiente são de cálculo difícil e pouco preciso devido ao desconhecimento do valor dos activos vulneráveis. Uma auditoria interna teria as condições ideias para correcta classificação das métricas do grupo ambiental. Apesar do âmbito do teste ser *grey box*, não existe o conhecimento do negócio para cálculo de pontuação na métrica de ambiente.

4.3 Análise de risco

Em seguida, apresentamos a classificação das vulnerabilidades mais severas encontradas.

	A3 - Cross site Scripting - XSS (3)	**A2 - Broken authentication and session management (1)**	**A4 - Insecure direct object reference (3)**	**A8 – Cross site request forgery - CSRF (6)**	**A10 - Unvalidated forward and response (1)**
Métricas Base					
Vector de ataque	Rede	Físico	Rede	Rede	Rede
Complexidade de ataque	Baixa	Baixa	Baixa	Baixa	Baixa
Privilégios necessários	Nenhuns	Nenhuns	Baixo	Nenhuns	Nenhuns
Interação p/ parte utilizador	Necessária	Desnecessária	Desnecessária	Necessária	Necessária
Âmbito	Alterado	Alterado	Alterado	Alterado	Alterado
Confidencialidade	Alta	Alta	Alta	Baixa	Baixa
Integridade	Baixa	Alta	Baixa	Alta	Baixa
Disponibilidade	Não afectada	Não afectada	Não afectada	Baixa	Baixa
Métricas Temporais					
Maturidade exploração	Alta	Alta	Alta	Funcional	Alta
Correção disponível	Alternativa	Alternativa	Alternativa	Resolução Oficial	Alternativa
Confiança reprodução	Confirmada	Confirmada	Confirmada	Razoável	Confirmada
Total (base)	**8,2 (Alta)**	**7,3 (Alta)**	**8,5 (Alta)**	**8,8 (Alta)**	**7,1 (Alta)**
Total (temporal)	**8,0 (Alta)**	**7,1 (Alta)**	**8,3 (Alta)**	**7,8 (Média)**	**6,9 (Média)**

Tabela 5 - Classificação CVSS de vulnerabilidades de severidade alta

As pontuações calculadas são valores elevados que classificam estas vulnerabilidades com severidade alta. Contudo são ataques que requerem interacção por parte do utilizador e, apesar de fáceis de identificar, não são tão simples de executar.

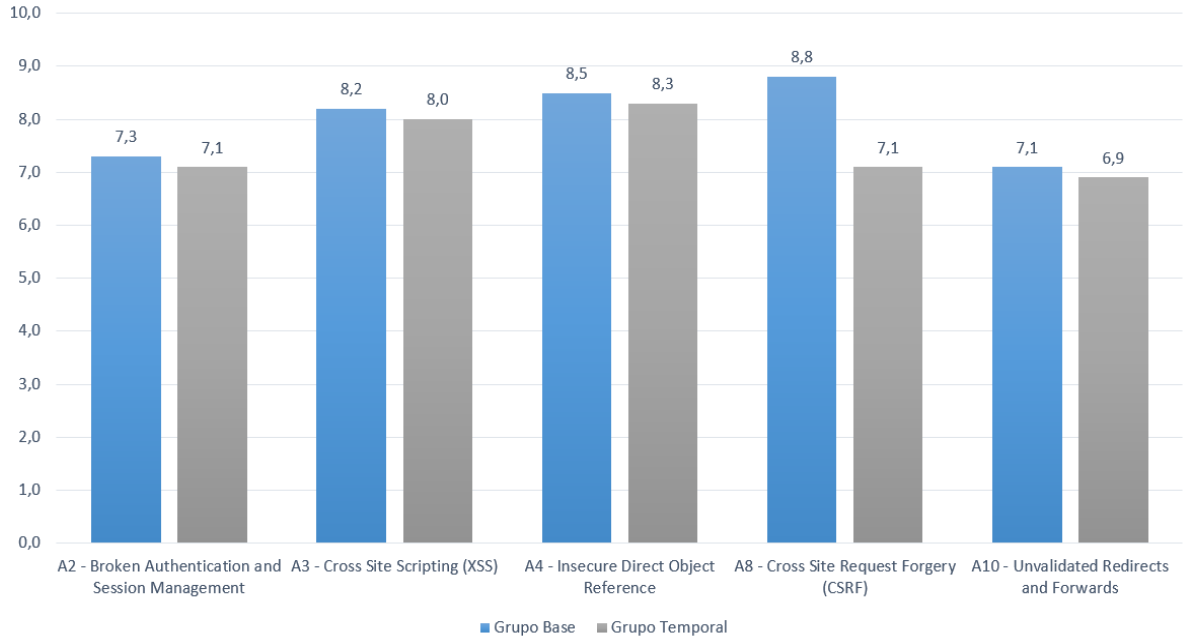


Figura 4.9 - Pontuação CVSS para vulnerabilidades de severidade alta (OWASP Top 10)

Relativamente a vulnerabilidades médias, temos a seguinte informação:

	Application error disclosure (3)	**Frameable response (potential Clickjacking) (23)**	**SSL cookie without secure flag set (4)**	**X-Frame option header not set (141)**
Métricas Base				
Vector de ataque	Rede	Rede	Rede	Rede
Complexidade de ataque	Baixa	Baixa	Baixa	Baixa
Privilégios necessários	Baixo	Nenhuns	Baixo	Nenhuns
Interação p/ parte utilizador	Desnecessária	Necessária	Necessária	Necessária
Âmbito	Inalterado	Alterado	Inalterado	Alterado
Confidencialidade	Baixa	Nenhuma	Baixa	Baixa
Integridade	Não afectada	Nenhuma	Não afectada	Nenhuma
Disponibilidade	Não afectada	Baixa	Não afectada	Baixa
Métricas Temporais				
Maturidade exploração	Alta	Funcional	Não provada	Funcional
Correção disponível	Alternativa	Resolução oficial	Resolução Oficial	Resolução Oficial
Confiança reprodução	Confirmada	Razoável	Razoável	Razoável
Total (base)	**4,3 (Alta)**	**4,2 (Alta)**	**3,5 (Alta)**	**3,5 (Alta)**
Total (temporal)	**4,2 (Alta)**	**4,2 (Alta)**	**3,3 (Alta)**	**3,0 (Média)**

Tabela 6 - Classificação CVSS de vulnerabilidades severidade média

A Figura 4.10 apresenta as ditas pontuações.

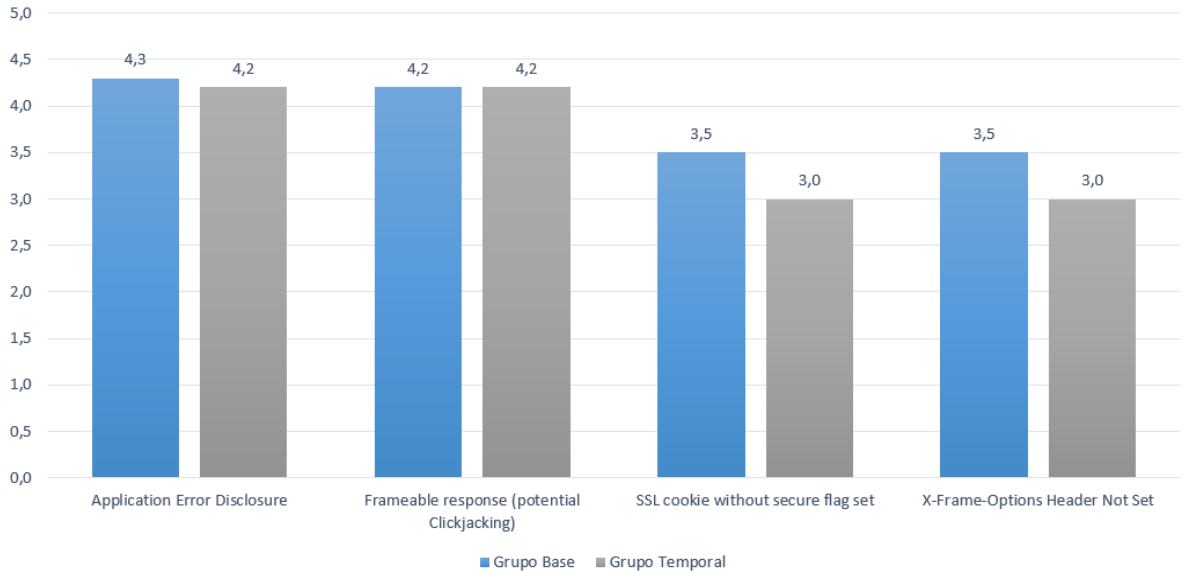


Figura 4.10 - Pontuação CVSS para vulnerabilidades encontradas de severidade média

Uma vulnerabilidade do tipo *application error disclosure* é interessante porque, permite a um atacante focar-se nas tecnologias utilizadas e ser mais objectivo nos testes. Nesta dissertação, dada a natureza do tipo de teste (*grey box)*, os testes foram orientados às tecnologias. À excepção do *application error disclosure* que depende da forma como as aplicações se encontram estruturadas, a resolução das restantes é relativamente simples.

A auditoria será encerrada com a construção do relatório e apresentação do mesmo com as vulnerabilidades encontradas, forma de exploração e mitigações. Resultados gerados através do *dashboard* construído para o efeito.

4.4 Precisão de *web scanners*

Analisando o total vulnerabilidades, ignorando as instâncias consideradas informação, temos a precisão total face aos *web scanners* utilizados na Figura 4.11.

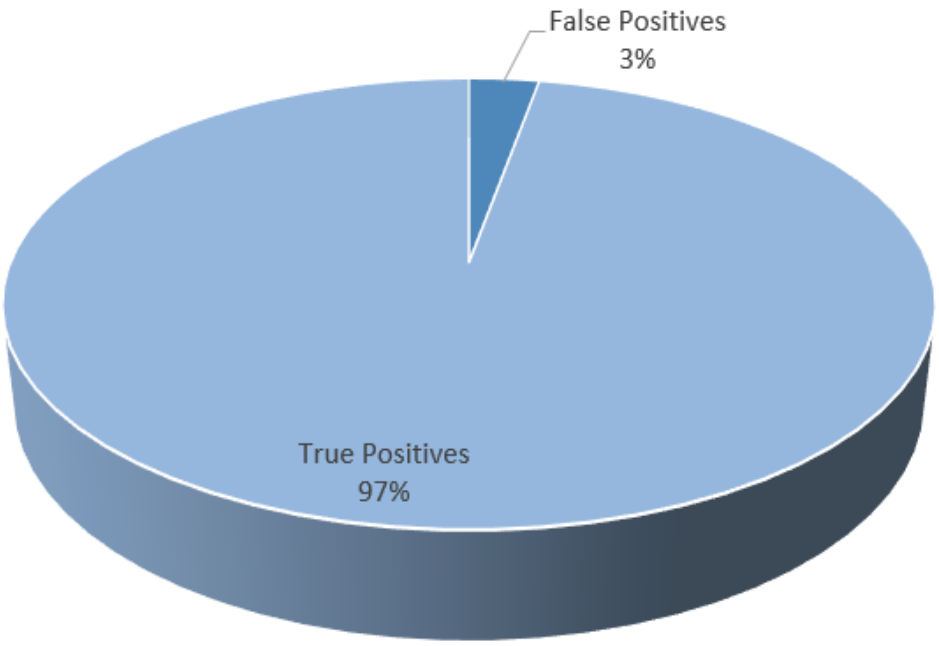


Figura 4.11 - Percentagem falsos positivos nos web scanners utilizados

Apesar destes resultados não significarem que as aplicações podem ser comprometidas em 97% das instâncias encontradas, os resultados dos *web scanners* são positivos, no sentido em que identificam situações de forma correcta. Situações que, individualmente podem comprometer uma aplicação ou, em conjunto com outras vulnerabilidades. Esta conclusão pode ser explicada pelo facto de existir conhecimento prévio sobre a infraestrutura de testes, isto permite configurar o *web scanner* a executar os testes baseado na tecnologia definida e por consequência aumentar a sua precisão.

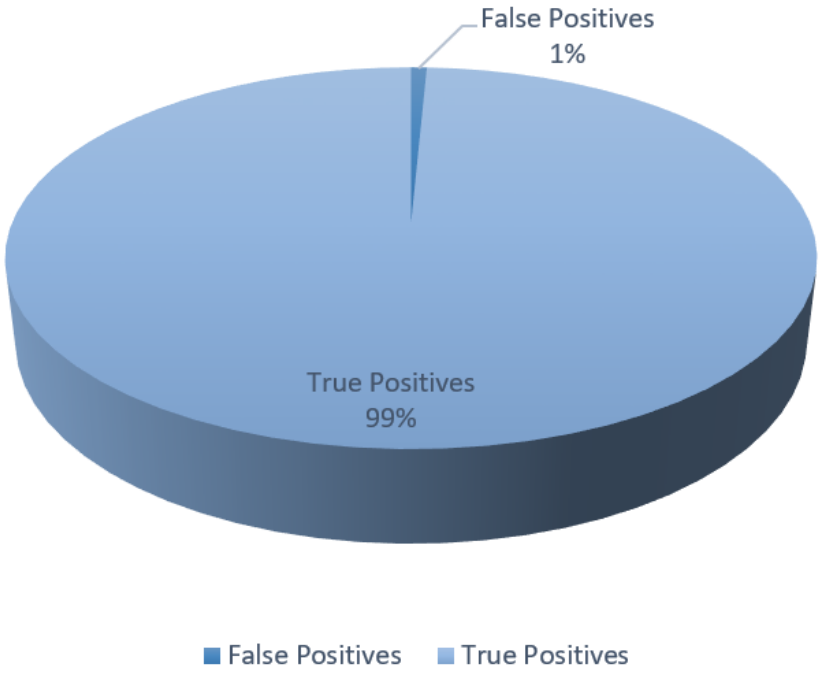


Figura 4.12 - Falsos positivos de web scanner ZAP

A percentagem de falsos positivos para o *web scanner* ZAP apresentada na Figura 4.12 é bastante reduzida. São resultados positivos para uma auditoria de segurança.

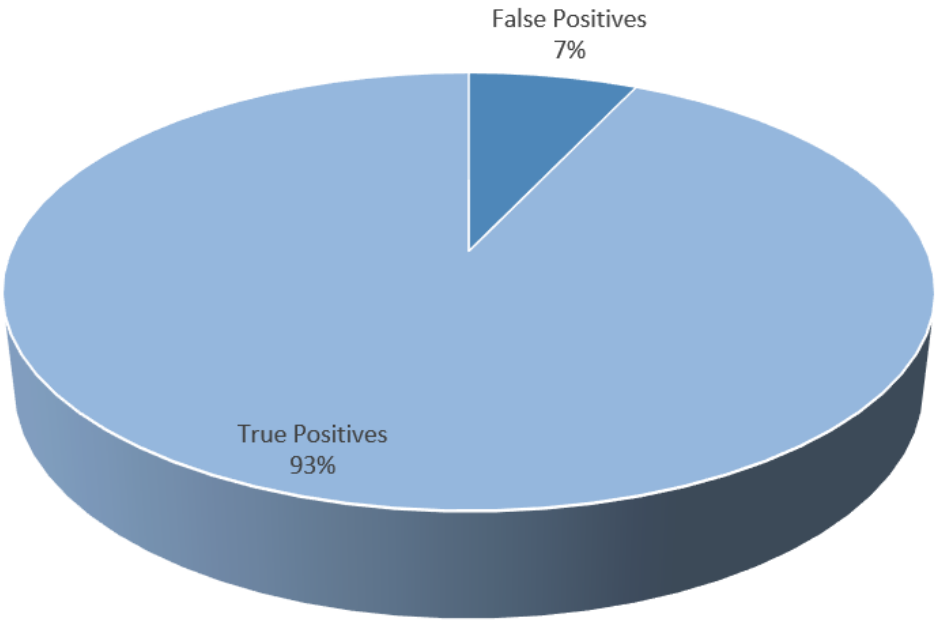


Figura 4.13 - Falsos positivos de web scanner BURP

Na análise de falsos positivos da ferramenta Burp, temos uma percentagem ligeiramente superior face ao ZAP mas, há que considerar que existem diferenças entre as vulnerabilidades identificadas por ambos. A grande maioria dos falsos positivos identificados pelo Burp são de severidade classificada como informação. Além disso, são situações que o ZAP não identificou sendo que, uma das vulnerabilidades tem uma percentagem de 19% das descobertas.

Os resultados obtidos pelas ferramentas utilizadas ZAP e Burp podem ser explicados pelo facto de existir conhecimento prévio sobre a infraestrutura de testes, isto permite configurar o *web scanner* a executar os testes baseado na tecnologia definida e por consequência aumentar a sua precisão.

Como explicado anteriormente, os falsos positivos podem ocupar bastante tempo do auditor na validação e confirmação da vulnerabilidade, mas, as aplicações *web* têm uma facilidade de reutilização de conteúdos programados muito grande. Isto significa que, em algumas situações de identificação de um grande número instâncias de uma vulnerabilidade, tanto a validação como a mitigação podem ser extrapoladas analisando apenas uma das várias instâncias encontradas.

Estes resultados provam que ainda há possibilidade de melhoria destas ferramentas, pois existem vulnerabilidades encontradas por apenas um dos *web scanners* e existem os falsos positivos no entanto, os resultados obtidos podem ser considerados óptimos face à diversidade tecnológica e complexidade de aplicações *web*.

Como nota, os resultados classificados como informação não foram incluídos devido a, por si só representarem um risco residual. Em exploração conjunta com outras vulnerabilidades, é o cenário que representam maior perdido. Não obstante, devem com certeza ser analisadas e corrigidas.

Capítulo 5 Validação do Dashboard de Vulnerabilidades (*VDashboard)*

Como forma de validação do aspecto gráfico e funcionalidades do Dashboard de Vulnerabilidades (VDashboard), a aplicação foi apresentada a especialistas da área da segurança de informação da instituição onde foram realizados os testes de intrusão, os quais se mostraram recetivos à utilização da mesma e contribuíram com sugestões.

A apresentação decorreu nas instalações da instituição, alvo desta auditoria e os envolvidos foram responsáveis do departamento de segurança, elementos de desenvolvimento e gestão de projectos que consistiam um total de 6 pessoas. Por motivos de indisponibilidade, vários dos convidados reunidos não puderam comparecer, portanto foi efectuada uma gravação (slides, aplicação e som) para que pudesse ser partilhada a outros interessados e estes partilharem a sua opinião mais tarde.

Este capítulo apresenta um resumo de todo o *feedback* recolhido face à apresentação da ferramenta *VDashboard* desenvolvida, assim como introduz algumas das sugestões para sua melhoria futura.

5.1 Entrevistas

Aproveitando o facto de ter uma equipa de especialistas ligados à área de segurança e desenvolvimento de software como plateia na apresentação do *dashboard*, foram realizadas entrevistas aos presentes de forma a identificar melhor as espectativas e necessidades dos vários tipos de utilizadores. As perguntas associadas à entrevista consistiram nas seguintes:

1. Já testou alguma aplicação desta natureza?
2. Utiliza aplicações com esta natureza profissionalmente?

3. Os gráficos apresentados no painel de instrumentos inicial são de fácil interpretação?
4. Como caracteriza a informação presente no painel inicial?
5. A informação caracterizadora das vulnerabilidades é suficiente para caracterizar a mesma?
6. Como caracteriza a organização da informação de uma auditoria de segurança?
7. (Se resposta negativa) Que informação considera estar omissa?
8. A informação contida nos relatórios é útil para um técnico?
9. (Se resposta negativa) Que informação considera estar omissa?
10. A informação contida nos relatórios é útil para um gestor?
11. (Se resposta negativa) Que informação considera estar omissa?
12. A informação contida nos relatórios é útil para tomada de decisões em relação à priorização de mitigações?
13. Como classifica o aspecto da aplicação?
14. Como classifica a facilidade de utilização da aplicação?
15. Tem sugestões ou melhorias?

O *feedback* recolhido e as opiniões facultadas resultam na seguinte compilação de resultados:

5.2 Feedback

Nesta secção é apresentado um resumo dos principais aspectos que os especialistas de segurança da informação elencaram aquando do seu contacto e teste com o VDashboard.

Após primeiro contacto com a aplicação, houve aceitação por parte dos membros que elogiaram e subscreviam algumas vantagens inerentes à utilização do *dashboard*:

- Interface muito apelativa;
- Categorização de serviços e módulos compatível com estrutura de projectos da instituição;
- Gráficos e informações associadas à OWASP Top 10 importantes;
- Classificação de vulnerabilidades familiar e *standard* da indústria;
- Notas de mitigação informações de falsos positivos bem aplicadas;
- Possibilidade de escolher vulnerabilidades presentes no relatório pertinente;
- Informação caracterizadora das vulnerabilidades e auditoria pertinente;

Tratando-se de uma primeira versão do *dashboard*, os pontos de melhoria apontados foram os seguintes:

Referências OWASP: foi considerado "quick win" relativamente expressivo incluir nas páginas onde se referencia as categorias OWASP, links diretos para as descrições da OWASP – ajuda tanto a quem categoriza as vulnerabilidades como a quem consulta o registo com o intento de as corrigir.

Relatório: incluir descrição do serviço no relatório das vulnerabilidades (para identificar univocamente onde reside a vulnerabilidade)

Usar escalas lineares em vez de adaptativas nos gráficos: configurar ferramenta de gráficos de forma que as escalas dos eixos sejam uniformes em vez de adaptativas aos resultados.

Tornar instrumentos visuais (gráficos) parametrizáveis: Vulnerabilidades por sistema pesquisável. Sistemas com vulnerabilidades pesquisável. Vulnerabilidades num determinado período pesquisável. Associar também a estas pesquisas OWASP Top 10.

Perfis de utilizador: restringir os conteúdos de acordo com as responsabilidades no *dashboard* através de sistema de autenticação.

Drill down: nos gráficos de sistema/módulo. Apresentar informação mais detalhada ao selecionar um elemento do gráfico.

Link para vulnerabilidades corrigidas: em cada registo de vulnerabilidade, link para mostrar registos de vulnerabilidades registadas já resolvidas dessa categoria.

Integração e conectividade: por último, a integração desta ferramenta com outras ferramentas de trabalho como por exemplo o email para notificações e alertas seria extremamente útil.

Conclusão

Este capítulo resume todo o trabalho efetuado ao longo desta dissertação levada a cabo com o intuito de realização de uma auditoria se segurança numa instituição de um determinado sector em Portugal e conclusão sobre os seus resultados.

Tendo como motivação a aprendizagem na área de segurança *web* e a contribuição para a protecção de activos da instituição, definiram-se os seguintes objectivos:

- Identificar guias de práticas para realização de uma auditoria de segurança;
- Escolher ferramentas automáticas para optimizar o processo anterior;
- Definir a classificação de risco que ajudaria a instituição a compreender as vulnerabilidades;
- Estruturar uma metodologia de testes de intrusão em aplicações *web* flexível e reutilizável;
- Realizar a auditoria de segurança respeitando de forma ética as preocupações da instituição;
- Investigar os ataques resultantes das vulnerabilidades encontradas e mitigações;
- Criar uma aplicação de apoio à decisão:
    - Com um painel de instrumentos resumindo o estado e riscos das aplicações testadas;
    - Capaz de apresentar os resultados formalmente – num relatório;

Esta investigação e trabalho realizados permitiu criar valor acrescentado a nível académico e também para a instituição através de três entregáveis: uma metodologia de testes de intrusão *web*, uma comparação entre *web scanners* e uma aplicação de apresentação de resultados.

Face aos objectivos iniciais propostos e aqui resumidos eis as seguintes conclusões desta auditoria:

Foram identificados através de estudos na área, *web scanners* poderosos com resultados considerados fiáveis e que alavancaram esta auditoria de segurança. Os resultados das ferramentas Burp e ZAP não permitem excluir a utilização de uma ferramenta, de facto, os resultados obtidos entre ambos, completam-se e corroboram as referências bibliográficas quanto à utilização de várias ferramentas. O número reduzido de falsos positivos, a fácil validação dos mesmos através da informação fornecida pelos *web scanners* e a capacidade de

configurar novos testes rapidamente para validação são uma clara vantagem deste tipo de ferramentas.

A existência de falsos positivos pode em alguns casos ser explicada por boas práticas de desenvolvimento, em camadas com vários níveis de segurança. Neste caso, a maioria dos resultados traduziam-se em incorrectas classificações, mas de rápida validação.

A auditoria de segurança seguiu as normas e boas prácticas dos guias em cooperação com as ferramentas automáticas que agilizaram e aceleraram o processo de auditoria. Permitindo assim criar uma metodologia flexível e reutilizável para futuras auditoria a aplicações *web*. A utilização dum guia ou *checklist* permitiu uma abrangência de testes mais confiante num processo moroso e vasto que são os testes de intrusão.

Comprovou-se que o planeamento de ataques com base num guia e com auxilio de ferramentas automáticas obtém melhores resultados em comparação com ataques de força bruta por parte dos *web scanners*. Muito devido à capacidade de o auditor identificar o comportamento da aplicação e da orientação por parte do guia de testes de intrusão. Produzem-se assim resultados mais incisivos.

Em relação às aplicações e vulnerabilidades, foram testadas 4 aplicações com diversos módulos focadas em tecnologias Microsoft, detectaram-se 25 vulnerabilidades distintas com 554 instâncias das mesmas. A distribuição de vulnerabilidades para severidade alta foi de 3%, 36% para as médias e 61% para as baixas. Com severidade alta foram encontradas 14 instâncias, com severidade média, 171 instâncias e com severidade baixa, foram encontradas 294. Efectivamente, as mais críticas são constituintes do OWASP Top 10 e podem ter consequências mais severas na confidencialidade, integridade ou disponibilidade. As aplicações, individualmente apresentam poucas vulnerabilidades críticas, no entanto, para comprometer uma aplicação ou sistema apenas é necessária uma instância.

Outro contributo deste estudo é uma compreensão do nível de maturidade em termos de segurança para as aplicações *web* desenvolvidas sobre as tecnologias referidas e a realização de uma extrapolação para o nível de maturidade em entidades reguladoras e as suas consequências. Apesar do número reduzido de vulnerabilidades, foram identificadas vulnerabilidades com potenciais consequências severas, a sua mitigação é imprescindível. Podemos também retirar deste estudo que existe maturidade no desenvolvimento de aplicações *web* e preocupações de segurança mas, o erro humano existe e devem ser aplicadas garantias de segurança como testes de intrusão mas, não só.

Algo que este estudo não permite classificar, são os danos na reputação da instituição. No entanto, através da severidade das vulnerabilidades encontradas, podemos concluir que podem existir consequências graves. Além do mais, agindo como regulador, existe comunicação com outras entidades. Assim sendo, a violação de confidencialidade compromete fortemente não só a segurança da informação, mas permite que seja usada a favor ou contra outras instituições causando danos nos lucros e reputação.

O último contributo deste estudo é uma aplicação *web* capaz de registar e apresentar informação sobre vulnerabilidades no âmbito *web* com um interface intuitivo e fácil de utilizar. Trata-se de uma ferramenta suficientemente avançada e bem-recebida pela instituição para suporte à decisão num departamento de segurança. As funcionalidades presentes na aplicação desenvolvida permitem um acompanhamento de vulnerabilidades, criação de relatórios, priorização de mitigações e constroem assim, uma base de dados de histórico de vulnerabilidades que reúnem boas práticas e controlos para desenvolvimento em aplicações futuras. É sem dúvida uma mais valia num ambiente empresarial onde o tempo de reacção é importante e cujos custos inerentes à resolução e à identificação de vulnerabilidades são muito elevados.

Finalmente, no âmbito deste estudo, foi submetido e aceite na conferência *International Conference for Security Technology and Secured Transactions (ICITST) a realizar de 5 a 7 de Dezembro de 2016,* o caso de estudo levado a cabo nesta auditoria de segurança e os resultados obtidos: *Web security in the finance sector – case study* (Vieira & Serrão, 2016).

Trabalho Futuro

A dissertação realizada teve como âmbito uma auditoria de segurança a uma instituição reguladora constituindo um nicho muito específico no país. Estando condicionada a disponibilidade das aplicações alvo da auditoria pelos calendários de desenvolvimento e testes, este trabalho abrangeu uma amostra reduzida. Tendo sido avaliada apenas uma instituição, apenas é possível extrapolar resultados face à segurança de aplicações *web* com tecnologias deste estudo.

É possível avaliar que os objectivos propostos foram alcançados com limitações que possibilitam este trabalho ser continuado.

Em primeira instância e pelo facto de ter sido realizado numa instituição, apenas pode ser extrapolado o nível de maturidade de segurança no sector. Para melhores conclusões, a abrangência neste sector poderá ser aumentada a fim de incluir na auditoria de segurança, outras entidades reguladoras de forma a verificar efetivamente o grau de maturidade de segurança no sector, o que implicaria um âmbito internacional.

Adicionalmente, a realização de uma nova auditoria às mesmas aplicações, poderia ser uma análise interessante de forma a determinar, se as vulnerabilidades identificadas foram de alguma forma mitigadas, quer se os resultados obtidos com novas versões dos *web scanners* seriam mais eficazes, isto é, na identificação de vulnerabilidades e redução de falsos positivos e analisar a evolução de maturidade deste tipo de processo na instituição

Em relação ao *dashboard*, bastantes desenvolvimentos poderão ser adicionados para complementar a ferramenta. A ferramenta pode evoluir em dois sentidos, orientada às necessidades de uma instituição em específico ou num sentido mais genérico e abrangente. Por esses motivos, poderiam ser feitos estudos comparativos com outras ferramentas a nível de funcionalidade, usabilidade e medir o impacto na gestão de vulnerabilidades de equipas de desenvolvimento e auditores de segurança.

De forma a comprovar as funcionalidades e usabilidade do *dashboard*, proceder à realização de inquéritos através de questionários a um número maior de utilizadores e profissionais da área para aperfeiçoamento da ferramenta.

Finalmente, realizar auditorias de segurança em outras áreas ou aplicações com outras tecnologias, por exemplo, Java. Que pudessem aferir a eficácia da metodologia em outros âmbitos.

Bibliografia

Acunetix. (2011, Janeiro). *How to choose a web vulnerability scanner*. Retrieved from Acutinex: https://www.acunetix.com/blog/articles/how-to-choose-web-vulnerability-scanner/

Alrodham, W., & Mitchell, C. (2011). Enhancing User Authentication in Claim-Based Identity Management.

Austin, A., & Williams, L. (2011). *One Technique is not Enough: A Comparison of Vulnerability Discovery Techniques.*

Bozic, j., & Wotawa, F. (2015). PURITY: A Planning-based secURITY Testing Tool.

Buchler, M., Oudinet, J., & Pretschner, A. (2012). Semi-automatic security testing of web applications from a secure model.

CAPEC View: Domain Attacks. (2015). Retrieved from CAPEC Common Attack Pattern Enumeration and Classification: https://capec.mitre.org/data/definitions/117.html

Center for Internet Security. (2015). Retrieved from CISSecurity.org: https://www.cisecurity.org/about/

CIS Critical Security Controls. (2015). Retrieved from SANS.org: https://www.sans.org/critical-security-controls

Client Server Architecture. (2016). Retrieved from CCM: http://ccm.net/contents/151-networking-3-tier-client-server-architecture

COBIT. (2014). Retrieved from ISACA: https://cobitonline.isaca.org/

Curphey, M., & Arawo, R. (2006). Web Security - Web Application Security Assessment Tools.

CVSS. (2015, Junho). Retrieved from FIRST: https://www.first.org/cvss

Diniz, B., & Serrão, C. (2014). Using PTES and open-source tools as a way to conduct external footprinting security assessments for intelligence gathering.

Doupé, A., Cova, M., & Vigna, G. (2010). Why Johnny Can't Pentest: An Analysis of Black-box Web Vulnerability Scanners.

Dukes, L., Yuan, X., & Akowuah, F. (2013). A Case Study on Web Application Security Testing with Tools and Manual Testing.

EC Council. (2015). Retrieved from EC Council: http://www.eccouncil.org/

Exploit Database. (2015). Retrieved from Exploit Database: https://www.exploit-db.com/

Faraday. (2016). *FaradaySec*. Retrieved from Faraday: https://www.faradaysec.com/

Fong, E., Gaucher, R., Okun, V., & Black, P. E. (2008). Building a Test Suite for Web Application Scanners.

Guide for Applying the Risk Management Framework to Federal Information Systems. (2010). Retrieved from SP 800-37: http://csrc.nist.gov/publications/nistpubs/800-37-rev1/sp800-37-rev1-final.pdf

Hasan, W., & Sajib, C. (2015). Testing A Comparative Overview on Penetration Testing.

Hoff, J. (2014). OWASP WebGoat.NET. Retrieved 2016, from https://github.com/jerryhoff/WebGoat.NET

ISO 27000. (2013). Retrieved from ISO 27000: http://www.27000.org/

ISO 27002. (2013). Retrieved from ISO: http://www.27000.org/iso-27002.htm

Kali Linux. (2015). Retrieved from Kali Linux: https://www.kali.org/

KPMG Risk Management. (2015). Retrieved from KPMG: https://home.kpmg.com/xx/en/home/services/advisory/risk-consulting.html

Lyon, G. (2015). *Sectools: Top 125 Network Security Tools*. Retrieved from Sectools: http://sectools.org/

Makino, Y., & Klyuev, V. (2015). Evaluation of Web Vulnerability Scanners. *The 8th IEEE International Conference on Ittenligent Data acquisition and Advanced Computing Systems: Technology and Applications*.

Malhotra, Y. (2014). Global Banking & Finance Networks VoIP Protocols Yogesh Malhotra , PhD Griffiss Cyberspace , Global Risk Management Network , LLC According to computer

scientists at Columbia University , " A vulnerability inside all current Cisco IP phones allows hackers.

McGraw, G., Migues, S., & West, J. (2015). *BSIMM6*. Retrieved from BSIMM: https://www.bsimm.com/wp-content/uploads/2015/10/BSIMM6.pdf

Metasploit. (2015). Retrieved from Metasploit: http://www.metasploit.com/

MITRE. (2015). Retrieved from MITRE: http://www.mitre.org/

Mullins, M. (2005). *Choose the best penetration testing method for your company*. Retrieved from Tech Republic: http://www.techrepublic.com/article/choose-the-best-penetration-testing-method-for-your-company/5755555/

Nessus Vulnerability Scanner. (2015). Retrieved from Nessus: http://www.tenable.com/products/nessus-vulnerability-scanner

NIST 800-30 Risk Management Guide for Information Technology Systems. (2002, Julho). Retrieved from NIST: http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf

NIST SP 800-115 Technical Guide to Information Security Testing And Assessment. (2008). Retrieved from NIST SP 800-115: http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf

NIST.org. (2015). Retrieved from NIST.

Offensive Security Training, Certifications and Services. (2015). Retrieved from Offensive Security: http://www.offensive-security.com/

OSSTMM. (2015). Retrieved from ISECOM.org: http://www.isecom.org/research/osstmm.html

OSSTMM v3.0 - The Open Source Security Testing Methodology Manual. (2010). Retrieved from ISECOM: http://www.isecom.org/mirror/OSSTMM.3.pdf

OWASP Application Security Verification Standard Project. (2015, Outubro). Retrieved from OWASP.org.

OWASP Code Review V2 Project. (2008). Retrieved from OWASP: https://www.owasp.org/images/2/2e/OWASP_Code_Review_Guide-V1_1.pdf

OWASP Developers Guide. (2014). Retrieved from OWASP Developers Guide: https://www.owasp.org/index.php/OWASP_Guide_Project

OWASP Foundation. (2015). Retrieved from OWASP: https://www.owasp.org/index.php/About_OWASP#The_OWASP_Foundation

OWASP Mutillidae. (2016). Retrieved 2015, from https://sourceforge.net/projects/mutillidae/

OWASP Risk Rating Methodology. (2015). Retrieved from OWASP: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

OWASP Testing Guide V3. (2008). Retrieved from OWASP: https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf

OWASP Testing Guide v4. (2014). Retrieved from OWASP Testing Guide v4: https://www.owasp.org/images/5/52/OWASP_Testing_Guide_v4.pdf

OWASP Top 10. (2013). Retrieved from OWASP: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

OWASP Vulnerable Web Applications Directory Project. (2015). Retrieved from https://www.owasp.org/index.php/OWASP_Vulnerable_Web_Applications_Directory_Project#tab=Main

OWASP ZAP. (2015). Retrieved from OWASP: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

PCI Security Standards Council. (2015). Retrieved from PCI Security Standards Council: https://www.pcisecuritystandards.org/index.php

Prandini, M., & Ramillo, M. (2010). Towards a practical and effective security testing methodology.

PTES. (2014). Retrieved from Pentest Standard.org: http://www.pentest-standard.org/index.php/Main_Page

PTES Technical Guidelines. (2012). Retrieved from Pentest Standard.org: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines

Razzaq, A., Hur, A., Haider, N., & Ahmad, F. (2009). Multi-Layered Defense against Web Application Attacks.

SAMM. (2009). Retrieved from OWASP: https://www.owasp.org/index.php/Software_Assurance_Maturity_Model_(SAMM)

SANS. (2015). Retrieved from SANS: https://www.sans.org/

SANS Network Penetration Testing Ethical Hacking. (2015). Retrieved from SANS.org: https://www.sans.org/course/network-penetration-testing-ethical-hacking

SANS Web App Penetration Testing Ethical Hacking. (2015). Retrieved from SANS: https://www.sans.org/course/web-app-penetration-testing-ethical-hacking

Scarfone, k., & Mell, P. (2009). An analysis of CVSS version 2 vulnerability scoring.

Schneier, B. (2000). *Computer Security: Will We Ever Learn?* Retrieved from Schneier on Security: https://www.schneier.com/crypto-gram/archives/2000/0515.html

Serpico. (2016). Serpico Project.

Skipfish. (n.d.). Retrieved from Google Code: https://code.google.com/archive/p/skipfish/

Stuttard, D., & Pinto, M. (2011). *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws.* Wiley.

Tenable. (2016). *Tenable*. Retrieved from Security Center Dashboards: https://www.tenable.com/sc-dashboards

Teodoro, N., & Serrão, C. (2011). Web Application Security.

The CIS Critical Security Controlls for Effective Cyber Defense version 6.0. (2015, Outubro).

ThreadFix. (2016). *ThreadFix*. Retrieved from https://www.threadfix.it/

Tung, Y.-H., Tseng, S.-S., Shih, J.-F., & Shan, H.-L. (2014). W-VST: A Testbed for Evaluating Web Vulnerability Scanner.

Vieira, T., & Serrão, C. (2016). Web security in the finance sector - Case study.

Vulnerability Scanning Tools. (2015). Retrieved from OWASP: https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools

Walker, M. (2012). *CEH Certified Ethical Hacking All-in-one Exam Guide* (Segunda ed.). The McGraw Hill. Retrieved 2016

WASC Threat Classification. (2010). Retrieved from Web Application Security Consortium : http://projects.webappsec.org/f/WASC-TC-v2_0.pdf

Web Application Security Consortium. (2014). Retrieved from http://www.webappsec.org/

Web Application Security Scanner Evaluation Criteria. (2009).

Wheeler, E. (2011). *Security Risk Management Building an Information Security Risk Management Program from the Ground Up*. ELSEVIER.

Younis, A., & Malaiya, Y. (2015). Comparing and Evaluating CVSS Base Metrics and Microsoft Rating System.

Zitta, S., Marik, O., & Neradova, S. (2014). Conducting effective penetration testing.

Anexos

A. Terminologia BSIMM6

Activity: Actions carried out or facilitated by the SSG as part of a practice. Activities are divided into three levels in the BSIMM.

Domain: The domains are: governance, intelligence, secure software development lifecycle (SSDL) touchpoints, and deployment. See the SSF section on page 10.

Practice: One of the 12 categories of BSIMM activities. Each domain in the Software Security Framework has three practices. Activities in each practice are divided into three levels. See the SSF section on page 10.

Satellite: A group of interested and engaged developers, architects, software managers, testers, and similar roles who have a natural affinity for software security and are organized and leveraged by a Software Security Initiative.

Secure Software Development Lifecycle (SSDL): Any SDLC with integrated software security checkpoints and activities.

Security Development Lifecycle (SDL): A term used by Microsoft to describe their Secure Software Development Lifecycle.

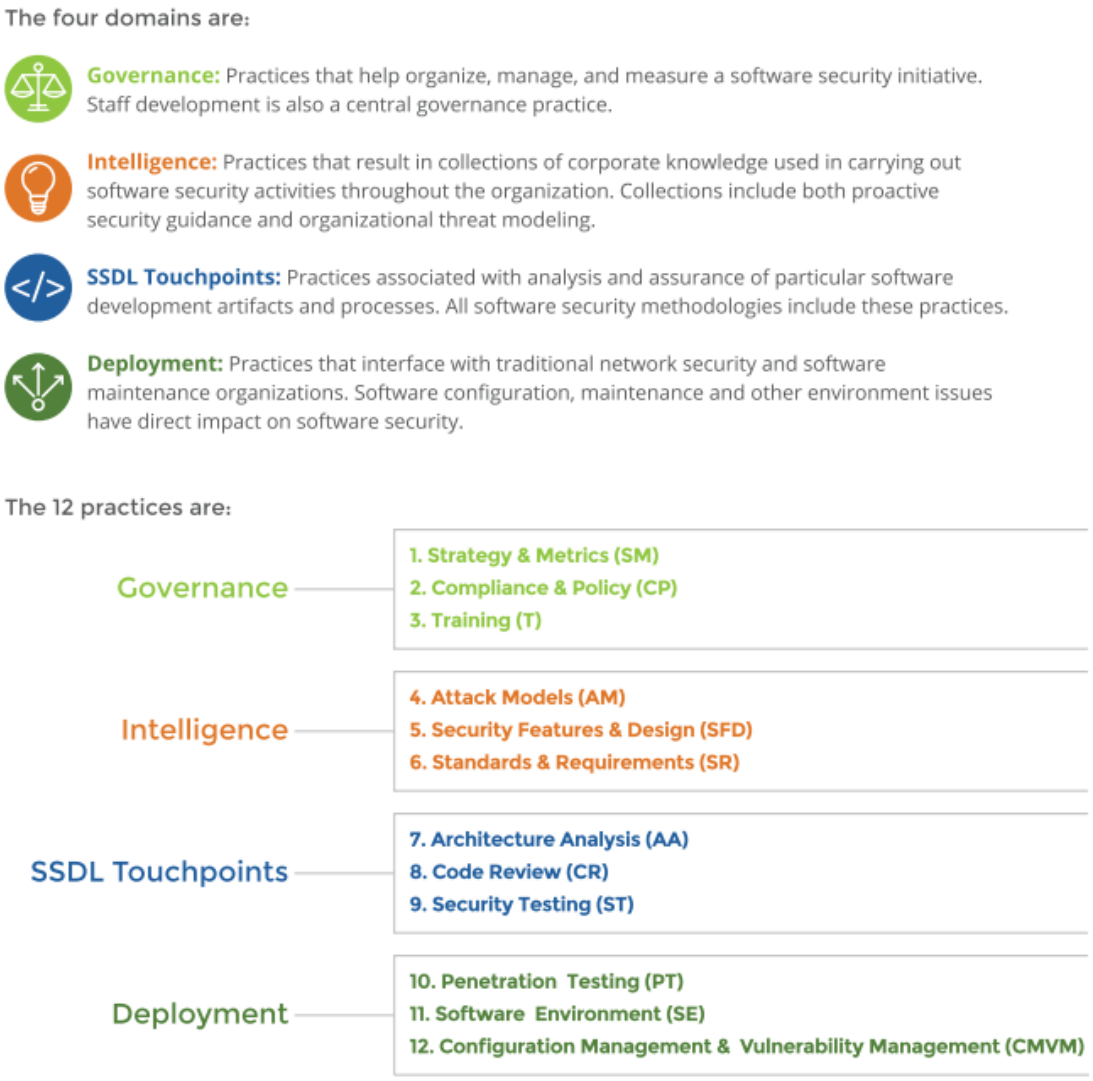
Software Security Framework (SSF): The basic structure underlying the BSIMM, comprising 12 practices divided into four domains. See the SSF section on page 10.

Software Security Group (SSG): The internal group charged with carrying out and facilitating software security. According to our observations, the first step of a Software Security Initiative is forming an SSG.

Software Security Initiative: An organization-wide program to instill, measure, manage, and evolve software security activities in a coordinated fashion. Also known in the literature as an Enterprise Software Security Program (see chapter 10 of *Software Security: Building Security In*).

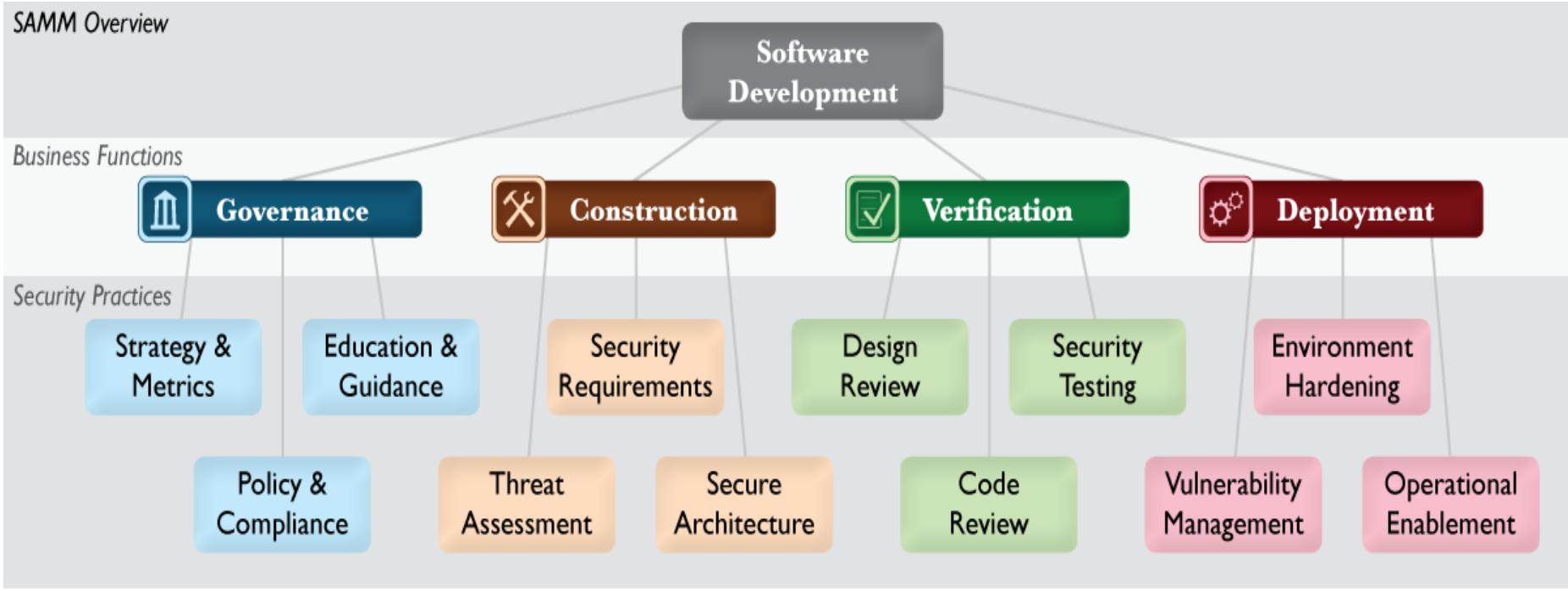
Anexo 1 - Conceitos BSIMM6 (McGraw, Migues, & West, 2015)

B. Estrutura da Framework BSIMM6



Anexo 2 - Domínios e práticas BSIMM6 (McGraw, Migues, & West, 2015)

C. Domínios SAMM



Anexo 3 - Domínios e Práticas SAMM (SAMM, 2009)

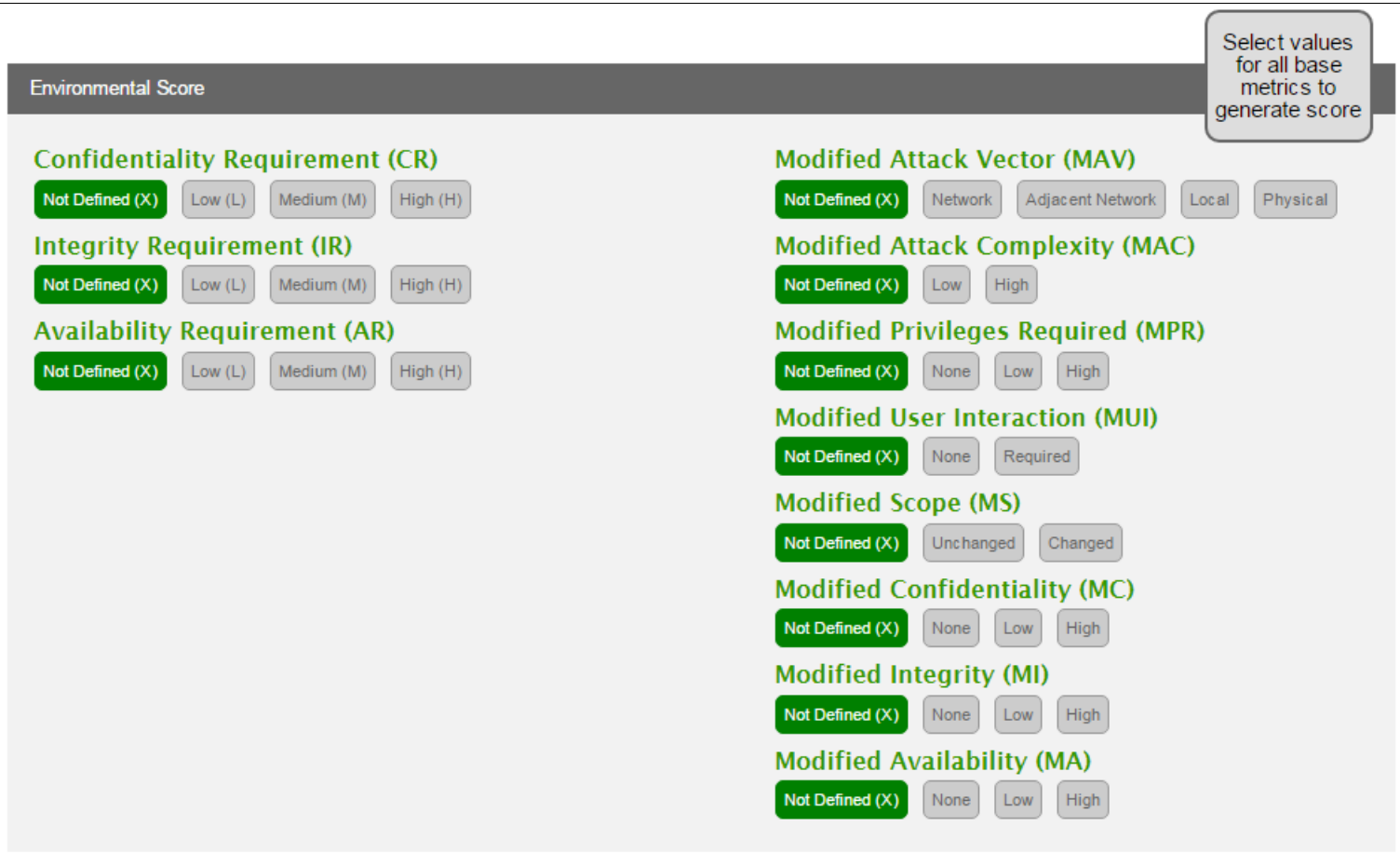
D. CVSS Grupo Base

Anexo 4 - Métricas Base CVSS (CVSS, 2015)

E. CVSS Grupo Temporal

Anexo 5 - Métricas Temporais CVSS (CVSS, 2015)

F. CVSS Grupo Ambiental



Anexo 6- Métricas Ambientais CVS (CVSS, 2015)

1. Declaração de confidencialidade

Declaração de Confidencialidade para Colaboradores Do/a [NOME DO PRESTADOR DE SERVIÇOS], ao Serviço de Clientes

Pela presente, o(a) Sr(a) [NOME DO COLABORADOR], portador do Bilhete de Identidade/Cartão de Cidadão _____, de __/__/____, emitido por _____, de agora em diante designado abreviadamente por COLABORADOR, obriga-se a garantir o rigoroso sigilo e confidencialidade, não divulgando, nem permitindo o seu conhecimento, as informações relativas à [INSTITUIÇÃO], a que tenha acesso no âmbito da execução Contrato de Prestação de Serviços celebrado entre o/a [NOME DO PRESTADOR DE SERVIÇOS] e a [INSTITUIÇÃO], adiante designado como CONTRATO,

Para efeitos da presente Declaração, considera-se informação confidencial toda e qualquer informação obtida ou entregue, documentos ou conteúdo total ou parcial dos mesmos, dados ou factos transmitidos entre o/a [NOME DO PRESTADOR DE SERVIÇOS] e o [INSTITUIÇÃO], entre o COLABORADOR e a [INSTITUIÇÃO], ou a que qualquer das partes tenha acesso no âmbito da execução do CONTRATO, por escrito, oralmente ou por qualquer outra forma de comunicação, podendo incluir, planos de negócio, abordagens metódicas e de projetos, invenções, descobertas, processos, protótipos, informações sobre clientes, marcas e qualquer outro tipo de informação comercial, financeira, técnica ou estratégica.

O COLABORADOR obriga-se, em particular, a não revelar, qualquer informação oral, escrita ou digital, confidencial ou não, relativa a dados, recursos, projetos, técnicas, métodos e procedimentos utilizados na [INSTITUIÇÃO] de que venha a ter conhecimento durante a execução do CONTRATO, incluindo-se nesta obrigação os meios e recursos postos à sua disposição, nomeadamente números de telefone, endereços de rede, programas informáticos e demais dispositivos de segurança relativos à [INSTITUIÇÃO].

O COLABORADOR obriga-se a que a utilização das informações acima referidas seja limitada à execução dos serviços objeto do CONTRATO e a não fazer nada, de forma ativa ou passiva, que possa afetar a confidencialidade ou integridade das referidas informações.

O COLABORADOR obriga-se, ainda, a não comunicar, completa ou parcialmente, qualquer informação obtida no âmbito da execução do contrato entre o/a [NOME DO PRESTADOR DE SERVIÇOS] e a [INSTITUIÇÃO], a quem quer que seja, exceto com o acordo escrito prévio do/da [NOME DO

PRESTADOR DE SERVIÇOS] e/ou da [INSTITUIÇÃO], devendo comunicar quer à/ao [NOME DO PRESTADOR DE SERVIÇOS], quer à [INSTITUIÇÃO] toda a violação de confidencialidade ou segurança de que tem conhecimento.

Quando terminar a sua prestação ou em caso de suspensão desta, o COLABORADOR obriga-se a devolver a totalidade dos elementos recebidos ou adquiridos, em particular, os documentos fornecidos para a execução da sua prestação.

Uma vez terminada a prestação de serviços, o COLABORADOR obriga-se a continuar a tratar como confidenciais, bem como a não usar, as informações a que tenha tido acesso para e durante a sua prestação.

O COLABORADOR reconhece que o presente compromisso de confidencialidade não lhe confere nenhum direito de propriedade, em particular, industrial ou intelectual, sobre as informações fornecidas ou a que teve acesso, os quais continuam propriedade da [INSTITUIÇÃO].

O COLABORADOR reconhece que lhe é aplicável, atendendo à natureza da [Instituição], o regime legal do dever de segredo das autoridades de supervisão previsto no artigo 80.º do Regime Geral das Instituições de Crédito e Sociedades Financeiras, pelo que está sujeito, nos termos da lei, ao dever de segredo sobre os factos cujo conhecimento lhe advenha exclusivamente da execução dos serviços objeto do CONTRATO, designadamente aqueles

relativos à vida das instituições sujeitas à supervisão da [INSTITUIÇÃO] ou à relação destas com os seus clientes.

O COLABORADOR obriga-se a ressarcir o/a [NOME DO PRESTADOR DE SERVIÇOS] e/ou a [INSTITUIÇÃO] por todos os danos diretos que possam resultar da violação da obrigação de confidencialidade contida no presente documento, sem prejuízo de eventual responsabilidade disciplinar e/ou criminal que daí possa resultar.

À presente Declaração aplica-se o direito Português.

Lisboa, __ de _____ 20__

Nome por extenso:

Assinatura:

Anexar fotocópia do Bilhete de Identidade/Cartão do Cidadão.