INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Estudo de vulnerabilidades em aplicações web e o seu reflexo em domínios Portugueses

NUNO MIGUEL DA SILVA MONTEIRO
Novembro de 2015

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Estudo de vulnerabilidades em aplicações web e o seu reflexo em domínios Portugueses

Nuno Miguel da Silva Monteiro

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Arquiteturas, Sistemas e Redes

Orientador: Nuno Pereira

Júri:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, Outubro 2015

Aos meus Pais.

Resumo

Muito se tem falado sobre revolução tecnológica e do aparecimento constante de novas aplicações Web, com novas funcionalidades que visam facilitar o trabalho dos utilizadores. Mas será que estas aplicações garantem que os dados transmitidos são tratados e enviados por canais seguros (protocolos)? Que garantias é que o utilizador tem que mesmo que a aplicação utilize um canal, que prevê a privacidade e integridade de dados, esta não apresente alguma vulnerabilidade pondo em causa a informação sensível do utilizador? Software que não foi devidamente testado, aliado à falta de sensibilização por parte dos responsáveis pelo desenvolvimento de software para questões de segurança, levam ao aumento de vulnerabilidades e assim exponenciam o número de potenciais vítimas. Isto aliado ao efeito de desinibição que o sentimento de invisibilidade pode provocar, conduz ao facilitismo e consequentemente ao aumento do número de vítimas alvos de ataques informáticos. O utilizador, por vezes, não sabe muito bem do que se deve proteger, pois a confiança que depõem no software não pressupõem que os seus dados estejam em risco. Neste contexto foram recolhidos dados históricos relativos a vulnerabilidades nos protocolos SSL/TLS, para perceber o impacto que as mesmas apresentam e avaliar o grau de risco. Para além disso, foram avaliados um número significativo de domínios portugueses para perceber se os mesmos têm uma vulnerabilidade específica do protocolo SSL/TLS.

Palavras-chave: Ataques informáticos, Vulnerabilidades, Ameaças, Segurança informática, Aplicações Web.

Abstract

Much has been said about the technological revolution and the constant appearance of new web applications, with new features that are designed to facilitate the work of users. But do these applications ensure that the transmitted data are processed and sent via secure channels (protocols)? What guarantees does the user have to even if the application use a channel, which provides privacy and data integrity, and does not present any vulnerability endangering sensitive information? Software that has not been properly tested, combined with a lack of awareness on the part of developers to security issues, lead to an increase of vulnerabilities and with this the number of potential victims. This coupled with the disinhibition effect that the feeling of invisibility may bring, leads to certain lightness and consequently to an increase the in number of victims. The user sometimes does not know very well what it should protect, because he trusts in the software and does not assume that the data are at risk. In this context, historical data about vulnerabilities in the SSL/TLS protocols was collected, to realize the impact they have and assess the degree of risk. In addition, a significant number of Portuguese domain´s were assessed to verify if they have a specific vulnerability of the SSL/TLS protocol.

Keywords: Cyber attacks, Vulnerabilities, Threats, Computer Security, Web Applications.

Agradecimentos

Agradeço especialmente aos meus pais, por todos os sacrifícios que fizeram e que sempre me apoiaram e motivaram para a conclusão de mais esta etapa.

Agradeço ao meu orientador pela colaboração no desenvolvimento deste trabalho.

Agradeço ainda a todos que, dentro e fora do ambiente académico, me incentivaram para a conclusão do Mestrado.

A todos, muito abrigado.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

API	Application Programming Interface
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scorring System
CWE	Common Weakness Enumeration
DNS	Domain Name System
DOM	Document Object Model
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IETF	Internet Engineering Task Force
IP	Internet Protocol
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MAN	Metropolitan Area Network
MIME	Multipurpose Internet Mail Extensions
NVD	National Vulnerability Database
OSI	Open Systems Interconnection
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URI	Uniform Resource Identifiers
URL	Uniform Resource Locator
WAN	Wide Area Network

WHATWG	Web Hypertext Application Technology Working Group
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XSS	Cross Site Scripting

1 Introdução

A internet é hoje utilizada para a realização das mais diversas atividades contendo um universo de conteúdos que está sujeito a uma serie de ameaças físicas e/ou virtuais, colocando em risco a privacidade e segurança dos utilizadores. Apesar da existência de mecanismos de que visam a sua proteção o mesmo nem sempre se verifica. A web proporciona um acesso unificado fácil, de conteúdo dinâmico apenas através de um simples navegador. A mesma página pode incorporar várias tecnologias que combinadas fornecem um conjunto de serviços que há alguns anos atrás seria impensável. Contudo com o aparecimento de novas tecnologias aparecem também novas ameaças. Estas ameaças degeneram em vulnerabilidades que podem ser exploradas. Marc Wiser previu há alguns anos atras que os computadores estariam nos mais vulgares objetos (Weiser, 1991). A verdade é que nos dias de hoje consultamos email, fazemos compras, navegamos na internet, etc. apenas com recurso a um telemóvel. No entanto esta banalização das tecnologias de informação trás igualmente outros constrangimentos ao nível da segurança, o aparecimento de vulnerabilidades. Este capítulo descreve os principais objetivos deste trabalho, os fatores motivacionais e como esta tese está estruturada.

1.1 Enquadramento

Este documento tem a finalidade de servir como relatório final da disciplina de Tese de Mestrado ministrada no segundo ano do curso de Mestrado em Engenharia Informática no Instituto Superior de Engenharia do Porto para obtenção do Grau de Mestre na área de área de especialização em Arquiteturas, Sistemas e Redes.

Esta disciplina vem proporcionar ao aluno a possibilidade de aplicar ou adquirir novos conhecimentos sobre um determinado tema. Assim, no âmbito desta disciplina, foi elaborado um estudo sobre vulnerabilidades e relacionadas com os protocolos SSL/TLS.

1.2 Objectivos do Projecto

Os objetivos deste trabalho situam-se em dois níveis: Geral e Especifico.

1.2.1 Objetivos Gerais

Descrever alguns aspetos em envolvem as aplicações web de hoje em dia, alguns dos seus principais atores, passando por modelos de representação de conhecimento como é caso das taxonomias e que já foi feito nesta área relacionada com segurança. Pretende-se igualmente dar o foco com conceito de vulnerabilidade, os seus impactos e apresentar algumas vulnerabilidades mais comuns nas aplicações Web.

1.2.2 Objetivos Específicos

O primeiro objetivo específico desta Tese é a utilização de dados históricos como base para analisar as vulnerabilidades existentes nos protocolos SSL/TLS desde 2010 até 2015. Para isso foram recolhidos dados de uma fonte pública (National Vulnerability Database) e armazenados numa base de dados criada para o efeito.

O segundo objetivo específico desta Tese é fazer um teste à segurança de alguns domínios portugueses a uma vulnerabilidade específica do protocolo SSL/TLS.

1.3 Motivação para a Realização do Projeto

Num mundo em crescendo estamos cada dia que passa mais dependentes de sistemas de computacionais, com isto, mas num ritmo mais lento, vamos tomando consciência para questões de segurança e na forma como nos expomos na rede. Software evoluiu ao longo dos anos as aplicações sofreram grandes melhorias em termos de funcionalidades. Esta evolução, alcançada como o aparecimento de projetos maiores, inovadores, que facilitam a vida das pessoas, mas que não podem ser resolvidos por uma única pessoa. A complexidade aumentou

assim como a necessidade de cooperar e coordenar esforços por parte das equipas de desenvolvimento. Em alguns casos podemos estar a desenvolver algo com base em software de terceiros, recorrendo a componentes que podem conter vulnerabilidades de segurança. Em complemento as empresas sofrem constantemente a pressão da competitividade e são levadas a desenvolver aplicações num espaço temporal que não lhes permite atestar eficientemente a qualidade do software desenvolvido. Assim as aplicações são lançadas e os erros corrigidos posteriormente com o lançamento de novas versões. Estes erros acompanharam a evolução das aplicações e são naturalmente mais complexos de resolver levando a o que traduz numa maior exposição ao risco de exploração de uma vulnerabilidade.

Os utilizadores colocam no software uma confiança que ficam expostos às vulnerabilidades que estas apresentam. Áreas sensíveis como serviços bancários, serviços de saúde, governo, assumem uma importância na confiabilidade dos seus sistemas que podem por em causa a continuidade do negócio se estes apresentarem algum desvio no seu comportamento. As aplicações Web permitem fazer grande parte dos negócios de hoje, tornaram-se omnipresentes, os utilizadores só precisam de um navegador e de acesso à internet para receber uma interface independentemente do sistema que usam e estarem habilitados a enviar e receber dados e com isto a ficarem expostos. Para além dos erros de programação existem outras fontes de vulnerabilidades, tais como erros de instalação ou configuração insegura que colocam os utilizadores vulneráveis a possíveis ataques.

O foco nas vulnerabilidades é um ponto fundamental nesta tese e em particular nas vulnerabilidades relacionadas com os dois protocolos SSL/TLS. Para dar resposta a este tema, e numa primeira parte da tese, foram recolhidas dados históricos de uma fonte pública National Vulnerability Database (NVD) no período de 2010 a 2015. Cada entrada na base de dados da NVD contém, entre outras coisas, informação sobre o software afetado, versões afetadas e grau de gravidade da vulnerabilidade. Na segunda parte desta tese é testada a exposição de um número significativo de domínios português a uma vulnerabilidade em particular.

1.4 Tecnologias Utilizadas

Durante o projeto foram utilizadas diversas tecnologias para o desenvolvimento do mesmo das quais se destacam as seguintes:

- SQL Server 2014 Manegement Studio[1]
- Nmap 6.49BETA5[2]

1.5 Contributos deste Trabalho

A análise e identificação de vulnerabilidades já revelou ser importante na segurança e desenvolvimento de novas aplicações, neste ponto de vista este trabalho permitiu:

- Reunir e estruturar informação sobre vulnerabilidades Web mais comuns.
- Realização de um estudo das vulnerabilidades em aplicações Web mais comuns.
- Permitir avaliar o risco associado a determinada vulnerabilidade.
- Quantificar o número de software afetado por determinada vulnerabilidade e respetivas versões.
- Perceber o impacto que uma vulnerabilidade tem a nível de software.
- Estabelecer critérios de comparação entre diferentes versões de software.

Com os estudos de vulnerabilidades aqui apresentados, espera-se aumentar a consciência para a importância da segurança no desenvolvimento de software seguro e a relevância da correta aplicação de regras que visam mitigar algumas delas. A comparação de diferentes vulnerabilidades assim como os impactos que as mesmas provocam deverá ser útil como referência futura. O uso eficaz da informação assim como a facilidade de acesso a essa mesma informação deverá cimentar a importância e a viabilidade de técnicas de desenvolvimento de software seguro.

1.6 Estrutura do Relatório

De forma a perceber melhor a forma como o presente documento foi elaborado segue uma descrição sumária dos temas abordados em cada capítulo.

Capítulo 1 – Introdução: Apresentação o projeto desenvolvido, os objetivos e suas etapas. Efetuado de forma sumária a apresentação do trabalho assim como os motivos da sua elaboração.

[1] https://www.microsoft.com/en-us/download/details.aspx?id=42299
[2] https://nmap.org/

Capítulo 2 – Segurança em Aplicações Web: Neste capítulo é apresentada alguns atores que participam de forma ativa nas aplicações Web de hoje em dia.

Capítulo 3 – Segurança de informação: Este capítulo apresenta diferentes abordagens sobre a identificação e catalogação de vulnerabilidades e ataques.

Capítulo 4 – Vulnerabilidades Comuns em Aplicações Web: Descreve três vulnerabilidades comuns em aplicações Web que fazem parte relatório da OWASP 2013.

Capítulo 5 – Estudo de Vulnerabilidades: Este capítulo espelha o resultado do estudo desenvolvido e os passos que foram dados para o seu desenvolvimento no âmbito das vulnerabilidades SSL/TLS.

Capítulo 6- Vulnerabilidades em Domínios Portugueses: Neste capítulo é presentado o resultado efetuado aos domínios português.

Capítulo 7 – Conclusão: Resume o trabalho elaborado ao longo deste projeto sendo feitas considerações sobre o trabalho desenvolvido e referindo possíveis eixos de melhoria.

Capítulo 8 – Bibliografia: Mencionam-se as referências utilizadas ao longo do projeto.

2 Segurança em Aplicações Web

Com a evolução de internet e com o crescente aumento do número de aplicações Web não restam dúvidas que a segurança Web é um tema corrente nos dias de hoje. A nível empresarial a internet tornou-se numa forma de aumentar as receitas e de chegar a um público que até então não era possível. Desta forma é fundamental passar para o exterior uma imagem de segurança e confiabilidade das suas aplicações. Poucos utilizadores querem fazer qualquer transação com uma aplicação que não é segura, muito menos se para isso for necessário o fornecimento de informação sensível. Por outro lado as empresas não querem dar a conhecer as suas vulnerabilidades de segurança sob pena de isso ter alguns custos e principalmente denegrir a sua imagem. Desta forma não é uma tarefa fácil obter informação confiável sob o estado de segurança de uma aplicação Web. O objetivo deste capítulo é fazer uma presentação geral da Web, indicar alguns dos seus principais atores e algumas considerações sobre segurança e como esta é cada vez mais importante nos dias de hoje. Este capítulo pretende apresentas alguns dos principais atores na (in)segurança web, fatores motivacionais que levam alguém a explorar uma vulnerabilidades e identificar alguns protagonistas numa aplicação web.

2.1 Evolução das Aplicações Web

No início a WWW consistia em apenas um conjunto de páginas estáticas que serviam essencialmente como repositórios de informação, funcionando apenas num único sentido, do servidor para o cliente. A segurança nessa altura a segurança não era um bem essencial, as páginas não tinham autenticação, porque não precisavam, não existia segmentação de

informação, todos os utilizadores eram tratados da mesma forma o que resultavam que todos tinham os mesmos acessos e consequentemente a informação apresentada era a mesma. Os conteúdos das páginas não podiam ser alterados, apenas os programadores podiam fazer alterações. Os utilizadores eram apenas consumidores de informação. Estávamos no tempo do que se dominou a Web 1.0. Apesar de uma realidade completamente diferente da que é hoje permitiu uma maior visibilidade e expansibilidade para particulares e empresas, proporcionando acesso a um público que até agora não era possível assim como novas oportunidades de negócio. Se um servidor fosse alvo de um ataque, normalmente, não comprometia informação sensível porque a informação que constava no servidor era aberta ao público, no entanto, a alteração ou manipulação da informação que nele constava poderia ser algo tentador para um atacante.

Atualmente a WWW é totalmente diferente do que era há alguns anos. As páginas Web são na verdade aplicações bidirecionais, em que o fluxo de informação entre servidor e cliente é nos dois sentidos. Passamos de consumidores de informação para produtores de informação. Existe uma dinâmica de informação entre utilizador e aplicação muitas vezes adaptado a cada utilizador. Informação sensível passou a fazer parte da rede atraindo pessoas mal-intencionadas na obtenção dessa informação para seu proveito. A segurança na rede adquire um papel de grande importância na salvaguarda da informação dos utilizadores. Ninguém utiliza uma aplicação se correr o risco da sua informação poder ser divulgada a terceiros. Algumas tecnologias como o Ajax foram fundamentais para o desenvolvimento de páginas Web como as conhecemos atualmente. Estas tecnologias trouxeram novas formas de desenvolvimento de páginas Web, novas ideias surgiram, novos serviços de hospedagem, comunidades baseadas na Web e a promoção da criatividade e da informação partilhada. Com o desenvolvimento de novas tecnologias proporcionou um crescente interesse por aplicações Web trazendo com isso novas ameaças à segurança dos utilizadores. Novas vulnerabilidades surgiram e com isso uma necessidade de repensar de estar na rede. As aplicações podem ser desenvolvidas em varias linguagens e para sistemas operativos diferentes o que pode ser uma problema em termos de segurança. Uma das ameaças à segurança está na falta de sensibilidade dos programadores para este tema o que provoca um conjunto de lacunas que podem provocar prejuízos difíceis de colmatar. Um conjunto teórico de princípios de programação segura não resolve as lacunas de segurança que possam aparecer no código. Aplicações empresariais com interligação a outras aplicações, que contêm informação sensível, devem ter um conjunto de requisitos que

salvaguardem a integridade e a confidencialidade da informação, sob pena que utilizadores mal-intencionados executem ações com prejuízo para a empresa e seus colaboradores.

2.2 Motivação para Ataques Web

As motivações para atacar um sistema informático podem ser tão numerosos como as formas de os executar. Pelo simples desafio, pela emoção da pirataria, questões sociais, status ou simplesmente para ter um proveito financeiro. Para Harvey existem um desejo de descobrir, de mexer, de saber o que houver para saber (Harvey, 1985). O maior aliado dos ataques Web será a mais-valia financeira que estes podem proporcionar. Com o simples roubo de cartões de crédito o ganho pode ser imediato. Roubo de documentos, distribuição de spam, espionagem, obtenção de passwords, são alguns exemplos de ataques que podem ocasionar um proveito financeiro. O desafio de "derrotar" um sistema, principalmente se este for considerado como seguro, é um autoestímulo para abarcar maiores desafios. Naturalmente que algumas motivações mudam ao longo do tempo. A Microsoft afirma que uma das motivações prende-se com o sentimento de invisibilidade e com a obtenção de grandes lucros. Alguns atacantes são apenas curiosos querem ver o que podem aprender sobre determinada empresa não tendo qualquer intenção maliciosa e desconhecendo que estão a violar a lei. Outros têm uma intenção maliciosa relacionada com atividades criminosas. Apoderam-se de informação sensível para venda a terceiros ou pelo simples desfruto de destruir o trabalho dos outros. Existem ainda, segundo a Microsoft, um outro tipo de motivação, o de ser útil. A realização de tarefas de "emergência" permite que as políticas e diretrizes de segurança sejam ignoradas (Microsoft, 2013). Para a Cronkhite a principal motivação dos atacantes é o status. Quanto maior for o desafio e mais sofisticado for o ataque mais prestigio acarreta obrigando a atingir um nível de conhecimento e a demonstra-lo para os seus pares, tendo, eventualmente, a comunicação social como um difusor da informação (Cronkhite & McCullough, 2001). Vários atacantes tornaram-se famosos por desenvolverem ataques a grandes empresas e instituições governamentais tornando-se uma referência no seu tempo.

Scambray definiu um conjunto de características das aplicações Web que motivam os atacantes a desenvolver ataques informáticos (Scambray, Liu, & Sima, 2011) :

- **Ubiquidade** – As aplicações Web estão em todo o lado, redes privadas e públicas e numa grande variedade de ambientes. Encontram-se disseminadas por toda a parte

e disponíveis a todo o momento. É, por isso, improvável que encontrar uma escassez de alvos.

- **Técnicas Simples** - Ao contrário da criação de aplicações complexas usadas, por exemplo, no sistema financeiro ou sistemas operativos as técnicas de ataque na Web são facilmente compreendidas uma vez que a sua maioria é baseada em texto, o que facilita a entrada num aplicativo bastante simples.
- **Anonimato** – Existe ainda muitos locais onde é possível permanecer anonimo sem o perigo de ser detetado. Atacantes mais experientes recorrem a técnicas mais sofisticadas para dificultar o seu rastreamento, utilizando um proxy para cada pedido. Esta é uma das principais razões para a proliferação e aparecimento de novos atacantes.
- **Politicas das Firewalls** – Por regra as firewall têm predefinido um conjunto de políticas que definem o que passa ou não na rede. HTTP(S) é por uma dessas regras que é considerada como segura. Com a popularidade das redes sociais, blogs, etc, não sabemos o que esta por traz e se eventualmente estamos a dar autorização com ser executado código malicioso.
- **Código Personalizado** – Com o acesso facilitado a plataformas de desenvolvimento Web, a maioria destas são cridas por desenvolvedores com pouca experiencia recorrendo por vezes a funções que não seguras e que podem originar falhas de segurança com a consequente exploração.
- **Seguração Imatura** - A autenticação é um dos problemas das aplicações, atualmente as plataformas de desenvolvimento Web disponibilizam um conjunto de mecanismos de gestão de sessões e que tentam colmatar os erros de programação com a criação de sessões mais comuns.
- **Constante Mudança –** Muitas pessoas têm acesso facilitado às aplicações Web, desde os próprios desenvolvedores como os administradores de sistemas, gestores de conteúdos, etc. É por isso difícil definir uma política de segurança que seja aplicada de forma consistente para garantir que as mudanças sejam feitas com sentido de responsabilidade. A consciencialização de segurança nas alterações efetuadas é um bem necessário.
- **Dinheiro –** Um dos fatores motivacionais mais importantes nos dias de hoje. Atacantes comuns, anónimos, singulares degeneram em empresas criminosas desenvolvendo esforços na propagação dos seus ataques tendo em vista um bem

essencial. O phishing ou extorsão são dos crimes mais conhecidos resultando em alertas de diversas empresas, nomeadamente a banca, para proteção dos seus clientes.

2.3 Arquitectura das Aplicações Web

Uma arquitetura básica de uma aplicação Web inclui navegadores, uma rede, e um servidor Web. Os navegadores são responsáveis por enviar os pedidos para os servidores Web que por sua vez processam e executam os pedidos. Cada página contém um conjunto de instruções de formatação expressas em HTML. O HTML foi a linguagem que popularizou as páginas Web, recorrendo a uma variedade de tags e atributos permite definir a estrutura e o layout da página. O protocolo de envio de mensagens HTTP foi outra tecnologia popularizada pela World Wide Web. De uma forma lata esta tecnologia define um conjunto de regras de como as mensagens são formatadas e transmitidas e que tipo de ação os servidores e os navegadores devem ter perante determinados comandos. Apesar destas duas tecnologias serem as mais reconhecidas para a maior parte dos utilizadores existem muitas outras que permitem dinamizar as aplicações *Web*. O recurso a scripts, por exemplo, do lado do cliente também é usual. Os scripts definem o comportamento dinâmico da página e muitas vezes interagem com o navegador e o conteúdo da página. Para o seu correto funcionamento algumas páginas Web necessitam de plug-ins adicionais para que determinado conteúdo funcione.

Do ponto de vista do cliente uma página Web é sempre um documento HTML. Em contrapartida, do lado do servidor, uma página pode manifestar-se de maneiras diferentes. A informação que circula hoje na rede é bem diferente de alguns anos, os utilizadores são cada vez mais exigentes e os conteúdos das páginas cada vez elaborados. Servidores mais robustos, com mais segurança, incluem recursos como a administração remota ou gestão de estado do cliente. O desempenho aliado a características de segurança e escalabilidade são pontos fortes nos servidores atuais permitindo ter várias aplicações com uso intensivo a dados.

2.4 Aspetos de Rede

A forma como comunicamos está dependente de como os elementos que compõem uma rede se interligam entre si. Conseguimos interagir com pessoas do outro lado do mundo com a mesma facilidade com as que estão mais próximas. A rede é o veiculo para a quem quer atacar

e para quem se quer defender. Os próximos pontos descrevem alguns conceitos base de uma rede.

2.4.1 Estrutura

Na estrutura de uma rede está definida os elementos que a compõem. De uma forma lata temos elementos ativos que são os servidores, máquinas clientes, routers, etc. E os elementos passivos que são aquelas que não participam de forma ativa na comunicação mas que são essenciais para o sucesso da mesma, a cablagem e materiais de suporte.

2.4.2 Topologia

A organização de cada componente na rede caracteriza a sua topologia (pontos de conexão, cablagem, etc.). Do ponde de vista de abrangência da própria rede está subjacente os seguintes conceitos: LAN, MAN e WAN. Do ponto de vista do layout físico existem quatro formas base: ponto-a-ponto, barramento, em anel e estrela.

2.4.3 Modelo Cliente-Servidor

O modelo que se pretende, resumidamente, descrever é muito popular nas redes de computadores e utiliza dispositivos específicos cujo propósito se encontra devidamente definida. O modelo cliente-servidor pode ser usado na Internet como em outros contextos como por exemplo redes locais (LANs). Na Web, e como exemplo, este modelo incluí os navegadores, servidores Web, clientes e servidores de FTP (File Transfer Protocol), DNS (Domain Name System), entre outros. Os clientes são tipicamente computadores com aplicações de redes instaladas que recebem e enviam informação para os servidores. O cliente, numa arquitetura web, é uma aplicação executada num host que envia pedidos para outra aplicação. No contexto da arquitetura de uma aplicação web o cliente é o navegador. Este tem como função comunicar com o servidor através de protocolos de comunicação. O mais comum é o HTTP, no entanto, existem outros que neste modelo também são igualmente validos como o FTP, SMTP, SSH, DNS, etc.

Os servidores são entidades que passivamente esperam pedidos dos clientes e então agir sobre esses pedidos. Um servidor armazena, normalmente, aplicações mais complexas que permitem dar resposta aos pedidos dos clientes (ex: bases de dados). Para satisfazer estes pedidos têm

uma maior capacidade de processamento, mais memória e mais discos rígidos de maior capacidade. Os servidores prestam um serviço especifico pelo não é de todo seguro expor o seu conteúdo ao publico, desta forma, tentam encapsular o serviço prestado para que o estado do servidor seja protegido. Existem casos em que os servidores também são clientes para outros servidores permitindo que tarefas sejam divididas em sub-tarefas e para cada uma das sub-tarefas exista um servidor que a executa.

O Instituto de tecnologia e gestão de estudos, Sharada Vikas, definiu um conjunto de características que define um sistema cliente- servidor (Vikas):

- **Serviço** – é uma relação entre processos que se encontram em execução em máquinas distintas. Desta forma o cliente consome um serviço e o servidor é o provedor do serviço. Existe uma separação logica de funções entre cliente e servidor tendo por base o conceito de serviço.
- **Partilha de Recursos** – um servidor gere os recursos compartilhados com os vários clientes que simultaneamente os solicitam.
- **Protocolo Assimétricos –** existe uma relação de muitos para um entre cliente e servidor. Neste diálogo o cliente é quem tem a iniciativo de enviar um pedido ao servidor. Os servidores são passivos e aguardam as solicitações dos clientes. Em alguns casos os clientes tornam-se servidores quando fazem referência a uma *callback* que invoca um serviço.
- **Localização Transparente –** o servidor é um processo que pode residir na mesma máquina do cliente ou em outra maquina que ligada através da rede. Um programa pode adquirir o papel de ciente, servidor ou ambos.
- **Independência** – o software cliente-servidor deve ser independente de plataformas de hardware ou de sistemas operativos.
- **Baseado na Transmissão de Mensagens –** clientes e servidores são mecanismo que interagem através da troca de mensagens. O recurso a mensagens permite solicitar e entregar respostas entre serviços. O cliente e servidor devem ser independentes, isto é, não deve ser condição que para o cliente corra o servidor esteja também a correr.
- **Encapsulamento de Serviços –** o servidor é responsável pelo tratamento e implementação dos serviços, assim como na forma como os implementa. Podem sofrer atualizações desde que a interface das mensagens não seja alterado.

- **Escalabilidade –** a evolução dos sistemas cliente-servidor pode ser feita através da remoção ou adição de novas máquinas (escalamento horizontal) quer por evolução de máquinas mais potentes e consequentemente como um maior desempenho (escalamento vertical).
- **Integridade –** o código e dos dados do servidor deve ser gerido centralmente. Isto permite a salvaguarda da integridade dos dados assim como uma redução dos custos de manutenção.

Existem vários modelos de cliente-servidores desde um modelo primitivo na troca de informação como são os servidores de ficheiros, ou os servidores de base de dados que fazem a troca de informação através de instruções SQL. No contexto desta tese vamos focar-nos sobre os servidores Web. É um modelo relativamente recente que consiste em servidores muito pesados que tentam dar reposta a milhares de solicitações dos clientes. Estes por contrapartida são muito leves e não são apenas computadores pessoais. Os dispositivos os móveis têm tido um papel preponderante fruto de um desenvolvimento tecnológico que lhes permite tirar partido de toda a Web. Cada vez mais rápidos com melhores processadores, mais rápidos e mais potentes, em alguns casos, do que os processadores de computadores de secretaria de há algum tempo. Beneficiando de um custo cada vez mais reduzido e de um largura de banda cada vez maior, atualmente como o 4G podemos atingir os 40MB num dispositivo móvel, permite estar online em todo lado e ter cada vez mais um conjunto de aplicações para quase tudo. As chamadas ao servidor são feitas através destas aplicações ou o pelo navegador do cliente via HTTP. Apesar de um servidor Web parecer uma coisa simples e de fácil implementação, importa referir que as vulnerabilidades foram sendo descobertas ao longo do tempo resultando na sua exploração. O resultado disto é a proeminência para a segurança Web.

2.4.4 Protocolos

Os protocolos representam a sintaxe da rede. Definem as regras para a comunicação entre dispositivos de rede. Incluem mecanismos para que dispositivos possam identificar e fazer conexões entre si bem como regras de formatação que especificam como as mensagens são enviadas e recebidas. Como as comunicações podem ser de várias formas existem vários protocolos que definem a forma de comunicar. Os protocolos existem em vários níveis numa comunicação e por isso não é possível desassocia-los dos modelos padrão OSI (Open Systems Interconnection) e TCP/IP (Transmission Control Protocol / Internet Protocol) sendo que para

cada um dos níveis existem protocolos diferentes e com funções distintas. Naturalmente que também a este nível existem protocolos mais seguros que outros e que por isso assumem um papel relevante na troca de dados. A escolha deve recair em protocolos que garantam que por natureza garantam canais seguros numa comunicação. A este nível o capítulo 5 desta tese descreve as vulnerabilidades existentes nos protocolos SSL/TLS nos últimos anos.

2.5 HTTP(S)

Hypertext Transfer Protocol é o protocolo usado para transferir informação na Web. Foi criado no início dos anos 90 e os navegadores utilizam este protocolo para trocar praticamente toda a informação. Os navegadores são os clientes que interpretam os arquivos de acordo com o tipo de arquivo. Para isso recorrem a um componente essencial na troca de informação, a norma MIME (Multipurpose Internet Mail Extensions), que para além de definir o formato das mensagens de correio eletrónico é um componente fundamental de comunicação do protocolo HTTP, assegurando desta forma que a informação trocada obedeça aos mesmos padrões da troca de correio eletrónico. A W3C define o HTTP como um protocolo ao nível da aplicação para sistemas de informação distribuídos, colaborativos e hipermédia. É um protoloco genérico que pode ser usado para muitas tarefas para além da sua utilização para hipertexto, tais como servidores de nomes e sistemas de gestão de objetos distribuídos (Fielding, et al., 1999). HTTP define o mecanismo para solicitar um recurso, através das URI´s (Uniform Resource Identifiers), e o servidor retorna esse recurso caso ele exista. Os recursos podem ser muito variados e no contexto das páginas Web podem variar desde uma página estática, como acontecia no tempo da Web 1.0, até conteúdo dinâmico com o aparecimento da Web 2.0. As comunicações são efetuadas, por regra, através da porta 80 do protocolo TCP (Transmission Control Protocol). Quase todos os navegadores Web utilizam esta porta, por definição, para comunicarem via Web. O conteúdo da troca de informação entre cliente e servidor ocorre através do cabeçalho das mensagens HTTP, que definem os paramentos de funcionamento de uma operação. São por natureza a parte central destas solicitações e contêm informação sobre o navegador do cliente, da página solicitada, do servidor entre outras.

A figura seguinte mostra um cabeçalho de um pedido via HTTP para determinada página Web.

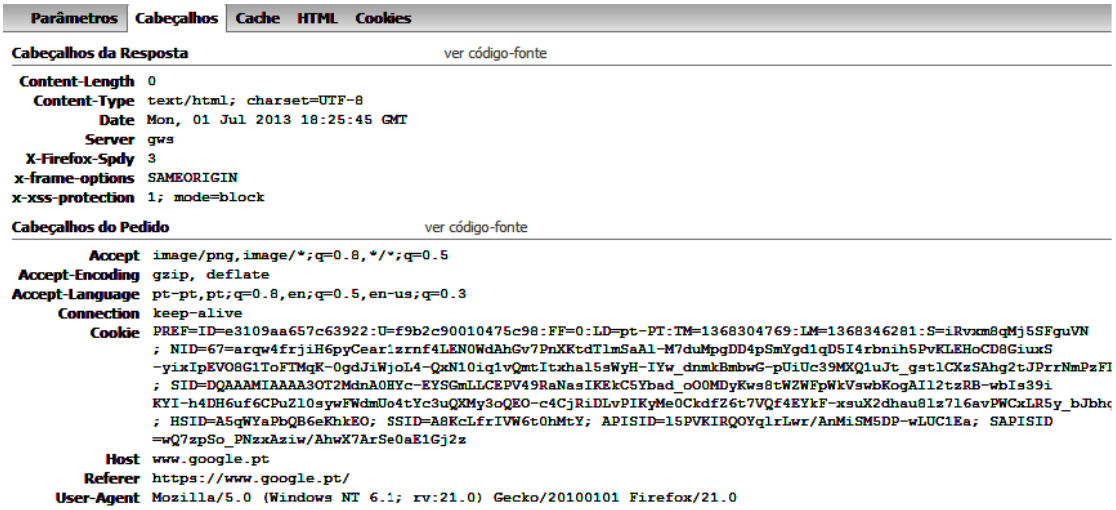

Figura 1 - Cabeçalho HTTP

Conforme mostra a figura não é necessário grandes conhecimentos para que o conteúdo das mensagens não possa ser facilmente compreendido por um utilizador mais curioso e sem grande esforço. Outra propriedade deste protocolo é que não guarda o estado da sessão do utilizador o que significa que dois pedidos distintos executados num curto espaço de tempo são considerados como pedidos novos e únicos. O HTTP para estes dois pedidos envia para o servidor a identidade do utilizador assim como o estado do cookie. Com o intuito de tornar as comunicações mais seguras surgiu o HTTPS (HyperText Transfer Protocol Secure) ou SSL (Secure Sockets Layer). Este protocolo, criado pela Netscape em 1994, envia informação encriptada entre cliente e servidor de forma a que terceiros não consigam, de uma forma simples, aceder ao seu conteúdo. Para isso o HTTPS usa certificados para garantir a identidade do servidor e para proteger a ligação com o navegador do cliente. Para habilitar o HTTPS é necessário um certificado que identifica o servidor durante a conexão. Este certificado pode ser valido apenas para a sessão ou guardado no computador do utilizador de forma a garantir a conexão das futuras sessões com o servidor. Este tipo de ligação é normalmente utilizada para troca de informação sensível e de forma a garantir, para o utilizador, uma imagem de segurança nas transações eletrónicas. A confidencialidade e integridade dos dados é um requisito fundamental no envio de informação sensível na rede e desta forma este protocolo teve varias versões sendo a ultima lançada em finais de 95 ficando com a designação de SSL 3.0. Nos finais dos anos 90 aparece um novo protocolo designado por TLS (Transport Layer Security). Criado pela IETF (Internet Engineering Task Force) é, segundo Stephen Thomas, uma renomeação da última versão do SSL apresentando poucas diferenças entre eles.

	SSL v3.0	TLS v1.0
Protocol version in messages	3.0	3.1
Alert protocol message types	12	23
Message authentication	ad hoc	standard
Key material generation	ad hoc	PRF
CertificateVerify	complex	simple
Finished	ad hoc	PRF
Baseline cipher suites	includes Fortezza	no Fortezza

Figura 2 - Diferenças entre SSL e TLS[3]

Segundo a sua especificação o protocolo TLS apresenta 4 objetivos principais (Dierks & Allen, 1999):

- **Segurança criptográfica** – a sua utilização permite uma conexão segura entre dois pontos.
- **Interoperabilidade** – a utilização do TLS deve poder ser usada por programadores independentes sem que seja necessário a divulgação do código para o estabelecimento da conexão.
- **Extensibilidade** – criação de uma framework para a incorporação de novas chaves públicas e métodos quando necessário.
- **Eficiência Relativa –** operações criptográficas tendem a consumir muito CPU pelo que este protocolo permite reduzir o número de ligações que precisam de ser estabelecidas.

Importa referir que estes protocolos só protegem o caminho, a integridade e confidencialidade dos dados, não protegem o computador do utilizador ou os servidores. Se algum destes elementos for alvo de ataques de software malicioso as informações podem estar comprometidas pelo que o recurso a outro tipo de software atualizado como antivírus, firewall e outros *softwares* de segurança, é uma condição obrigatória.

[3] Fonte - Securing the Web, Thomas (2000).

2.6 HTML

HyperText Markup Language mais conhecido por HTML é uma linguagem de programação utilizada na construção de páginas Web e um dos impulsionadores no desenvolvimento da internet como a conhecemos hoje. Encontra-se especificada pela World Wide Web Consortium (W3C) que a define como sendo um dos principais componentes da plataforma Open Web. Criada nos anos 90 por Tim Berners-Lee, veio introduzir um conjunto de elementos que que permitiam descrever uma página Web. Apesar de ser uma linguagem relativamente recente, o impacto que teve junto dos utilizadores permitiu que a mesma tivesse um conjunto de atualizações num curto período de tempo. A primeira versão surgiu em 1995, aquando da criação da W3C, e ficou conhecida como HTML 3.0, rapidamente apareceu a versão 3.2, em 1997, e um ano depois surgiu a versão que conhecemos por HTML4. Neste ano surgiu a especificação XHTML 1.0 (Extensible HyperText Markup Language**)** que visa juntar o XML (Extensible Markup Language) com o HTML. A W3C define o XHTML como uma família de tipos de documentos atuais e futuros e os módulos que reproduzem, englobam e estendem o HTML4 **Erro! A origem da referência não foi encontrada.**. Basicamente o XHTML 1.0 e XHTML 1.1 tem s mesmas funcionalidades que o HTML4, com exceção de algumas exclusões e extensões que o HTML mas que foram reformulados para o XML.

Como linguagem de marcação, o HTML é definido pelos elementos ou tags que definem um ampla variedade de formato ou as funcionalidades dos elementos no documento. Quando exibidas num navegador as tags são interpretadas e é dada a funcionalidade definida pelas tags.

A última versão é o HTML5 que introduz um conjunto de novas funcionalidades assim como aborda questões de segurança. Esta última versão foi desenvolvida por um grupo constituído por empresas que desenvolvem navegadores e outras entidades que tentam evoluir a especificação HTML. Este grupo denominado de WHATWG (Web Hypertext Application Technology Working Group) surgiu em 2004 é constituído pelas empresas Apple Computer, Inc, Mozilla Foundation e a Opera Software ASA. Como a especificação do HTML5 é efetuada principalmente por este grupo pode-se pensar que esta versão está influenciada por quem desenvolve os navegadores e não por aqueles que a usam. Importa saber até que ponto isto pode influenciar a segurança na rede dos utilizadores. Um responsável da W3C dizia, em 2010, que o HTML5 não estava pronto a ser utilizada em aplicações Web modernas por questões de interoperabilidade (Krill, 2010). O que colocava em causa princípios de segurança para os

utilizadores. Provavelmente nesta altura já estarão ultrapassados, no entanto, a Web está em constante mutação pelo que o aparecimento de novas vulnerabilidades e novos ataques devem ser sempre equacionados na forma que se desenvolvem aplicações.

Um dos princípios associados ao HTML5 é o de manter a compatibilidade com as versões anteriores, o que parece fácil mas se pensarmos que existem milhares de páginas e que o HTML tem mais de 20 anos e para além disso nem todas as páginas respeitaram as regras definidas nas edições anteriores, é uma tarefa árdua. Se por um lado esta nova versão traz novas funcionalidades também introduz algumas questões de segurança que precisam de ser considerados. O recurso a cookies é um deles. Se por um lado libertam o utilizador de um conjunto de conjunto de tarefas (ex: autenticação) por outro levantam alguns princípios de segurança. Os cookies tentam sempre correlacionar as sessões do utilizador com várias tecnologias permitindo armazenar identificadores únicos no computador do utilizador**Erro! A rigem da referência não foi encontrada.**. No entanto esta informação não é consensual segundo o sítio Security Focus a vulnerabilidade dos sistemas é praticamente inexistente. Os cookies apenas podem dizer a um servidor Web se o utilizador já visitou o site e passar informação que, em princípio, não terá efeito danoso para os clientes, como por exemplo o nome do utilizador. Tipicamente um cookie é composto por 5 parâmetros (InfosecWriters, 2015) :

- O Nome do cookie.
- O Valor atribuído ao *cookie*.
- A data de expiração.
- O domínio para que são válidos.
- O caminho do cookie.

Nos últimos anos apareceram algumas tecnologias que tentavam resolver o problema de armazenamento de informação do lado do cliente. Se por um lado há quem defenda que o recurso a cookies pode ser um problema de segurança, existe também que a quantidade de dados armazenada é insuficiente para guardar a informação pretendida. Com isto foram desenvolvidos alguns plugins que permitiam armazenar informação do lado do cliente e esta ser consumida apenas pela página que a armazenou. Com o HTML5 aparece um novo conceito denominado com Web Storage. Este conceito é mais seguro e rápido. A informação não é incluída em cada solicitação do servidor, mas apenas quando solicitada. Também é possível

armazenar, no lado do cliente, grandes quantidades de dados, sem afetar o desempenho da página (W3SCHOOLS, 2015). Cada capítulo da especificação HTML5 tem um subcapítulo correspondente à segurança. O envio de pedidos HTTP encriptados, a segurança relacionada com elementos de canvas ou forms. Estes subcapítulos tendem a cobrir pontos que precisam ser bem pensados aquando da programação das partes correspondentes tendo como objetivo principal identificar as vulnerabilidades dessas mesmas partes e como evitar ser alvo de um ataque. Com base neste princípio surgem duas novas funcionalidades:

- **Web Messaging**: trata-se de uma forma de documentos em contextos diferentes compartilharem dados sem estarem expostos a ataques do tipo cross-site scripting. Basicamente permite que a árvore do DOM de um documento HTML não fique exposta **Erro! A origem da referência não foi encontrada.**.
- **Sandboxing** : limita as capacidades dos IFrames assim como não permitir a execução de JavaScripts, forms, plugins ou links para outras páginas. (Microsoft, 2015).

Apesar de existir uma preocupação com segurança na especificação do HTML5 importa dizer que com o aparecimento de novas funcionalidades não é necessariamente linear que a segurança tenha aumentado, isto porque, se por um lado com a chegada de novas funcionalidades pode trazer uma imagem de maior seguração para as aplicações não podemos descurar que por outro lado podem aparecer novos ataques e consequentemente novas ameaças para os utilizadores. Os programadores têm um papel preponderante na segurança das suas aplicações, pois os recursos de segurança não podem ser usados de forma insegura desvirtuando o conceito para que foram pensados. Uma programação insegura trás novas vulnerabilidades e ameaças abrindo caminho para novos ataques com os custos que podem trazer.

2.7 JavaScript

Interatividade numa página Web é um pressuposto fundamental do sucesso da aplicação, o HTML, sendo uma linguagem de marcação, não dispõem dessas capacidades permitindo apenas definir a estrutura da página. Neste contexto e face às necessidades emergentes de dar vida às aplicações Web surgiu a linguagem JavaScript. Resultado da parceria entre a Sun Microsystems e a Netscape foi lançada a primeira versão em 1995 e impulsionada aquando do lançamento do

Netscape Navigator 2.0 em 1996. Apenas com HTML não conseguimos fazer algo simples como enviar dados para o servidor ou processar informação, é necessário o recurso a uma linguagem que consiga manipular esta informação e processa-la. Existem várias linguagens que o podem fazer como PHP ou ASP no entanto estas linguagens foram pensadas para que a informação fosse processada do lado do servidor. Remotamente ou localmente é sempre necessário um servidor para que as aplicações funcionem. Em contrapartida o JavaScript, através de um interpretador alojado no navegador do cliente, possibilita que a informação seja processada e o seu funcionamento depende das funcionalidades do navegador do utilizador, podendo aceder a um conjunto de API´s disponíveis implementadas pelo navegador. Estas API´s permitem que o *script* comunique com outros elementos da página, assim como aceder a cookies do utilizador, servidores remotos e de manipular outros elementos da página. Programas em JavaScript são essencialmente em embutidos em documentos HTML e executados nas máquinas dos clientes. Com recurso ao Document Object Model (DOM) os programas em JavaScript podem ter acesso ao sistema do cliente e melhorar as funcionalidades da página.

Segundo um estudo realizado por Yue e Wang em 2009 sobre a medição de práticas inseguras da utilização do JavaScript verificaram que das 6805 páginas visitadas 96,9% utilizavam JavaScript (Yue & Wang, 2009). O que demostra a popularidade desta linguagem. No entanto, se por um lado o JavaScript veio dinamizar as aplicações Web por outro abriu uma porta a potenciais ataques que tentam obter proveito de algum desconhecimento por parte de utilizadores mais ingénuos. Quando o código JavaScript se encontra incluído numa aplicação Web, este por sua vez tem os mesmos privilégios que o código originalmente embutido na aplicação. Estes privilégios podem executar um conjunto de ações maliciosas colocando em causa os dados e a privacidade dos utilizadores. O Javascript é frequentemente usado para abrir uma janela no cliente para daí tirar proveito do sistema ou através da injeção de vulnerabilidades. As vulnerabilidades resultam de erros, não detetados, dos autores das aplicações e que podem levar o sistema a um nível de insegurança e consequentemente acarretar alguns custos para os utilizadores. É comum que pela visita a determinada página o utilizador seja confrontado com a abertura de conteúdos não desejados (*pop-ups*) que podem ser difíceis de controlar. Não sabemos qual o código que se encontra escondido na página e que efeito terá se clicar em determinada zona dessa página. Podemos de uma forma involuntária estar a dar autorização ao sistema para abertura de uma porta que vai ser usada por um atacante. Com consequências mais gravosas os ataques de phishing têm explorado as

potencialidades do JavaScript onde, por exemplo, tentam esconder a origem da página através da ocultação da barra de localização ou com barras de localização falsas. A OWASP define este tipo de ataque como a deturpação, onde o atacante recorre à engenharia social para aparecer com uma identidade confiável. Aproveitam a confiança para obter informações valiosas, detalhes de contas ou informações suficientes para abrir contas, ou adquirir bens através de sítios de comércio eletrónico (OWASP, Phishing, 2009). Muitos ataques recorrem a vulnerabilidades conhecidas e que ainda não forma solucionadas pelas empresas de software, permitindo a injeção de código malicioso num servidor vulnerável. Ataques do tipo XSS (Cross-Site Scripting) são muito comuns e dos mais eficazes. Consistem numa falha de validação de parâmetros de entrada e permite que o código persista na página visitada pelo utilizador. O navegador não reconhece este código como malicioso e a proteção é contornada, permitindo ter acesso a informação do utilizador com por exemplo o cookie da sessão.

A partir dos vários tipos de ataques Shacham e Jhala realizaram um estudo e catalogaram os ataques em 4 tipos (Jang, Jhala, Lerner, & Shacham, 2010):

- **Roubos de cookies** – como os scripts podem aceder a todos os objetos do navegador numa aplicação Web da mesma forma podem ter acesso ao *cookie* da sessão. Desta forma um atacante pode ter acesso a um conjunto de identificadores únicos e através de um script transmiti-los para uma aplicação remota sem que o utilizador tome consciência disso.
- **Localização por sequestro** – um código malicioso pode revelar a localização de um documento ou através do endereço de URL. Isto permite que o script navegue na página de uma aplicação web sem o consentimento do utilizador.
- **Roubo do histórico de navegação** - permite através de código malicioso verificar se o utilizador já visitou determinada página ou aceder às páginas visitadas.
- **Análise do comportamento** – um script executado numa página web pode recolher informações sobre o comportamento do utilizador nessa página. Desde os cliques e movimentos do rato, que partes da página são destacadas ou qual o conteúdo na área de transferência. Um script malicioso pode enviar esta informação para um servidor remoto permitindo uma análise comportamental do utilizador por parte de alguém mal-intencionado.

Em geral o JavaScript tem sido explorado para a construção de vários tipos de ataques, a construção de exploits tentam tirar partido de falhas de software desde o sistema operativo até às aplicações Web. Até que estas falhas sejam corrigidas ou até divulgadas os utilizadores podem ser atacados sem se aperceberem, pelo que é fundamental a constante instalação das atualizações de segurança. Se pensarmos que quando fazemos um download de um ficheiro, de origem não muito confiável, podemos sempre cancelá-lo ou até apagar o ficheiro sem o abrir, quando visitamos uma página Web não sabemos se por trás tem código malicioso que é executado com um simples clique.

2.8 Conclusão

A internet é atualmente o principal meio de comunicação dos cidadãos permitindo estar online de forma constante e interruptamente. Temos um leque de soluções que tornam a vida de cada um mais simples permitindo poupar tempo e deslocações. Transações monetárias, realização de compras, partilha de vídeos, músicas e ideias são alguns exemplos do que podemos fazer. E se esta informação se encontra disponível na rede significa que a privacidade dos utilizadores pode estar em risco. Toda esta informação, principalmente a mais sensível, deve ser assegurado por um conjunto de garantias que permitam a troca de informação de forma segura e sem consequências indesejadas para os utilizadores. No entanto, a consciencialização da segurança é da responsabilidade de e todos e não apenas dos que produzem aplicações. Atos inconscientes como desativar mecanismos de defesa ou abrir ficheiros executáveis, que não sejam de fontes seguras, podem propiciar a propagação de software malicioso e tornar vulnerável o próprio sistema. Do lado dos programadores a validação dos inputs e outputs deve ser verificado, acrescentando verificações extras se necessário. A verificação e/ou filtragem da informação via URL permite evitar ataques do tipo Cross-Site Scripting que comumente aparece nos relatórios da OWASP como sendo um dos mais perigosos.

Neste capítulo tentou-se dar uma visão geral da arquitetura das aplicações Web, seus componentes, alguns dos seus principais atores, como o HTTP e o HTML, assim como algumas preocupações que foram tidas em conta no desenvolvimento de novas versões de protocolos mais seguros e linguagens mais adaptadas à realidade de hoje com mecanismos de segurança que não existiam.

3 Segurança de Informação

A necessidade de combater ataques informáticos é cada vez mais importante. Qualquer computador ligado em rede é um alvo potencial. Software malicioso, como vírus ou worms, e ataques em rede são cada vez mais sofisticados tronando-se mais difíceis de mitigar. Erros de programação, utilizadores incautos, inexistência de mecanismos de defesa são alguns exemplos que dificultam a mitigação destes ataques e motivam a propagação de novos. Qualquer pessoa pode colocar uma página web com código malicioso na rede tornando-se desde logo acessível a um número ilimitado de utilizadores. Novas realidades, novas vulnerabilidades e desenvolvimentos tecnológicos oferecem um conjunto de oportunidades para que os atacantes consigam que as suas ações atinjam os objetivos desejados. A engenharia social é um dos fenómenos que ajuda na concretização destes prepósitos e com isto levar a cabo um conjunto de ações, para de alguma forma, obter um proveito. Este capítulo pretende apresentar alguns conceitos relacionados com segurança assim como formas de categorização e representação de vulnerabilidades e ataques.

3.1 Ameaças

No contexto das tecnologias de informação uma ameaça refere-se a qualquer circunstância que tenha potencial de causar danos ao sistema. As ameaças podem degenerar em vulnerabilidades que por sua vez poem ser exploradas colocando em risco indivíduos, sistemas informáticos, etc. Podem ser mal-intencionadas, quando existe a vontade de prejudicar algo ou alguém, ou por erro humano.

O NIST define uma ameaça como "qualquer circunstancia ou evento com o potencial de afetar negativamente as operações organizacionais, indivíduos, outras organizações através de um sistema de informação por via de acessos não autorizados." (National Institute of Standards and Technology, 2012).

Existem quatro tipos principais de ameaças (Cisco Systems, 2010):

- Ameaças não estruturadas – são indivíduos inexperientes que através de ferramentas facilmente acessíveis tentam por em práticas as habilidades de um *hacker* podendo por em causa a operacionalidade do um sistema.
- Ameaças estruturadas – este tipo de ameaças vêm de indivíduos tecnicamente evoluídos que conseguem desenvolver ataques e usar técnicas de invasão sofisticadas.
- Ameaças externas – partem de pessoas individuais ou coletivas externas à empresa sem acesso direto à mesma.
- Ameaças internas – partem de indivíduos com acesso autorizado à empresa e que usufruem de um posição privilegiada para disferir um ataque.

A avaliação do risco é um ponto fundamental nas organizações permitindo entender como pode ser atacada e que impactos que podem provocar. A criação de um modelo, adaptado à empresa, permite classificar a ameaças e a definir regras de atuação. A OWASP propõem um modelo que permite identificar, quantificar e tratar os riscos de segurança associadas a uma aplicação (OWASP, 2015):

- Decompor a aplicação – o objetivo é perceber como a aplicação interage com as outras entidades. Identificação dos inputs da aplicação, que são um ponto por onde o invasor pode tirar partido de alguma vulnerabilidade. Assim como identificar os tipos de acessos que serão concedidos a entidades fidedignas.
- Determinar e classificar ameaças – representa um método de identificação e categorização de uma ameaça. O objetivo é tentar identificar as ameaças do ponto de vista do atacante assim como da perspetiva do defensor e classifica-las de ponto de vista do risco.
- Identificar medidas e mitigação – a implementação de medidas corretivas que mitiguem as ameaças identificadas. Como as ameaças tem um valor de risco predefinido é possível priorizar as regras que visem a sua correção e assim fazer uma

correta avaliação do risco. Não significa a sua correção desde que a avaliação do risco não a justifique.

As ameaças não são apenas a nível de computacional, existem igualmente ameaças físicas como desastres, incêndios, falta de energia, etc. Assim como ameaças humanas como roubo, suborno, invasões, etc. Numa organização todo tipo de ameaças devem ser identificadas assim como estar previsto no plano de emergência.

3.2 Vulnerabilidades

De uma forma lata vulnerabilidade significa um ponto fraco do de algo ou alguém capaz de ser atacado ou danificado. Nos sistemas de informação são uma das principais ameaças à integridade do próprio sistema representando uma preocupação para quem o defende e um desafio para quem pretende tirar proveito. É também comumente definida como uma falha em um componente de software que pode ser explorada, representando uma ameaça para o próprio sistema reduzindo seu valor ou mesmo inutilizando-o. O Glossário de Termos Chave de Segurança de Informação criado pela NIST define vulnerabilidade como uma debilidade do sistema de informação, nos procedimentos de segurança do sistema, controles internos ou na aplicação que pode ser explorada por uma fonte de ameaça (NIST, 2013).
Segundo Correia as vulnerabilidades podem ser classificadas de três tipos (Correia & Sousa, 2010):

- Projeto – Resulta de um falha aquando o sistema ou software ainda está na fase de projeto. Por exemplo não considerar um mecanismo de segurança e a troca de informação na rede pode ser acedida indevidamente.
- Codificação – é um tipo de vulnerabilidade introduzida durante a codificação do software.
- Operacional – trata-se de uma vulnerabilidade causada pelo ambiente no qual o sistema é executado.

A capacidade de detetar vulnerabilidades no próprio sistema é um meio de o tornar mais seguro e confiável e como o número de aplicações Web é cada vez maior o número de vulnerabilidades tende também a aumentar. Para além disto as aplicações são acedidas por um vários utilizadores que acedem de diferentes ambientes podendo estes desencadear um conjunto de

ações cujas consequências e custos são imprevisíveis. Novas tecnologias trazem mais vulnerabilidades e consequentemente novos ataques serão criados. Podendo o invasor ter acesso privilegiado ao sistema de tal forma que não seja possível para o invadido ter consciência que está a ser atacado. Compreender as vulnerabilidades, recolher informações e correlaciona-las com os ataques permite entender as ameaças que estas representam e corrigir os erros de forma a mitigar a fonte de ameaça. Uma melhor compreensão da natureza das vulnerabilidades, suas manifestações e mecanismos que podem ser utlizados para prevenir ou eliminar pode ser atingido através do estudo das mesmas e da correta classificação segundo os seus atributos. Taxonomias tem sido utilizadas para classificar o conhecimento nas mais diversas áreas e, com a evolução tecnológica, a necessidade de as aplicar na área de segurança de informação.

Segundo o relatório de divulgado em 2014 pela Secunia, em 2013 foram descobertas 1208 vulnerabilidades nos 50 programas mais comuns em computadores pessoais (Secunia, 2014). Uma das empresas com maior representação é a Microsoft que é responsável por um vasto número de vulnerabilidades nos seus produtos, no entanto, os programas de terceiros são os que mais representam a fonte de ameaça ao próprio sistema representando 76% dessas vulnerabilidades. Torna-se crucial desenvolver mecanismos de classificação de vulnerabilidades e que serviam de modelos para situações mais atuais e com maior impacto na segurança de informação.

3.2.1 Introdução às Taxonomias de Vulnerabilidades

A palavra taxonomia vem do grego “tassis” que significa ordem e “nomos” que significa norma. Podemos dizer que uma taxonomia é um sistema de classificação para um sistema pré-determinado. O termo taxonomia varia consideravelmente, na verdade o uso deste termo é relativamente recente.

Podemos dizer que uma taxonomia é um sistema de classificação que permite distinguir conceitos e organizar esses mesmos conceitos de uma forma hierárquica, isto é permite agrupar objetos de acordo com as suas semelhanças. É usada para aceder a determinada informação sobre determinado tema de uma forma simples e sem ambiguidade. No âmbito da Gestão de Informação e segundo Gordon, citado por Vital, “as taxonomias são definidas como elementos estruturantes, estratégicos e centrais para negócios baseados em informação e conhecimento para classificar e facilitar o acesso à informação” (Vital & Café, 2009).

Segundo Dutra & Busch "uma taxonomia não é perfeita, mas melhora significativamente a pesquisa e navegação imediatamente que é colocada em prática" (Dutra & Busch, 2003).
Hardy define uma taxonomia como sendo "uma estrutura hierárquica em que os documentos podem ser organizados" (Hardy, 1996).
Atualmente as taxonomias são estruturas de classificação que tem por finalidade servir de instrumento para a organização da informação assim como a recuperação de informação.
Com a sobrecarga de informação e para de alguma forma controlar essa mesma informação é necessário filtrar, categorizar e "etiquetar" a informação que dispomos. Pode-se dizer que necessitamos de novas profissões no sentido em que é imprescindível organizar a informação de uma forma simples e clara. Um exemplo disso mesmo são as páginas Web. Na construção de uma página Web ou na sua reestruturação a organização e a gestão da informação deve ser uma prioridade assim como a recuperação dessa mesma informação. Isto normalmente envolve o uso de conceitos baseados em princípios de catalogação como classificação ou indexação.
As taxonomias também estão a ser usadas no campo da segurança de redes informáticas. Estas taxonomias incluem uma lista de termos, uma lista de categorias e uma lista de resultados. Não têm necessariamente que centrar sobre os ataques aos computadores ou à rede podem se centrar nas falhas de segurança ou vulnerabilidades do sistema que podem ser utilizados para ataques. Do ponto de vista do atacante a sua motivação quando pretende atacar um sistema informático é alcançar os seus objetivos que pode ser simplesmente entrar no sistema ou até causar danos irreparáveis no próprio sistema. Na área da segurança informática é aconselhado o uso de termos mais descritivos em vez de termos mais genéricos.

Para Howard a base para o desenvolvimento de uma taxonomia obedece ao conjunto das seguintes propriedades (Howard & Longstaff, 1998):

- Mutuamente Exclusiva – as categorias não se sobrepõem o que significa que a classificação em um grupo afasta a classificação num outro.
- Exaustiva – todas as possibilidades encontram-se definidas nas suas categorias.
- Exata – a classificação deve ser clara e precisa eliminado a ambiguidade, independentemente do que estar a ser classificado.
- Repetível – a mesma classificação é atribuída a aplicações repetidas.
- Aceitável – logica e intuitiva de forma a ser comumente aceite.
- Útil – a obtenção de conhecimento pode ser usado através do campo de pesquisa.

O desenvolvimento de uma taxonomia pressupõem uma aproximação da realidade onde se insere pelo que o seu desenvolvimento implica o conhecimento e definição das caraterísticas onde esta se enquadra. Quanto maior for a diversidade do ambiente em estudo maior será o número de características estruturais a identificar. O exemplo clássico é classificação dos seres vivos que pela sua grande diversidade é constituído por uma grande número de grupos. A imprecisão ou uma incorreta classificação degenera numa estrutura taxonómica cujo conteúdo é impreciso levando a equívocos.

3.2.2 Taxonomia de Krsul

A taxonomia de Krsul tem por base a taxonomia de Aslam mas com algumas melhorias no processo de classificação das vulnerabilidades. Segundo o autor a as taxonomias existentes têm um número limitado de recursos taxonómicos pelo que sugere novos recursos para melhorar a classificação de vulnerabilidades com base em observações ou medições afim de se evitar ambiguidades. As pressuposições em que a aplicação vai ser executada, o seu comportamento ou o comportamento de outros subsistemas representa uma falha de segurança da própria aplicação que pode degenerar no aparecimento de vulnerabilidades violando as políticas de segurança. Segundo o autor existem quatro classes de suposições que os programadores fazem no desenvolvimento de software e que são responsáveis pelo aparecimento de vulnerabilidades:

- Design – Representa a fase de design do projeto e a compreensão dos requisitos do sistema e do seu ambiente. Uma vulnerabilidade poderá surgir se o designer não compreender a complexidade da aplicação ou se assumir que o ambiente onde vai ser executada tem determinadas características.
- Suposições do Ambiente – resulta da assunção de determinadas características do ambiente em que a aplicação vai ser executada. A vulnerabilidade ocorre quando o programador assume que a aplicação vai correr sobre determinado ambiente.
- Falhas na Codificação – a vulnerabilidade surge por falhas na programação.
- Erros de Configuração – o resultado de uma má configuração com parâmetros impróprios pode representar o surgimento de uma vulnerabilidade.

Tendo por base estes quatro grandes pilares, que se subdividem em múltiplos subníveis, a presente taxonomia é o resultado das suposições que os programadores fazem ao longo de um projeto e cada uma das características da taxonomia pode assumir um de quatro valores possíveis: Sim, Não, Não se Aplica e Desconhecido.

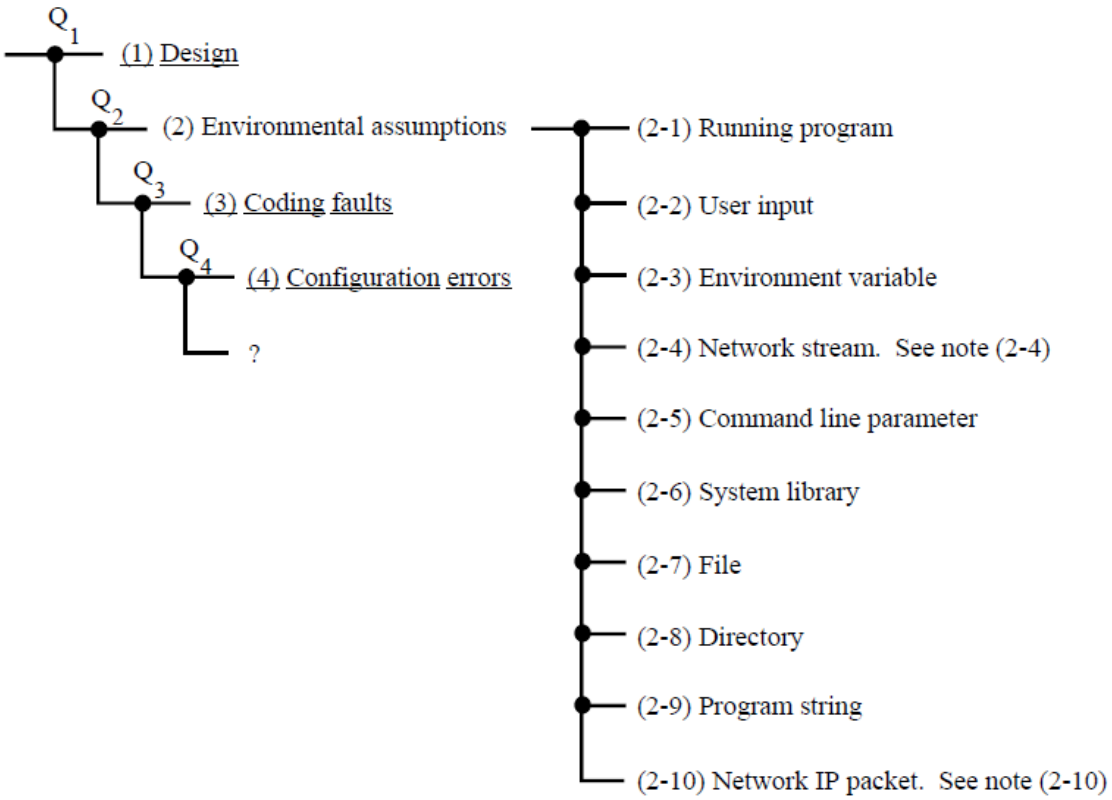

Figura 3 - Taxonomia de Krsul

3.3 Ataques

Qualquer computador conectado à rede é um potencial alvo de um ataque. De forma organizada ou incontrolados, com mais ou menos habilidade ou com mais ou menos conhecimento traduzem-se num prejuízo para quem é alvo. A engenharia social tem contribuído para aproximar atacantes das vítimas e assim facilitar um possível ataque. Os ataques de phishing estão muito ligados a esta técnica que permite enganar utilizadores mais incautos. Os ataques estão diretamente relacionados com as ameaças e vulnerabilidades pelo compreendendo-os, compreende-se melhor os ataques.

A Cisco Systems classifica os ataques como sendo de quatro tipos possíveis (Cisco Systems, 2010):

- Reconhecimento – representa o mapeamento da empresa (serviços, vulnerabilidades, sistemas). O objetivo é tentar obter a maior informação possível sobre o alvo e tentar identificar possíveis vulnerabilidades. Os ataques normalmente caraterizam-se por ter como objetivo obter informações de portas, informações se determinado IP está disponível, assim como espionagem.
- Acesso – representa a capacidade de obter acesso a algo para o qual não está autorizado, tirando partido de alguma vulnerabilidade. A exploração das vulnerabilidades pode permitir o acesso a contas ou serviços não autorizados. Um ataque comum é o de força bruta recorrendo a programas que tentam efetuar login num recurso partilhado (ex.: a utilização de dicionários ou métodos de computação). Ataques de phishing, man-in-the-middle ou engenharia social também estão enquadrados neste tipo de ataque.
- Negação de serviço – significa a intenção de alguém de impossibilitar a disponibilidade de um serviço aos seus utilizadores. Este tipo de ataque significa a intencionalidade de derrubar o sistema ou de provocar ruturas nos níveis de serviço. O esforço para realizar este tipo de ataques é inferior quando comparado com os restantes.
- Worms, vírus e cavalos de troia – código malicioso é inserido no sistema com o intuito de provocar danos e, no caso dos Worms, de se propagarem. Têm a caraterística de se propagarem no sistema conseguindo contaminar um número indeterminável de vítimas. São utilizados, normalmente, para recolher informação sensível dos utilizadores como palavras-chave, no entanto, também podem causar danos significativos na carga e na fluidez do sistema.

De uma forma mais lata podemos ainda classificar os ataques como passivos ou ativos. Os ataques passivos caraterizam-se por não alterarem o fluxo normal da informação, não existe alteração nem corrupção dos dados. A informação é trocada entre emissor e recetor sem que os mesmos tomem consciência que estão a ser monitorizados. O objetivo neste tipo de ataques é apenas recolher informação pela simples escuta de pacotes na rede. Não deixa indícios da sua presença o que torna mais difícil a sua deteção face há ausência de alterações nos dados transmitidos. Os ataques ativos, por sua vez, atuam de forma ativa no fluxo de informação,

criando ou modificando os dados trocados. São o oposto dos ataques passivos. Ataques ativos são difíceis de prevenir, existem mecanismos que dificultam e mitigam de ataques de determinado tipo, no entanto, a grande variedade de aplicações assim como a disponibilidade de acesso faz aumentar o número de vulnerabilidades potenciando ataques deste tipo. Quando combinados os dois tipo os ataques tornam-se mais eficazes.

3.4 Categorização dos Ataques

A categorização dos ataques surge da necessidade de compreender, analisar e classificar os ataques numa rede de computadores, para isso e de forma a explicitar este conceito existem quatro taxonomias que foram desenhadas e que agora se descrevem.

As taxonomias, de uma forma em geral, não foram criadas apenas por um grupo de pessoas, em vez disso evoluíram ao longo do tempo com investigação e desenvolvimento por grupos que pretendiam chegar a um ponto mais alto. Com o desenvolvimento de novas tecnologias surge a necessidade de melhorar as taxonomias até agora desenvolvidas e adaptá-las a novas realidades. A evolução tecnológica e o consequente aumento de informação faz-nos prever alterações a níveis de ambientes Web. Perante este cenário e motivado pelo crescente aumento das tecnologias utilizadas em que os utilizadores adquirem cada vez mais um papel essencial no uso da informação assim como na sua classificação ou categorização dessa mesma informação mas também na produção de informação.

3.4.1 Taxonomia de Álvarez e Petrovic

Tendo por base o ciclo de vida de um ataque Álvarez e Petrovic definiram uma taxonomia para os ataques Web (Álvarez & Petrovic, 2003).

Ao conjunto de passos de uma atividade maliciosa é entendido como sendo um ciclo de vida conforme mostra a Figura 4. O seu objetivo é, com base na análise e classificação de ataques Web, extrair informação que permitisse que os desenvolvedores pudessem construir aplicações mais seguras. Segundos os autores o desenvolvimento de uma taxonomia permitiria prever a fonte de vulnerabilidades das aplicações e consequentemente a sua correção. Tendo por base ataques mais comuns assim como algumas vulnerabilidades estes podem fornecer linhas orientadoras que permitam a construção de novos sistemas que estarão livres destes erros. A taxonomia desenvolvida é multidimensional em que cada dimensão representa uma

característica de um ataque. A sua classificação é horizontal e cada ataque é classificado de acordo com determinadas características. Segundo os autores todos os pontos de entrada têm uma vulnerabilidade que ameaça um serviço que pode ser explorado por uma ação usando uma entrada contra um alvo com o intuito de obter determinados privilégios.

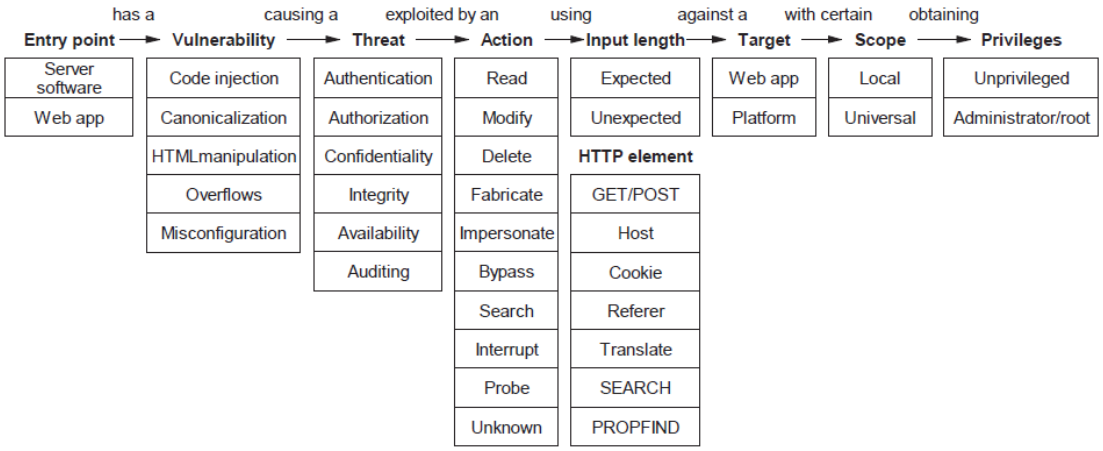

Figura 4 - Taxonomia de Ataques Web

- **Ponto de entrada –** é a origem do ataque e pode ser executado tendo por base uma vulnerabilidade no código do servidor ou da aplicação Web.
- **Vulnerabilidade -** é uma fraqueza do sistema que resulta numa ação não autorizada. No âmbito desta taxonomia uma vulnerabilidade pode ser injeção de código, canonização, manipulação de código HTML, overflows e configurações incorretas.
- **Ameaça** - representa o risco contra vulnerabilidades. A autenticação, confidencialidade, integridade e disponibilidade representam um conjunto de premissas que todos os sistemas devem satisfazer para serem considerados seguros.
- **Ação** – resultante de uma vulnerabilidade a ação entra-se relacionada com o facto de perceber o que é feito na intrusão.
- **Tamanho dos Input´s** – o tamanho dos argumentos passados através de pedidos de HTTP. Entre os ataques mais comuns encontra-se o de buffer overflow sendo, segundo os autores, dos mais fáceis de explorar.
- **Elementos HTTP** – um conjunto de verbos ou métodos que se encontram incluídos no cabeçalho de um pedido HTTP, representam um pré-requisito para um ataque Web.

- **Alvo** – é a vítima do ataque que se encontra diferenciado pela aplicação Web ou plataforma.
- **Âmbito –** é área de impacto do ataque, estes podem ser locais atingindo um grupo pequeno de utilizadores ou universais atingindo todos os utilizadores.
- **Privilégios –** representam os privilégios obtidos após o ataque tendo como objetivo principal a obtenção de privilégios a nível da administração.

3.4.2 Taxonomia de Howard

Esta taxonomia, apresentada por John Howard sobre ataques informáticos e de rede, é baseada em processos tendo em conta as motivações e objetivos dos atacantes (Howard & Longstaff, 1998). Para isso criou cinco etapas para esta taxonomia: atacantes, ferramentas, acessos, resultados e objetivos. Os atacantes são quem pode lançar os ataques. Os meios de obter acesso são representados pelas ferramentas, sendo que os acessos podem ser obtidos através de vulnerabilidades do próprio sistema. Os resultados provêm do sucesso do ponto anterior e podem ser numerosos, desde o simples acesso a informação sensível como a manipulação de informação. Verificando-se os quatro pontos anteriores o atacante atinge os seus objetivos que podem variar desde o simples acesso privilegiado á aplicação como a infeção do próprio sistema com código malicioso.

A classificação desta taxonomia difere do que habitualmente se encontra definido e, com base na literatura, comumente aceite, ou seja, é baseada em processos. Isto traduz algumas críticas que, segundo Lough (Lough, 2001), esta taxonomia não garante a exclusão mutua e algumas categorias podem sobrepor-se. Segundo este, em alguns casos pode não ser possível distinguir um vândalo de um terrorista. Esta taxonomia é baseada no atacante e na forma como prepara e efetua o ataque ao contrário da taxonomia apresentada no ponto 3.4.1 que se fundamenta nas vulnerabilidades do próprio sistema.

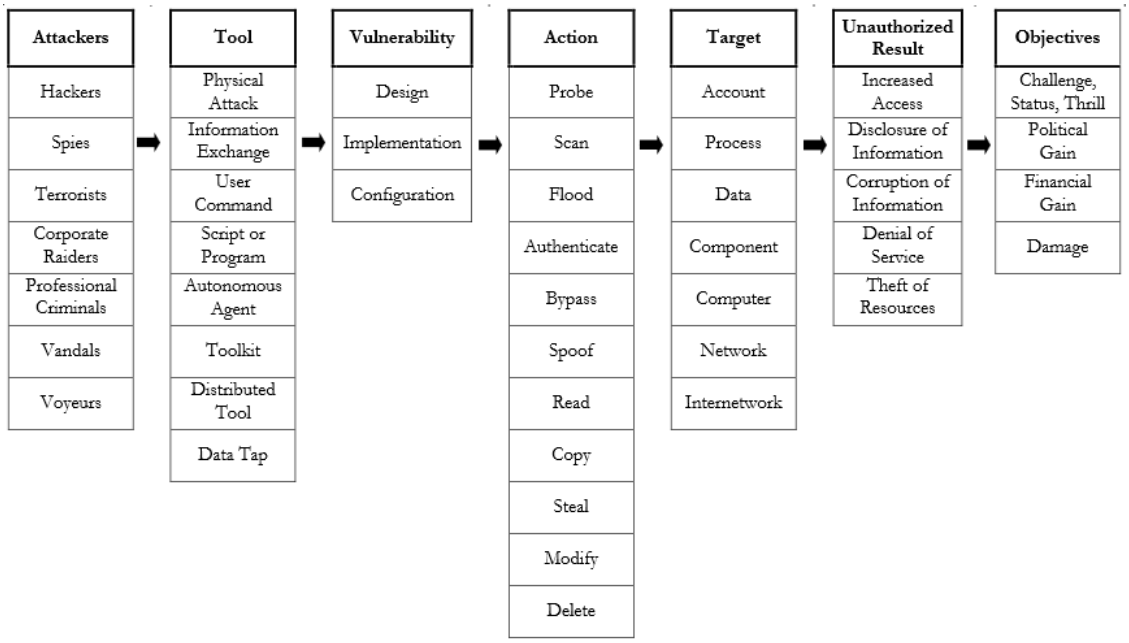

Figura 5 - Taxonomia de Howard

3.4.3 Taxonomia de Bishop

Outra abordagem de uma estrutura taxonómica é apresentada por Matt Bishop que em 1995 apresentou uma taxonomia de vulnerabilidades para o sistema Unix (Bishop, 1995). Esta taxonomia é baseada em seis eixos em que as vulnerabilidades encontradas são utilizadas para criar um esquema de classificação. Desta forma definiu os seguintes princípios:

- **Natureza da Falha** - Tem por base a descoberta e catalogação de padrões de erro de forma a permitir a deteção automática de falhas de segurança.
- **Período de Introdução** – Representa quando a vulnerabilidade foi introduzida no sistema. Normalmente durante o período de desenvolvimento da aplicação.
- **Exploração** – Os resultados que são obtidos através da exploração da vulnerabilidade.
- **Área do Resultado** – É a área que pode ser afetada pela vulnerabilidade.
- **Número Mínimo** – Representa os componentes mínimos necessários para explorar uma vulnerabilidade. Pode ser, por exemplo, um processo com acesso a determinada área da memória ou como a utilização de uma porta.
- **Fonte** – Quem identificou ou descobriu a vulnerabilidade.

No âmbito do desenvolvimento e estruturação desta taxonomia Bishop realizou um estudo comparativo entre taxonomias existentes, algumas que são estruturadas com base nos ataques e outras em vulnerabilidades, e conclui "que muitas vezes ataques e vulnerabilidades são considerados o mesmo problema, no entanto, a distinção dos ataques permite distinguir o mecanismo de aproveitamento da vulnerabilidade subjacente" (Bishop & Bailey, 1996).

3.4.4 Taxonomia de Ataques Web

Uma abordagem semelhante é a que é apresentada por Lai (Lai, Wu, Chen, Wu, & Yang, 2008). Tendo por base o relatório da OWASP concluíram que o problema da segurança na rede se focava na sua grande maioria na existência de um grande número de vulnerabilidades nas aplicações Web e propuseram uma taxonomia baseada em métodos HTTP e separa a linguagem de marcação usada assim como a tecnologia do lado do servidor. Como todos os ataques têm um custo o desenvolvimento desta taxonomia, segundo os autores, permitiria reduzir as perdas daí inerentes. Desta forma foram analisados um conjunto de ataques e definido um conjunto de procedimentos que leva à sua classificação de acordo com as suas características.

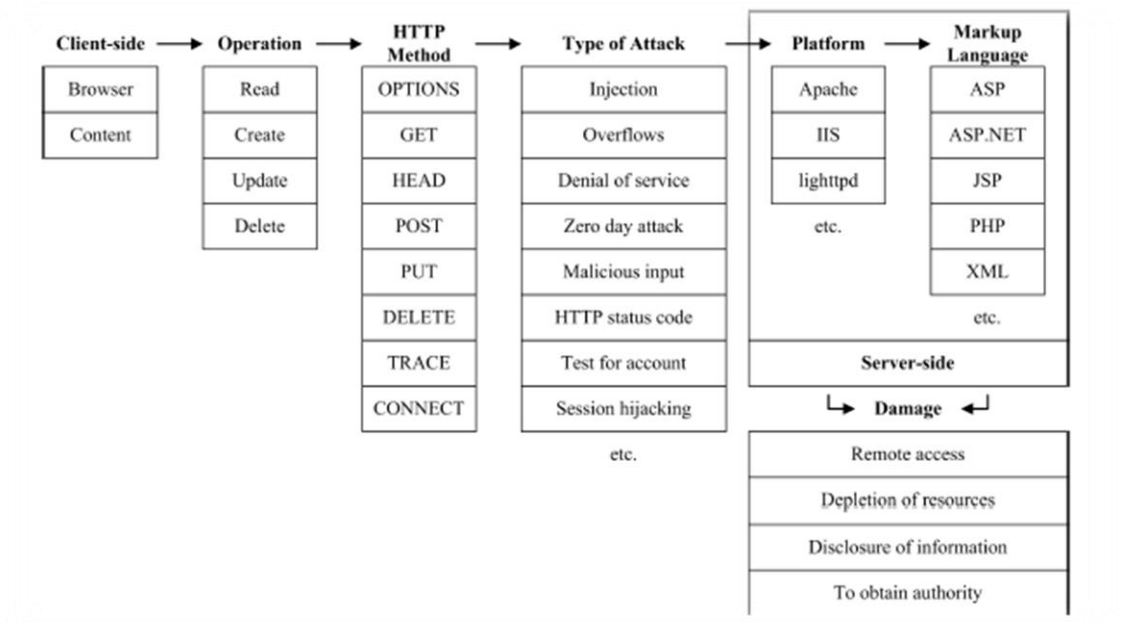

Figura 6 - Taxonomia de Ataques Web

- **Cliente** – A popularidade da internet permitiu o desenvolvimento de novas tecnologias e de novos navegadores. Cada navegador tem particularidades

diferentes na forma de interagir com o ambiente o que poderá traduzir-se em vulnerabilidade diferentes de navegador para navegador.

- **Operação** – Representa o nível de acesso que o utilizador tem para desempenhar determinadas operações. Os acessos a aplicações Web são, por norma, limitados no entanto existe informação que pode ser acedida e alterada mediante o perfil de cada um.
- **Método HTTP** – A troca de informação entre cliente e servidor é efetuado através de métodos HTTP. Os pacotes contêm algum tipo de informação que pode identificar o utilizador de forma unívoca. A forma como os pedidos são efetuados pode simplificar o acesso à informação por parte de um atacante. Por exemplo se para troca de informação sensível utilizamos o método GET esta informação é visível através do URL.
- **Tipo de Ataque** – A perceção e conhecimento de como um ataque ocorre pode incitar o desenvolvimento das defesas necessárias para a sua mitigação.
- **Plataforma** – Existem vulnerabilidades associadas a várias plataformas e o conhecimento do tipo de servidor que a aplicação está a aceder pode facilitar a exploração de uma vulnerabilidade já existente.
- **Linguagem de Marcação** – Representa as linguagens de mais utilizadas para obter, de forma mal-intencionada, informação dos utilizadores.
- **Danos –** Segundo os autores os danos ocorrem por ataques no lado do servidor. O acesso a informação sensível, dos utilizadores ou do servidor, pode provocar perdas difícil de recuperar.

3.5 Conclusão

A informação tem hoje um valor incalculável e um impacto nas organizações que pode colocar em causa a continuidade do negócio. A avaliação e controlo do risco é algo que cada um terá de fazer, não é mensurável visto de fora. As taxonomias, como forma de classificação e catalogação de conhecimento, ajudam a tentar perceber de que tipo de vulnerabilidades e ataques podem comprometer a segurança de informação. Mas não é um tema estanque, novas tecnologias são lançadas, novas vulnerabilidades vão aparecer e novos ataques vão surgir, o quer dizer que este tema carece de acompanhamento e atualização constante. Os estudos aqui

elencados podem ser um ponto de partida para tentar desenvolver novos mecanismos que sirvam de modelo para situações mais atuais e com maior impacto na segurança de informação.

4 Vulnerabilidades Comuns em Aplicações Web

Este capítulo pretende descrever três classes de vulnerabilidades que representam alguns dos maiores riscos das aplicações Web. Para isso, e tendo por base o último relatório da OWASP relativo ao ano de 2013, foram escolhidas 3 tipos de vulnerabilidades para as quais se pretende descrever com mais pormenor. A OWASP tem como objetivo a publicação periódica de um relatório com as 10 vulnerabilidades que representam o maior grau de risco para todos que navegam na internet. Importa referir que as classes de vulnerabilidades não se vão alterando muito de relatório para relatório, apenas vão mudando de posição entre si. Cada vulnerabilidade tem um grau de risco associado sendo que caberá a cada um, e no seu contexto, atribuir um grau de importância à vulnerabilidade presente e avaliar a existência de agentes com capacidade e/ou motivação para explorar a vulnerabilidade. A escolha recaiu pela primeira posição do top (Injeção de falhas), pela terceira posição do top (Cross Site Scripting) por andar nos lugares cimeiros nos últimos relatórios (em 2010 estava em segundo lugar) e por último a exposição de dados sensíveis por ser um tema abordado ao longo desta Tese e que representa, como seria expectável, o maior número do tipo vulnerabilidades no âmbito do estudo descrito no capítulo 5.

4.1 Injecção de Falhas

4.1.1 Descrição

"Injeção de falhas é uma classe de vulnerabilidades de segurança que permite a um utilizador sair do contexto da aplicação web. Se a aplicação web receber informação do utilizador e se inserir esses dados numa base de dados, ou se executar um comando do sistema operativo a aplicação pode ser vulnerável a uma falha de infeção." (Hardin, 2015)

As injeção de falhas podem verifica-se a vários níveis, as mais comuns são as seguintes: Injeção de SQL, XML, LDAP. A injeção de SQL é uma vulnerabilidade que tira partido da aplicação interagir com uma base de dados e com isto o atacante conseguir manipular a informação inserida na aplicação de forma a passar instruções SQL. Este tipo de ataque tem o objetivo de extrair, inserir e adicionar informação na base de dados, assim como identificar o esquema relacional. A aplicação web fica totalmente exposta caso o ataque tenha sucesso.

A injeção de XML é uma técnica usada nas aplicações baseadas em XML, desta forma o atacante tenta manipular as tags do XML comprometendo o seu normal funcionamento. A injeção do tipo LDAP é muito semelhante à do SQL mas o alvo é um repositório LDAP e não uma base de dados.

4.1.2 Defesa e Prevenção

Como prevenção de ataques deste tipo passar por parametrizar os comandos de entrada que são aceites, separando o que é o comando dos dados. Não permitir a inserção de caracteres especiais, caso seja necessário recorrer a algum mecanismo de higienização dos dados (data sanitization). Esta técnica permite garantir a correta normalização da informação removendo carateres perigosos ou dados não expectáveis (uma aplicação que receba comentários dos utilizadores como entrada, não deverá aceitar, por exemplo, marcações de HTML).

4.2 Cross Site Scripting

4.2.1 Descrição

Cross Site Scripting, ou tipicamente identificado pelo acrónimo XSS, é um dos tipos de ataques mais comuns, a nível da aplicação, que os hackers usam para tirar partido de uma vulnerabilidade das aplicações Web. Tem sido nos últimos anos uma classe de vulnerabilidades dominante nas aplicações Web. A razão subjacente a esta vulnerabilidade encontra-se o facto das linguagens de marcação Web, do lado do cliente, não oferecerem mecanismos de segurança que permita o isolamento seguro dos dados. Esta técnica de ataque permite que uma pagina Web contenha código malicioso, que, em seguida, é executado do lado do utilizador. "Um invasor pode usar XSS para enviar um script malicioso para um utilizador desprevenido. O navegador do utilizador final não tem forma de saber se o código é confiável e executará o código malicioso que poderá aceder a informação sensível do utilizador." (OWASP, 2014)

Tipicamente, mas nem sempre, o código escrito em HTML e/ou JavaScript não é executado do lado do servidor mas sim no navegador do lado do cliente, sendo o servidor apenas o hospedeiro do código malicioso. O atacante aproveita-se da confiabilidade que o utilizador tem na página Web como um meio para atingir um fim, recorrendo a atos nefastos que vão desde a invasão de conta, gravação de teclas, roubo de cookies, etc. Na Common Weakness Enumeration uma vulnerabilidade do tipo XSS é descrita como "uma das mais prevalentes, obstinadas e perigosas em aplicações Web. É praticamente inevitável quando combinada com HTTP, scripts em HTML, a troca de informação entre aplicações Web, diversas formas de codificação e diversidade de recursos disponibilizada pelos navegadores. Se não tiver cuidado pode ser injetado código malicioso que o navegador executará. A página é acedida por outros utilizadores cujos navegadores executarão o código malicioso. " (CWE/SANS, 2011).

Os principais tipos de vulnerabilidades XSS são: por reflexão (não persistentes), por armazenamento (persistentes) e baseados em DOM.

4.2.2 XSS por Reflexão

XSS por reflexão (ou não persistente) ocorre quando o script injetado é executado imediatamente. Parâmetros de consulta HTTP ou formulários HTML são formas comuns de aproveitamento deste tipo de vulnerabilidades, por exemplo num processo de autenticação

caso se verifique que os dados introduzidos não sejam válidos, e a página devolver os valores inseridos pelo utilizador, estamos perante uma vulnerabilidade por reflexão. Qualquer autenticação não deve retornar ou fazer referência aos valores introduzidos. Uma técnica comum neste tipo de ataques passa por enviar uma mensagem para a vítima com uma hiperligação para a página que usualmente utilizam. Nessa mensagem colocam um script que quando acedido será executado. Existe uma relação direta neste tipo de ataques e a engenharia social pois a vítima é levada a crer que a informação recebida é legítima. Da perspetiva da vítima limita-se a seguir um URL de um sítio Web em que confia. Este ataque podes ser confundido com um ataque de phishing, no entanto, estes tentam iludir a vítima através de um URL parecido com que a vítima conhece não obstante este tipo de vulnerabilidades pode potenciar ataques de phishing mais perigosos.

Um a vulnerabilidade deste tipo encontra-se identificada na base de dados da NVD com o identificador CVE-2014-1879. Esta vulnerabilidade encontra-se dentro da interface de importação de ficheiros phpMyAdimin que é muito utilizado por programadores para manipulação de bases de dados. Esta vulnerabilidade pode ser explorada carregando o código malicioso através da interface de importação que por sua vez irá processar código. No entanto a execução do código malicioso ocorre dentro do navegador do utilizador.

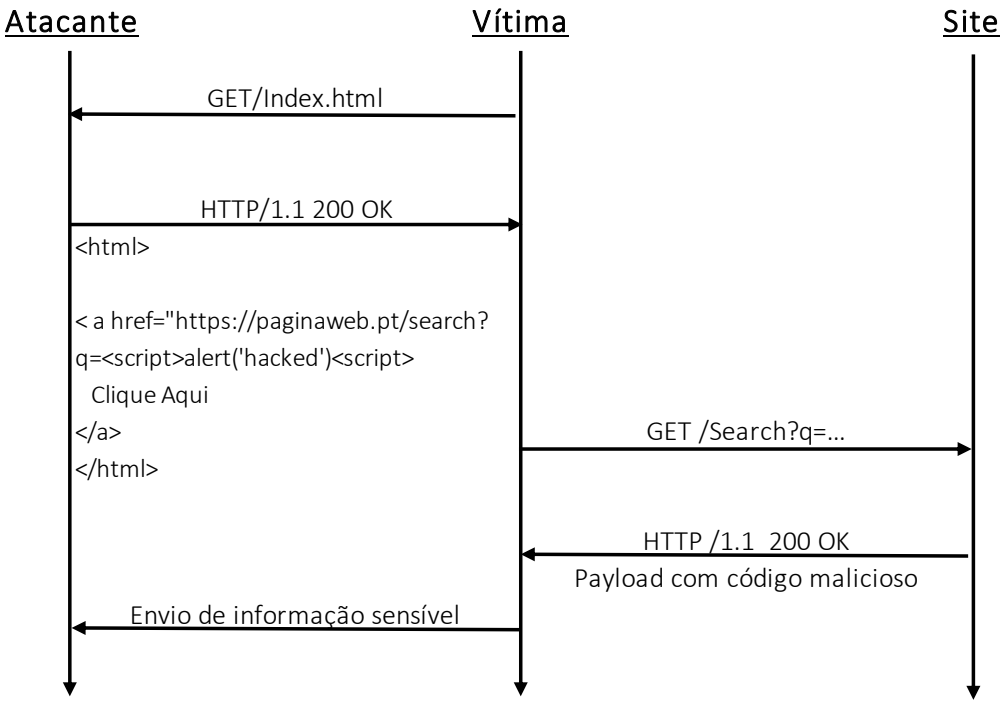

Figura 7 – Fluxo XSS por Reflexão

4.2.3 XSS por Armazenamento

Este tipo de vulnerabilidades, também conhecida como XSS persistente, é semelhante à anterior, no entanto, o sítio Web vulnerável armazena o código malicioso e envia estes dados para qualquer utilizador que aceda a esses dados. Quer isto dizer que apenas a forma de atacar muda relativamente anterior, visto que o código malicioso pode ser o mesmo. Este tipo de ataques encontra-se particularmente presente em sítios onde a validação de inputs se encontre de alguma forma reduzida como fóruns ou blogs, permitindo chegar a número de potenciais alvos bastante alargado, podendo inclusivamente ser usado código de autopropagação de forma a exponenciar o número de vitimas. Em 2005 uma vulnerabilidade deste tipo permitiu que um worm XSS, denominado Samy Worm, infetasse mais de um milhão de utilizadores em apenas 24 horas na rede social MySpace (Auger , 2011).

A vulnerabilidade CVE-2014-3988 da base de dados na NVD é um exemplo do tipo XSS por armazenamento. O produto por detrás desta vulnerabilidade é a aplicação de código aberto KCFinder que é um gestor de ficheiros web. Para esta vulnerabilidade um utilizador que tenha acesso a um servidor, e que utilize esta aplicação para gerir os seus ficheiros via web, pode ser explorado através desta vulnerabilidade. Para isso o código malicioso é depositado no servidor e quando um utilizador navegue até ao local do ficheiro, este é executado no seu navegador. Para além disso esta aplicação é utilizada para como um plug-in para estender funcionalidades de outras aplicações o que significa que estas aplicações ficarão igualmente vulneráveis.

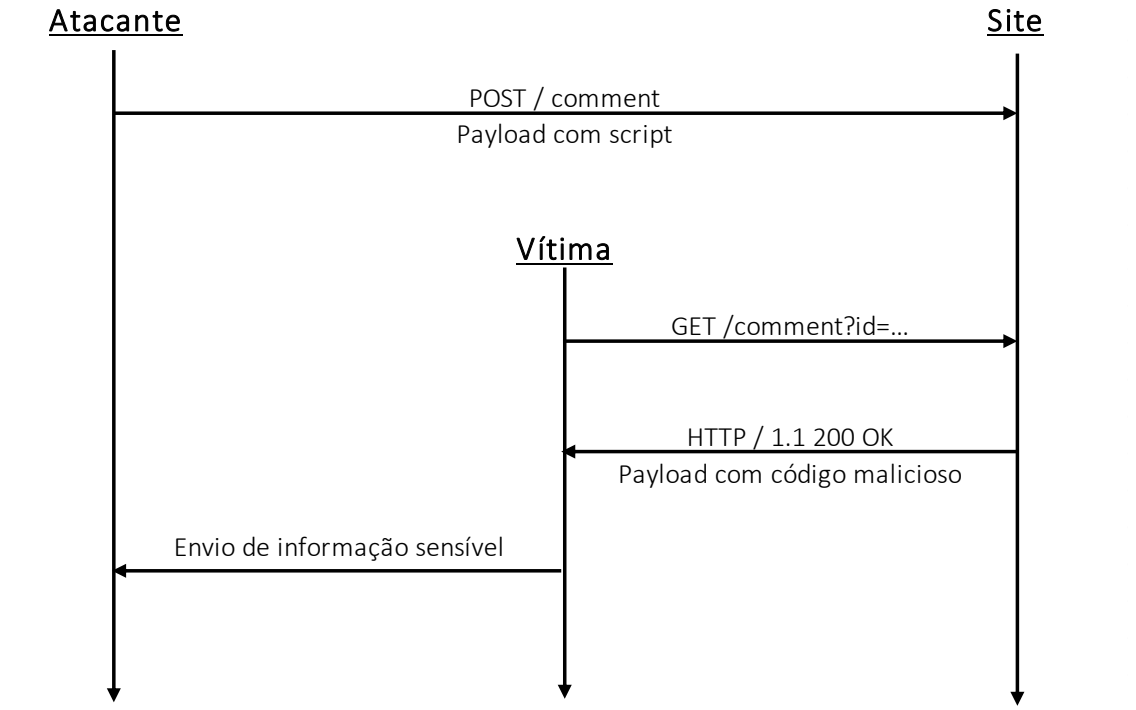

Figura 8 – Fluxo XSS por Armazenamento

4.2.4 XSS Baseado em DOM

Este tipo de vulnerabilidades é caraterizado por uma fragilidade em algum dos componentes da página Web ao invés de ser embutido no código fonte. O ataque baseado em DOM permite alterar diretamente as propriedades dos objetos DOM em vez de se encontrar em alguma parte do HTML. Ao contrário dos exemplos anteriores em que o código malicioso está dentro da página quando é executado, este resulta de uma alteração de um objeto DOM no navegador da vítima, ou seja, a página em si não muda mas o código do lado do cliente é executado de forma diferente devido às alterações ocorridas no ambiente DOM.

Nos XSS anteriores o código malicioso é executado quando a página é carregada com fazendo parte do HTML enviado pelo servidor, enquanto que no XSS baseado em DOM o código é executado em algum momento após o carregamento da página.

4.2.5 Defesa e Prevenção

Qualquer defesa ou mecanismo de prevenção de uma vulnerabilidade tem subjacente, antes de mais, a sensibilização do programador para as questões de segurança, quer no tratamento de dados não confiáveis quer na própria programação segura. Nas vulnerabilidades do tipo XSS

o a validação dos inputs e a codificação dos outputs são essenciais para mitigar situações que podem trazer consequências de danosas para a aplicação e/ou utilizadores. Desta forma, e por princípio, não se deve confiar nos dados inseridos por terceiros. A validação dos dados implica que a própria aplicação assegure que os dados inseridos são os corretos pelo que qualquer dado que não seja o esperado deve ser descartado. Poderá recorrer higienização dos dados conforme referido no tipo de vulnerabilidades do ponto 4.1.

Embora exista um número inimaginável de proceder a ataques XSS, a que acresce a existência de ferramentas que possibilitam verificar a existência de vulnerabilidades deste tipo, a OWASP definiu um conjunto de regras que estabelecem um conjunto de boas práticas contra ataques deste tipo, a saber (OWASP, 2015):

a) Nunca insira dados não confiáveis exceto em locais definidos para o efeito. O princípio desta regra é negar tudo e principalmente não aceitar código JavaScript de uma fonte desconhecida e a seguir executa-lo.
b) Valide os carateres de escape antes de os inserir dentro de elementos HTML. A não validação dos inputs pode permitir a injeção de código malicioso dentro da aplicação. Suponhamos que esta a desenvolver uma aplicação do tipo fórum ou blog em que se espera que os utilizadores insiram comentários num determinado local. Pode ser possível a inserção de algo como “O meu primeiro post <script src = "// sitemalicioso.pt/RoubodeCookiez.js"> </ script>!”. O que aparecerá publicado será “O meu primeiro post!” mas o navegador também irá ver o código Javascript que lhe permite roubar os cookies para um outro local.
c) Valide os carateres de escape antes de os inserir código dentro dos atributos HTML. Esta regra só se aplica aos atributos mais comuns como “name” ou “value”. Atributos como “src”, “href” ou “class” estão foram do âmbito desta regra.
d) Não corra o risco de inserir dados não confiáveis em strings de Javascript. Se a aplicação permitir gerar dados dinamicamente deverá precaver que a mesma tenha o comportamento que se espera que tenha.
e) Valide os dados antes de os colocar dentro das propriedades CSS. Ao inserir dados não confiáveis em etiquetas CSS é importante ter em consideração que estes podem ser usados para ataques pelo que devem ser devidamente validados e só devem ser colocados nos valores das propriedades.

f) Valide os parâmetros do URL e verifique a informação que envia nos pedidos HTTP, os escapes hexadecimais usados nos URL´s como por exemplo %25, em que 25 é o código ASCII do caráter "%".
g) Evite ataques de injeção de HTML recorrendo a bibliotecas que analisam o HTML inserido como por exemplo: HtmlSanitizer, OWASP Java HTML Sanitizer ou PHP Html Purifier.

Concluindo a validação do input e a codificação do output são os dois pilares para a mitigação deste tipo de vulnerabilidades. O tipo de estratégia adotado para implementar este princípio é que pode não ser de fácil realização. Conforme defendido por Kang existem três estratégias de validação. A criação de uma lista de dados confiáveis, a criação de uma lista de dados não confiáveis e a higienização dos dados (Kang, et al., 2005). A estratégia da primeira é de apenas aceitar dados que se encontrem devidamente identificados, esta opção e particularmente útil quando o número de possibilidades é reduzido, por exemplo se a entrada é o dia do mês. A segunda é o oposto da primeira o que apresenta logo à partida um problema que é a identificação de todos os casos em que não é possível a aceitação de dados, podendo ficar alguns casos de fora. A terceira consiste em identificar o que é problemático nos dados e proceder à sua remoção ou codificação (tradicionalmente os metacaracteres).

4.3 Exposição de Dados Sensíveis

4.3.1 Descrição

Esconder informação é algo que não é novo e já se fazia no passado muito antes de se ouvir falar em computadores. As principais diferenças para os dias de hoje prendem-se com a tecnologia aplicada e não forma como enviamos as mensagens. Hoje em dia todos nós utilizamos a Web para o envio de informação sensível no nosso dia-a-dia. As aplicações web são responsáveis por zelar que esta informação chega ao seu destinatário sem ser manipulada ou fornecida a terceiros e por isso quase todas as aplicações precisam de garantir que esta informação é guardada de forma segura. Para garantir isto têm obrigatoriamente que implementar mecanismos que mitiguem ou dificultem o acesso a dados. Neste âmbito os algoritmos de encriptação são responsáveis pela transformação de informação original, numa

ilegível para terceiros. Este mecanismo tem como objetivo assegurar que a informação enviada, apenas é possível a sua descodificação por pessoas que possuam a chave de desencriptação. “Algoritmos de criptografia definem transformações de dados que não podem ser invertidas facilmente por utilizadores não autorizados. Nenhum algoritmo é ideal para todas as situações.” (Microsoft, 2015).

Neste processo existem dos conceitos base, a encriptação que é o processo de transformação de dados de forma a torná-los inteligíveis e a desencriptação que faz o processo inverso. No desenvolvimento de aplicações construir uma aplicação sem criptografia é mais simples, no entanto, a segurança pode ficar seriamente afetada. Existem dois tipos de criptografia: simétrica e assimétrica. A criptografia simétrica utiliza apenas um chave que é do conhecimento do emissor e recetor de informação e que é usada para cifrar e decifrar. Tem a vantagem de ser um algoritmo mais rápido mas não apresentam o mesmo nível de segurança da assimétrica. Na criptografia assimétrica temos duas chaves, a publica e a privada. A chave pública é usada para encriptar e a chave privada é usada para desencriptar. Apenas a chave publica é partilhada com o recetor. Um ataque comum neste tipo de vulnerabilidade é do tipo Man-in-the-Middle e em Outubro de 2014 foi revelada uma vulnerabilidade que coloca em risco a integridade dados dados dos utilizadores (CVE-2014-3566). Os seus descobridores Bodo Möller, Thai Duong e Krzysztof Kotowicz deram-lhe o nome de Poodle (Möller, Duong, & Kotowicz, 2014). Esta vulnerabilidade tira partido da versão 3.0 do protocolo SSL comprometendo a sua criptografia. Apesar dos navegadores mais recentes utilizarem o TLS nas suas comunicações, a versão SSLv3 ainda existe num grande número de aplicações sendo possível forçar a utilizar uma ligação SSLv3. A vulnerabilidade pode ser explorada por induzir o navegador do cliente a fazer vários pedidos sobre HTTPs com SSLv3 e a inferir detalhes sobre o conteúdo criptografado. Segundo investigadores da Websense Security esta vulnerabilidade é crítica, que pode ser explorada e o roubo de dados pode ser significativo (Websense Security Labs, 2015). A investigação indica que esta vulnerabilidade, em princípio, só se verifica do lado do cliente. Esta é uma vulnerabilidade que afeta o protocolo SSLv3 em si e não um aplicação em particular (ex: Sistema operativo). Segundo a Microsoft a única forma de mitigar esta vulnerabilidade é desativar o SSLv3, sendo que na ultima versão de Internet Explorer 11 lançada este ano a mesmas já se encontra desabilitada (Microsoft, 2015). No entanto uma nova variação do Poodle surgiu em Dezembro de 2014 e tira proveito de uma falha nos protocolos TLS 1.2 (CVE-2014-8730). Isto significa que um atacante já não tem de fazer o downgrade para SSLv3 para executar um ataque. Convém

referir que não é só o comum HTTPS que implementa os protocolos SSL\TLS, outros protocolos utilizam o SSL/TLS para asseguram uma comunicação segura como é o caso do FTP (porta 989).

4.3.2 Defesa e Prevenção

Neste tipo de vulnerabilidades o atacante tem alguma dificuldade em conseguir atingir o seu objetivo, principalmente se os algoritmos usados forem bastantes fortes, no entanto, a OWASP indica 5 recomendações para minimizar os perigos de uma comunicação insegura (OWASP, 2013) :

- Criptografar toda a informação sensível e avaliar que de ameaças se pretende proteger.
- Evitar o armazenamento de dados sensíveis quando não se justifica.
- Usar chaves de encriptação fortes e gerir estas chaves.
- As palavras passe devem ser armazenadas com um algoritmo designado para o efeito.
- Os formulários não devem guardar em cache informação do utilizador.

Se aliado às boas práticas acimas descritas, forem utilizados algoritmos criptográficos testados e confiáveis aquando do desenvolvimento de software, termos mais aplicações confiáveis. Desaconselha-se reinventar algoritmos criptográficos.

4.4 Conclusão

A Web revolucionou a forma de comunicar assim como veio alterar a perceção de privacidade, integridade e confidencialidade da informação. Possibilita de uma forma quase infinita a partilha de informação. No entanto, se de um modo veio contribuir para o desenvolvimento da comunidade virtual, por outro esta troca de informação permite ter acesso a informação a um ritmo cada vez maior. O acesso a ficheiros passa a ser acessível a todos de uma forma nunca antes alcançada. O desenvolvimento de aplicações Web em muito contribuiu para isto. Mas se esta informação se encontra disponível online, isto significa que a nossa privacidade pode estar em risco. Esta questão vem levantar cada vez mais o problema da segurança das aplicações web, o acesso a informação sensível e de carácter confidencial por terceiros cujo objetivo é tirar partido desses dados para uso indevido. Boas práticas de programação devem ser implementadas e testes de vulnerabilidade devem ser executados antes de colocamos em produção uma aplicação. Validação de dados, implementação de criptografia na comunicação,

utilização de chaves seguras, utilização de protocolos seguros são alguns exemplos que visam aumentar a segurança das aplicações, mesmo que isto possa causa alguma perda no seu desempenho.

5 Estudo de Vulnerabilidades

Este capítulo explica a metodologia adotada no estudo de vulnerabilidades, com um foco particular sobre a forma como um conjunto de dados foi selecionado, filtrado e processada a informação. O capítulo inclui uma serie de analises sobre vários pontos de vista das vulnerabilidades que afetam os protocolos SSL/TLS. Descreve ainda alguns componentes, que não sedo alvo de estudo, são importantes para a retirada de conclusões. No final do capítulo resume-se os resultados.

5.1 Fonte de Informação

As vulnerabilidades representam uma das principais causas de insegurança nos dias de hoje, permitindo destabilização e roubo de informação dos utilizadores. Internet, Bluetooth, infravermelhos são alguns exemplos de meios de acesso a informação que podem propiciar a exposição de dados, violando a privacidade e expondo informação confidencial. A National Vulnerability Database (NVD) é uma fonte de dados pública que mantém informação padronizada sobre vulnerabilidades publicando mais de 50000 vulnerabilidades desde a sua criação. A NVD agrega informação de quase 100 empresas de segurança, grupos de consultadoria e organizações (MITRE, 2014). Para além disso é complementada com os componentes que inclui Common Vulnerabilities and Exposures (CVE), Common Weakness Enumeration (CWE), Common Platform Enumeration (CPE) e Common Vulnerability Scorring System (CVSS), estes explicados com mais detalhe nos pontos 5.1.1,5.1.2,5.1.3 e 5.1.4.

Todos os dados são disponibilizados em formato XML, estando nesta altura disponíveis as vulnerabilidades desde 2002 a 2015. Para o estudo desta tese apenas foram consideradas as vulnerabilidades reportadas no período de 2010 a 2015. Cada entrada (vulnerabilidade) tem um identificador no formato CVE-ANO-NUMERO, cada vulnerabilidade encontra-se ordenada pela data da sua publicação, os produtos afetados pela vulnerabilidade, o score CVSS que contém um conjunto de métricas que visam quantificar o nível de gravidade da vulnerabilidade em diversos parâmetros, uma descrição da vulnerabilidade, o CPE assim como a classificação da vulnerabilidade segundo o seu tipo (representado no XML por CWE). Para o tratamento e análise da informação constante nos diversos ficheiros XML foi desenvolvida uma base de dados em SQL de forma a agregar toda a informação necessária. A Figura 9 exemplifica uma entrada do ficheiro XML disponibilizado pela NVD.

```
<entry id="CVE-2015-0017">
    <vuln:vulnerable-configuration id="http://www.nist.gov/">
      <cpe-lang:logical-test operator="OR" negate="false">
        <cpe-lang:fact-ref name="cpe:/a:microsoft:internet_explorer:6"/>
        <cpe-lang:fact-ref name="cpe:/a:microsoft:internet_explorer:7"/>
        <cpe-lang:fact-ref name="cpe:/a:microsoft:internet_explorer:8"/>
        <cpe-lang:fact-ref name="cpe:/a:microsoft:internet_explorer:9"/>
        <cpe-lang:fact-ref name="cpe:/a:microsoft:internet_explorer:10"/>
        <cpe-lang:fact-ref name="cpe:/a:microsoft:internet_explorer:11:-"/>
      </cpe-lang:logical-test>
    </vuln:vulnerable-configuration>
    <vuln:vulnerable-software-list>
      <vuln:product>cpe:/a:microsoft:internet_explorer:7</vuln:product>
      <vuln:product>cpe:/a:microsoft:internet_explorer:9</vuln:product>
      <vuln:product>cpe:/a:microsoft:internet_explorer:6</vuln:product>
      <vuln:product>cpe:/a:microsoft:internet_explorer:8</vuln:product>
      <vuln:product>cpe:/a:microsoft:internet_explorer:10</vuln:product>
      <vuln:product>cpe:/a:microsoft:internet_explorer:11:-</vuln:product>
    </vuln:vulnerable-software-list>
    <vuln:cve-id>CVE-2015-0017</vuln:cve-id>
    <vuln:published-datetime>2015-02-10T22:00:34.527-05:00</vuln:published-datetime>
    <vuln:last-modified-datetime>2015-02-18T22:00:15.320-05:00</vuln:last-modified-
datetime>
    <vuln:cvss>
      <cvss:base_metrics>
        <cvss:score>9.3</cvss:score>
        <cvss:access-vector>NETWORK</cvss:access-vector>
        <cvss:access-complexity>MEDIUM</cvss:access-complexity>
        <cvss:authentication>NONE</cvss:authentication>
        <cvss:confidentiality-impact>COMPLETE</cvss:confidentiality-impact>
        <cvss:integrity-impact>COMPLETE</cvss:integrity-impact>
        <cvss:availability-impact>COMPLETE</cvss:availability-impact>
        <cvss:source>http://nvd.nist.gov</cvss:source>
        <cvss:generated-on-datetime>2015-02-11T13:55:59.750-05:00</cvss:generated-on-
datetime>
      </cvss:base_metrics>
    </vuln:cvss>
    <vuln:cwe id="CWE-399"/>
    <vuln:references xml:lang="en" reference_type="VENDOR_ADVISORY">
      <vuln:source>MS</vuln:source>
      <vuln:reference href="http://technet.microsoft.com/security/bulletin/MS15-009"
xml:lang="en">MS15-009</vuln:reference>
    </vuln:references>
    <vuln:references xml:lang="en" reference_type="UNKNOWN">
      <vuln:source>SECTRACK</vuln:source>
      <vuln:reference href="http://www.securitytracker.com/id/1031723" xml:lang="en">
1031723</vuln:reference>
    </vuln:references>
    <vuln:references xml:lang="en" reference_type="UNKNOWN">
      <vuln:source>BID</vuln:source>
      <vuln:reference href="http://www.securityfocus.com/bid/72402" xml:lang="en">72402
</vuln:reference>
    </vuln:references>
    <vuln:summary>Microsoft Internet Explorer 6 through 11 allows remote attackers to
execute arbitrary code or cause a denial of service (memory corruption) via a crafted
web site, aka "Internet Explorer Memory Corruption Vulnerability," a different
vulnerability than CVE-2015-0020, CVE-2015-0022, CVE-2015-0026, CVE-2015-0030, CVE-2015-
0031, CVE-2015-0036, and CVE-2015-0041.</vuln:summary>
  </entry>
```

Figura 9 - Exemplo de uma entrada no ficheiro NVD

5.1.1 Common Platform Enumeration

É uma especificação que descreve um sistema de nomenclatura estruturado no âmbito das tenologias de informação (hardware, sistemas operativos e aplicações). A sua especificação inclui uma descrição de como construir nomes CPE´s através de informação do produto, um dicionário que coleta todos os nomes CPE ´s conhecidos assim como informações de diagnostico. (Buttner & Ziring, 2009).

```
cpe:/ <part>:<vendor>:<product>:<version>:<update>:<edition>:<language>
```

Figura 10 – Estrutura de um nome CPE

A Figura 10 mostra a estrutura do um nome CPE, de seguida descreve-se cada campo com mais detalhe e o seu significado:

- **Part –** Define o sistema de deteção e pode obter um dos seguintes valores:
  - a – Aplicação
  - h – Hardware
  - o – Sistema Operativo
- **Vendor –** Inclui o nome da organização/empresa que desenvolveu o produto. Por existir diversas formas de expressar o nome do vendedor o nome utilizado é o do domínio DNS.
- **Product** – Inclui o nome do produto que foi detetada a vulnerabilidade sendo utilizado o mais comummente aceite.
- **Version** – Indica versão do produto que é alvo da vulnerabilidade.
- **Update** – Pode ser confundida com a anterior, no entanto, pretende indicar a última atualização do produto (ex: Service Pack 2 – apareceria no nome CPE “SP2”).
- **Edition –** Mostra a edição de uma plataforma. Por exemplo uma edição para hardware pode ser x64 ou x86, enquanto que a nível aplicacional pode ser Linux ou Windows.
- **Language –** Mostra o idioma detetado.

O conhecimento desta informação é uma mais-valia, essencialmente para as empresas, visto permitir adotar políticas de gestão que visam garantir uma maior segurança dos seus ativos, possibilitando aos administradores de sistemas uma maior visibilidade e controle sobre os problemas e solução dos mesmos.

5.1.2 Common Vulnerability Scoring System

A Common Vulnerability Scorring System (CVSS) é um sistema de pontuação que permite classificar uma vulnerabilidade de uma forma padronizada. Permite estabelecer uma medida do grau de uma vulnerabilidade em comparação com outras de forma a ajudar a priorizar a resolução das mesmas por parte das empresas. As pontuações tem por base um conjunto de métricas que definem o grau de gravidade da vulnerabilidade. Alguns dos componentes representados na NVD são descritos abaixo (Mell, Scarfone, & Romanosky, 2007):

- **Access Vector –** Representa a forma como a vulnerabilidade pode ser explorada da perspetiva a rede. Assume três valores: Localmente, Rede e Rede Adjacente. Localmente caso o atacante tenha de ter acesso físico ao sistema vulnerável ou conta local. Caso a vulnerabilidade possa ser explorada remotamente assume o valor de Rede. Caso o atacante tenha de ter acesso ao domínio de broadcast (ex: bluetooth) esta métrica assume o valor Rede Adjacente.
- **Access Complexity –** Representa o grau de dificuldade em explorar uma vulnerabilidade dividida em três níveis: alta, média e baixa.
- **Authentication –** Indica se um atacante tem de se autenticar para tirar partido da vulnerabilidade. Os valores possíveis são: Múltiplas instâncias caso o atacante tenha de se autenticar duas ou mais vezes, instância única nos casos em que apenas carece de uma única autenticação e nenhuma caso não seja necessária autenticação para explorar a vulnerabilidade. Esta métrica não mede o esforço necessário para ganhar a autenticação.
- **Confidentiality Impact –** Esta métrica mede o impacto no acesso a informação sensível existente no sistema. Pode assumir três valores: Nenhuma, Parcial e Completo. Caso se verifique que a informação acedida na exploração de vulnerabilidade seja irrelevante é atribuída o valor mais baixo, nenhuma. Parcial se o acesso à informação que o atacante teve acesso não põem em risco o próprio sistema (Ex: acesso apenas a algumas tabelas de uma base de dados). Completa quando há divulgação integral proporcionando o acesso a qualquer dado do sistema.
- **Integrity Impact –** Esta métrica mede o impacto sobre a integridade de um sistema explorado com sucesso. Pode assumir três valores: Nenhuma, Parcial e Completo. Se não tiver qualquer impacto no sistema assume o valor de nenhum. Caso o

alcance do atacante possa ser limitado e sem controle do que possa ser alterado é atribuída a classificação de Parcial. Quando a integridade do sistema se encontra comprometida com a possibilidade de alteração do sistema é atribuída o nível mais alto, Completa.

- **Availability Impact –** Esta métrica mede o impacto à disponibilidade do sistema alvo de exploração, ataques do tipo denial of service estão enquadrados nesta métrica. Pode assumir três valores: Nenhuma, Parcial e Completo. Caso não tenha impacto na disponibilidade do sistema assume o valor de nenhuma, se existir uma quebra no desempenho do sistema é classificada como parcial e caso se verifique a perda total do sistema é atribuído o valor de completo.
- **Score -** Vulnerabilidades com uma pontuação entre 0-3.9 são consideradas de gravidade baixa, entre 4 e 6.9 de gravidade média e de 7 a 10 de gravidade alta. (National Institute of Standards and Technology, 2015).

Empresas como Microsoft, Cisco Systems, Symantec, Oracle, CERT têm adotado este sistema de pontuação, embora a sua interpretação possa variar.

5.1.3 Common Vulnerabilities and Exposures

Os identificadores CVE ou nomes CVE são identificadores únicos que foram criados pela Mitre Corporation e que são comuns em vulnerabilidades de segurança de informação. É um padrão de identificação de vulnerabilidades e que permite referenciar uma vulnerabilidade.

A identificação de vulnerabilidades apenas é aplicada a software que se encontre disponível ao público, desde versões de pré-lançamento até novas atualizações. Software feito à medida para uma empresa em particular não entra nesta classificação. Para além disso serviços também não entram na classificação (ex.: um provedor de email baseado na Web) a menos que a vulnerabilidade exista num *software* subjacente a esse produto e que se encontre disponível ao público.

5.1.4 Common Weakness Enumeration

É um projeto que visa a catalogação de falhas e vulnerabilidades de software e que tem como objetivo o entendimento destes defeitos no desenvolvimento de aplicações e na criação de

ferramentas automatizadas que podem ser usadas para identificar, corrigir e prevenir estas lacunas de segurança. Este projeto é também suportado pela Mitre Corporation que segundo estes o uso de ferramentas e serviços de segurança permitem uma melhor compreensão dos pontos fracos de software relacionados com a arquitetura e design (The MITRE Corporation, 2015). Para além a acessibilidade na identificação de ameaças permite, uma vez identificado, a possibilidade de mitigar essa ameaça.

5.2 Selecção dos Dados

A NVD disponibiliza uma grande quantidade de informação sobre cada vulnerabilidade para os fins do presente estudo estamos interessados apenas no nome, data da publicação, métricas do CVSS, descrição e software vulnerável (toda informação inerente a este elemento, vendedor, sofware, etc.). O modelo de dados encontra-se ilustrado na Figura 11 pelo que a seguir se descreve, de forma resumida, o conteúdo de cada uma das tabelas. Informação mais pormenorizada sobres as tabelas e seu conteúdo encontra-se no Anexo A desta Tese.

- **tbl_cwe** – Armazena os tipos de vulnerabilidades classificadas para cada uma das entradas. Cada tipo está formatado em CWE-NUMBER. Esta tabela não foi criada com base no ficheiro da NVD mas surgiu da necessidade de conseguir atribuir ao CWE o tipo de vulnerabilidade correspondente.
- **tbl_entries** – Tabela que armazena as entradas de novas vulnerabilidades. Contém a descrição de cada uma das vulnerabilidades assim como a data da sua publicação entre outros.
- **tbl_entry-cwe** – É uma tabela que associa o identificador da vulnerabilidade ao identificador CWE.
- **tbl_metrics** – Tabela com o resultado da aplicação das métricas nos mais diversos parâmetros analisados (grau de gravidade da vulnerabilidade, complexidade, autenticação, integridade da informação, entre outros).
- **tbl_part_description** – tabela que armazena o tipo de código e a descrição do sistema de deteção.
- **tbl_references** – tabela que armazena as referências e o tipo de referencias para cada uma das vulnerabilidades.

- **tbl_vulnerable-software** – esta tabela armazena os produtos que são afetados por cada vulnerabilidade. Para além dos produtos contem todas as versões do software afetado e as empresas que o desenvolveram.

O recurso ao desenvolvimento de uma base de dados para armazenamento do conteúdo do ficheiro XML apresenta, desde logo, algumas vantagens quando comparado com a análise direta do ficheiro XML. Tratando-se de um volume muito grande de informação torna-se crucial ter informação de forma estruturada e que permita acrescentar campos necessários ao enriquecimento da informação. Por outro lado surge a necessidade de uniformizar a informação contida no XML, por exemplo, os nomes dos vendedores nem sempre são representados da mesma forma (ex: Microsoft ou MS, Internet Explorer ou IE). O tratamento e análise da informação seria impossível de realizar se não fosse possível tratar e segmentar toda a informação contida em cada um dos ficheiros. Também para tratamento da informação foi necessário recorrer a software de produtividade para criação de tabelas e gráficos.

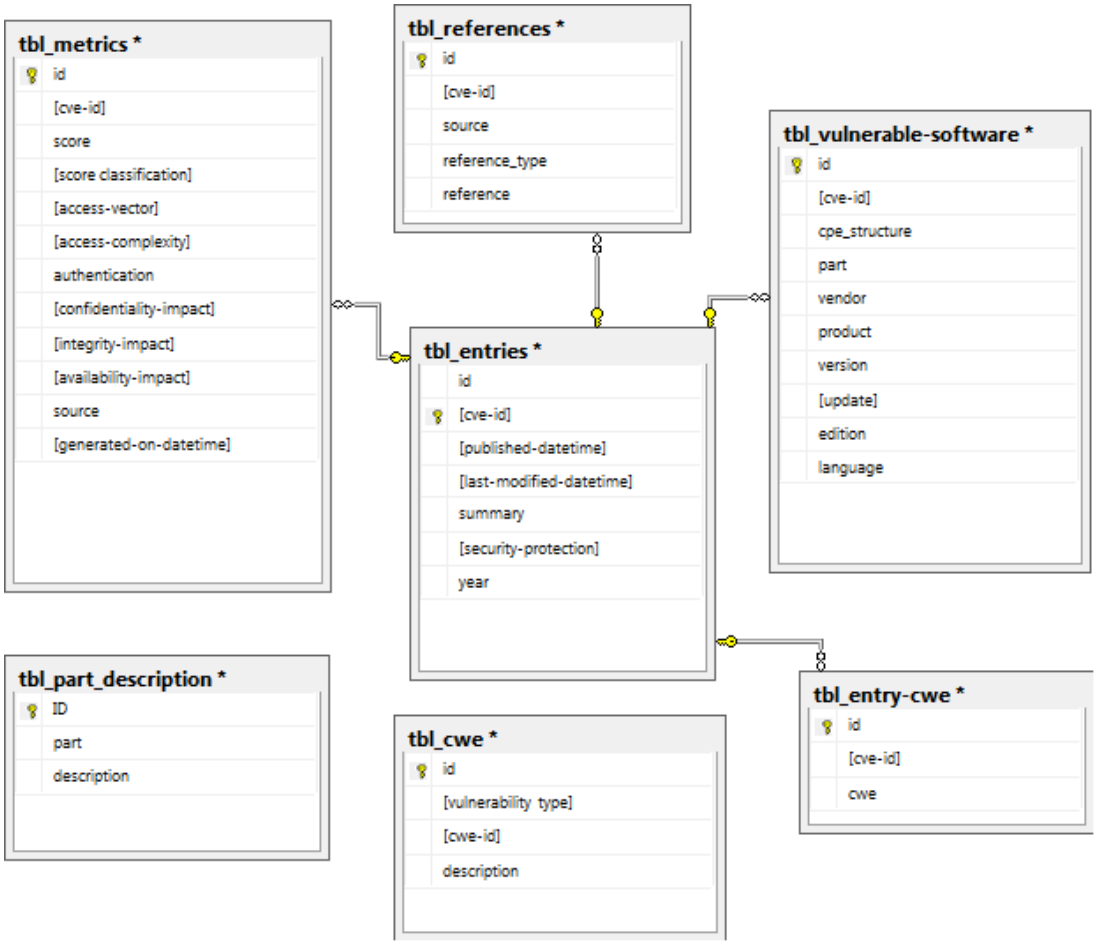

Figura 11 – Diagrama EER da base de dados utilizada para armazenar e analisar informação da NVD

5.3 Filtragem de Informação

Das mais de 30000 vulnerabilidades publicadas desde 2010 até 2015 foram selecionadas 2073 vulnerabilidades. Estas vulnerabilidades estão relacionadas com os protocolos SSL e TLS. Para a sua seleção foram analisadas a sua descrição e constatado que se trata de uma vulnerabilidade relacionada com estes protocolos. Existem vulnerabilidades que estão marcadas como "**DISPUTED**" que significa que os fornecedores dos produtos não estão de acordo quanto à existência dessas vulnerabilidades. Atendendo ao reduzido número, apenas 6, e à incerteza quanto à confirmação destas vulnerabilidades, as mesmas não foram consideradas para o presente estudo. Existem ainda 30 vulnerabilidades que se encontram marcadas como "Unspecified" cujo significado resulta de um desconhecimento por parte da NVD em saber a quando as mesmas ocorrem ou se efetivamente existem, no entanto, alimentam o ficheiro por

terem sido de alguma forma referenciadas como sendo uma vulnerabilidade. Estas vulnerabilidades também não foram objeto de estudo. A Tabela 1 mostra as empresas que possuem maior número de vulnerabilidades acumuladas nos seus produtos. No total temos que das 2037 vulnerabilidades, que reúnem condições de legibilidade, e representam 1362 empresas. Como nos estamos referir a vulnerabilidades de protocolos de comunicação é natural que a mesma vulnerabilidade possa ser comum a mais que um produto de vendedores distintos. Este pressuposto é confirmado pela tabela abaixo onde o número total de vulnerabilidades é superior ao objeto de estudo. Numa primeira análise verificamos que as primeiras 15 posições são ocupadas por empresas com um peso muito grande no mercado e que significa que milhares de clientes finais podem ser vítimas destas vulnerabilidades. De referir que 56.3% das empresas registam apenas 1 vulnerabilidade e 21.2% entre 10 e 2 vulnerabilidades as restantes 22.5% têm todas mais de 10 vulnerabilidades registadas sendo que o as empresas que se encontram nos 10 lugares do topo representam 17.9% das vulnerabilidades existentes.

Tabela 1 – Distribuição das Vulnerabilidades por Vendedor

Vendedor	**DISPUTED**	**Unspecified**	**Legíveis**	**Total**
Openssl	1		84	85
Cisco		10	57	67
IBM		1	56	57
Magzter			46	46
Apple			30	30
Redhat			24	24
Google			23	23
Mozilla	1		22	23
Pocketmags			22	22
Microsoft			20	20
Gnu			17	17
Apache			16	16
Canonical			15	15
Playscape			13	13
Nobexrc			12	12
Outras Empresas	4	27	1677	1708
% Total	**0.28%**	**1.74%**	**97.98%**	

5.4 Distribuição das Vulnerabilidades

Da análise às 2037 vulnerabilidades que foram examinadas, encontram-se agora distribuídas por ano da sua divulgação e por classes de deteção e cujo resultado se resume na Tabela 2 e na Tabela 3 respetivamente. A primeira tabela revela que até 2013 o aparecimento de vulnerabilidades deste tipo era em número reduzido, no entanto, em 2014 apareceram o maior número de vulnerabilidades. A Tabela 3 mostra que o foco destas vulnerabilidades está ao nível da aplicação. Ocupando a OpenSSL a primeira posição, não é de todo de estranhar pois o foco desta organização está no desenvolvimento de ferramentas de implementação dos protocolos SSL/TLS. A Cisco é a única empresa que tem uma distribuição mais equitativa pelos três sistemas. Empresas como a Apple, Google, Mozilla e Microsoft deixam antever que os seus navegadores devem apresentar um número bastante expressivo de vulnerabilidades deste tipo. Como o número de empresas em análise é bastante grande (1362) não faria sentido, nem acrescentaria valor ao documento, apresentar todas empresas na tabela, desta forma definiu-se o top 20 e as restantes foram agrupadas na penúltima linha da tabela como Outras Empresas. Aqui temos 1199 empresas com apenas uma vulnerabilidade e as restantes 143 apresentam entre 9 e 2 vulnerabilidades. Na última linha encontra-se o valor percentual para cada sistema de deteção.

Tabela 2 – Distribuição das Vulnerabilidades por Ano

Ano	Score Classification			Total	% Total
	High	Medium	Low		
2010	17	37	5	**59**	**2,90%**
2011	10	64	2	**76**	**3,73%**
2012	14	115	10	**139**	**6,82%**
2013	30	93	9	**132**	**6,48%**
2014	19	1521	9	**1549**	**76,04%**
2015	9	69	4	**82**	**4,03%**
Total	99	1899	39	**2037**	
% Total	**4,86%**	**93,23%**	**1,91%**		

Tabela 3 – Distribuição das Vulnerabilidades por Sistema de Deteção

Vendedor	Aplicação	Hardware	Sistema Operativo	Total
Openssl	84			84
Cisco	37	16	13	66

IBM	51	7	2	60
Magzter	46			46
Apple	15		21	36
Redhat	21		7	28
Google	21		2	23
Mozilla	22			22
Pocketmags	22			22
Microsoft	12		9	21
Gnu	17			17
Apache	16			16
Canonical			15	15
Playscape	13			13
Nobexrc	12			12
Paypal	12			12
Siemens	3	2	7	12
EMC	11			11
Juniper	5	1	5	11
Gcspublishing	10			10
Outras Empresas	1591	9	30	1630
% Total	**93,26%**	**1,62%**	**5,12%**	

A Tabela 4 apresenta distribuição das vulnerabilidades segundo sistema de pontuação CVSS. Numa primeira leitura verificamos que em todos os pontos observados o maior número de vulnerabilidades está centrado com o grau de gravidade média, sendo que de seguida se apresenta o grau mais elevado e no final o grau mais baixo, este quando comparado com os anteriores apresenta pouca expressão no número de vulnerabilidades registadas. Do ponto de vista do atacante este vai ter de alguma forma de aceder ao seu alvo e para isso, e segundo a perfectiva da rede, verificamos que as vulnerabilidades que tem uma pontuação mais elevada podem ser acedidas remotamente e apenas para um pequeno número (0.64%) o atacante tem de ter acesso físico ao sistema. Constatamos igualmente que é necessário algum nível o nível de conhecimento para explorar as vulnerabilidades, o que quer dizer que não será o utilizador comum a tirar partido desta vulnerabilidade. Isto significa que o atacante pode ter de recolher algumas informações antes de efetuar o ataque, podendo recorrer à engenharia social para enganar os utilizares mais incautos (ex.: ataques de phishing). Do ponto de vista da autenticação constatamos que de forma expressiva, em 98.58% dos casos, não é necessário o atacante autenticar-se. Este pode ser um fator motivador, pois para além de não ser necessário o esforço para obter as credenciais de acesso, consegue atacar sem a necessidade de se autenticar. Do ponto de vista da confidencialidade do próprio sistema constatamos que na sua maioria (86.16%) não põe em risco o próprio sistema, isto é o atacante não tem acesso total à

informação com a exploração da vulnerabilidade, no entanto, não significa que o acesso não autorizado a dados parciais não cause prejuízo à sua vítima. O mesmo se verifica do ponto de vista da integridade dos dados o atacante tem acesso à informação mas não a consegue alterar. Por último verifica-se que em 76.53% dos casos existe quebra no desempenho do sistema, embora apenas num pequeno numero (3.68%) a quebra seja total, não deixa de ser expressivo a redução de performance ou possíveis interrupções no funcionamento dos recursos.

Tabela 4 – Distribuição de Vulnerabilidades por CVSS

Metrics		Score Classification			Total	% Total
		High	Medium	Low		
Access Vector	ADJACENT_NETWORK	2	1396	5	1403	68,88%
	LOCAL	0	2	11	13	0,64%
	NETWORK	97	501	23	621	30,49%
Access Complexity	HIGH	6	28	18	52	2,55%
	MEDIUM	35	1727	13	1775	87,14%
	LOW	58	144	8	210	10,31%
Authentication	MULTIPLE_INSTANCES	0	1	0	1	0,05%
	SINGLE_INSTANCE	9	11	8	28	1,37%
	NONE	90	1887	31	2008	98,58%
Confidentiality Impact	COMPLETE	40	5	0	45	2,21%
	PARTIAL	29	1701	25	1755	86,16%
	NONE	30	193	14	237	11,63%
Integrity Impact	COMPLETE	40	2	0	42	2,06%
	PARTIAL	29	1706	9	1744	85,62%
	NONE	30	191	30	251	12,32%
Availability Impact	COMPLETE	66	9	0	75	3,68%
	PARTIAL	26	1528	5	1559	76,53%
	NONE	7	362	34	403	19,78%

Segundo a Netcraft 66% das páginas Web ativas utilizam OpenSSL para criptografar as suas comunicações (Netcraft, 2015). Segundo a Builtwith mais de 65 milhões de páginas ativas utilizam OpenSSL (BuiltWith, 2015) . Isto vai de encontro ao resultado obtido nas duas próximas duas tabelas (Tabela 5 e Tabela 6) onde a esta organização lidera as duas frentes. Verificamos igualmente que empresas como a Google, Mozilla e Apple se encontram nos lugares cimeiros, aqui as diferentes versões dos navegadores são os principais responsáveis pelo número elevado de vulnerabilidades. De salientar que a Microsoft, empresa concorrente às anteriores, apenas regista 125 vulnerabilidades no total, sendo que o Internet Explorer apenas registou 19. Apesar

de 2014 ter sido o ano onde se registaram mais entradas novas, este ano foi, para as versões dos navegadores anteriores, aquele em que se registou menos versões afetadas, no global apenas afetaram 12 versões dos navegadores. Em 2011 apenas 3 vulnerabilidades (CVE-2011-3106, CVE-2011-3389 e CVE-2011-0782) foram responsáveis pelo que a Google tivesse 1196 versões diferentes afetadas do seu navegador.

Tabela 5 – Empresas com Maior Número de Vulnerabilidades

Vendedor	Score Classification			Total
	High	Medium	Low	
Openssl	398	2456	118	2972
Mozilla	686	1837	67	2590
Google	253	1435	2	1690
Cisco	1376	245	8	1629
GNU	187	826	0	1013
IBM	82	758	85	925
Apple	193	556	36	785
Tor	0	458	0	458
Digium	0	416	0	416
Opera	0	293	112	405
Outras Empresas	725	5852	370	6947
%Total	**19,67%**	**76,31%**	**4,02%**	

Tabela 6 – Produtos com Maior Número de Vulnerabilidades

Produto	Score Classification			Total
	High	Medium	Low	
Openssl	401	2462	118	2981
Chrome	213	1420	2	1635
Firefox	268	681	16	965
Gnutls	187	766	0	953
Seamonkey	218	575	51	844
IOS	813	7	8	828
Thunderbird	196	449	0	645
Tor	0	477	0	477
Opera Browser	0	293	112	405
php	76	209	89	374
Outras Empresas	1528	7793	402	9723
%Total	**19,67%**	**76,31%**	**4,02%**	

5.5 Conclusão

O objetivo principal do estudo de vulnerabilidades deste tipo é perceber o impacto que as mesmas têm no quotidiano e que tipo de implicações é que podem causar. Para isso foi utilizada uma fonte de dados publica (NVD) e de seguida foram selecionadas e filtradas as vulnerabilidades objeto do estudo. Várias perspetivas foram mostradas sobre os dados em análise e algumas conclusões resumem-se a seguir:

- Para a grande maioria das vulnerabilidades serem exploradas carecem de conhecimentos técnicos e que não estão presentes no utilizador comum.
- Em mais de 75 % dos casos os impactos na sua exploração, embora possam causar danos, são parciais e não inviabilizam a continuidade do negócio.
- Os potenciais alvos das fragilidades das aplicações representam mais de 2/3 das aplicações web ativas.
- A mesma vulnerabilidade pode ser comum a mais do que um produto/empresa.
- Não foi possível quantificar o número de vezes que uma vulnerabilidade é explorada.
- Não existe uma relação de consequência entre o aparecimento de novas vulnerabilidades e o aumento do software afetado.

6 Vulnerabilidades em Domínios Portugueses

Todos nós utilizamos diariamente a internet nos seus mais diversos sentidos, como ferramenta de trabalho, lazer, na busca e partilha de informação, etc. A existência de plataformas que permitem o desenvolvimento de páginas Web minimamente profissionais e sem um nível de conhecimento muito elevado, facilitam que qualquer um possa promover o seu negócio e construir uma imagem online completa e credível. O objetivo deste capítulo é verificar se as páginas Web de domínios português (.pt ou .com.pt) apresentam vulnerabilidades do tipo SSL/TLS. Mais concretamente qual o resultado do teste à porta 443, porta utilizada pelo protocolo HTTPS e descrito mais pormenorizadamente no ponto 2.5, e que implementa numa camada adicional de segurança na comunicação. Para além disso pretende-se verificar se as mesmas são vulneráveis a possíveis ataques do tipo Poodle. Para isso recolheu-se, junto de fontes públicas, o maior número de domínios possíveis para a realização deste teste, mais concretamente 2290 páginas Web. Para a realização deste teste foram executados alguns scans de que visam recolher informação dos domínios para efeitos de estatística. Obviamente que o principal interesse está relacionado com a segurança que as mesmas apresentam para os utilizadores e não com qualquer tipo de tentativa de obter qualquer tipo de informação sensível de qualquer página. Desta forma não será individualizada nenhuma página mas sim efetuada a apresentação das conclusões no global. Sabemos que algumas foram construídas e suportadas por profissionais outras por utilizadores curiosos, no entanto, todas estão na rede e são acessíveis.

6.1 Selecção de Domínios

Os domínios foram selecionados através de fontes públicas mais concretamente do repositório da Internet-Wide Scan Data Repository. Este repositório é suportado pela Universidade de Michigan que hospeda dados não só a sua próprios informação como a de outros pesquisadores que periodicamente recolhem informação da rede. Da muita informação que este repositório disponibiliza apenas nos interessa informação de domínio português. De salientar que muita informação carece de software específico, algum pago, pelo que a escolha recaiu para os dados que podiam ser lidos. Para além deste repositório foi utilizada a informação disponibilizada pela Netcraft das 500 páginas mais visitadas com domínio português. Os duplicados foram eliminados.

6.2 Obtenção de Dados

Para obter os dados para analise, e após recolha e compilação do objeto de estudo, foi necessário recorrer a um programa para analisar informação. Por ser muito conhecido e claramente aceite na área de auditoria e segurança a escolha recaiu pela utilização do NMAP. Entre muitas vantagens é uma ferramenta que se encontra disponível de forma gratuita e que dispõem de uma licença open source. Para além de dispor de uma grande flexibilidade pode ser executado na grande maioria dos sistemas operativos incluindo uma interface que possibilita a execução de scan´s sem ser pela linha de comando, assim como a leitura facilitada de alguma informação.

Não pondo em causa a eficácia e a qualidade dos resultados no NMAP, mas para obter um grau de confiança que permitisse certificar os resultados obtidos foram utilizados dois sites que, de forma aleatória e penas para um universo reduzido de domínios, confirmaram que o resultado era o mesmo. Para a confirmação se o domínio era vulnerável a ataques do tipo Poodle foi utilizado a página da COMODO[4] e para confirmar a localização geográfica o site da HCI Data[5].

Também para tratamento e manuseio da informação foi necessário recorrer a software de produtividade para criação de tabelas e gráficos.

[4] https://sslanalyzer.comodoca.com
[5] http://www.hcidata.info/host2ip.htm

6.3 Resultados Obtidos

A primeira analise que é feita recai nos países as páginas estão alojadas. As 2290 páginas estão alojados em 29 países diferentes pelo que no gráfico representado na

Figura 12 apenas foram considerados aqueles que têm mais de 40 páginas alojados no seu país. Com Portugal a representar ao maior número de alojamentos, com cerca de 1123 páginas, não deixa ser expressivo o resultado dos Estados Unidos que detêm 461 páginas, correspondendo a 20% do universo em estudo, uma explicação para este número é o facto da sua representação ser composta, na sua maioria, por blogs (blogspot) cujo servidor de alojamento se encontra neste país. Os 3% dos Outros Países somam 70 páginas distribuídas por 20 países tais como República Checa, Ucrânia, China ou Canadá. Para dois domínios não foi possível identificar a sua localização geográfica, apenas sabemos que estão localizados na europa. Estes domínios estão incluídos nos Outros Países.

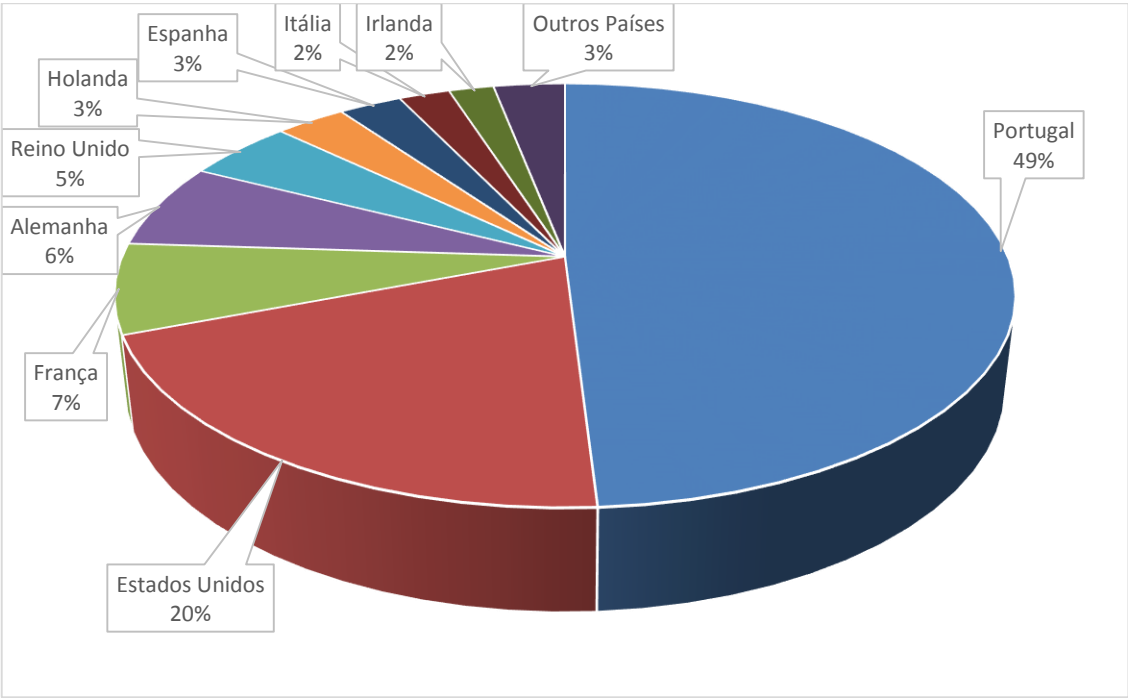

Figura 12 – Distribuição das páginas Web por países

Do teste à vulnerabilidade SSL verificamos que 37% são vulneráveis, verifica-se ainda que na sua grande maioria têm a porta 443 aberta, em 121 foi filtrada pela firewall e em 18 domínios encontra-se encerrada. Este ponto confirma o já indicado no decorrer do ponto 4.3 onde foi indicado que este tipo de vulnerabilidade não afeta apenas a porta mais popularizada em termos estabelecer um canal seguro de comunicação, mas também outras portas que utilizado SSL encontram-se igualmente vulneráveis. Dos 856 domínios vulneráveis 188 são blogs da blogspot das quais 158 tem a porta 443 aberta e as restantes o pedido foi filtrado. Não deixa de ser curioso que alguns domínios são de empresas da área de Tecnologias de Informação e outras de empresas cotadas no mercado como confiáveis.

Do lado das não vulneráveis verificamos que 0.2% bloqueou o pedido, 17.1% dos domínios porta 443 está fechada e que em 24.4% o pedido de conexão foi filtrado, no entanto, os restantes 58.3% tem a porta aberta e nem por isso são vulneráveis a ataques deste tipo.

Tabela 7 – Resultado dos Testes SSL

Resultado Teste Poodle (SSL)	**Resultado Teste Porta 443**					**Total**	**% Total**
	blocking ping	**closed https**	**filtered https**	**open https**	**Host Inacessível**		
Ligação recusada	69	2	6	1	0	**78**	**3,41%**
Host Inacessível	0	0	0	0	5	**5**	**0,22%**
Não Vulnerável	3	231	329	786	0	**1349**	**58,91%**
Não foi possível estabelecer uma conexão SSL	0	0	0	2	0	**2**	**0,09%**
Vulnerável	0	18	121	717	0	**856**	**37,38%**
Total	**72**	**251**	**456**	**1506**	**5**	**2290**	**100,00%**
%Total	**3,14%**	**10,96%**	**19,91%**	**65,76%**	**0,22%**	**100,00%**	

6.4 Conclusão

Em resumo concluímos que apesar de uma empresa, com uma imagem de marca confiável no mercado e com boa reputação, não é condição suficiente para que a comunicação e partilha de dados entre as duas partes seja efetuado de forma segura. Embora o objeto de estudo seja apenas o universo português naturalmente que as conclusões se estendem a todos os domínios. Também convém referir que a para muitas empresas, com pouca representação no mercado, não têm consciência para situações deste tipo, o mesmo já não é aceitável quando o suporte é dado por profissionais.

7 Conclusão

Neste capítulo é feita uma breve conclusão do projeto, fazendo um ponto de situação final, identificando se os objetivos definidos inicialmente foram atingidos ou não, melhoramentos futuros, conhecimento adquirido e dificuldades encontradas.

7.1 Ponto de situação final

A leitura e interpretação dos dados que foi feita no capítulo 5 poderia ter sido de muitas outras formas. Apesar da NVD ser uma fonte de dados publica ter inúmeras contribuições de fontes externas e ser periodicamente atualizada não é imune a criticas, no entanto, na opinião do autor, é a mais completa fonte de informação sobre o tema. Ozment também teceu algumas considerações sobre a informação que a NVD disponibiliza entre as quias o perdido cronológico em que as vulnerabilidades foram descobertas e quando o código vulnerável foi lançado. (Ozment, 2007). Por outro lado verificamos que ao longo do tempo os ficheiros disponibilizados pela NVD foram sendo alterados, o que dificulta a análise para períodos temporais maiores. Para além disso a falta de documentação clara levanta algumas dúvidas sobre a interpretação a dar aos campos do ficheiro. Uma outra limitação da NVD é que a mesma não refere se a vulnerabilidade pode ser explorada ou se já existe algum exploit para a vulnerabilidade. Isto representa uma dificuldade acrescida de avaliar o risco que a vulnerabilidade representa. Por conseguinte o autor considerou que todas as vulnerabilidades podem ser exploradas. Também não fica claro se todos os produtos são medidos pelo mesmo critério, isto é, muitas empresas utilizam o identificador CVE para identificar vulnerabilidades, no entanto, diferentes

interpretações podem gerar resultados diferentes. Por fim apesar de esta ser uma fonte de dados pública não sabemos se os fornecedores indicam todas as vulnerabilidades existentes no seu software ou apenas algumas o que pode sobrevalorizar a qualidade dos seus produtos.

7.2 Objetivos

Conforme indicado no 1.2 do presente documento existiam três objetivos na realização deste trabalho. O primeiro, mais lato, consistia em descrever alguns aspetos em envolvem as aplicações web de hoje em dia, alguns dos seus principais atores, passando por modelos de representação de conhecimento como é caso das taxonomias e que já foi feito nesta área relacionada com segurança. Pretendia-se igualmente dar o foco no conceito de vulnerabilidade, os seus impactos e apresentar algumas vulnerabilidades mais comuns nas aplicações Web. O autor considera que este objetivo foi atingido com sucesso. Para suportar esta opinião elencamos os capítulos 2, 3 e 4 que se focam nesta matéria com vários exemplos e com diferentes perspetivas (atacante e defesa).

O segundo objetivo era a utilização de dados históricos como base para analisar as vulnerabilidades existentes nos protocolos SSL/TLS no período de 2010 até 2015. O autor considera que este objetivo foi igualmente atingido com sucesso. De suporte a esta opinião temos o capítulo 5 onde foi feito o levantamento de um conjunto de vulnerabilidades, efetuada a sua análise, espelhados os resultados e elencadas as conclusões.

Um outro objetivo deste trabalho era um teste à segurança de alguns domínios portugueses a uma vulnerabilidade específica do protocolo SSL/TLS. Este objetivo também foi cumprido. O capítulo 6 espelha o resultado do sucesso do cumprimento deste objetivo.

Face ao exposto, e para o autor, o resultado final é positivo e foi alcançado com sucesso.

7.3 Conhecimentos adquiridos

Apesar do trabalho a realizar ter-se revelado mais complexo do que inicialmente se previa, a realização do mesmo, concretizado neste relatório, representou um verdadeiro desafio. Isto porque construir algo a partir do zero para analisar um tema que não sabia se existia matéria para o desenvolver foi algo arriscado.

Desta forma, para além dos objetivos inicialmente propostos, foi possível aprender:

- Formas de representação e classificação de vulnerabilidades.
- A importância da segurança no desenvolvimento de software.
- A importância do uso de técnicas adequadas para prevenir e mitigar vulnerabilidades.

Para o autor a realização deste Projeto revelou-se bastante produtiva no sentido que aprendeu bastante sobre vulnerabilidades e os seus impactos, no entanto com plena consciência de esta matéria está em constante mutação e que carece de acompanhamento rápido.

8 Bibliografia

Álvarez, G., & Petrovic, S. (2003). *A new taxonomy of web attacks suitable for efficient encoding.* Obtido de https://lists.oasis-open.org/archives/was/200308/pdf00000.pdf

Álvarez, G., & Petrovic, S. (2008). *Encoding a Taxonomy of Web Attacks with Different-Length Vectors.* Obtido de http://arxiv.org/pdf/cs/0210026.pdf

Aslam, T., & Krsul, I. (1996). *Use of A Taxonomy of Security Faults.* Obtido de http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=2304&context=cstech

Auger , R. (2011). *Cross Site Scripting*. Obtido de The Web Application Security Consortium: http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting

Bishop, M. (1995). A Taxonomy of UNIX System and Network Vulnerabilities. pp. 9-11. Obtido de http://cwe.mitre.org/documents/sources/ATaxonomyofUnixSystemandNetworkVulnerabilities%5BBishop95%5D.pdf

Bishop, M., & Bailey, D. (1996). A Critical Analysis of Vulnerability Taxonomies. Obtido de http://seclab.cs.ucdavis.edu/projects/vulnerabilities/scriv/ucd-ecs-96-11.pdf

BuiltWith. (2015). *OpenSSL Usage Statistics*. Obtido de BuiltWith: http://trends.builtwith.com/Server/OpenSSL

Buttner, A., & Ziring, N. (2009). *Common Platform Enumeration (CPE) – Specification.* The MITRE Corporation and National Security Agency. Obtido de http://cpe.mitre.org/files/cpe-specification_2.2

Cisco Systems. (2010). Vulnerabilities, Threats, and Attacks. 17-39. Obtido de http://www.lovemytool.com/files/vulnerabilities-threats-and-attacks-chapter-one-7.pdf

COMODO . (2015). *COMODO SSL Analyzer*. Obtido de COMODO : https://sslanalyzer.comodoca.com/

Correia, M. P., & Sousa, P. J. (2010). Em *Segurança no Software* (pp. 12-16 ;136-138; 166-169). FCA.

Cronkhite, C., & McCullough, J. (2001). Understanding Hackers and How They Attack. Em *Access Denied: The Complete Guide to Protecting Your Business Online* (pp. 1-16). The McGraw-Hill Companies Book. Obtido de http://books.mcgraw-hill.com/downloads/products/0072133686/0072133686_ch01.pdf

CWE/SANS. (2011). *2011 CWE/SANS Top 25 Most Dangerous Software Errors*. Obtido de CWE Common Weakness Enumeration: http://cwe.mitre.org/top25/index.html

Dierks, T., & Allen, C. (1999). *The TLS Protocol Version 1.0.* Janeiro. Obtido de http://www.ietf.org/rfc/rfc2246.txt

Dutra, J., & Busch, J. (2003). *Enabling knowledge discovery: Taxonomy Development for NASA.* Obtido de http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/7825/1/03-2121.pdf

Fielding, R., Irvine, U., Gettys, J., Mogul, J., Frystyk, H., & Berners-Lee, T. (Junho de 1999). *Hypertext Transfer Protocol -- HTTP/1.1*. Obtido de http://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf

Giacobbi, G. (Novembro de 2006). *The GNU Netcat project*. Obtido de The GNU Netcat: http://netcat.sourceforge.net/

Hardin, B. (2015). *Injection Flaws*. Obtido de CONSTANTLY LEARNING - Programming, Information Security, Startups, and Learnings.: http://bretthard.in/2009/07/injection-flaws/

Hardy, D. (Julho de 1996). *Resource Description Messages (RDM)*. Obtido de W3C: http://www.w3.org/TR/NOTE-rdm.html#taxonomy

Harvey, B. (1985). *What is a Hacker*. Obtido de University of California: http://www.cs.berkeley.edu/~bh/hacker.html

HCI Data Ltd. (2015). *Convert Host/Domain Name to IP Address and vice versa*. Obtido de HCIDATA: http://www.hcidata.info/host2ip.cgi

Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Navara, E. D., O'Connor, E., . . . Marquis, M. (2015). *HTML 5.1 Nightly*. Obtido de W3C: http://www.w3.org/html/wg/drafts/html/master/

Holman, P. (2014). *The Hacker Motivation*. Obtido de http://www.homepagedaily.com/Pages/article3553-the-hacker-motivation.aspx

Howard, J. D., & Longstaff, T. A. (1998). *A Common Language for Computer Security Incidents.* Obtido de http://prod.sandia.gov/techlib/access-control.cgi/1998/988667.pdf

InfosecWriters. (2015). *An Insight into Cookie Security.* Obtido de InfosecWriters: http://www.infosecwriters.com/text_resources/pdf/Cookie_Security_ADoraiswamy.pdf

Jang, D., Jhala, R., Lerner, S., & Shacham, H. (2010). *An Empirical Study of Privacy-Violating Information Flows in JavaScript Web Applications.* Obtido de University of California: http://cseweb.ucsd.edu/~lerner/papers/ccs10-jsc.pdf

Kang, A., Wiesmann, A., Russell, A., Klein, A., Hau, W., Smith, T., & Huseby, S. (2005). Data Validation. Em *A Guide to Building Secure Web Applications and Web Services* (pp. 161-165). Obtido de http://www.salientsecurity.com/resources/files/OWASP_Guide_2-0-1.pdf

Krill, P. (Outubro de 2010). *Hold off on deploying HTML5 in websites*. Obtido de infoworld.com: http://www.infoworld.com/article/2626533/html5/w3c--hold-off-on-deploying-html5-in-websites.html

Krsul, I. V. (1998). *Software Vulnerability Analysis.* Obtido de http://www.krsul.org/ivan/articles/main.pdf

Krsul, I., & Spafford, E. (1998). *A Classication of Software Vulnerabilities That Result From Incorrect Environmental Assumptions.* Purdue University. Obtido de http://cardscorporate.com/c40/segurid/segik13.pdf

Lai, J.-Y., Wu, J.-S., Chen, S.-J., Wu, C.-H., & Yang, C.-H. (2008). Designing a Taxonomy of Web Attacks. Obtido de http://security.nknu.edu.tw/psnl/publications/2008ICHIT_taxonomy.pdf.

Lough, D. L. (2001). A TAXONOMY OF COMPUTER ATTACKS WITH APPLICATIONS TO WIRELESS NETWORKS. pp. 59-60. Obtido de http://scholar.lib.vt.edu/theses/available/etd-04252001-234145/unrestricted/lough.dissertation.pdf

Mell, P., Scarfone, K., & Romanosky, S. (2007). *A Complete Guide to the Common Vulnerability Scoring System Version 2.0.* Obtido de https://www.first.org/cvss/cvss-v2-guide.pdf

Microsoft. (2013). *Motivations of a Criminal Hacker*. Obtido de Microsoft: https://technet.microsoft.com/en-us/library/cc505924.aspx

Microsoft. (2015). *Defense in Depth: HTML5 Sandbox*. Obtido de Microsoft: http://ie.microsoft.com/testdrive/HTML5/sandbox/Default.html

Microsoft. (2015). *Escolher um algoritmo de criptografia*. Obtido de Developer Network: https://msdn.microsoft.com/pt-br/library/ms345262(v=sql.120).aspx

Microsoft. (2015). *Microsoft Security Advisory 3009008*. Obtido de TechCenter de Segurança: https://technet.microsoft.com/library/security/3009008

MITRE. (2014). *Organizations with CVE Identifiers in Advisories*. Obtido de Common Vulnerabilities and Exposures: http://cve.mitre.org/compatible/alerts_announcements.html

Möller, B., Duong, T., & Kotowicz, K. (2014). *This POODLE Bites: Exploiting The SSL 3.0 Fallback.* Google. Obtido de https://www.openssl.org/~bodo/ssl-poodle.pdf

National Institute of Standards and Technology. (2012). Risk Models. Em *Information Security* (pp. 8-9). Obtido de http://csrc.nist.gov/publications/nistpubs/800-30-rev1/sp800_30_r1.pdf

National Institute of Standards and Technology. (2015). *NVD Common Vulnerability Scoring System Support v2*. Obtido de National Vulnerability Database: https://nvd.nist.gov/cvss.cfm

Netcraft. (2015). *April 2014 Web Server Survey*. Obtido de Netcraft: http://news.netcraft.com/archives/2014/04/02/april-2014-web-server-survey.html

NIST. (2013). *Glossary of Key Information Security Terms.* (R. Kissel, Ed.) Obtido de http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf

NMAP. (2015). *Nmap Security Scanner*. Obtido de NMAP: https://nmap.org/

OWASP. (Abril de 2009). *Phishing*. Obtido de OWASP: https://www.owasp.org/index.php/Phishing

OWASP. (2013). *OWASP*. Obtido de Top 10 2013-Top 10: https://www.owasp.org/index.php/Top_10_2013-Top_10

OWASP. (2013). *Top 10 2013-A6-Sensitive Data Exposure*. Obtido de OWASP: https://www.owasp.org/index.php/Top_10_2013-A6-Sensitive_Data_Exposure

OWASP. (Abril de 2014). *Cross-site Scripting (XSS)*. Obtido de OWASP: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

OWASP. (2015). Application Threat Modeling. Obtido de https://www.owasp.org/index.php/Application_Threat_Modeling

OWASP. (2015). *DOM based XSS Prevention Cheat Sheet*. Obtido de OWASP: https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet

OWASP. (2015). *XSS (Cross Site Scripting) Prevention Cheat Sheet*. Obtido de OWASP: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet#XSS_Prevention_Rules

Ozment, A. (2007). Vulnerability Discovery & Software Security. pp. 39-42. Obtido de http://andyozment.com/papers/ozment_dissertation.pdf

Scambray, J., Liu, V., & Sima, C. (2011). *Hacking Exposed Web Aplications 3.* McGrawHill.

Secunia. (2014). *Secunia Vulnerability Review.* Obtido de https://secunia.com/?action=fetch&filename=secunia_vulnerability_review_2014.pdf

SecurityFocus. (Março de 1998). *Internet Cookies*. Obtido de SecurityFocus: http://www.securityfocus.com/advisories/1083

Shreeraj , S. (2014). *Detecting Web Application Security Vulnerabilities*. Obtido de ONLamp: http://www.oreillynet.com/pub/a/sysadmin/2006/11/02/webapp_security_scans.html?page=1

Smith, S. (Junho de 2015). *Secure Node Apps Against OWASP Top 10 - Cross Site Scripting*. Obtido de http://scottksmith.com/: http://scottksmith.com/blog/2015/06/22/secure-node-apps-against-owasp-top-10-cross-site-scripting/

The MITRE Corporation. (2015). *Common Platform Enumeration: CPE Specifications*. Obtido de Common Platform Enumeration: https://cpe.mitre.org/

Thomas, S. A. (2000). SSL & TLS Essentials. Em *Securing the Web* (pp. 117-121). Wiley Computer Publishing. Obtido de http://imcs.dvfu.ru/lib.int/docs/Web/SSL%20&%20TLS%20Essentials.%20Securing%20the%20Web.pdf

Vikas, S. (s.d.). Client Server Architecture. pp. 7-8. Obtido de http://www.sharadavikas.com/CourceMeterials/bca34.PDF

Vital, L. P., & Café, M. L. (2009). Ontologias e taxonomias: diferenças. Obtido de http://www.scielo.br/scielo.php?pid=S1413-99362011000200008&script=sci_arttext

W3C. (Agosto de 2002). *The Extensible HyperText Markup Language (Second Edition)*. Obtido de W3C: http://www.w3.org/TR/xhtml1/#xhtml

W3C. (Novembro de 2010). *HTML5 Web Messaging*. Obtido de W3C: http://www.w3.org/TR/2010/WD-webmessaging-20101118/

W3SCHOOLS. (2015). *HTML5 Local Storage*. Obtido de w3schools: http://www.w3schools.com/html/html5_webstorage.asp

Websense Security Labs. (2015). *SSLv3 "POODLE" Vulnerability CVE-2014-3566*. (I. Websense, Editor) Obtido de Websense Security Labs Blog: http://community.websense.com/blogs/securitylabs/archive/2014/10/15/sslv3-quot-poodle-quot-vulnerability-cve-2014-3566.aspx

Weiser, M. (1991). The Computer for the 21st Century. Obtido de https://www.ics.uci.edu/~corps/phaseii/Weiser-Computer21stCentury-SciAm.pdf

Yue, C., & Wang, H. (2009). *Characterizing Insecure JavaScript Practices on the Web.* Obtido de http://www2009.eprints.org/97/1/p961.pdf

ZMap. (2015). *Internet-Wide Scan Data Repository*. Obtido de ZMap · The Internet Scanner: https://scans.io/

ZMap. (2015). *Scanning Best Practices*. Obtido de ZMap · The Internet Scanner: https://zmap.io/documentation.html#bestpractices

Anexo A

Tabela tbl_cwe

Esta tabela armazena o tipo de vulnerabilidade possível de encontrar no ficheiro da NVD, desta forma, esta tabela não resulta da importação do conteúdo do ficheiro XML da NVD mas sim da necessidade de conseguir identificar, através do CWE-ID, o tipo de vulnerabilidade.

Nome Campo	**Tipo**	**Exemplo**
id	AutoNumber	10
vulnerability type	Short Text	Cryptographic Issues
cwe-id	Short Text	CWE-310
description	Long Text	Weaknesses in this category are related to the use of cryptography.

Tabela tbl_entries

Esta tabela armazena informação resumida das vulnerabilidades do período em análise. Os campos resultantes desta tabela derivam, na sua grande maioria, do conteúdo do ficheiro da NVD, por opção nomes dos campos são os mesmos dos ficheiros XML. Nem todos os campos têm conteúdo conforme se verifica no exemplo abaixo.

Nome Campo	Tipo	Exemplo
id	AutoNumber	5745
cve-id	Short Text	CVE-2014-3566
published-datetime	Date/Time	14-10-2014
last-modified-datetime	Date/Time	16-07-2015
summary	Long Text	The SSL protocol 3.0, as used in OpenSSL through 1.0.1i and other products, uses nondeterministic CBC padding, which makes it easier for man-in-the-middle attackers to obtain cleartext data via a padding-oracle attack, aka the "POODLE" issue.
security-protection	Short Text	
year	Number	2014

Tabela tbl_entry-cwe

Tabela que armazena os CVE-ID´s e o correspondente código do tipo de vulnerabilidade. Uma vulnerabilidade pode ter de mais do que um tipo.

Nome Campo	Tipo	Exemplo
id	AutoNumber	5752
cve-id	Short Text	CVE-2014-3566
cwe	Short Text	CWE-310

Tabela tbl_metrics

Tabela que armazena o resultado da aplicação das métricas nos mais diversos parâmetros analisados (grau de gravidade da vulnerabilidade, complexidade, autenticação, integridade da informação, entre outros). O conteúdo dos campos resultam do conteúdo do ficheiro XML da NVD.

Nome Campo	Tipo	Exemplo
id	AutoNumber	5467
cve-id	Short Text	CVE-2014-3566
score	Number	4.3
score classification	Short Text	Medium
access-vector	Short Text	NETWORK
access-complexity	Short Text	MEDIUM
authentication	Short Text	NONE
confidentiality-impact	Short Text	PARTIAL
integrity-impact	Short Text	NONE
availability-impact	Short Text	NONE
source	Short Text	http://nvd.nist.gov
generated-on-datetime	Date/Time	15-10-2014

Tabela tbl_part_description

Tabela que armazena o tipo de código e a descrição do sistema de deteção. O ficheiro da NVD apenas tem o código (part) do sistema de deteção, optou-se por criar uma tabela com a descrição do respetivo código.

Nome Campo	Tipo	Exemplo
ID	AutoNumber	1
part	Short Text	a
description	Short Text	Aplication

Tabela tbl_references

Tabela que armazena as fontes e o tipo de referências para cada uma das vulnerabilidades.

Nome Campo	Tipo	Exemplo
id	AutoNumber	16307
cve-id	Short Text	CVE-2014-3566
source	Short Text	MISC
reference_type	Short Text	VENDOR_ADVISORY
reference	Long Text	https://www.openssl.org/~bodo/ssl-poodle.pdf

Tabela tbl_vulnerable-software

Esta tabela armazena os produtos que são afetados por cada vulnerabilidade. Para além dos produtos contém todas as versões do *software* afetado e as empresas que o desenvolveram. Nem todas as vulnerabilidades têm os campos update, *edition* ou *language* com valores.

Nome Campo	Tipo	Exemplo
id	AutoNumber	109252
cve-id	Short Text	CVE-2014-3566
cpe_structure	Short Text	cpe:/a:openssl:openssl:1.0.0:beta2
part	Short Text	a
vendor	Short Text	openssl
product	Short Text	openssl
version	Short Text	1.0.0
update	Short Text	beta2
edition	Short Text	
language	Short Text	