INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Estudo de Segurança dos Protocolos SSL/TLS

FILIPE JOSÉ TEIXEIRA DIAS
Outubro de 2016

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Estudo de Segurança dos Protocolos SSL/TLS

Filipe José Teixeira Dias

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Nuno Alexandre Magalhães Pereira

Co-orientador: Jorge Manuel Canelhas Pinto Leite

Júri:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

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

Porto, Outubro 2016

Aos meus pais, família e amigos

Filipe Dias

Resumo

Esta dissertação surge da constante evolução por parte das tecnologias e serviços e a dependência que temos destes sistemas. Com esta evolução, aparecem também novas oportunidades para os atacantes. Inúmeras vezes durante o dia acede-se ao correio eletrónico, navega-se pela Internet e efetuam-se operações sensíveis como transações bancárias com os protocolos SSL/TLS sempre presentes, confiando – pelo menos os utilizadores mais atentos – na sua eficácia na proteção que oferecem.

Mas afinal o que são estes protocolos e que segurança oferecem? Nos domínios web que os portugueses frequentam assiduamente, a inclusão destes protocolos é suficiente e confiável ou existe exposição às vulnerabilidades?

Para responder a estas perguntas, efetuou-se um estudo sobre os protocolos SSL/TLS, *cipher suite* associada, certificados digitais e vulnerabilidades relacionadas, tendo como objetivo ilustrar boas práticas ou medidas de prevenção na aplicação destes protocolos.

Examinou-se soluções que ajudassem a identificar os problemas já mencionados com o intuito de recolher informações para executar uma análise e apresentar os resultados obtidos.

A solução preconizada irá auxiliar fundamentalmente os Administradores de Sistemas para elucidar eventuais ameaças com as configurações atuais e pontos a melhorar.

Propôs-se, portanto, um método de seleção, recolha e análise para avaliar a exposição dos portugueses aos domínios web que frequentam assiduamente e chegou-se à conclusão que não é possível garantir a partilha de informação de forma segura entre as partes envolvidas.

É importante reforçar e alertar não só os utilizadores portugueses como a população em geral para tomarem consciência e mais cuidado quando navegam na Internet, nunca desprezando as boas práticas tanto para administradores de sistemas (ou quem configura e disponibiliza os serviços) como para os utilizadores.

Palavras-chave: TLS, SSL, HTTPS, Vulnerabilidades, Segurança, Internet

Abstract

This thesis comes from the constant evolution of technologies and services, as well as from our reliance on those systems. With this evolution, new opportunities are created for attackers as well. We check our e-mail, browse the internet or perform sensitive operations online, like bank payments, several times throughout the day, with SSL/TLS protocols always there, while trusting - at least the savvier users - in their efficiency and protection.

But what are these protocols and what safety do they guarantee? On the web domains the Portuguese usually browse, is the use of these protocols sufficient and trustable or are they being exposed to vulnerabilities?

To answer these questions, the author performed a study on the SSL/TLS protocols, cipher suite, algorithms, digital certificates and related vulnerabilities, having, as the main objective, to highlight good practices or preventive measures in these protocols' application.

The proposed solution will mainly assist System Administrators on what are the main threats with current configurations and where to improve these.

It was proposed, as such, a method of collecting, processing and analysis to evaluate the exposure of Portuguese users to the web domains more frequently browsed which permitted to conclude that it is not currently possible to guarantee a safe flow of information between all parties involved.

It is very important to improve and to alert not only the Portuguese users but also other users in general to have a more cautious approach when browsing the internet, never dismissing on best practices, either for system administrators (or whoever is responsible for configuring and providing those services) or for end users.

Keywords: TLS, SSL, HTTPS, Vulnerabilities, Security, Internet

Agradecimentos

Em primeiro lugar gostaria de agradecer aos meus pais, por todo o apoio e carinho que me deram ao longo do meu percurso académico e também pelo enorme esforço que fizeram para que eu pudesse concluir o mesmo.

Ao resto da minha família e a todos os amigos pelo incentivo e críticas construtivas ao longo do Mestrado, que me tornaram mais sapiente.

Aos meus orientadores, nomeadamente os Engenheiros Nuno Pereira e Jorge Pinto Leite, pela proposta, confiança e apoio depositado em mim em abraçar esta dissertação.

À BLIP, pelo espaço de trabalho.

A Robert Dell'Immagine, Diretor da Comunidade da Qualys SSL Labs e Ivan Ristic, Diretor do Departamento de Engenharia e Segurança da Qualys e autor da Qualys SSL Labs API, pela permissão dada para usufruir da ferramenta em questão para a recolha de dados cruciais para esta dissertação.

A todos os que não mencionei por lapso e aos acima descritos, o meu sincero OBRIGADO!

Filipe Dias

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

AES	*Advanced Encryption Standard*
API	*Application Programming Interface*
BEAST	*Browser Exploit Against SSL/TLS*
BREACH	*Browser Reconnaissance and Exfilltration via Adaptive Compression of Hypertext*
CBC	*Cipher-Block Chaining*
CRIME	*Compression Ratio Info-leak Made Easy*
CRL	*Certificate Revocation Lists*
CSP	*Content-Securicy-Policy*
DH	*Diffie–Hellman key exchange*
DROWN	*Decrypting RSA Obsolete and Weakened eNcryption*
DSA	*Digital Signature Algorithm*
ECDH	*Elliptic curve Diffie–Hellman*
ECDSA	*Elliptic Curve Digital Signature Algorithm*
FREAK	*Factoring RSA Export Keys*
HMAC	*Hash-based Message Authentication Code*
HPKP	*HTTP Public Key Pinning*
HSTS	*HTTP Strict Transport Security*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IETF	*Internet Engineering Task Force*
JSON	*JavaScript Object Notation*
MAC	*Message Authentication Code*

MD5	*Message-Digest algorithm 5*
MITM	*Man-in-the-middle*
MVC	*Model-View-Controller*
OCSP	*Online Certificate Status Protocol*
OSI	*Open Systems Interconnection*
PCBC	*Propagating Cipher Block Chaining*
PKI	*Public Key Infrastructure*
POODLE	*Padding Oracle On Downgrade Legacy Encryption*
PRF	*Pseudorandom Function*
RC4	*Rivest Cipher 4*
RSA	*Rivest-Shamir-Adleman*
SHA-1	*Secure Hash Algorithm 1*
SOAP	*Simple Object Access Protocol*
SSH	*Secure Shell*
SSL	*Secure Sockets Layer*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
UDDI	*Universal Description, Discovery and Integration*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
VPN	*Virtual Private Network*
WSDL	*Web Service Description Language*
XML	*eXtensible Markup Language*
XSS	*Cross-Site Scripting*

1 Introdução

O presente capítulo tem como objetivo expor o problema e contextualizá-lo no âmbito desta dissertação. Irá ser apresentada uma abordagem preconizada da solução e uma breve análise de valor. A última secção contém a estrutura do documento.

A Internet nos dias de hoje é utilizada para os mais variados serviços e atividades. Há uma constante evolução por parte das tecnologias e serviços disponibilizados e estamos cada vez mais dependentes destes sistemas. Contudo, com o aparecimento de novas tecnologias, aparecem também novas oportunidades para os atacantes.

Áreas sensíveis como serviços bancários e saúde, comércio e instituições governamentais são apenas alguns exemplos que assumem uma importância na confiabilidade dos seus sistemas e que podem por em causa a continuidade do negócio.

Num ritmo mais moroso, os utilizadores vão tomando consciência para as questões de segurança e boas práticas na sua utilização, mas isto nem sempre se verifica, principalmente a quem disponibiliza e configura esses serviços. Surgem assim vulnerabilidades!

Esta dissertação incide e descreve principalmente os protocolos SSL/TLS bem como os algoritmos ou criptografia associada. Pretende-se igualmente dar foco às vulnerabilidades e ataques, os seus impactos e ilustrar as boas práticas ou medidas de prevenção na aplicação destes protocolos.

Serão investigados os domínios web que os portugueses frequentam assiduamente e serão apresentadas conclusões sobre os resultados obtidos.

1.1 Contexto

Os protocolos *SSL/TLS* são muito importantes no nosso dia a dia. Sem dar por isso, acedemos várias vezes ao correio eletrónico durante o dia, navegamos pelas redes sociais, consultamos

informação por toda a Internet e ainda fazemos pagamentos bancários com estes protocolos sempre presentes no fluxo da informação. Mas o que representam na realidade?

Estes protocolos são muito importantes na proteção de informação sensível transacionada entre o cliente e o servidor quando a confidencialidade e integridade dos dados é um requisito fundamental. O *SSL/TLS* permitem comunicações encriptadas, fornecem autenticação e evitam ataques do tipo *man-in-the-middle – MITM*[1].

Sendo assim é possível continuar a navegar na Internet sem preocupações?

Desde as primeiras versões do protocolo *SSL*[2] que têm vindo a ser identificadas várias vulnerabilidades[3].

Em alguns casos, tirando partido destas vulnerabilidades, é possível aos atacantes roubar informações sensíveis tais como palavras-chave, acessos bancários e acesso ao correio eletrónico, por exemplo, sem que a pessoa afetada tenha conhecimento.

Para minimizar e prevenir o sucesso dos atacantes existem soluções para evitar ou reduzir a exposição destas vulnerabilidades, que passam essencialmente por uma configuração de segurança informada e uma utilização cuidada dos serviços.

Tendo uma importância relevante, as configurações destes protocolos devem receber mais atenção e, de uma forma geral, expor mais a dimensão e frequência de problemas relacionados com os mesmos.

1.2 Problema

Tem vindo a surgir ao longo dos últimos anos, uma série de vulnerabilidades de segurança informática associada aos protocolos *SSL/TLS*.

Embora existam boas práticas recomendadas para a configuração segura destes protocolos e soluções conhecidas para diagnosticar configurações, por vezes não estão implementadas nos serviços oferecidos aos utilizadores.

Há pouca divulgação nos *media* sobre o assunto e a falta de visibilidade leva a um desleixo/ignorância por parte de quem disponibiliza os serviços ao cliente.

Quão exposto estão os domínios *web* portugueses relativamente às vulnerabilidades ou ataques através dos protocolos referidos?

[1] Para mais informação consultar a secção 3.3
[2] Para mais informação consultar a subsecção 2.3.1
[3] Para mais informação consultar o capítulo 3

É imprescindível expor este tema, para que se possua mais conhecimento sobre o problema e vulnerabilidades existentes. Tomar medidas de prevenção contra eventuais ataques, de modo a manter a confiança nos serviços e, por último, ter consciência de boas práticas a implementar, sendo pró-ativo.

1.3 Análise de Valor

A presente dissertação pretende fornecer um método, um processo de recolha, tratamento e análise sobre o estado da arte dos domínios web que os portugueses frequentam assiduamente. Esta solução irá auxiliar fundamentalmente os Administradores de Sistemas para elucidar eventuais ameaças com as configurações atuais e pontos a melhorar. Tem a mais valia da capacidade de agregar outras ferramentas e validar os dados recolhidos.

Pode-se consultar com mais rigor toda a informação acerca da análise de valor desta dissertação no APÊNDICE A.

1.4 Objetivos

Esta dissertação tem como objetivo visar o estado da arte dos serviços oferecidos através dos protocolos SSL/TLS e responder à pergunta: quão exposto estão os domínios web que os portugueses utilizam assiduamente relativamente às vulnerabilidades ou ataques através destes protocolos?

Com base nesta pergunta, esta dissertação irá:

- Propor e desenvolver métodos para coletar as informações necessárias, como:
    - Suporte a *HTTPS;*
    - Detalhes sobre certificados, nomeadamente a sua validade, algoritmo criptográfico e chave associada, autoridade de certificação, etc.;
    - Protocolos suportados, designadamente a sua versão;
    - *Cipher suite*;
    - Cabeçalhos *HTTP*.
- Analisar os dados recolhidos, considerando as características de segurança avaliadas.

Esta análise poderá ter várias iterações, onde podem ser recolhidos mais dados de acordo com os resultados obtidos ou analisados.

Como solução do problema a resolver, foi projetada a coleta de dados adequada e a definição do âmbito do mesmo, que posteriormente será a base para a conclusão sobre a segurança desses serviços.

Para isso vai ser recolhida, organizada e avaliada informações sobre os protocolos *SSL/TLS* normalmente utilizados bem como os seus problemas, boas práticas para serviços mais seguros e avaliadas/comparadas ferramentas e métodos para coletar informações sobre os protocolos *SSL/TLS*.

Reunidas essas condições, as ferramentas a serem implementadas irão recolher informações sobre os protocolos já referidos, tais como: protocolos suportados, inspeção dos certificados, fases de troca de chaves e nível/força dos algoritmos de segurança. O estudo incluirá também a descoberta, seleção e classificação por tema/área de negócio dos domínios *web* portugueses.

1.5 Restrições Existentes

A principal restrição imposta desde o início da dissertação é recolher dados públicos dos domínios web em questão e não ser intrusivo sobre os mesmos. Desta forma e por questões legais, não é possível explorar as eventuais vulnerabilidades associadas.

Também por questões jurídicas e legais, não é possível discriminar nem tornar público, qualquer vulnerabilidade encontrada.

1.6 Estrutura do Documento

De forma a contextualizar melhor o leitor, a estrutura do presente documento está organizada por capítulos com as respetivas secções e subsecções.

Cada um dos pontos abaixo representa um capítulo com uma breve descrição sumária.

- No Capítulo 1– *Introdução*, são apresentados o contexto e o problema (resumidos) que esta dissertação irá abordar, bem como uma breve análise de valor e os objetivos idealizados e respetivas restrições existentes.

- No Capítulo 2 – *Estado da Arte*, descreve-se, mais pormenorizadamente, o contexto e problema e o estado da arte em tecnologia relevante (protocolos, algoritmos criptográficos, certificados digitais e cabeçalhos HTTP). São descritas também boas práticas de configuração nos serviços prestados através dos protocolos SSL/TLS.

- No Capítulo 3 – *Vulnerabilidades / Ataques*, apresentam-se de forma bastante sucinta algumas das vulnerabilidades que o autor considera importantes para o desenrolar da dissertação.

- No Capítulo 4 – *Ferramentas de Análise Criptográfica*, enumeram-se soluções existentes, dá-se destaque às vantagens e desvantagens das soluções e apresentam-se métodos de validação das ferramentas para a informação recolhida.

- No Capítulo 5 – *Design da Solução*, reflete-se a arquitetura da solução concebida, os casos de uso e explicações sobre as decisões de *design* de software tomadas. Aborda-se também as tecnologias usadas e é exposto o levantamento de requisitos funcionais e não funcionais.

- No Capítulo 6 – *Avaliação da Solução*, pretende-se explicar o método de seleção, recolha e análise dos dados. É realizada uma conclusão sobre os resultados obtidos e é apresentada a exposição dos domínios web relativamente às vulnerabilidades descritas no Capítulo 3.

- No Capítulo 7 – *Conclusões*, apresenta-se o ponto de situação final, objetivos alcançados bem como as limitações e trabalho futuro.

2 Estado da Arte

É importante perceber como surgem as vulnerabilidades, mas principalmente saber onde é que estas existem.

Para compreender a informação a analisar é imprescindível ter conhecimento sobre os protocolos a analisar, a forma como se estabelece uma ligação entre estes ou mesmo os algoritmos criptográficos associados.

Este capítulo visa descrever, mais pormenorizadamente, o estado da arte à volta dos protocolos *SSL/TLS*.

Será abordado o processo de estabelecer uma comunicação "segura" e tudo o que envolve esse procedimento, nomeadamente a troca de chaves, o papel dos certificados digitais, os cabeçalhos *HTTP* nos pedidos e respostas e a *cipher suite* associada aos protocolos.

A próxima secção aborda mais minuciosamente o contexto e problema.

2.1 Detalhes sobre Contexto e Problema

No mundo atual em que vivemos é praticamente indispensável o acesso à Internet. A Figura 1 mostra o número de assinantes do acesso à Internet em Portugal, conhecidos até à data.

Vivemos numa sociedade da informação (Feather, 2013) onde o Homem assiste a um desenvolvimento tecnológico que lhe permite melhorar os seus atuais padrões de vida. O comércio e a banca *online*, as redes sociais, o correio eletrónico e muitos outros serviços que a Internet disponibiliza, estão acessíveis para todos à distância de um clique.

Dadas todas estas evoluções, sem precedentes e a um ritmo vertiginoso, estão a ser tomadas medidas de segurança adequadas? Será que os serviços que utilizamos diariamente estão protegidos contra as ameaças à segurança na Internet? Sabe-se que existem ameaças à

segurança da Internet que foram responsáveis por inúmeros roubos de identidade e fraudes financeiras, que causaram prejuízos incalculáveis aos lesados (Feather, 2013). Esses roubos são hoje em dia um negócio lucrativo e controlado pelo crime organizado.

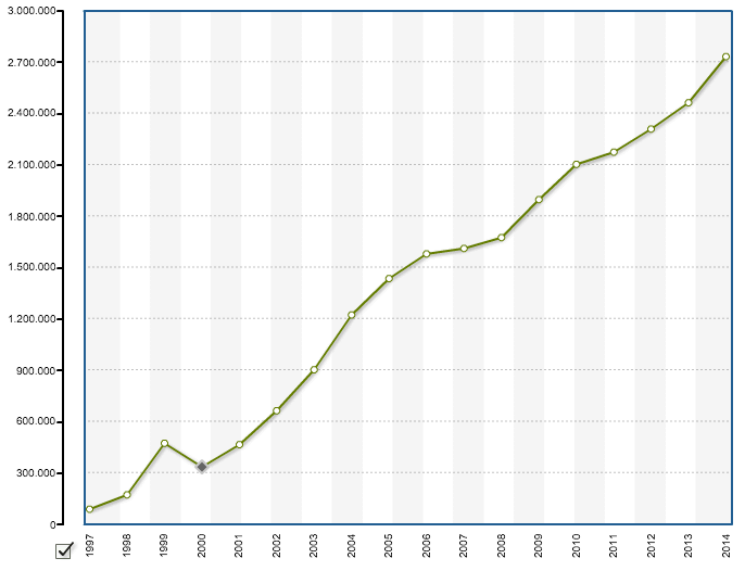


Figura 1 - Assinantes do acesso à Internet

(PORDATA, 2016)

Para ajudar os utilizadores da Internet a estarem protegidos, foi criada uma camada de segurança adicional (*SSL/TLS*) que opera através do *HTTPS*, que encripta os dados trocados entre o cliente e um servidor *web*. Podemos então estar tranquilos enquanto navegamos na Internet e utilizamos este protocolo de comunicação? Como iremos ver ao longo do documento, a resposta é: não. Cada vez existem mais ataques considerados graves a estes protocolos de comunicação, nomeadamente à configuração dos mesmos bem como os algoritmos utilizados (Sheffer et al., 2015).

A secção seguinte explica a origem do *HTTP* e expõe o *HTTPS* nas comunicações seguras.

2.2 HTTP e HTTPS

No início dos anos 90, Tim Bernard-Lee criou o primeiro cliente e servidor *web*, estabelecendo depois um protocolo para transferir informação na *web*, que deu origem ao *HTTP* (Berners-Lee and Fischetti, 1999).

A W3C[4] define o *HTTP* como um protocolo genérico ao nível da camada de aplicação, segundo o modelo OSI (International Organization for Standardization, 1994), para sistemas de informação distribuídos, colaborativos e hipermédia.

O conteúdo da troca de informação entre o cliente e servidor ocorre através do cabeçalho da mensagem *HTTP* (secção 2.6), onde se podem definir parâmetros para o seu funcionamento.

A Figura 2 apresenta os cabeçalhos de uma resposta *HTTP* para uma determinada página *web*.

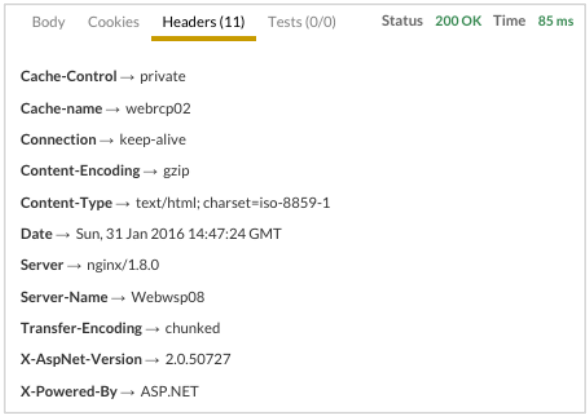


Figura 2 - Cabeçalho HTTP

Para evitar que a informação transmitida entre o cliente e o servidor seja intercetada por terceiros, foi implementado o protocolo *SSL/TLS* sobre o protocolo *HTTP*, servindo assim uma camada adicional de segurança por forma a tornar as comunicações mais seguras. Esta camada adicional permite que os dados sejam transmitidos por meio de uma ligação criptografada, onde se verifica a autenticidade do servidor e do cliente através de certificados digitais.

O *HTTPS* é, portanto, a combinação do protocolo *HTTP* com o *SSL/TLS.* É o processo mais comum atualmente de trocar informação sensível e de garantir, para o utilizador, uma imagem de segurança nas transações *online* (Symantec, 2016).

A confidencialidade e a integridade dos dados são um requisito fundamental aquando o envio de informação sensível e ao longo dos tempos, os protocolos *SSL/TLS* sofreram atualizações.

A próxima secção, descreve estes protocolos e mostra essencialmente, entre outra informação, como funcionam.

[4] https://www.w3.org/

2.3 SSL/TLS

Na secção anterior mencionou-se os protocolos *SSL* e *TLS*, mas ainda não foram referidas as diferenças entre eles bem como o seu funcionamento. Esta secção pretende desmistificar, de forma simples e clara, as eventuais dúvidas que possam ter surgido.

A subsecção adjacente elucida sobre a evolução do protocolo ao longo dos tempos.

2.3.1 Versões

O *TLS* sofreu várias alterações ao longo do tempo, como se pode verificar através da Figura 3.

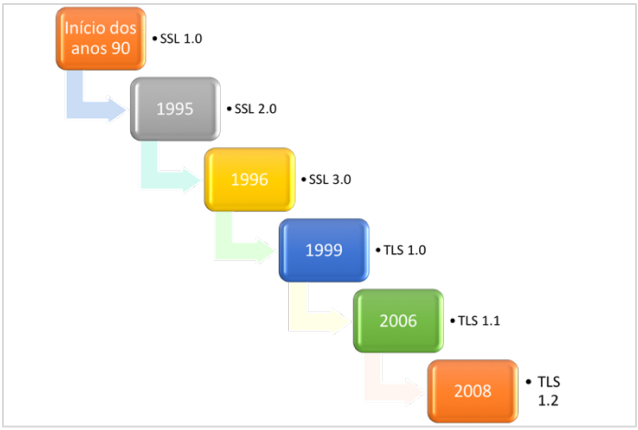


Figura 3 - Versões do SSL/TLS

A primeira versão do protocolo, *SSL* 1.0, foi criada no início dos anos 90, não havendo uma data específica de lançamento visto que o protocolo nunca foi público devido às evidentes falhas de segurança.

O *SSL* 2.0, que surgiu em 1995, foi descontinuado em 2011 (Turner and Polk, 2011) também por graves falhas de segurança (Claessens et al., 2002) nomeadamente:

- Chaves criptográficas semelhantes para a autenticação e encriptação das mensagens;
- Construção débil da *Message Authentication Code (MAC)* que usa a função para o cálculo de *hash* do algoritmo criptográfico *MD5*[5] com um prefixo secreto, tornado vulnerável a ataques do tipo *length extension*[6].

[5] Para mais informação consultar a secção 2.4
[6] Para mais informação consultar a secção 3.2

- Ausência de proteção a ataques ao estabelecer uma sessão (*handshake*), nomeadamente a ataques do tipo *Man-in-the-middle - MITM*[7] fazendo um *downgrade attack*[8].

- O SSL 2.0 usa *TCP connection close*, para indicar o fim de informação. Isto indica-nos que está suscetível a *truncation attacks*[9].

- Esta versão do protocolo assume um serviço único e certificado de domínio fixo, que contraria o *standard* de *virtual hosting* nos servidores *web*.

Por sua vez o *SSL* 3.0, que surgiu em 1996, foi descontinuado em Junho de 2015 (Barnes et al., 2015). Traz melhorias relativamente à versão anterior, adicionando suporte em algoritmos criptográficos baseados em *SHA-1*, para autenticação de certificados. No entanto foi descoberta uma vulnerabilidade que deu origem, em Outubro de 2014, a um *POODLE Attack*[10].

De seguida emergiu o *TLS* 1.0, em 1999, uma atualização do *SSL* 3.0, como já foi exposto. Esta versão do protocolo tem poucas diferenças relativamente ao seu antecessor. Ainda assim está vulnerável ao modo como opera o algoritmo *cipher-block chaining – CBC*, que usa a *block cipher* para disponibilizar a confidencialidade ou autenticidade da informação.

O *TLS* 1.1, definido em Abril de 2006, já está protegido contra as vulnerabilidades da *CBC:* a inicialização implícita de vetor passou para explícita inicialização de vetor[11].

Desde Agosto de 2008 até aos dias de hoje, seguiu-se o *TLS* 1.2. A principal diferença inclui a combinação de algoritmos criptográficos através de *pseudorandom function (PRF)* serem substituídos por *cipher suite*[12] *PRF's* especificados.

Existe já um *Internet-Draft*[13], anunciado em Janeiro de 2016, para a próxima versão do protocolo, o *TLS 1.3*[14].

De seguida explica-se a proveniência do *SSL*.

2.3.2 SSL

O *SSL* é um *standard* na transmissão segura de informação via Internet e permite criptografar os dados que transmitimos através de uma comunicação entre cliente e servidor. Pode-se também mencionar que se trata de um protocolo, ou seja, o *SSL* descreve como os algoritmos

[7] Para mais informação consultar a secção 3.3
[8] Para mais informação consultar a secção 3.4
[9] https://www.blackhat.com/us-13/briefings.html#Smyth
[10] Para mais informação consultar a secção 3.7
[11] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38761
[12] Para mais informação consultar a secção 2.4
[13] https://www.ietf.org/id-info/
[14] https://tlswg.github.io/tls13-spec/

de encriptação devem ser usados, quais as variáveis de encriptação durante a transmissão de informação, certificados, entre outros.

Foi desenvolvido pela Netscape Communications [15] e tem a função de providenciar uma transmissão segura na Internet, através de autenticação e encriptação dos pacotes entre o servidor e cliente.

A subsecção adjacente informa acerca da evolução do SSL.

2.3.3 TLS

O *TLS* é uma atualização mais segura do *SSL*. Ainda se costuma utilizar este acrónimo por ser o mais comum. Daqui em diante no documento será usado o termo *TLS*, exceto quando houver a necessidade de especificar uma versão específica do *SSL* (subsecção 2.3.1).

Relativamente à sua representação no modelo OSI, este insere-se entre a camada 4 e a camada 7. No modelo TCP/IP está presente entre a camada 2 e a camada 3, como se pode verificar através da Figura 4.

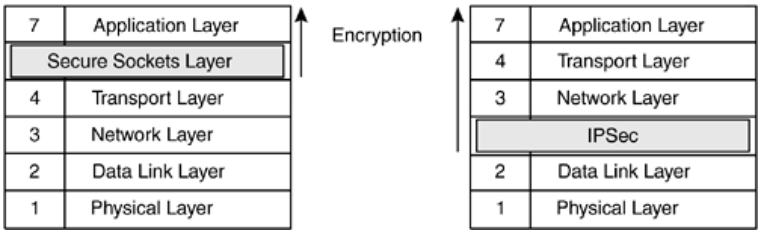


Figura 4 - TLS no modelo OSI (à esquerda) e no modelo TCP/IP (à direita)
(Syme and Goldie, 2004)

De seguida, irá ser explicada o processo de comunicação e autenticação numa ligação *TLS*.

2.3.4 Handshake

Tanto o *SSL* como o *TLS* usam uma combinação de criptografia de chave pública e chave simétrica. A criptografia de chave simétrica é mais rápida que a criptografia de chave pública, no entanto esta última fornece melhores técnicas de autenticação (Symantec, 2014).

Para estabelecer uma sessão *TLS* é necessária haver previamente uma troca de mensagens, chamada de *Handshake*.

O *Handshake* permite que o servidor se autentique com o cliente usando uma chave pública. De seguida, permite que o cliente e o servidor cooperem na criação das chaves simétricas

[15] http://isp.netscape.com/

utilizadas para uma rápida encriptação, desencriptação e deteção de intrusos durante a sessão. Opcionalmente o *Handshake* também permite que o próprio cliente se autentique no servidor.

Antes de haver troca de dados entre o cliente e o servidor é necessário negociar o canal criptográfico a usar. O cliente e o servidor devem concordar a versão do protocolo *TLS*, escolher a *cipher suite*[16] e validar os certificados[17], se necessário. Cada uma destas etapas requer um novo pacote de dados, o que aumenta o tempo de resposta na inicialização de todas as ligações através de *TLS*.

A Figura 5 ilustra o *Handshake* em *TLS*.

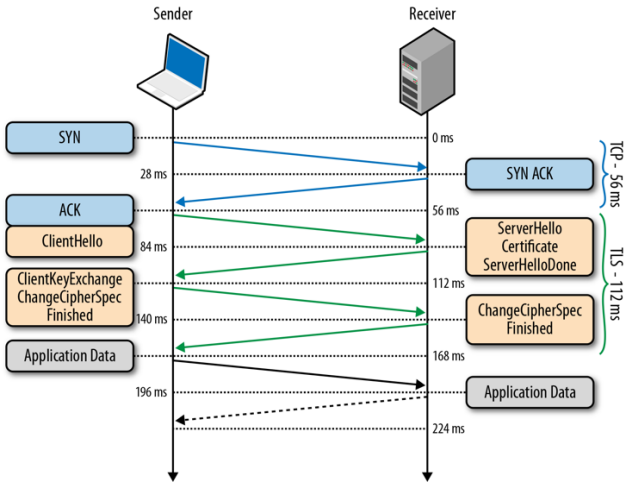


Figura 5 - TLS Handshake
(Grigorik, 2013)

A próxima secção explica o enquadramento da c*ipher suite* numa comunicação *TLS.*

2.4 Cipher Suite

Esta secção pretende explicar o termo *cipher suite* e enumerar com breve detalhe os principais algoritmos de encriptação.

Cipher suite é uma combinação de nomes entre algoritmos de autenticação, encriptação, *MAC* e algoritmos de troca de chaves, usados para a negociação numa conexão sobre o protocolo *TLS*.

Por exemplo TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, em que:

[16] Para mais informação consultar a secção 2.4
[17] Para mais informação consultar a secção 2.5

- **ECDHE_RSA**, representa o algoritmo de troca de chaves, usado para determinar como é que o cliente e o servidor irão autenticar-se durante o *Handshake*.

- **AES_128_GCM**, retrata o algoritmo de criptografia, usado para encriptar a *message stream*. Inclui também o tamanho da chave e comprimento dos vetores de inicialização, explícitos e implícitos.

- **SHA256**, retrata o algoritmo *MAC*, usado para criar a *hash* criptográfica de cada bloco da *message stream*.

Faltando referir a ***pseudorandom function (PRF)***, que utiliza a função de *hash* do algoritmo *MAC* para criar a *master secret*, um segredo de 48 *bytes* partilhado entre os dois intervenientes da ligação. O objetivo principal deste segredo é ser utilizado como fonte de imprevisibilidade ao criar as chaves de sessão, tal como utilizado para criar o algoritmo *MAC*.

Lista-se brevemente alguns algoritmos criptográficos que o autor considera importantes.

- **RSA**[18] **(autenticação):** a chave de encriptação é pública e difere da chave privada, que é mantida em segredo. Neste sistema de encriptação, a assimetria é baseada na dificuldade de factorização de dois números primos grandes. Apenas quem souber os números primos é que consegue descodificar a mensagem encriptada. Parágrafo adaptado (Rivest et al., 1978)

- **Diffie-Hellman**[19] **(troca de chaves):** na troca de chaves é partilhado um segredo entre as partes envolvidas, que pode ser usado numa comunicação segura para troca de informação numa rede pública. Parágrafo adaptado (Merkle, 1978)

- **MD5**[20] **(MAC):** é uma função de *hash* que gera um valor de 128 bits. Esta função foi considerada imprópria para utilização por ser criptograficamente frágil. Parágrafo adaptado (CMU Software Engineering Institute, 2008)

Existem muitos mais algoritmos criptográficos para além dos acima mencionados. Temos como exemplo o **ECDH**[21], o **PSK**[22], o **DSA**[23], o **ECDSA**[24], o **AES**[25] e o **SHA**[26].

A próxima secção irá abordar os certificados digitais.

[18] Para mais informações consultar o RFC 2437 (Kaliski and Staddon, 1998)
[19] Para mais informações consultar o RFC 2631 (Rescorla, 1999)
[20] Para mais informações consultar o RFC 1321 (Rivest, 1992)
[21] Para mais informações consultar o RFC 6090 (McGrew et al., 2011)
[22] Para mais informações consultar o RFC 4279 (Eronen and Tschofenig, 2005)
[23] Para mais informações consultar o RFC 2792 (Blaze et al., 2000)
[24] Para mais informações consultar o RFC 4754 (Fu and Solinas, 2007)
[25] Para mais informações consultar o RFC 3394 (Schaad and Housley, 2002)
[26] Para mais informações consultar o RFC 3174 (Eastlake and Jones, 2001)

2.5 Certificados Digitais

Um certificado digital atesta a quem pertence a chave pública através de um atributo do certificado. Isto permite aos outros confiarem em assinaturas feitas pela chave privada, que corresponde na prática à chave pública certificada.

Por outras palavras, um certificado digital funciona como um "passaporte" com autorização para troca de informação de forma segura, usando uma chave pública, do tipo *Public Key Infrastructure (PKI)*.

Assim como um passaporte, o certificado digital tem identificação, é resistente à falsificação e pode ser verificado porque foi autorizado por uma entidade fidedigna.

O certificado digital contém o nome do titular, número de série, data de validade, uma cópia da chave pública e a assinatura digital da autoridade que emite a certificação (*Certificate Authority - CA*), de modo que o destinatário possa validar se o certificado é autêntico.

A Figura 6 ilustra o exemplo de um certificado digital válido.

Figura 6 - Exemplo de um certificado digital

A maioria dos certificados digitais usa o *standard* X.509[27].

O OCSP é um protocolo usado para obter o status de revogação dos certificados X.509 (Santesson et al., 2013) e é o substituto do CRL.

[27] https://tools.ietf.org/html/rfc5280

Desta forma é possível perceber se o certificado está válido ou já expirou, por exemplo.

De seguida, explica-se a relevância dos cabeçalhos HTTP numa comunicação *TLS*.

2.6 Cabeçalhos HTTP

Cabeçalhos *HTTP* são componentes que se podem adicionar nos pedidos e respostas entre o cliente e o servidor (ou vice-versa) e definem comportamentos numa comunicação *HTTP* (Fielding et al., 1999).

No entanto, embora sejam importantes e conhecidos, por vezes também são desprezados. Quando implementados, podem tornar a aplicação mais versátil e até mesmo segura ("OWASP Secure Headers Project - OWASP," 2016).

As próximas subsecções apresentam os cabeçalhos *HTTP* que o autor considera importantes implementar ao nível de segurança.

2.6.1 HTTP Strict Transport Security (HSTS)

O HTTP Strict Transport Security (*HSTS*) é uma funcionalidade de segurança em que o domínio *web* obriga o *browser* (cliente) a comunicar-se apenas em *HTTPS* em vez de *HTTP* (Hodges et al., 2012). Uma das principais razões é mitigar os ataques de *downgrade* (forçar uma ligação *HTTP*) ao *HTTPS*.

Para disponibilizar esta funcionalidade deve-se respeitar as seguintes configurações (Hodges et al., 2012), como se pode verificar através da Figura 7.

```
Strict-Transport-Security: max-age=expireTime [; includeSubDomains]
```

Figura 7 - Configuração do cabeçalho HSTS

Em que os parâmetros:

- **max-age**: define o tempo, em segundos, que o *browser* deve respeitar e ligar-se apenas em HTTPS;
- **includeSubDomains** (opcional): quando especificado, a regra também se aplica aos subdomínios *web*.

A Figura 8 exemplifica uma configuração deste cabeçalho:

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

Figura 8 - Exemplo de uma configuração HSTS para 1 ano, incluindo os subdomínios

2.6.2 HTTP Public Key Pinning (HPKP)

O *HTTP Public Key Pinning* (*HPKP*) é uma funcionalidade de segurança em que o domínio *web* diz ao cliente *web* para associar uma chave criptográfica específica por forma a prever ataques do tipo *MITM* a certificados fraudulentos (Evans et al., 2015).

Para disponibilizar esta funcionalidade deve-se respeitar as seguintes configurações (Evans et al., 2015), como se pode verificar através da Figura 9.

```
Public-Key-Pins: pin-sha256="base64=="; max-age=expireTime [; includeSubDomains][; report-uri="reportURI"]
```

Figura 9 - Configuração do cabeçalho HPKP

Em que os parâmetros:

- **pin-sha256**: tem de estar codificado em *Base64*[28] com impressão digital *SPKI*[29]; É possível especificar vários "Pins" para diferentes chaves públicas;
- **max-age:** define o tempo, em segundos, que o *browser* deve lembrar que o cliente *web* apenas deve ser acedido usando um dos "Pins";
- **includeSubDomains** (opcional): quando especificado, a regra também se aplica aos subdomínios *web;*
- **report-uri** (opcional): quando especificado, permite reportar erros de validação para um determinado *URL*.

A Figura 10 exemplifica uma configuração deste cabeçalho:

```
Public-Key-Pins: pin-sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM=";
pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g="; report-
uri="http://example.com/pkp-report"; max-age=10000; includeSubDomains
```

Figura 10 - Exemplo de uma configuração HPKP

[28] https://tools.ietf.org/html/rfc4648
[29] https://www.rfc-editor.org/rfc/rfc2693.txt

2.6.3 Content-Security-Policy (CSP)

O *Content-Securicy-Policy* (*CSP*) é uma funcionalidade de segurança que instrói o cliente *web* do tipo de recursos que são permitidos carregar e a sua proveniência. (Mike West, 2016).

Um dos principais objetivos do *CSP* é mitigar e reportar ataques do tipo *XSS*[30].

A lista de parâmetros[31] configuráveis pode por exemplo, definir a origem de todos os *scripts* que o site pode executar, como se pode verificar através da Figura 11.

```
Content-Security-Policy: script-src 'self'
```

Figura 11 - Exemplo de uma configuração CSP

2.6.4 X-XSS-Protection

O cabeçalho *X-XSS-Protection* ativa o filtro de proteção contra os ataques do tipo *XSS* (Scott Helme, 2015).

As configurações que se podem aplicar são:

- **0**: filtro desativo;
- **1**: filtro ativo. Se o *browser* detetar um ataque do tipo *XSS* a página é limpa;
- **1; mode=block**: filtro ativo. Se o *browser* detetar um ataque do tipo *XSS* impede a página de renderizar;
- **1**; **report=http://[YOURDOMAIN]/your_report_URI**: filtro ativo. O browser limpa a página e reporta a tentativa de ataque para um determinado *URL*.

A Figura 12 exemplifica uma configuração deste cabeçalho:

```
X-XSS-Protection: 1; mode=block
```

Figura 12 - Exemplo de uma configuração X-XSS-Protection

[30] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
[31] https://www.w3.org/TR/CSP3/

2.6.5 X-Frame-Options

O *X-Frame-Options* é uma funcionalidade que declara ao *browser* que o domínio *web* não deve ser apresentado dentro de uma *frame*[32] ou dentro de outros domínios *web* (Ross and Gondrom, 2013).

Previne também ataques do tipo *clickjacking*[33].

As configurações que se podem aplicar são:

- **deny:** não carregar a página dentro de uma frame;
- **sameorigin:** não carregar a página se a origem não coincidir.
- **allow:from: DOMAIN:** permite carregar a página num domínio especificado.

A Figura 13 exemplifica uma configuração deste cabeçalho:

```
X-Frame-Options: deny
```

Figura 13 - Exemplo de uma configuração X-Frame-Options

2.6.6 X-Content-Type-Options

O *X-Content-Type-Options* é uma funcionalidade que previne o *browser* de interpretar *MIME types* desconhecidos (Freed and Borenstein, 1996).

A sua configuração contém apenas um valor:

- **nosniff:** previne que o *browser* interprete tipos de conteúdo desconhecidos.

A Figura 14 exemplifica uma configuração deste cabeçalho:

```
X-Content-Type-Options: nosniff
```

Figura 14 - Exemplo da configuração X-Content-Type-Options

[32] https://tools.ietf.org/html/rfc7540
[33] https://www.owasp.org/index.php/Clickjacking

2.7 Boas Práticas na Configuração dos Serviços via TLS

Esta secção pretende enumerar boas práticas na configuração dos serviços oferecidos via TLS e baseia-se no guia da Qualys SSL Labs ("SSL and TLS Deployment Best Practices," 2016). Este guia não dispensa uma verificação regular do mesmo e de outras fontes com o mesmo tipo de informação.

- Chave Privada e Certificado:
    - Preferir Chaves Privadas RSA de 2048-bit (no mínimo);
    - Proteger as Chaves Privadas com palavras-chave;
    - Renovar os Certificados anualmente;
    - Evitar *wildcards* na criação de certificados;
    - Adquirir os Certificados de uma *Certification Authority* (CA) fidedigna.
- Configuração:
    - Usar todos os certificados fornecidos pela CA (em caso de ser necessário utilizar mais do que um Certificado), de modo a não ter problemas na cadeia de Certificados;
    - Suportar apenas a versão TLS 1.2.
    - *Cipher Suites*:
        - Preferir *AEAD*[34] que fornece autenticação, troca de chaves, suporta *Forward Secrecy* e encriptação no mínimo com 128 *bits*.
        - Evitar *Cipher Suites ADH* (*Anonymous Diffie-Hellman*);
        - Evitar RC4;
        - Evitar 3DES.
    - Utilizar *Forward Secrecy*;
    - Preferir o algoritmo ECDHE com a curva elíptica "P-256" para a troca de chaves.
- Implementar Cabeçalhos HTTP:
    - HSTS;

[34] https://www.rfc-editor.org/rfc/rfc5116.txt

- HPKP;
- CSP;
- X-XSS-Protection;
- X-Frame-Options;
- X-Content-Type-Options.

2.8 Conclusões

Nos dias de hoje a Internet é seguramente um dos meios de comunicação mais utilizados, permitindo efetuar inúmeras tarefas à distância de uns cliques. Efetuar transações bancárias ou comprar roupa, torna-se ainda mais comum a cada dia que passa. Há uma constante troca de informação, por vezes sensível, e é crucial manter a integridade e confidencialidade em todas as comunicações. Quem disponibiliza os serviços, deve assegurar a troca de informação de forma segura e sem consequências indesejadas para os utilizadores.

Neste capítulo foi dada uma visão geral sobre o estado da arte à volta dos protocolos TLS. Tentou-se dar a entender como se estabelece uma comunicação segura e perceber o papel dos certificados digitais, bem como a *cipher suite* associada neste processo.

É possível, a quem de direito, aplicar boas práticas nas configurações dos serviços, nomeadamente na configuração dos protocolos TLS e nos cabeçalhos HTTP, com o intuito de despistar possíveis vulnerabilidades e assegurar uma maior segurança a quem utiliza esses serviços.

3 Vulnerabilidades / Ataques

Este capítulo irá descrever de forma sucinta, as várias vulnerabilidades que existem no protocolo *TLS*.

Existem vulnerabilidades que podem ser exploradas por dois tipos: falhas de implementação e falhas de protocolo.

Uma implementação é uma parte de um software que interpreta um protocolo. O *OpenSSL*[35], por exemplo é uma implementação.

As seguintes secções dão a conhecer os principais ataques.

3.1 Heartbleed attack

Este ataque não é uma falha de protocolo, mas sim uma falha de implementação que afeta algumas versões do *OpenSSL*. Este tipo de ataque afeta as versões do protocolo *TLS 1.0, TLS 1.1 e TLS 1.2.* Este tipo de ataque permite roubar informação protegida, tais como nomes de utilizador e palavra-passe, por exemplo. Compromete as chaves privadas usadas para encriptar a ligação, sendo possível espiar as comunicações realizadas (Codenomicon, 2014).

As versões *1.0.1, 1.0.1f, 1.0.2-beta* e *1.0.2-beta1* do *OpenSSL* estão afetadas. Para estar protegido contra esta vulnerabilidade basta atualizar a versão do *OpenSSL* para a *1.0.1g* ou superior.

[35] Para mais informação consultar a subsecção 4.1.1

3.2 Length extension attack

Este tipo de ataque acontece quando certos tipos *hash* são usados indevidamente como *MAC*, o que permite ao atacante adicionar informação extra à mensagem.

Algoritmos criptográficos para o cálculo de *hash* baseados na construção de "Merkle-Damgård", tais como MD5, SHA-1 e SHA-2, por exemplo, estão vulneráveis a este ataque.

Os algoritmos MD2, SHA-224, SHA-3 e SHA-384, por exemplo, não são vulneráveis a este ataque.

Para estar protegido contra esta vulnerabilidade, para além dos algoritmos não vulneráveis descritos no parágrafo anterior, pode-se substituir o *MAC* por *HMAC*, visto utilizar uma construção diferente nos algoritmos criptográficos.

3.3 Man-in-the-middle attack (MITM)

Este tipo de ataque permite ao atacante intercetar-se no meio de uma ligação e fazer-se passar por ambas as partes (cliente e servidor, por exemplo), onde toda a informação trocada é passada pelo atacante, como se pode verificar na Figura 15. Ao ter sucesso, o atacante pode manipular os dados trocados entre as partes envolvidas.

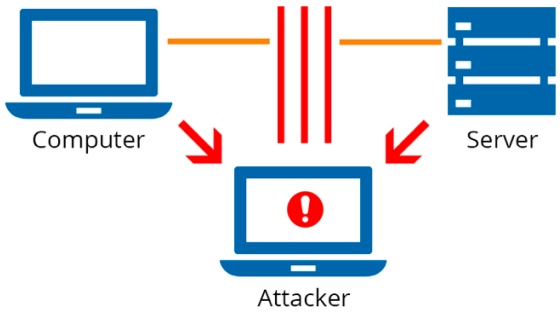


Figura 15 - Exemplo do ataque man-in-the-middle
(StarForce Technologies, 2014)

Uma das formas de prevenir este tipo de ataques é usar técnicas de implementação *PKI*, ou *Certificate and Public Key Pinning* [36], por exemplo. Para os clientes, é aconselhável o uso de VPN's.

[36] https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning

3.4 Downgrade attack (FREAK e Logjam)

Este tipo de ataque pode forçar os servidores e clientes a negociarem uma ligação usando chaves de encriptação fracas, geralmente *Diffie-Hellman*.

Um ataque *Factoring RSA Export Keys* (*FREAK* [37]), juntamente com a técnica *MITM*, permite que os atacantes intercetem as ligações *HTTPS.* Para prevenir este tipo de ataques, basta desativar a opção "*TLS export chiper suites*".

Já o ataque *Logjam*, juntamente com a técnica *MITM*, permite igualmente modificar os dados durante uma ligação sobre o protocolo *TLS*, usando uma encriptação *Ephemeral Diffie-Hellman (DHE)* de 512-bit. Este ataque acontece quando o servidor suporta "DHE_EXPORT". Para prevenir este tipo de ataque, deve-se desativar o suporte "DHE_EXPORT" e usar encriptações *Diffie-Hellman* de 2048-bit, ou então encriptações *Elliptic Curve Diffie-Hellman* (*ECDH*).

3.5 Renegotiation attack

Este tipo de vulnerabilidade acontece no processo de renegociação *TLS* que permite que o atacante injete texto nos pedidos da vítima. De notar que o atacante não consegue decifrar a comunicação entre o cliente e o servidor.

Uma das formas de prevenir este tipo de ataque é desativar o suporte a renegociação ou usar o *standard* de uma extensão de renegociação (Rescorla et al., 2010).

3.6 BEAST attack

O ataque Browser Exploit Against SSL/TLS (BEAST), afeta as versões anteriores do protocolo *TLS* 1.0 (inclusive). Este ataque aproveita as vulnerabilidades do *CBC, que juntamente com o ataque MITM* conseguem desencriptar e obter os *authentication tokens*, dando acesso (aos atacantes) aos dados transmitidos entre o servidor e o *browser* do cliente.

Para prevenir este tipo de ataque, basta utilizar as versões mais recentes do protocolo *TLS*, nomeadamente as versões 1.1 e 1.2.

[37] https://freakattack.com/

3.7 POODLE attack

O ataque *Padding Oracle On Downgraded Legacy Encryption - POODLE* [38] é um pouco semelhante ao ataque *BEAST*. Também aproveita as fragilidades do *CBC,* que juntamente com o ataque *MITM*, permitem desencriptar informação numa ligação com o protocolo *SSL* 3.0.

Uma das formas de prevenir este ataque é desativar o suporte à versão do protocolo *SSL 3.0*.

3.8 Bicycle attack

Esta recente vulnerabilidade demonstrada por Vraken, necessita que o atacante inspecione o tráfego *HTTPS* e seja capaz de determinar o comprimento de alguns pacotes transmitidos. Isso inclui calcular o comprimento dos cabeçalhos dos *cookies*, o comprimento da palavra-passe enviada através de um pedido *POST*, ou qualquer outra informação encapsulada no tráfego *TLS*. Este tipo de ataque é completamente indetetável e pode ser usado com efeitos retroativos sobre o tráfego *HTTPS* registado (Vranken, 2015).

Para este ataque triunfar é necessário serem preenchidos alguns requisitos: o *HTTPS* tem de usar algoritmos *stream-based* e depois determinar o comprimento dos restantes dados antes de extrair as partes específicas dos pacotes *HTTPS*. Dadas essas condições, basta capturar os pacotes que o utilizador gera numa operação de autenticação. Sabendo o nome de utilizador, o *URL* de *login* e com uma simples subtração é facilmente calculável o comprimento da palavra-passe do utilizador. Depois é possível recorrer a mecanismos de *brute-force* para descobrir a palavra-passe.

Para estar prevenido contra este tipo de ataque é recomendado desativar o suporte a algoritmos *TLS stream-based*, usar a última versão do protocolo (*TLS 1.2*) e adicionar um *padding* aos dados sensíveis de modo a mascarar o comprimento real dos dados a transmitir.

3.9 DROWN Attack

O ataque *Decrypting RSA with Obsolete and Weakened eNcryption – DROWN*, permite explorar vulnerabilidades nas comunicações *TLS* tirando partido da ligação *SSL 2.0* (Aviram et al., 2016).

É possível recorrer ao mecanismo semelhante utilizado no *POODLE Attack*[39] em conexões TLS para desencriptar a troca de informação. De notar que o *DROWN attack* pode ser utilizado em diferentes servidores, mesmo que não suportem *SSL 2.0*, uma vez que a maioria partilha a mesma chave RSA! (Prática comum entre o servidor web e o servidor de correio eletrónico, para o mesmo domínio)

[38] https://poodle.io/
[39] Para mais informação consultar a secção 3.7

Para prevenir este ataque é obrigatório desativar o suporte às versões *SSL 2.0* (ou inferior) ou garantir que as chaves privadas não são usadas em servidores que suportem as referidas versões.

3.10 BREACH e CRIME

Tanto o *Browser Reconnaissance & Exfiltration via Adaptive Compression of Hypertext – BREACH*[40] como o *Compression Ratio Info-leak Made Easy – CRIME* (Ritter, 2012) utilizam a técnica MITM para tirar partido dos mecanismos de compressão de dados, por forma a desencriptar e explorar a informação que é tratada (especialmente cookies) ("RFC 7457 - Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)," n.d.)

Para mitigar os ataques *CRIME* e *BREACH*, é imperativo desativar a compressão ao nível do TLS/SSL e a compressão ao nível do HTTP, respetivamente.

3.11 Forward Secrecy (FS) / Perfect Forward Secrecy (PFS)

Numa ligação encriptada o cliente e o servidor partilham uma chave de sessão, que é responsável por garantir a confidencialidade da comunicação. Quem intercetar essa comunicação e não conhecer a mesma, não consegue desencriptar os dados capturados.

O conceito de Perfect *Forward Secrecy e Forward Secrecy* (Quotes, 2016) são as propriedades que garantem que uma chave de sessão derivada de um conjunto de chaves públicas e privadas de longo prazo não comprometem as chaves de sessão criadas anteriormente e fornecem proteção para as chaves de sessão no futuro (Quotes, 2016).

Para garantir esta funcionalidade a nível de protocolo, é necessário garantir e preferir "*cipher suites*" ECDHE[41].

3.12 Conclusões

A informação tem hoje um valor incalculável e um impacto nas organizações que pode colocar em causa a continuidade do negócio. Como se pode verificar, existem inúmeras vulnerabilidades e a cada dia que passa, criam-se novas tecnologias e formas de contornar soluções que à partida estão estariam mitigadas com o lançamento de atualizações para prevenir certo tipo de ataques. É imperativo acompanhar de perto este tema e sobretudo dar mais visibilidade ao problema para que se possa reduzir os impactos negativos que advêm.

[40] http://breachattack.com/
[41] Para mais informações consultar a secção 2.4

4 Ferramentas de Análise Criptográfica

Este capítulo pretende enumerar algumas soluções existentes (dar ênfase às ferramentas que o autor considera mais importantes) para a recolha dos dados a analisar e demonstrar as vantagens e desvantagens de cada uma.

Existem no mercado diversas ferramentas de análise criptográfica, para uso privado, com o intuito de rever as configurações existentes e disponibilizar informação útil para o proprietário que disponibiliza os serviços. Contudo, nem todas elas são as que mais se adequam para este estudo.

Será explicada a escolha das ferramentas que mais se adequam ao propósito desta dissertação e será validada a recolha de informação entre as mesmas.

De seguida, enumera-se as soluções existentes no mercado.

4.1 Soluções Existentes

4.1.1 OpenSSL

O *OpenSSL* [42] é uma implementação *open-source* [43] do protocolo *TLS*. Esta ferramenta implementa serviços de criptografia e está disponível para maioria dos sistemas operativos. Atualmente encontra-se na versão *1.0.2j*.

Para utilizar esta ferramenta é necessária a instalação no sistema operativo do cliente a efetuar a recolha de dados.

[42] https://www.openssl.org/
[43] https://opensource.org/

4.1.2 COMODO SSL Analyzer

Esta ferramenta[44] permite recolher as principais informações sobre certificados e protocolos suportados. Inclui também esclarecimentos acerca da *cipher suite* suportada pelo servidor.

Para servir-se desta ferramenta basta utilizar um *browser* e indicar o endereço *HTTPS*[45] a testar e recolher manualmente a informação necessária.

4.1.3 Qualys SSL Labs API

Esta *API*[46] da Qualys[47] possui um leque completo de serviços que permite testar, de forma completa, o protocolo *TLS* num servidor.

É possível através desta ferramenta, recolher informações sobre certificados, protocolos e *cipher suite* suportada. Existe também a possibilidade de ser gerado um relatório com uma classificação atribuída ao teste elaborado. Esta classificação avalia, numa escala de A a F, os vários componentes acima descritos (certificados, protocolos e *cipher suite*).

Sendo uma *API*, é possível utilizar como *interface* num cliente de software, tendo em atenção os restritos Termos e Condições[48].

4.1.4 Mozilla Observatory

O *Mozilla Observatory*[49] é um projeto *open-source* que permite aos administradores de sistemas e profissionais de segurança, analisar a segurança dos servidores web.

Tal como a *Qualys SSL Labs API*, também usufrui de um vasto leque de informação para examinar e possui uma *API*, que torna mais fácil a integração num software.

4.1.5 Security Headers

A ferramenta *Security Headers*[50] foi criada por Scott Helme, um consultor de segurança, cujo principal objetivo é analisar os cabeçalhos de uma resposta HTTP. Par além da análise, também fornece documentação para mitigar os problemas encontrados.

[44] https://sslanalyzer.comodoca.com/
[45] Para mais informação consultar a secção 2.2
[46] https://www.ssllabs.com/projects/ssllabs-apis/index.html
[47] https://www.qualys.com/
[48] https://www.ssllabs.com/about/terms.html
[49] https://observatory.mozilla.org/
[50] https://securityheaders.io

4.2 Avaliação das Soluções

Este capítulo pretende dar destaque às ferramentas já referidas, que dados recolhem, vantagens e desvantagens entre elas e métodos de validação das ferramentas sobre a informação recolhida.

Após uma análise detalhada de cada uma das ferramentas e tendo em consideração a informação necessária para a recolha dos dados para efeitos de estudo, chegou-se à conclusão que é necessário obter as seguintes informações:

- Suporte a *HTTPS;*
- Detalhes sobre certificados, nomeadamente a sua validade, algoritmo criptográfico e chave associada, autoridade de certificação, etc.;
- Protocolos suportados, nomeadamente a sua versão;
- Cipher suite;
- Implementação de cabeçalhos *HTTP* críticos;[51]
- Simulação de *Handshake*[52] por *User Agent*[53] (opcional);
- Classificação global de segurança dada a análise de cada componente anterior (opcional).

A seguinte legenda permite classificar cada um dos componentes necessários para posterior validação das ferramentas a serem escolhidas, em que:

- ⊕, representa informação detalhada;
- ⊕, representa informação pouco detalhada e insuficiente para a análise ou difícil recolha dos dados pretendidos;
- ⊗, não suporte para obtenção de dados para análise.

A Tabela 1 relaciona as ferramentas de análise com o suporte à recolha de informação necessária para o tratamento dos dados.

[51] Para mais informação consultar a secção 2.6
[52] Para mais informação consultar a subsecção 2.3.4
[53] https://www.techopedia.com/definition/1614/user-agent-ua

Tabela 1 - Suporte para recolha de informação

	HTTPS	Certificados	Protocolos	Cipher Suite	Cabeçalhos HTTP
OpenSSL	⊕	⊕	⊕	⊕	⊕
COMODO SSL Analyzer	⊕	⊕	⊕	⊕	⊗
Qualys SSL Labs API	⊕	⊕	⊕	⊕	⊕
Mozilla Observatory	⊕	⊕	⊕	⊕	⊕
Security Headers	⊕	⊗	⊗	⊗	⊕

De um modo geral, a relação entre as ferramentas e o suporte à recolha de informação para cada coluna constata-se que tanto a *Qualys SSL Labs API* e o *Mozilla Observatory* disponibilizam detalhadamente a maior parte da informação necessária para a recolha e análise dos dados propostos, tais como o suporte a HTTPS, informação sobre os certificados, protocolos, *cipher suite* e os cabeçalhos *HTTP*.

Já para o *OpenSSL*, embora se consiga recolher praticamente toda a informação necessária é necessário um maior conhecimento e domínio na ferramenta para se conseguir obter a informação descrita, tornando mais difícil a sua recolha.

A ferramenta *Security Headers* reúne informação mais específica, como é o caso dos cabeçalhos *HTTP*.

O *COMODO SSL Analyzer* apresenta-se uma solução abaixo da média, comparativamente com a *Qualys SSL Labs API* e o *Mozilla Observatory*, com falta de informação ou com informação pouco detalhada.

Para uma análise de recolha de informação opcional, nomeadamente o *Handshake* e atribuição de uma classificação, a Tabela 2 apresenta as seguintes conclusões:

Tabela 2 - Suporte para recolha de informação opcional

	Handshake	Classificação
OpenSSL	⊕	⊗
COMODO SSL Analyzer	⊗	⊗
Qualys SSL Labs API	⊕	⊕
Mozilla Observatory	⊗	⊕
Security Headers	⊗	⊕

Nos opostos temos a *COMODO SSL Analyzer*, que não possui qualquer tipo de informação para recolha e a *Qualys SSL Labs API*, que faz exatamente o pretendido.

Tanto o *Mozilla Observatory* como o *Security Headers* atribuem classificação, mas não suportam simulação de *Handshake.*

Já o *OpenSSL* não possui qualquer tipo de classificação e é necessária uma conjugação de *scripts* para a simulação de *Handshake*, o que torna a recolha dos dados ainda mais difícil e suscetível a erros.

Numa perspetiva de desenvolvimento é igualmente importante verificar se as ferramentas possuem suporte à recolha de informação. A Tabela 3 ilustra esse mesmo suporte:

Tabela 3 - Ferramentas que suportam API

	Suporte API
OpenSSL	⊗
COMODO SSL Analyzer	⊗
Qualys SSL Labs API	⊕
Mozilla Observatory	⊕
Security Headers	⊗

Ao nível de recolha de informação para posterior análise, a *Qualys SSL Labs API* e o *Mozilla Observatory* disponibilizam uma API bem documentada e por esse motivo, respondem de forma muito positiva ao que é proposto.

Já o *COMODO SSL Analyzer* e o *Security Headers* não disponibilizam uma API, sendo necessária uma intervenção manual, o que torna a recolha dos dados mais demorada e suscetível a erros.

O *OpenSSL* não dispõe de uma API, contudo é possível criar e implementar *Scripts Bash* e manipular a informação para um documento para posterior análise.

4.3 Soluções a Usar

Após um profundo estudo acerca das vantagens e desvantagens de cada ferramenta, uma análise dos prós e contras tal como foi descrito nas tabelas da secção anterior e por forma a tirar as mais-valias de cada uma, segue-se a enumeração das ferramentas que o autor decidiu utilizar e uma breve explicação.

4.3.1 Qualys SSL Labs API

Por si só, a *Qualys SSL Labs API* seria uma das ferramentas que melhor se apropriavam para a recolha de dados, visto abranger praticamente todos os tópicos que são necessários ao propósito desta dissertação.

A informação recolhida irá abranger as colunas descritas na Tabela 1 para a ferramenta em questão.

A Figura 16 apresenta um excerto do relatório de análise a um determinado domínio web.

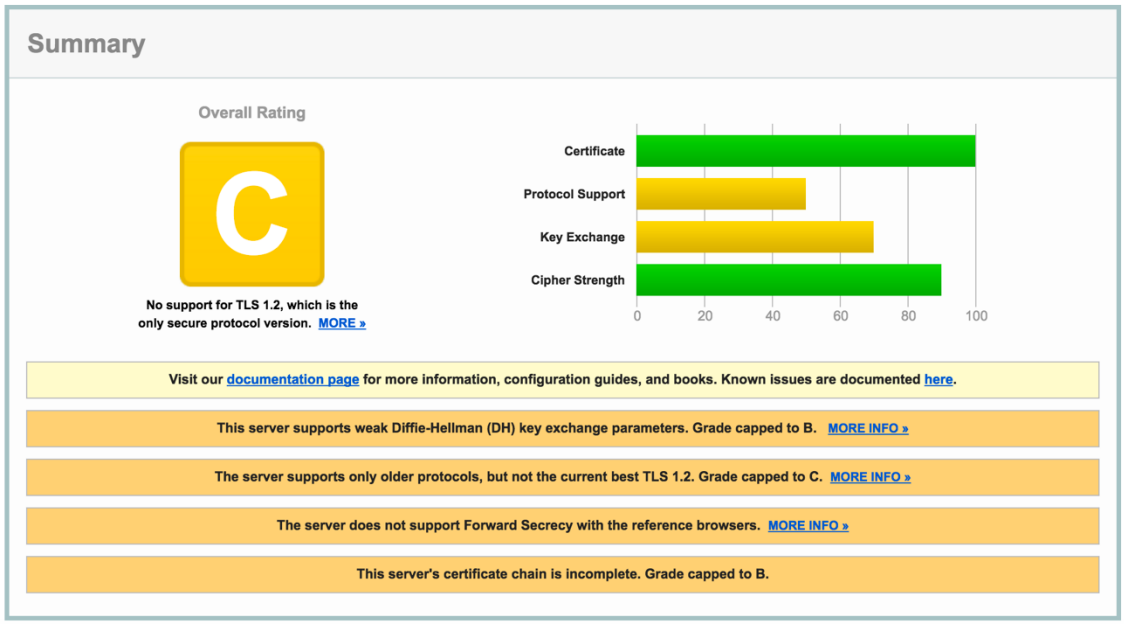


Figura 16 - Exemplo de um excerto do relatório do Qualys SSL Labs API

Contudo, após uma leitura cuidada dos Termos e Condições, foi possível verificar que sem a permissão expressa da própria entidade, neste caso da *Qualys SSL Labs*[54], não era possível efetuar a recolha dos dados a domínios web públicos.

Ao fim de algumas conversações entre *Robert Dell'Immagine,* Diretor da Comunidade da *Qualys SSL Labs* e *Ivan Ristic*, Diretor do Departamento de Engenharia e Segurança da *Qualys* e Autor da ferramenta em questão, foi possível para efeitos desta dissertação, removerem as restrições impostas pelos Termos e Condições.

É de realçar que esta ferramenta carece de informação completa sobre os cabeçalhos *HTTP*. Apenas apresenta alguns, estando ainda em fase *beta*.

[54] https://www.ssllabs.com/

4.3.2 Mozilla Observatory

Tal como a *Qualys SSL Labs API* esta ferramenta é bastante completa e vai abranger as colunas descritos na Tabela 1.

A Figura 17 ilustra um excerto de um relatório a um determinado domínio web.

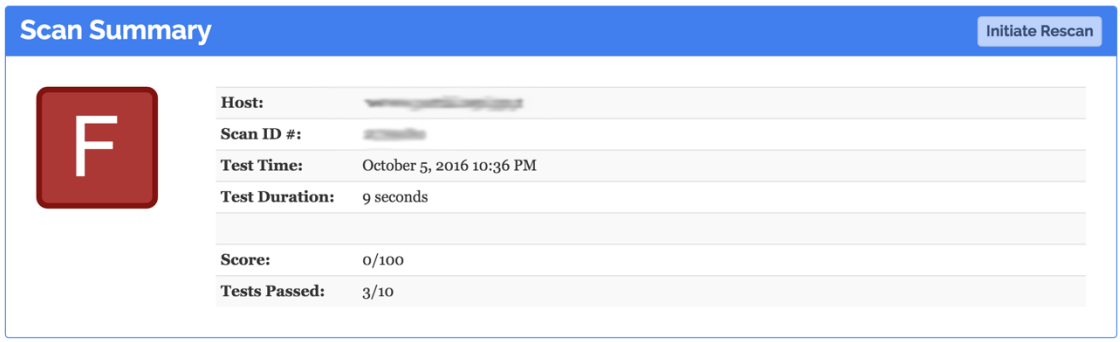


Figura 17 - Exemplo de um excerto do relatório do Mozilla Observatory

Outro critério para a sua seleção é o possuir uma API para integração num software.

4.3.3 Security Headers

Para analisar a resposta dos cabeçalhos *HTTP* esta é sem dúvida a ferramenta com informação mais detalhada e completa. Temos como exemplo a Figura 18:

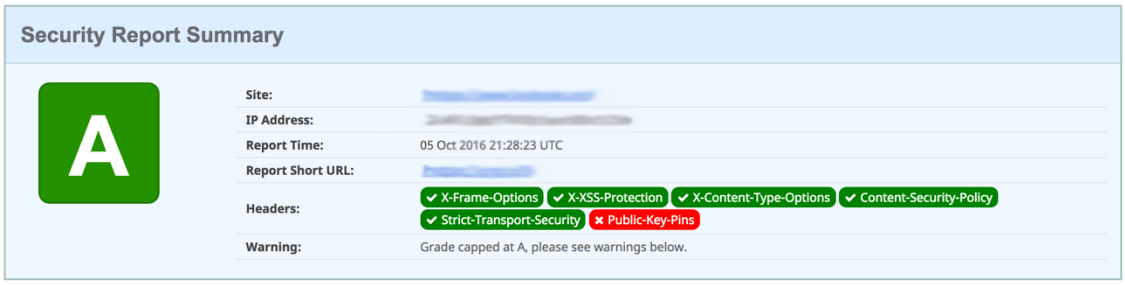


Figura 18 - Exemplo de um excerto do relatório do Security Headers

A única desvantagem é não possuir uma API.

O autor considera esta ferramenta crucial para a recolha de informação. Por esse motivo o próprio irá utilizar uma técnica de extração de informação de domínios web, denominada "*Web scraping*" ou "*Web data extraction*". ("Web Scraping Explained," n.d.)

4.3.4 OpenSSL

O uso do *OpenSSL* irá servir por exemplo para validar o suporte a *HTTPS*. Outra funcionalidade será validar informação retirada das outras ferramentas.

A Figura 19 apresenta apenas um excerto de uma resposta à ligação *HTTPS*.

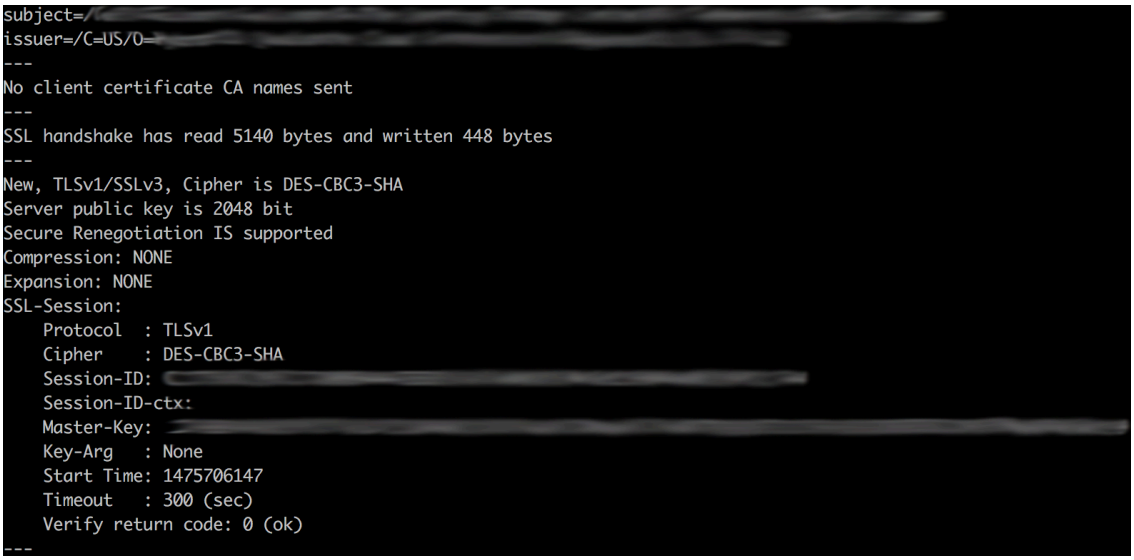
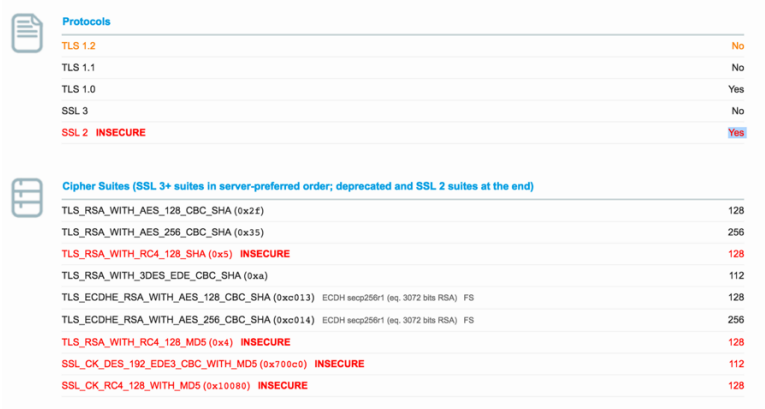


Figura 19 - Exemplo de um excerto do output gerado pelo OpenSSL

4.4 Validação das Soluções

De modo a poder legitimar e atestar a correta recolha dos dados, foram realizadas recolhas pontuais entre todas as ferramentas enumeradas na secção anterior a alguns domínios web que irão servir de comparação e validação dos dados recolhidos.

A Figura 20 e Figura 21 representam apenas um excerto de um exemplo usando a *Qualys SSL Server Test* e o *Mozilla Observatory* ao testar o suporte às *cipher suites* a um determinado domínio web.



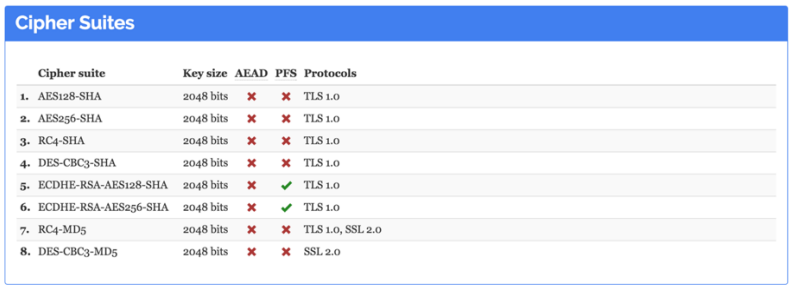
Protocols

TLS 1.2	No
TLS 1.1	No
TLS 1.0	Yes
SSL 3	No
SSL 2 **INSECURE**	Yes

Cipher Suites (SSL 3+ suites in server-preferred order; deprecated and SSL 2 suites at the end)

TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	128
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	256
TLS_RSA_WITH_RC4_128_SHA (0x5) **INSECURE**	128
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa)	112
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) ECDH secp256r1 (eq. 3072 bits RSA) FS	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_RSA_WITH_RC4_128_MD5 (0x4) **INSECURE**	128
SSL_CK_DES_192_EDE3_CBC_WITH_MD5 (0x700c0) **INSECURE**	112
SSL_CK_RC4_128_WITH_MD5 (0x10080) **INSECURE**	128

Figura 20 - Excerto de um teste às cipher suites suportadas utilizando a ferramenta Qualys SSL Server Test



Cipher Suites

	Cipher suite	Key size	AEAD	PFS	Protocols
1.	AES128-SHA	2048 bits	✗	✗	TLS 1.0
2.	AES256-SHA	2048 bits	✗	✗	TLS 1.0
3.	RC4-SHA	2048 bits	✗	✗	TLS 1.0
4.	DES-CBC3-SHA	2048 bits	✗	✗	TLS 1.0
5.	ECDHE-RSA-AES128-SHA	2048 bits	✗	✓	TLS 1.0
6.	ECDHE-RSA-AES256-SHA	2048 bits	✗	✓	TLS 1.0
7.	RC4-MD5	2048 bits	✗	✗	TLS 1.0, SSL 2.0
8.	DES-CBC3-MD5	2048 bits	✗	✗	SSL 2.0

Figura 21 - Excerto de um teste às cipher suites suportadas utilizando a ferramenta Mozilla Observatory

Para validar informação dos certificados, por exemplo, optou-se entre o *OpenSSL*, o *Qualys SSL Labs API* e o *Mozilla Observatory*, como se pode verificar na Figura 22, Figura 23, Figura 24 e Figura 25.

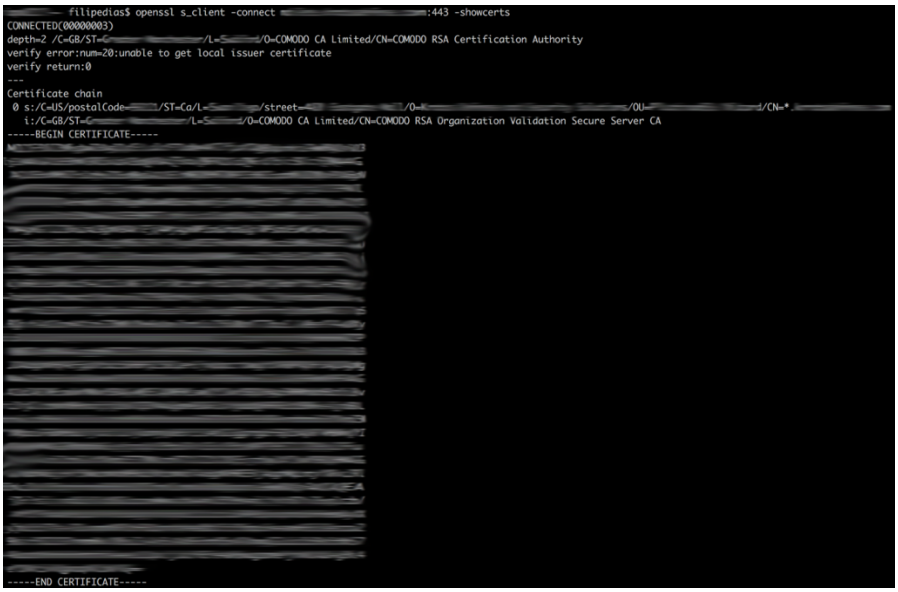


Figura 22 - Excerto de um teste aos certificados utilizando o OpenSSL

Figura 23 - Excerto de um teste à data de validade dos certificados utilizando o OpenSSL

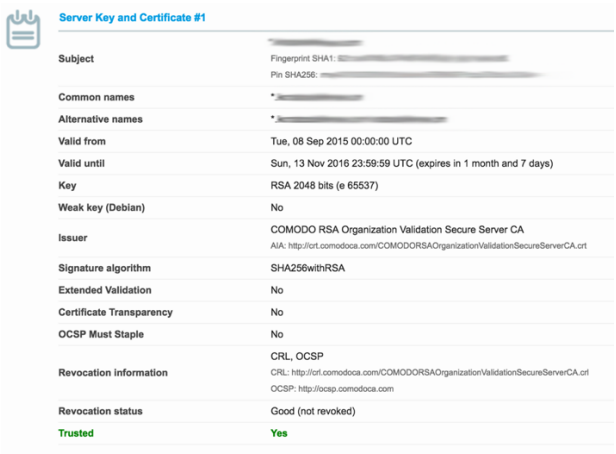
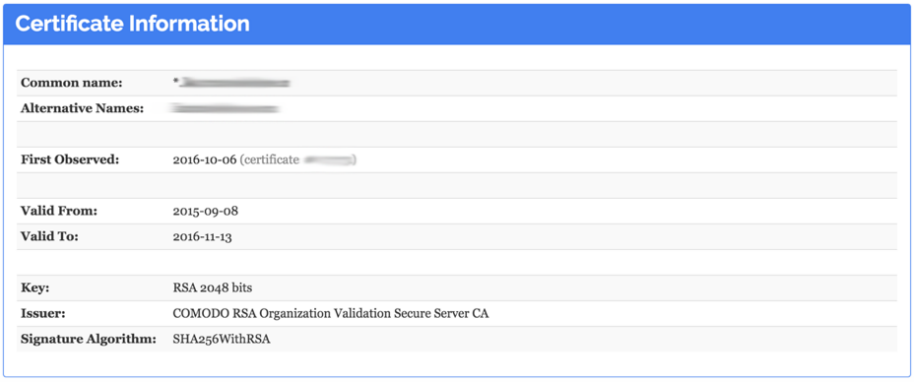


Figura 24 - Excerto de um teste aos certificados utilizando o Qualys SSL Labs API

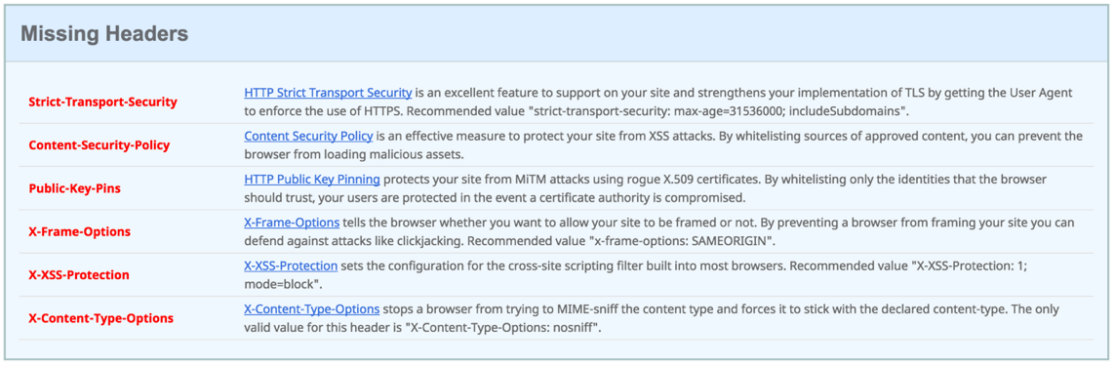


Certificate Information

Common name:	*
Alternative Names:	
First Observed:	2016-10-06 (certificate)
Valid From:	2015-09-08
Valid To:	2016-11-13
Key:	RSA 2048 bits
Issuer:	COMODO RSA Organization Validation Secure Server CA
Signature Algorithm:	SHA256WithRSA

Figura 25 - Excerto de um teste aos certificados utilizando o Mozilla Observatory

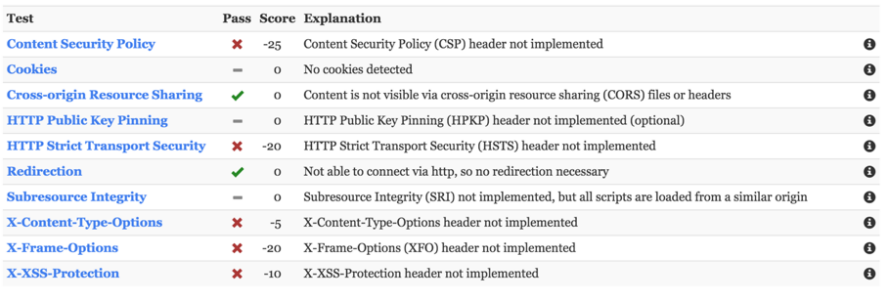
Para analisar os resultados dos cabeçalhos *HTTP*, utilizou-se a ferramenta *Security Headers* e a *Mozilla Observatory*, como se pode verificar através da Figura 26 e Figura 27.



Missing Headers

Strict-Transport-Security	HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to enforce the use of HTTPS. Recommended value "strict-transport-security: max-age=31536000; includeSubdomains".
Content-Security-Policy	Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets.
Public-Key-Pins	HTTP Public Key Pinning protects your site from MiTM attacks using rogue X.509 certificates. By whitelisting only the identities that the browser should trust, your users are protected in the event a certificate authority is compromised.
X-Frame-Options	X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking. Recommended value "x-frame-options: SAMEORIGIN".
X-XSS-Protection	X-XSS-Protection sets the configuration for the cross-site scripting filter built into most browsers. Recommended value "X-XSS-Protection: 1; mode=block".
X-Content-Type-Options	X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".

Figura 26 - Excerto de um teste aos cabeçalhos HTTP utilizando o Security Headers



Test	Pass	Score	Explanation
Content Security Policy	✖	-25	Content Security Policy (CSP) header not implemented
Cookies	–	0	No cookies detected
Cross-origin Resource Sharing	✔	0	Content is not visible via cross-origin resource sharing (CORS) files or headers
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)
HTTP Strict Transport Security	✖	-20	HTTP Strict Transport Security (HSTS) header not implemented
Redirection	✔	0	Not able to connect via http, so no redirection necessary
Subresource Integrity	–	0	Subresource Integrity (SRI) not implemented, but all scripts are loaded from a similar origin
X-Content-Type-Options	✖	-5	X-Content-Type-Options header not implemented
X-Frame-Options	✖	-20	X-Frame-Options (XFO) header not implemented
X-XSS-Protection	✖	-10	X-XSS-Protection header not implemented

Figura 27 - Excerto de um teste aos cabeçalhos HTTP utilizando o Mozilla Observatory

4.5 Conclusões

Neste capítulo instigou-se várias soluções para a análise de informação.

É possível validar e cruzar informação entre elas. Foram apresentadas vantagens e desvantagens de cada uma e o autor definiu um conjunto de soluções a utilizar na solução preconizada.

5 Design da Solução

Constituindo parte desta dissertação no desenvolvimento de uma ferramenta com o intuito de recolher os dados pretendidos, é essencial que a o *design* de software tenha uma estrutura sólida, ainda que se apresente um esboço no presente momento.

Neste capítulo será apresentada uma análise recorrendo-se a diagramas UML, para demonstrar a estrutura lógica da solução a implementar.

Serão expostos os requisitos funcionais e não funcionais, uma ilustração da arquitetura pretendida, bem como o principal caso de uso.

5.1 Tecnologias, Linguagens e Padrões de Software

Esta secção visa enumerar as tecnologias a utilizar na solução pretendida, os principais padrões de software e elucidar o leitor com alguns exemplos de *Web Services* a serem utilizados na aplicação.

5.1.1 Swift

Swift é uma linguagem de programação criada pela Apple para os dispositivos iPhone, Mac, Apple TV e Apple Watch. É *open-source* e neste momento está no Top 3 de linguagens preferidas pelos programadores ("Stack Overflow Developer Survey 2016 Results," 2016).

A razão pela qual o autor optou por esta linguagem ao invés de outras, deve-se à maior produtividade no desenvolvimento de software, como se pode verificar na Figura 28 e o facto de esta linguagem conseguir tirar melhores rendimentos a nível de *performance*, como é o caso de uma aplicação servidor ("Benchmarks for the Top Server-Side Swift Frameworks vs. Node.js," 2016).

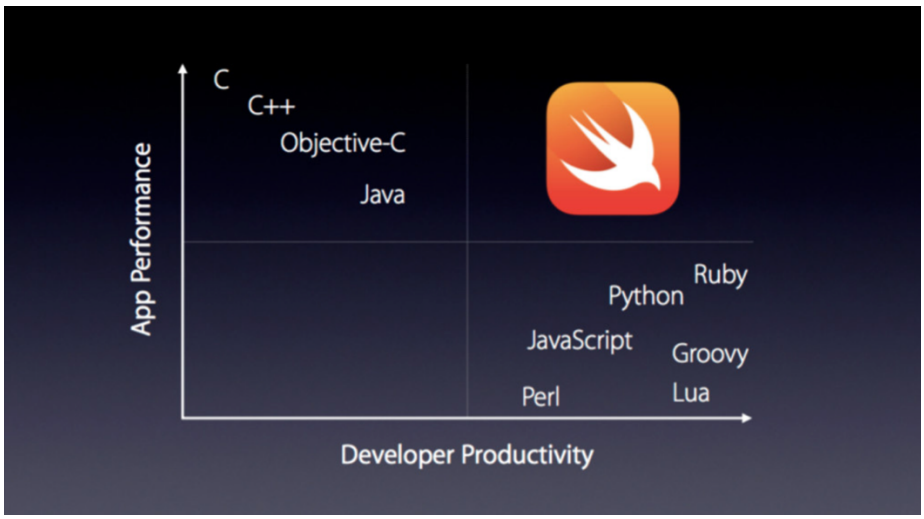


Figura 28 - Performance vs produtividade em Swift
(Edward Jiang, 2016)

A Figura 29 ilustra os resultados de performance entre Swift e Node.js, elaborados por *Ryan Collins*. De notar que tanto *Perfect, Vapor, Kitura e Zewo* são bibliotecas *Swift* para uso em aplicações servidoras.

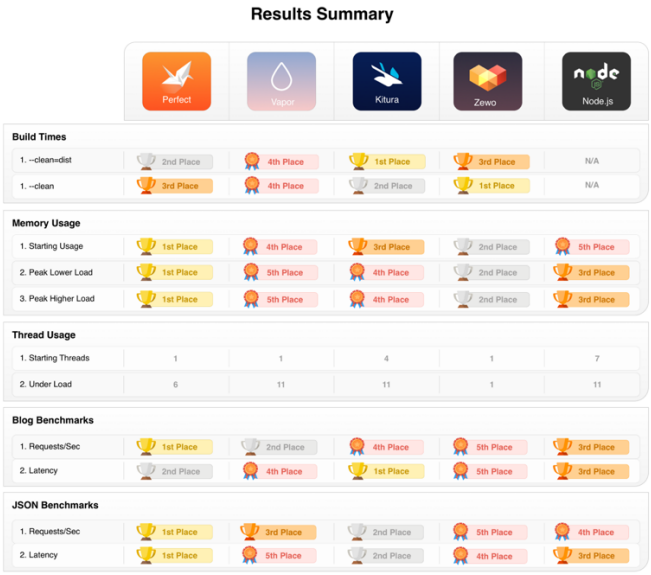


Figura 29 - Performance entre Swift e Node.js numa aplicação servidor
("Benchmarks for the Top Server-Side Swift Frameworks vs. Node.js," 2016)

5.1.2 Padrões de Software

Um padrão de software descreve uma solução geral de um problema e tende a ser reutilizável ao longo do desenvolvimento do projeto. É um modelo de como resolver um determinado tipo de problema e pode ser utilizado em situações distintas. Normalmente definem as relações e interações entre as classes ou objetos, sem especificar os detalhes das classes ou objetos envolvidos.

É importante utilizar padrões de software no desenvolvimento de uma aplicação, pois estes visam facilitar a comunicação, documentação e reutilizam soluções de design. Permitem baixo acoplamento e alta coesão, tornando melhor a separação de responsabilidades.

Desta forma, as subsecções seguintes, abordam a utilização dos padrões de software que o autor considera mais importante enumerar.

5.1.2.1 MVC

O padrão *Model-View-Controller*[55] *(MVC)* é bastante antigo.

É um padrão de alto nível que se preocupa com a arquitetura global de uma aplicação e classifica os objetos de acordo com as suas funções gerais.

O *Model* é usado para encapsular os dados, as *Views* mostram e alteram os dados e os *Controllers* são os mediadores da logica entre ambos. Ao separar as responsabilidades desta forma é mais fácil projetar, implementar e manter uma aplicação.

Existe, portanto, a separação entre a camada de dados, a camada logica e a camada de apresentação.

Programas orientados aos objetos que adaptam este padrão beneficiam de várias formas: muitos objetos desses programas tendem a ser mais reutilizáveis e as suas interfaces tendem a ser mais bem definidas. Em suma, os programas são mais adaptáveis às novas exigências.

A Figura 30 apresenta o padrão MVC.

[55] https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

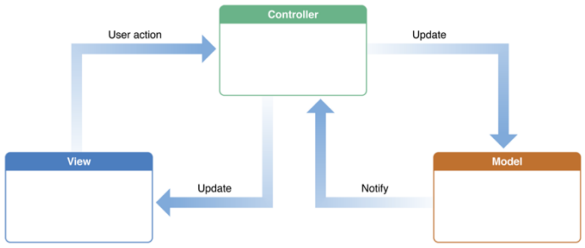


Figura 30 - Padrão MVC
(Apple Inc., 2015a)

Em que:

- *Model*: permite encapsular os dados e comportamentos básicos. Representa o conhecimento e define a lógica que manipula os dados.

- *View*: apresenta a informação ao utilizador. Esta não deve ser responsável por armazenar os dados que está a exibir. Uma *View* pode apresentar apenas uma parte contida no *Model* ou apresenta-lo por completo, ou ainda apresentar variada informação de diferentes *Models*. As *Views* são totalmente personalizáveis e reutilizáveis e fornecem consistência entre as aplicações.

- *Controller*: faz a ligação entre o *Model* e a *View* sendo intermediário entre estes dois. Como se pode verificar na figura, o *Model* e a *View* não comunicam entre si. É encarregado de aceder aos dados do *Model* e apresenta-los na View. Para além disso, se houver alterações no *Model*, atualiza na *View* ou vice-versa. Por fim, é também responsável por realizar uma pré-configuração e coordenação das tarefas bem como gerir o ciclo de vida dos objetos.

5.1.2.2 Delegation

Delegation[56] é um padrão simples e poderoso em que um objeto num programa age em nome de, ou em coordenação com outro objeto.

O objeto que delegou a tarefa mantém uma referência do outro objeto (o delegado) e, no momento apropriado, envia uma mensagem para ele. A mensagem informa o objeto delegado de um evento que o objeto que delega está prestes a manipular.

O objeto que delega, geralmente é uma *biblioteca* e o objeto delegado é, geralmente, um *Controller*.

[56] https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Delegation.html#//apple_ref/doc/uid/TP40008195-CH14-SW1

Este padrão é bastante utilizado com *Protocols*[57]. De uma forma muito resumida, um *Protocol* declara uma interface de programação que qualquer classe pode optar por implementar. Estes tornam possíveis a duas classes de software distantemente relacionadas por herança, comunicar por forma a atingir um objetivo. É uma alternativa, por exemplo, às subclasses de software. A Figura 31 descreve este último parágrafo.

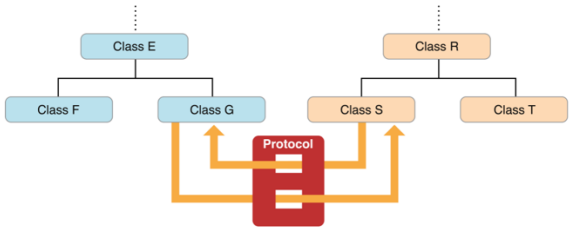


Figura 31 - Exemplo de uma aplicação orientada a Protocols
(Apple Inc., 2015b)

5.1.2.3 Singleton

Este padrão permite retornar a mesma instância de uma classe independentemente da quantidade de vezes que a aplicação o solicitar.

As classes de software que não implementam este padrão podem criar uma quantidade de instâncias da mesma classe, as vezes que for necessário. A Figura 32 ilustra a comparação entre uma classe "Típica" (que não implementa o padrão *Singleton*) e uma que implementa o padrão *Singleton*.

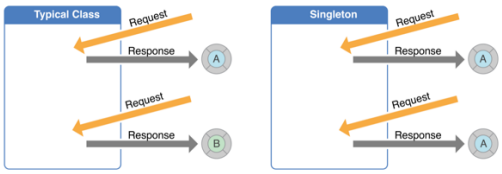


Figura 32 - Comparação entre uma classe típica e uma classe Singleton
(Apple Inc., 2015c)

[57] https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Protocol.html#//apple_ref/doc/uid/TP40008195-CH45-SW1

5.1.3 Web Services

A aplicação a ser desenvolvida que irá tornar possível a recolha de dados depende obrigatoriamente de *Web Services* e como tal, decidiu-se dar uma pequena explicação do que é e como funciona um *Web Service*.

Um *Web Service* é um conjunto de regras e processos, todos baseados em standards livres (XML, SOAP, WSDL, UDDI), utilizado para ligar serviços e aplicações *web* através da Internet.

A utilização de *Web Services* é uma mais-valia pois permite a interligação de plataformas distintas sem que haja a necessidade de nenhuma das partes, o serviço que disponibiliza o *Web Service* e o serviço que o consome, conhecer a lógica de negócio da outra.

Desta forma a aplicação criada irá consumir os *Web Services* (API's) que as ferramentas enunciadas na Tabela 3 suportarem. No futuro também é possível a própria aplicação suportar *Web Services* para comunicar com o exterior. (Por exemplo se a aplicação estiver alojada num domínio público)

Seguem-se dois exemplos de comunicação entre a aplicação desenvolvida e as respetivas ferramentas, onde a Figura 33 representa um exemplo de uma resposta enquanto que a Figura 34 ilustra um pedido.

Figura 33 - Excerto de uma resposta em formato JSON ao Web Service da Qualys SSL Labs API

```
## Request
curl -X "GET" "https://securityheaders.io/?q=[redacted]&hide=on&followRedirects=on"
\
     -H "upgrade-insecure-requests: 1" \
     -H "accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/
*;q=0.8" \
     -H "referer: https://securityheaders.io/?
followRedirects=on&hide=on&q=[redacted]" \
     -H "accept-encoding: gzip, deflate, sdch, br" \
     -H "authority: securityheaders.io"
```

Figura 34 - Excerto de um pedido em formato cURL à ferramenta Security Headers

5.2 Análise de Requisitos Funcionais

Nesta secção são enumerados os requisitos funcionais, ou seja, as funcionalidades que o sistema oferece. Algumas funcionalidades já foram enunciadas no capítulo 4.

Foi definido que a ferramenta concebida irá, para uma determinada entrada que neste caso pode ser uma lista de domínios web, recolher determinadas informações, nomeadamente:

Figura 35 - Informação a ser recolhida pela solução

No final da recolha, análise e tratamento dos de dados é solicitada ao utilizador a opção de exportar o relatório em dois formatos: "**.pdf**" ou "**.html**", podendo escolher ambas ou umas das visadas.

A Figura 36 representa o diagrama de caso de uso para um Administrador que pretenda verifica se os seus servidores estão corretamente configurados.

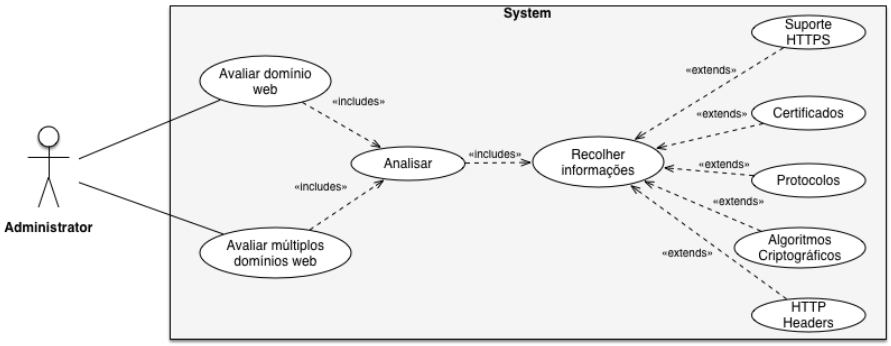


Figura 36 - Caso de Uso (UML) - Verificação de um ou múltiplos domínios web

A Tabela 4, descreve o caso de uso correspondente à verificação de um domínio web.

Tabela 4 - Descrição do Caso de Uso - Verificação de um domínio web

ID do Caso de Uso	**1**
Nome do Caso de Uso	Verificação de um domínio web
Descrição	Apresentação das configurações atuais do domínio web e respetiva classificação
Atores	Administrador
Pré-condições	O ator necessita de um domínio web válido
Pós-condições	O ator visualiza um relatório no formato **.pdf** ou **.html**, onde é atribuída uma classificação sobre os dados recolhidos

A Tabela 5, apresenta um fluxo básico de eventos do caso de uso descrito acima.

Tabela 5 - Fluxo básico de eventos - Verificação de um domínio web

Ações do ator	**Ações do sistema**
1. O ator introduz o domínio web no sistema	
	2. O sistema invoca as ferramentas de análise para recolherem informação necessária.
	3. O sistema analisa a informação recolhida
	4. O sistema trata a informação recolhida
5. O ator escolhe o formato em que pretende visualizar o relatório	
	6. É exportado um relatório final com a classificação dos dados recolhidos

5.3 Análise de Requisitos Não Funcionais

Os requisitos não funcionais são aqueles que afetam a ferramenta em termos de desempenho, segurança, usabilidade, entre outros. Estes requisitos influenciam os requisitos funcionais através de restrições e adicionam valor e qualidade à ferramenta de recolha da informação.

5.3.1 Desempenho

O desempenho da ferramenta pode ser medido através da eficácia e eficiência com que as operações são executadas, desde a rapidez que o ator tem ao seu dispor a informação necessária até aos recursos utilizados pela ferramenta ao nível de memória, carga de processamento e principalmente da ligação à Internet para recolher a informação.

Como a ferramenta está dependente de uma conexão à Internet para recolher a informação necessária, o desempenho desta varia de acordo com a qualidade da conexão. Será

apresentado ao ator, ao nível da *interface* de comunicação com o mesmo, um indicador de estado.

Outra mais valia será a implementação de computação paralela, tirando proveito do máximo número de *Threads*, o que quer dizer que para cada domínio web a ser analisado, todas as ferramentas estarão a trabalhar em paralelo. Desta forma, a aplicação irá analisar mais rapidamente e terá um melhor desempenho. Outra mais valia será guardar os resultados numa base de dados (por um tempo limitado) de forma a não sobrecarregar os serviços e demorar menos tempo a analisar e gerar a informação para apresentar ao utilizador.

5.3.2 Usabilidade

A *User Interface* pretende ser simples permitindo a qualquer ator, mesmo que não tenha conhecimentos sobre segurança informática, analisar um domínio web, de forma clara e concisa, apresentando ao utilizador informação sobre o progresso.

A Figura 37 e a Figura 38 apresentam a *User Interface* da aplicação.

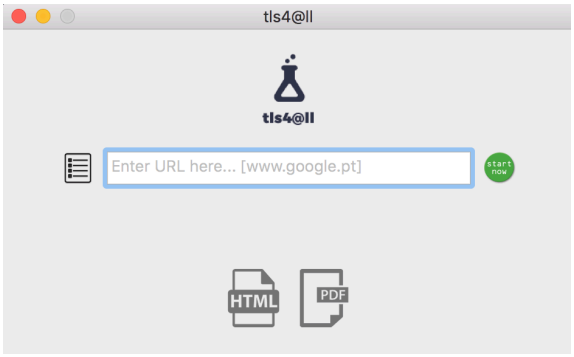


Figura 37 - User Interface da solução (1)

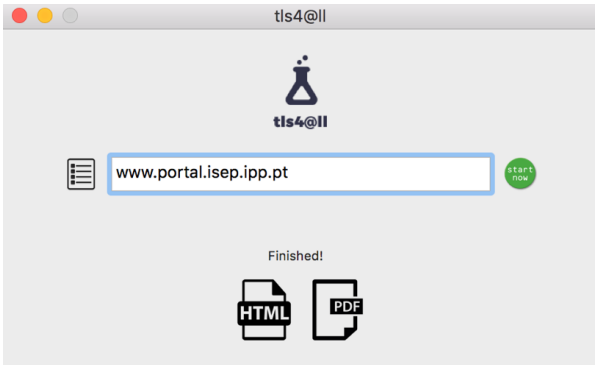


Figura 38 - User Interface da solução (2)

Na Figura 39 é possível ver a indicação do progresso.

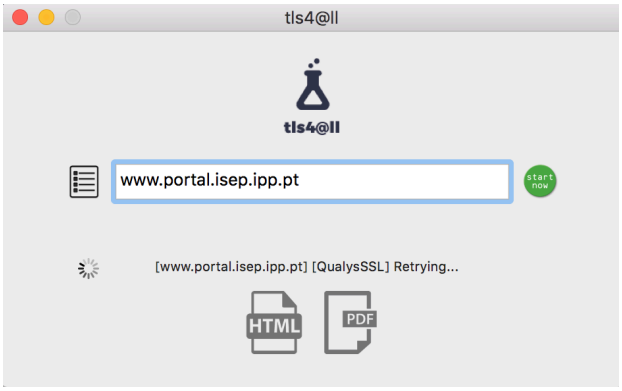


Figura 39 - Indicação de progresso

Em resumo, a Figura 40.

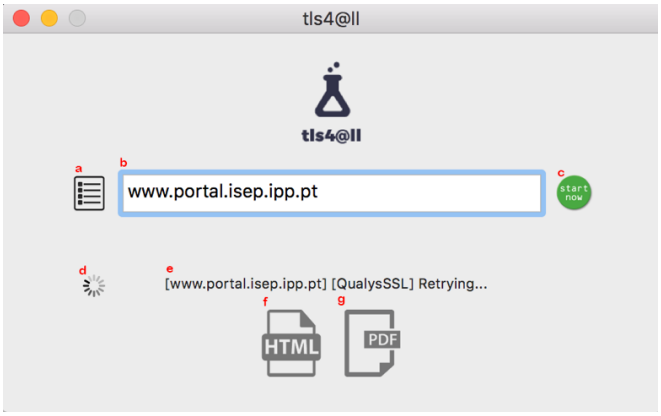


Figura 40 - Legenda da User Interface da aplicação

Em que:

a. Botão que ao clicar apresenta uma caixa de diálogo com o objetivo de carregar um ficheiro "*.json*" com domínios web a analisar;

b. Permite introduzir um domínio web, ou múltiplos domínios web, separados por ";";

c. Botão que ao clicar começa a análise dos domínios web;

d. Indicação que a aplicação está a analisar ou gerar o relatório;

e. Informação sobre o estado do progresso;

f. Botão que permite exportar o relatório para "*.html*". Só se encontra ativo no final da análise.

g. Botão que permite exportar o relatório para "*.pdf*". Só se encontra ativo no final da análise.

5.3.3 Flexibilidade

A implementação de uma nova ferramenta ou analisar um determinado parâmetro numa ferramenta já existente deverá ser bastante simples, recorrendo para isso aos padrões de software, mais especificamente a *Delegation* (subsecção 5.1.2.2).

5.3.4 Objetividade

A informação recolhida e examinada deverá gerar um relatório. A primeira parte do documento irá apresentar um resumo sobre o que foi analisado. Esta primeira parte do documento vai ser distinta em dois casos:

- Análise de apenas um domínio web (Figura 41)*;*
- Análise de múltiplos domínios web (Figura 41, Figura 42, Figura 43, Figura 44 e Figura 45).

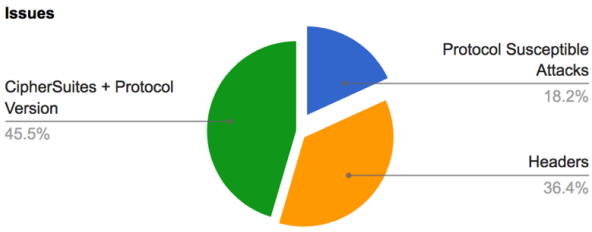


Figura 41 - Resumo de potenciais vulnerabilidades para um ou múltiplos domínios web

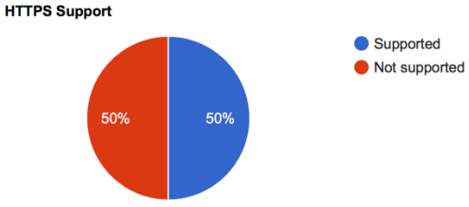


Figura 42 – Resumo do suporte a HTTPS múltiplos domínios web

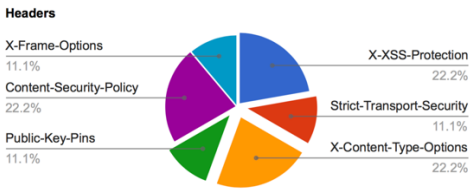


Figura 43 - Resumo de potenciais falhas nos cabeçalhos HTTP para múltiplos domínios web

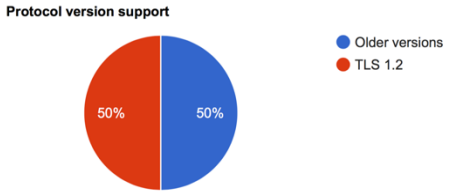


Figura 44 - Resumo do suporte a versões mais recentes do protocolo para múltiplos domínios web

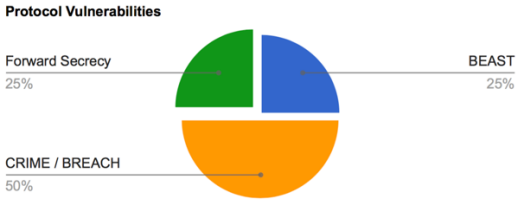


Figura 45 - Resumo potenciais vulnerabilidades a nível de protocolo para múltiplos domínios web

5.3.5 Confiabilidade

Para cada parâmetro a ser analisado pela solução, existem pelo menos 2 ferramentas que fazem a recolha e análise, de modo a garantir que os dados apresentados ao utilizador sejam fidedignos.

5.3.6 Robustez

Caso uma das ferramentas falhe a sua recolha ou análise, a solução pretendida irá continuar o seu processamento/funcionamento, apresentando no final os resultados com os dados que conseguiu recolher e analisar.

5.3.7 Uso de padrões de software

Tal como já foi referido na subsecção 5.1.2, esta solução está orientada ao baixo acoplamento e alta coesão, primando à melhor separação de responsabilidades.

A arquitetura da solução está separada por camadas, o que possibilita uma fácil manutenção e testabilidade. Mais informações sobre a arquitetura de software podem ser encontradas na secção 5.4.

5.4 Arquitetura de Software

A qualidade e longevidade de um software são determinadas pela sua arquitetura. A arquitetura de software representa uma abstração comum de um sistema que pode ser usada como base para compreensão, negociação ou até mesmo comunicação.

A arquitetura de software é uma peça fundamental para eliminar problemas de *design* ao nível de um sistema.

Cita-se uma definição de arquitetura de software (Bass et al., 2013):

The software architecture of a program or computing system is the structure or structures of the system, wich comprise software elements, the externally visible properties of those elements, and the relationships among them.

A arquitetura de software proposta tem como objetivo ser modular, conter um *design* que permita o baixo acoplamento e alta coesão. Isto significa um nível reduzido de dependência entre módulos, simplicidade na manutenção e reutilização (Martin, 2003).

A Figura 46 ilustra de um ponto de vista abstrato, a arquitetura do sistema proposto.

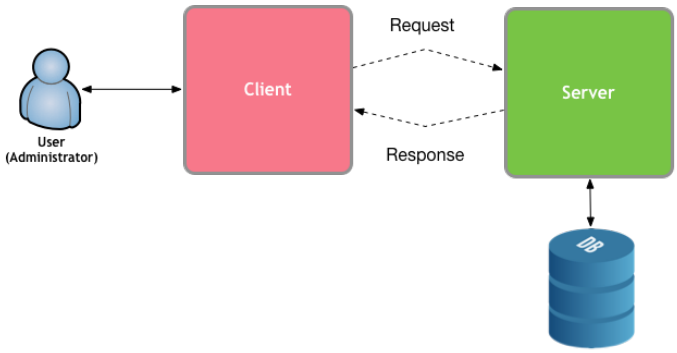


Figura 46 - Arquitetura abstrata do sistema

De um modo superficial o *Utilizador* irá interagir com um *Cliente*, que poderá ser uma página web, uma aplicação para computador ou um dispositivo móvel, por exemplo.

Esta interação consiste na inserção de um ou vários URL (página(s) web a analisar).

De seguida o *Cliente* irá fazer o pedido ao *Servidor*, no qual irá tratar de recolher e processar a informação necessária.

No fim deste processo é gerado um *Relatório* que será apresentado ao *Utilizador*.

Consegue-se elucidar com mais pormenor, o módulo *Servidor* através da Figura 47.

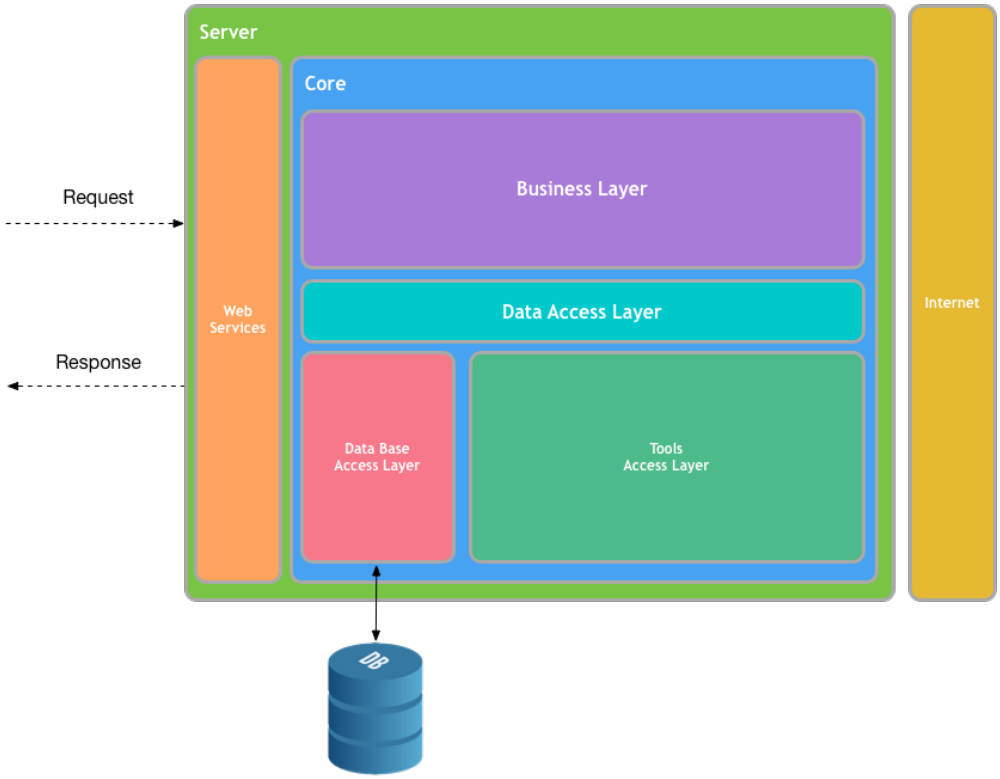


Figura 47 - Arquitetura do módulo Servidor

Este módulo (*Servidor*) tem diversos componentes:

- *Web Services*, módulo encarregue pela comunicação com o *Cliente*;
- *Core*, que engloba várias camadas:
  - Business Layer, responsável por toda a lógica de negócio e interação entre as restantes camadas;

- Data Access Layer, que abstrai de forma simples, acessos às camadas Database Access Layer e Tools Access Layer;
- Database Access Layer, incumbida de todos os procedimentos à Base de Dados;
- Tools Access Layer, camada orquestradora das várias ferramentas possíveis no sistema.

Transversalmente, temos ainda o módulo Internet.

A Figura 48 dá destaque à camada das ferramentas (*Tools Access Layer*).

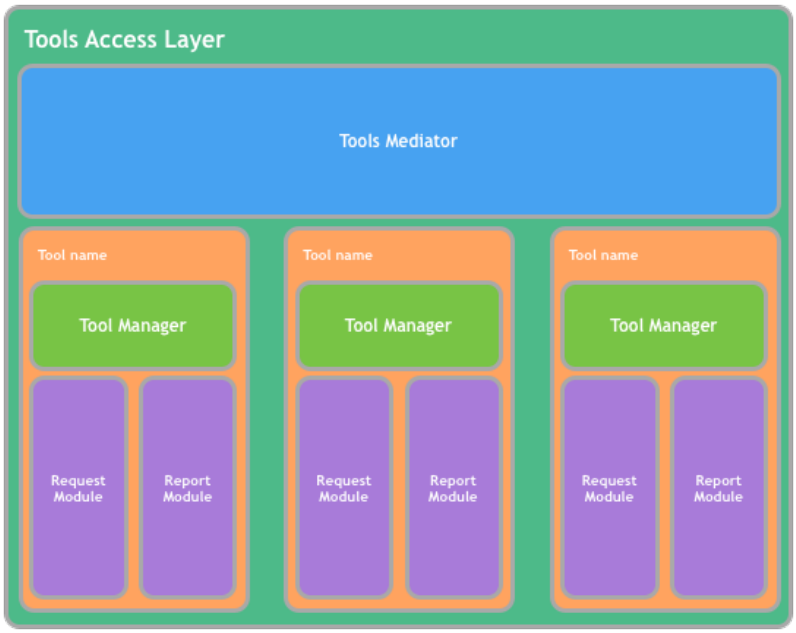


Figura 48 - Tools Access Layer

Esta camada possui um módulo *Mediador*, que irá definir protocolos às possíveis ferramentas existentes. A este módulo pertence a agregação dos *Relatórios* produzidos pelas possíveis ferramentas e conceção do modelo do *Relatório* final, que mais tarde será enviado para o módulo *Cliente*.

Cada ferramenta irá conter:

- *Tool Manager*, responsável pela comunicação com o módulo *Mediador* e interação entre o *Request Module* e *Report Module*.
- *Request Module*, cuja função será a recolha da informação necessária.
- *Report Module*, incumbido do tratamento da informação recolhida e produção de um *Relatório*, obedecendo ao protocolo previamente definido pelo *Mediador*.

Como se pode constatar, esta arquitetura apresenta-se bastante modular, tendo a hipótese por exemplo, de adicionar ou remover mais do que uma ferramenta sem que isso provoque alterações drásticas no sistema.

A Figura 49 apresenta com mais detalhe a arquitetura do módulo *Cliente*.

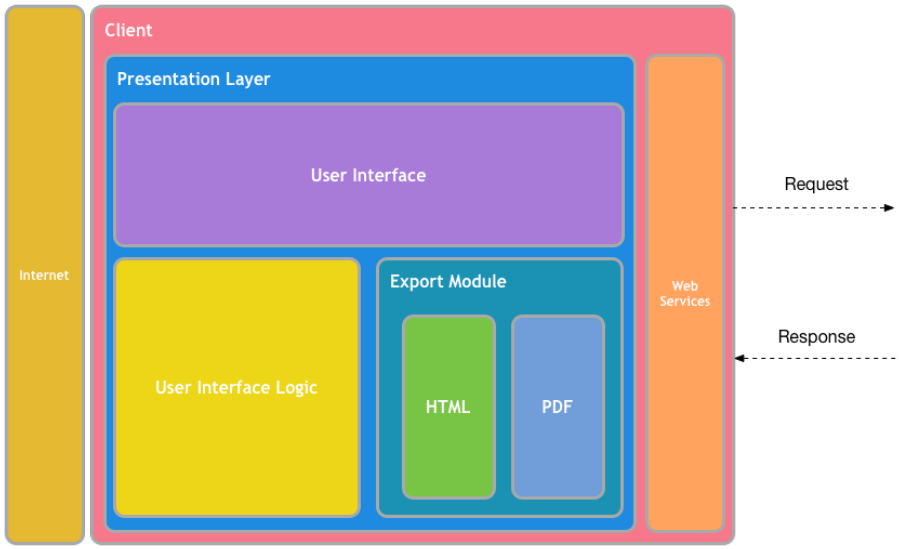


Figura 49 - Arquitetura do módulo Cliente

Este módulo (*Cliente*) tem diversos componentes:

- *Web Services*, módulo encarregue pela comunicação com o *Servidor*;
- *Presentation Layer*, que engloba várias camadas:
    - *User Interface*, responsável por mostrar informação ao utilizador;
    - *User Interface Logic* (*UI Logic*), que contém toda a lógica de apresentação;
    - *Export Module*, incumbido de transformar o modelo de relatório recebido pelo módulo *Servidor* e exportar para algo legível e de fácil leitura para o utilizador. Este módulo conta com 2 camadas, ou neste caso, 2 opções de exportação:
        - HTML;
        - PDF.

Transversalmente, temos ainda o módulo Internet.

5.5 Conclusões

Um dos objetivos principais do desenvolvimento de uma solução deste tipo é perceber o impacto das vulnerabilidades no quotidiano das pessoas.

A solução preconizada foi projetada de forma a ser modular e ter a possibilidade de reunir informação de outras soluções, com o propósito de fornecer ao utilizador um panorama mais aproximado da realidade e facultar informações úteis sobre a configuração dos serviços de forma apropriada.

Neste capítulo também se efetuou a análise de requisitos, expôs-se a arquitetura e o design da aplicação e fez-se uma breve abordagem sobre as tecnologias empregues no seu desenvolvimento.

6 Avaliação da Solução

De modo a apresentar uma ilação sobre os resultados analisados dos domínios web, é necessário existir um processo bem definido que permita a seleção, recolha e avaliação dos dados, tal como apresenta a Figura 50.

Figura 50 - Processo de seleção, recolha e análise de informação

O que traduz afinal este processo e que impacto tem?

O atual capítulo tem como foco principal apresentar o método de seleção e recolha dos dados presentes no processo, responder às perguntas anteriores e desmistificar qualquer dúvida relativamente à metodologia de avaliação.

No final serão expostos os resultados da análise feita.

6.1 Método de Seleção

Para efetuar a recolha de dados é necessária uma seleção prévia dos domínios web.

A amostra para este estudo incluirá os domínios web que os portugueses frequentam assiduamente, nomeadamente o top 300 definido pelo *Alexa*[58].

O propósito desta dissertação vai incidir sobre o top 100 da amostra descrita no parágrafo anterior.

Por forma a organizar melhor os resultados a apresentar, decidiu-se catalogar os resultados por área de negócio/tema. Desta forma é possível proteger as entidades individuais envolvidas e respeitar os termos legais associados.

Considerou-se através do *Google Trends*[59]*, Portal do Cidadão*[60], *Alexa* e *WebSimilar*[61] recolher informação sobre as categorias e/ou áreas de negócio mais relevantes para o autor, respetivamente, para catalogar os domínios web.

Do top 100 retirado do *Alexa* e confrontando os domínios web no *WebSimilar* com as respetivas categorias, decidiu-se aglutinar algumas delas de forma a não possuir categorias em excesso e não haver categorias com domínios web em número bastante reduzido.

Lista-se as categorias e/ou áreas de negócio:

- Adultos;
- Entretenimento (Artes, Animação, Humor, Celebridades, *Fashion*, TV, Jogos Online, Apostas);
- Finanças e Governo;
- Noticias;
- Outros (Trabalho, Indústria, Automóveis, Carreira e Educação, Viagens, Computadores e Eletrodomésticos).
- Shopping;
- Social (Internet e telecomunicações);

Paralelamente a este estudo será utilizada a amostra na sua totalidade para filtrar os domínios web nacionais (.pt) e efetuar uma comparação entre o top 100.

De seguida será explicado o método de recolha utilizado.

[58] http://www.alexa.com/
[59] https://www.google.pt/trends/explore
[60] https://bde.portaldocidadao.pt/evo/Services/balcaodoempreendedor/CatalogoLicencas.aspx#1
[61] https://www.similarweb.com/

6.2 Método de Recolha

Após uma análise detalhada de cada uma das ferramentas[62] e tendo em consideração a informação necessária para a recolha dos dados para efeitos de estudo (já averiguada na secção 4.2), enumera-se novamente as informações que são necessárias adquirir:

- Suporte a *HTTPS;*
- Detalhes sobre certificados, nomeadamente a sua validade, algoritmo criptográfico e chave associada, autoridade de certificação, etc.;
- Protocolos suportados, nomeadamente a sua versão;
- *Cipher suite*;
- Implementação de cabeçalhos *HTTP* críticos[63].

As próximas subsecções pretendem esclarecer quais as ferramentas quem recolhem cada parâmetro a analisar, com o propósito de demonstrar a autenticidade e veracidade dos dados.

6.2.1 Suporte a HTTPS

As ferramentas utilizadas para verificar o suporte a *HTTPS* são:

- OpenSSL
- Qualys SSL Labs API

A Figura 51 exemplifica a recolha deste parâmetro para múltiplos domínios web.

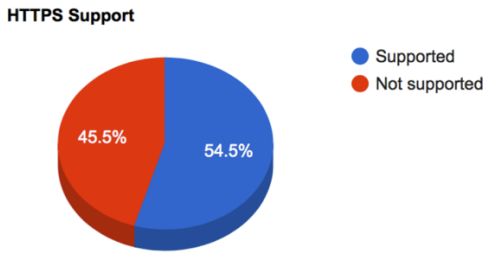


Figura 51 - Representação do suporte HTTPS para múltiplos domínios web

Já a Tabela 6 mostra todos os estados que este parâmetro pode apresentar.

[62] Para mais informação consultar a secção 4.1
[63] Para mais informação consultar a secção 2.6

Tabela 6 - Descrição de estados para o suporte HTTPS

Estado	Descrição
HTTPS: ✅ (Supported)	Domínio web suporta HTTPS
HTTPS: ❌ (Not supported)	Domínio web não suporta HTTPS
HTTPS: ⚠️ (Tools reported different results!)	As ferramentas encontram resultados diferentes
HTTPS: ❔ (Can't evaluate)	Não foi possível efetuar a análise ao parâmetro

6.2.2 Certificados

As ferramentas utilizadas para recolher informações sobre certificados são:

- OpenSSL
- Qualys SSL Labs API
- Mozilla Observatory

Enumera-se a informação que é recolhida:

- Subject
- Common Names
- Alternative Names
- Date
- Key
- Issuer
- Signature Algorithm
- Trusted

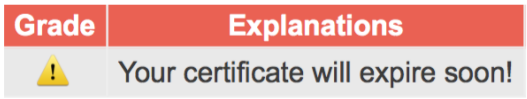
A Figura 52 apresenta a recolha efetuada para um domínio web.

1

Name	Description
Subject	[redacted]
Common Names	[redacted]
Alternative Names	[redacted]
Valid From	2016-04-04 00:00:00 +0000
Valid until	2018-04-04 12:00:00 +0000 (expires in 541d 15h 13m 0s)
Key	RSA 2048
Issuer	[redacted]
Signature Algorithm	SHA256withRSA
Trusted	Trust

Figura 52 - Representação da informação recolhida para certificados de um domínio web

Caso haja algum problema com os certificados é apresentada uma tabela no final com a devida explicação, como é exemplificado através da Figura 53.



Grade	Explanations
⚠	Your certificate will expire soon!

Figura 53 - Exemplo de um certificado cuja data de expiração é menos de 30 dias

Quando são analisados múltiplos domínios web é apresentado um gráfico com o tipo de erros existentes, como se pode verificar na Figura 54.

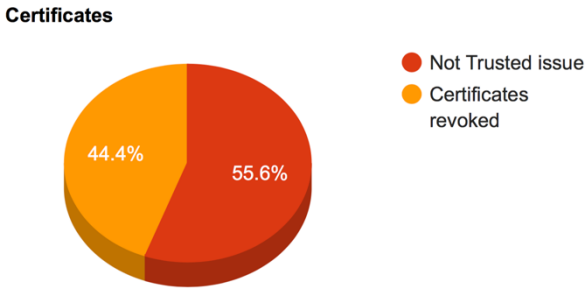


Figura 54 - Apresentação do tipo de erros nos certificados numa análise a múltiplos domínios web

6.2.3 Protocolos suportados

As ferramentas utilizadas para recolher informações sobre os protocolos suportados são:

- Qualys SSL Labs API
- Mozilla Observatory

É reunida informação sobre a versão dos protocolos suportados no domínio web.

A Figura 55 apresenta a recolha efetuada para um domínio web.

Protocol Version	Supported
TLS 1.2	false
TLS 1.1	false
TLS 1.0	true
SSL 3	false
SSL 2	false

Figura 55 - Exemplo de uma recolha aos protocolos suportados num determinado domínio web.

Se houve hipótese de melhoramento na configuração ou suporte no que toca às versões do protocolo é apresentada uma tabela com as respetivas recomendações, como se pode constatar na Figura 58.

Quando são analisados múltiplos domínios web é apresentado um gráfico, como se pode constatar através da Figura 56, com a percentagem de domínios web que suportam a versão do protocolo mais recente (TLS 1.2) e versões mais recentes.

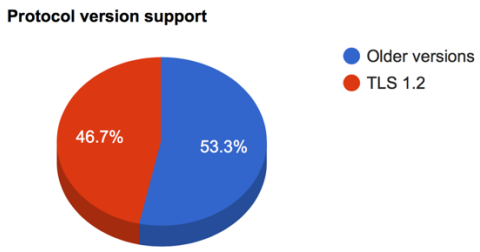


Figura 56 - Apresentação da percentagem de suporte às versões do protocolo, numa análise a múltiplos domínios web

6.2.4 Cipher Suite

As ferramentas utilizadas para recolher informações sobre os algoritmos criptográficos são:

- Qualys SSL Labs API
- Mozilla Observatory

São recolhidos todos os algoritmos criptográficos suportados pelo domínio web bem como a respetiva “força”. A Figura 57 ilustra um exemplo para um determinado domínio web.

Cipher name	Cipher Strength
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	128
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	128
TLS_RSA_WITH_AES_256_GCM_SHA384	256
TLS_RSA_WITH_AES_128_GCM_SHA256	128
TLS_RSA_WITH_AES_256_CBC_SHA256	256
TLS_RSA_WITH_AES_128_CBC_SHA256	128
TLS_RSA_WITH_AES_256_CBC_SHA	256
TLS_RSA_WITH_AES_128_CBC_SHA	128

Figura 57 - Exemplo da recolha dos algoritmos criptográficos para um determinado domínio web

Caso haja uma melhor alternativa no suporte e na configuração da *cipher suite* é apresentada uma tabela com as respetivas recomendações, como se pode constatar através da Figura 58.

Recommendations

remove ciphersuites ECDHE-RSA-AES256-SHA, ECDHE-RSA-AES128-SHA, AES256-GCM-SHA384, AES128-GCM-SHA256, AES256-SHA256, AES128-SHA256, AES256-SHA, AES128-SHA
consider adding ciphers ECDHE-ECDSA-AES256-GCM-SHA384, ECDHE-ECDSA-CHACHA20-POLY1305, ECDHE-RSA-CHACHA20-POLY1305, ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES256-SHA384, ECDHE-ECDSA-AES128-SHA256
remove protocols TLSv1, TLSv1.1
consider enabling OCSP stapling
use a certificate of type ecdsa, not RSA

Figura 58 - Exemplo de recomendações de configuração para protocolos e cipher suite

6.2.5 Cabeçalhos HTTP

As ferramentas utilizadas para recolher informações sobre os Cabeçalhos HTTP são:

- OpenSSL
- Qualys SSL Labs API
- Mozilla Observatory
- Security Headers

Cabeçalhos *HTTP* a analisar (enumerados na secção 2.6):

- HSTS
- HPKP
- CSP
- X-XSS-Protection

- X-Frame-Options
- X-Content-Type-Options

Caso a análise seja efetuada a múltiplos domínios web é apresentado um gráfico (Figura 59) especificando os cabeçalhos *HTTP* com maior impacto.

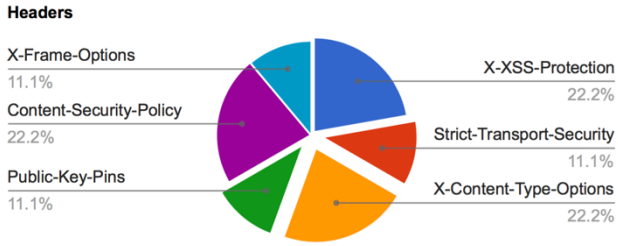


Figura 59 - Apresentação de um gráfico a ilustrar o impacto dos respetivos cabeçalhos HTTP

Para cada domínio web é mostrada uma tabela onde se verifica se os cabeçalhos *HTTP* estão a ser implementados e, caso não sejam contemplados na configuração atual é exposta informação adicional a elucidar o que são e como os configurar (Figura 60).

Headers (Implementation)

Name	Status	Explanations
Strict-Transport-Security	✅	Ok.
Content-Security-Policy	❌	Header not implemented! Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets.
Public-Key-Pins	✅	Ok.
X-Frame-Options	❌	Header not implemented! X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking. Recommended value "x-frame-options: SAMEORIGIN".
X-XSS-Protection	❌	Header not implemented! X-XSS-Protection sets the configuration for the cross-site scripting filter built into most browsers. Recommended value "X-XSS-Protection: 1; mode=block".
X-Content-Type-Options	❌	Header not implemented! X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".

Figura 60 - Exemplo da recolha dos cabeçalhos HTTP para um determinado domínio

Uma vez que são analisados os cabeçalhos *HTTP*, se existir possíveis melhoramentos é mostrada uma tabela com informação adicional, como se pode constatar na Figura 61.

Additional Info

Name	Description
X-Powered-By	X-Powered-By can usually be seen with values like "PHP/5.5.9-1ubuntu4.5" or "ASP.NET". Trying to minimise the amount of information you give out about your server is a good idea. This header should be removed or the value changed.
X-Content-Type-Options	X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".
X-AspNet-Version	X-AspNet-Version details specific information about your ASP.NET version and should be removed.
X-FRAME-OPTIONS	X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking.

Figura 61 - Informação adicional sobre os cabeçalhos HTTP para um determinado domínio

6.2.6 Potenciais Vulnerabilidades

Após a recolha indicada nas subsecções anteriores para cada domínio web, é possível reunir informação e averiguar se este está suscetível ao seguinte tipo de ataques:

- BEAST;
- CRIME / BREACH;
- POODLE;
- Heartbleed;
- FREAK;
- Logjam;
- Forward Secrecy;
- DROWN.

A Figura 62 apresenta a análise para um determinado domínio web. De notar que existe sempre informação adicional sobre a o ataque em questão e como prevenir/melhorar a atual configuração.

Name	Vulnerable	Explanations
BEAST	YES	✗ Insecure! (More info)
CRIME / BREACH	NO	✓ Ok
POODLE	NO	✓ Ok
Heartbleed	NO	✓ Ok
Freak	NO	✓ Ok
Logjam	NO	✓ Ok
Forward Secrecy	YES	✗ Insecure! (More info)
DROWN	NO	✓ Ok

Figura 62 - Exemplo da análise à suscetibilidade de ataques para um determinado domínio web

Se a análise incluir múltiplos domínios web é apresentado um gráfico (Figura 63) especificando os tipos de ataque com maior impacto.

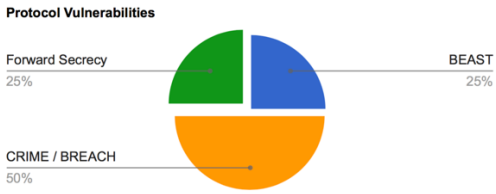


Figura 63 - Exemplo da análise à suscetibilidade de ataques para múltiplos domínios web

6.3 Método de Análise

Para poder efetuar uma análise dos dados recolhidos é crucial ter como base conhecimentos adquiridos no capítulo 2.

Como existem diferentes necessidades para cada domínio web, não é possível dizer que o "Domínio web A" ou o "Domínio web B" estejam corretamente configurados. Para isso seria necessário saber especificamente as necessidades de cada um.

Todas as soluções existentes descrevem nos termos e condições que não é possível garantir a veracidade e precisão da informação recolhida e como tal, a solução desenvolvida rege-se nos mesmos termos.

A solução desenvolvida vai recolher informações de outras ferramentas. Cada uma tem o seu próprio método de classificação dos resultados recolhidos (caso suporte classificação) e por esse motivo seria impraticável achar um método de classificação que se adaptasse a cada ferramenta (estando já implementada na solução ou não).

Pelas razões acima mencionadas, a solução irá classificar cada parâmetro com um dos seguintes estados (Figura 64):

Figura 64 - Classificação que a solução irá atribuir a cada parâmetro analisado

Os dados recolhidos vão ser classificados e avaliados de acordo com este guia porque produz uma classificação generalista e, exceto no caso em que está bem configurado, é sempre apresentada mais informação sobre a vulnerabilidade e como pode ser implementada.

Segue-se na secção seguinte as conclusões retiradas após a seleção, recolha e análise dos domínios web.

6.4 Resultados Obtidos

Esta secção apresenta os resultados adquiridos da ferramenta desenvolvida.

Com o intuito de proteger as entidades envolvidas não será divulgado o período de tempo em que a amostra foi recolhida.

Para a análise ocorrida foram reunidos os domínios web que os portugueses frequentam assiduamente, nomeadamente o top 300 definido pelo *Alexa.*

Na sua essência esta secção apresentará uma análise do top 100 e de seguida ocorrerá uma investigação aos domínios web nacionais (“.pt”), filtrados do top 300.

Por fim é ilustrada uma comparação entre o top 100 e os domínios web nacionais.

É importante salientar que cada domínio web pode conter um ou mais *hosts*, o que quer dizer que, por exemplo, para o domínio web “X”, dois *hosts* podem não ter problemas nos certificados e um ter.

A próxima subsecção expõe a investigação ao top 100.

6.4.1 Domínios Web – top 100

Esta subsecção indica a análise efetuada aos cem domínios web que os portugueses mais frequentam, já referida na secção 6.1.

A Figura 65 resume as vulnerabilidades encontradas na totalidade dos *hosts* analisados.

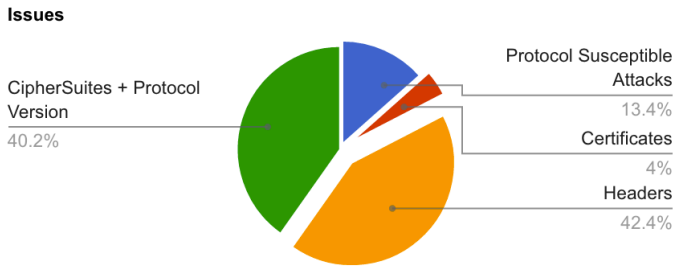


Figura 65 – Vulnerabilidades nos domínios web portugueses – top 100

Como se pode constatar a maior percentagem de incidências encontra-se ao nível dos Cabeçalhos HTTP (*Headers*) com 42,4% das vulnerabilidades encontradas. Muito próximo também são as configurações de protocolo e *cipher suite* (*Cipher Suite + Protocol Version*) com 40,2%. É alarmante o facto de 82,6% das incidências situarem-se apenas nos Cabeçalhos HTTP e nas configurações do protocolo, nomeadamente nas versões suportadas e respetiva *cipher suite*.

Relativamente ao protocolo TLS, a Figura 66 expõe a percentagem de suporte da versão mais recente (TLS 1.2) e versões antigas do protocolo TLS.

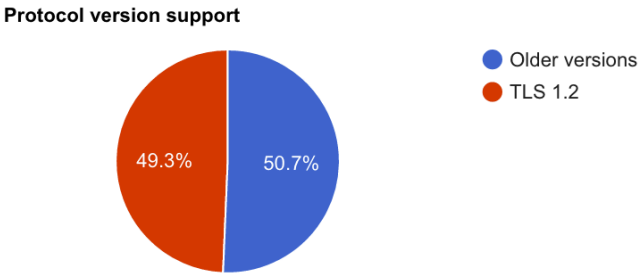


Figura 66 – Suporte às versões do protocolo TLS – top 100

Nota-se que ainda 50,7% suportam versões antigas do protocolo TLS, ou seja, versões com vulnerabilidades já identificadas.

De forma mais positiva encontra-se a configuração ao nível dos certificados (*Certificates*), apenas com 4% das ocorrências, como se revela na Figura 65.

A Figura 67 sintetiza as vulnerabilidades encontradas nos certificados.

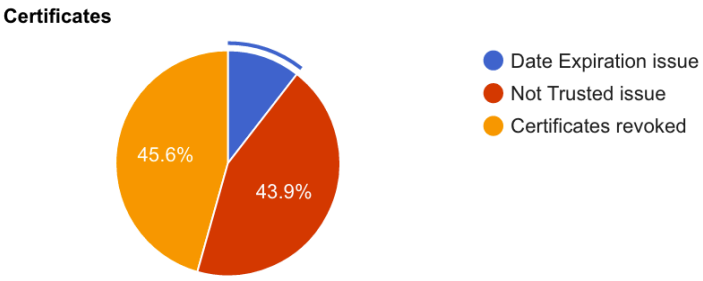


Figura 67 - Vulnerabilidades nos certificados – top 100

Cerca de 43,9% do universo total de 31 (4%) vulnerabilidades encontradas nos certificados são de fonte não fidedigna (*Not Trusted issue*) enquanto que 10,5% das vulnerabilidades estão relacionadas com a validade dos certificados (*Date Expiration issue*). Por fim, a maior percentagem de incidências encontra-se nos certificados revogados (*Certificates revoked*).

No que toca ao suporte HTTPS pode-se apurar através da Figura 68 que dos cem domínios web analisados, 47 ainda não suportam HTTPS (*Not supported*). Este dado é no mínimo preocupante, uma vez que praticamente metade dos cem domínios web mais utilizados ainda não disponibilizam uma comunicação "mais segura" com o utilizador.

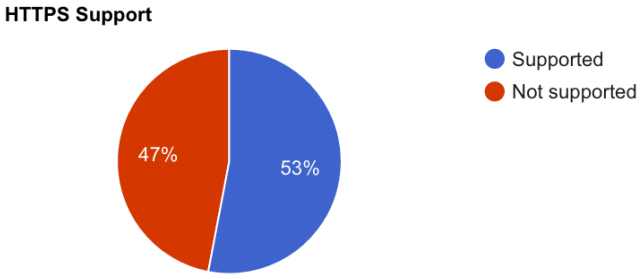


Figura 68 – Suporte HTTPS nos domínios web portugueses – top 100

Para um maior nível de detalhe, a Figura 69 representa o suporte HTTPS agrupado pelas áreas de negócio referidas na secção 6.1.

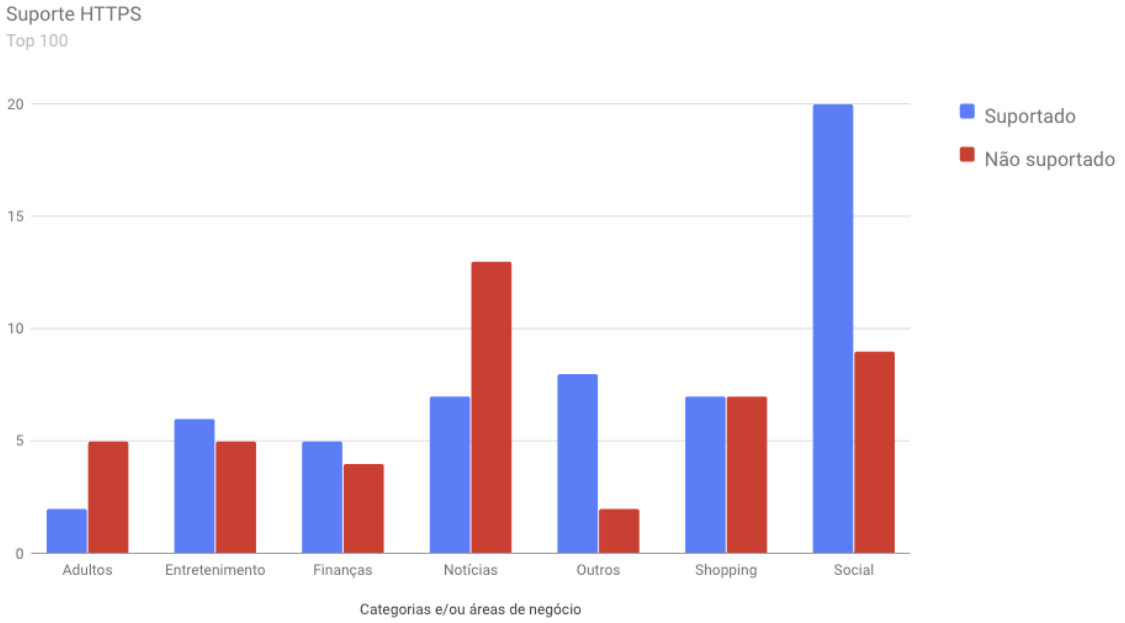


Figura 69 – Suporte HTTPS, agrupado por área de negócio – top 100

Para uma apreciação com mais rigor era necessário definir uma proporção dos respetivos domínios web e áreas de negócio. Esta figura pretende apenas dar uma ideia ao leitor da distribuição das ocorrências pelas respetivas áreas de negócio.

É de frisar que numa das áreas/categorias sensíveis como é a das Finanças e Governo, o suporte a HTTPS fica um pouco aquém das expectativas.

Mitigando as vulnerabilidades nos cabeçalhos HTTP que representam 42,4% das ocorrências, a Figura 70 apresenta com mais detalhe o resumo das vulnerabilidades encontradas.

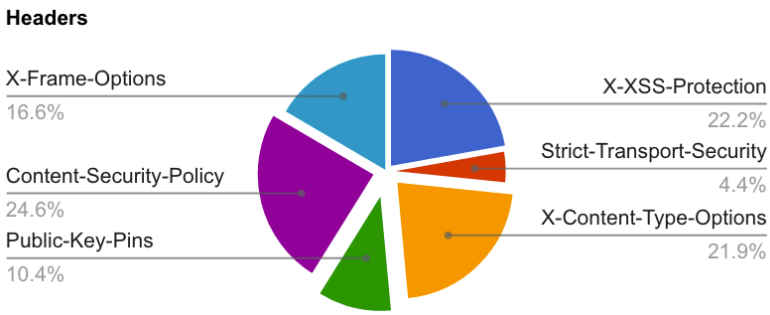


Figura 70 – Vulnerabilidades nos cabeçalhos HTTP – top 100

É de notar que 24,6% das ocorrências não implementam o cabeçalho CSP (*Content-Security-Policy*). Isto quer dizer que para um determinado domínio web não se consegue garantir que o tipo de recursos (bem como a sua proveniência) é conhecido pelo próprio domínio.

Em destaque encontra-se também a ausência de configuração do cabeçalho *X-XSS-Protection*, representado em 22,2%. Isto quer dizer que não é possível garantir, ao nível dos cabeçalhos HTTP, proteção contra os ataques do tipo *XSS*.

Para um maior nível de detalhe, a Figura 71 exibe as vulnerabilidades dos cabeçalhos HTTP agrupados pelas áreas de negócio referidas na secção 6.1.

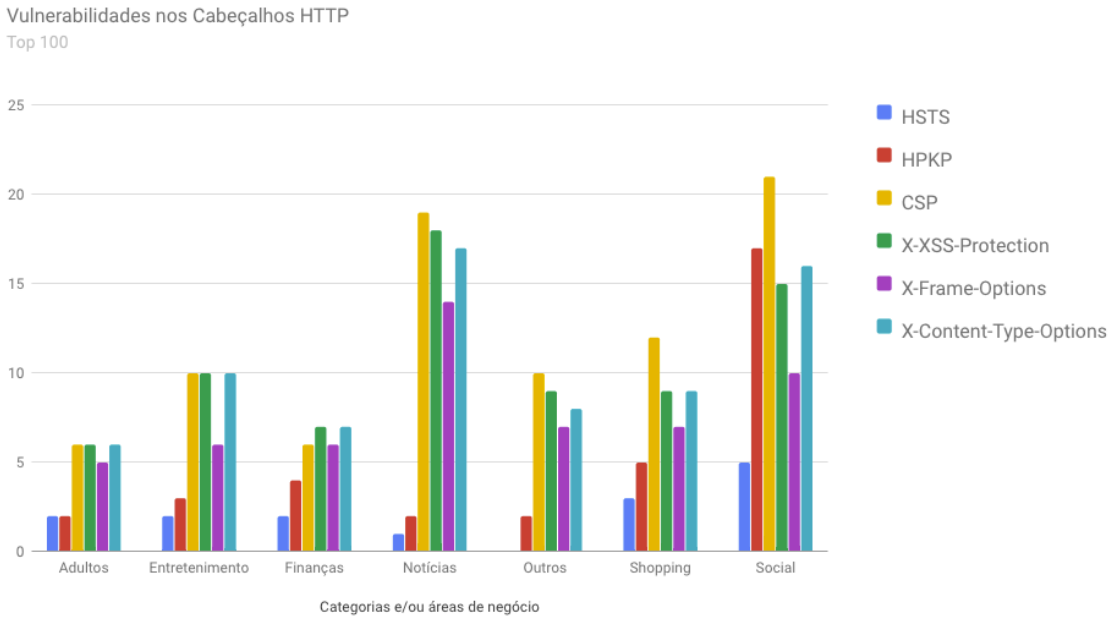


Figura 71 – Vulnerabilidades dos cabeçalhos HTTP agrupados por área de negócio – top 100

Para uma apreciação com mais rigor era necessário definir uma proporção dos respetivos domínios web e áreas de negócio. Esta figura pretende apenas dar uma ideia ao leitor da distribuição das ocorrências pelas respetivas áreas de negócio.

É no mínimo curioso verificar através da Figura 72, que no universo de 13,4% das vulnerabilidades encontradas ao nível do protocolo (Figura 65), cerca de 52,4% são referentes a ataques do tipo *CRIME* e *BREACH* e 40,5% representam ataques do tipo *BEAST*.

DROWN, *Forward Secrecy* e *POODLE* representam apenas 6,9% das vulnerabilidades encontradas.

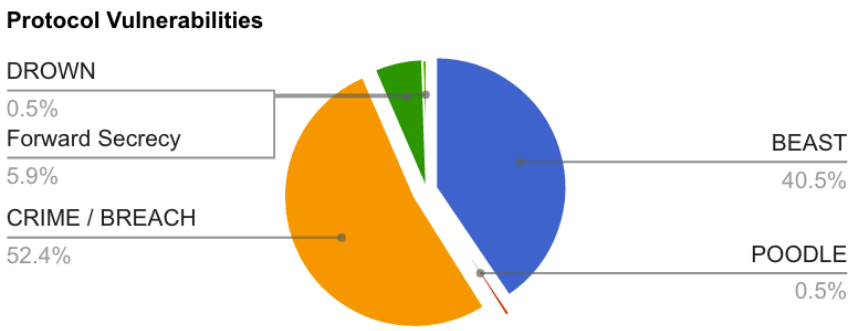


Figura 72 – Vulnerabilidades dos protocolos a ataques – top 100

Para um maior nível de detalhe a Figura 73 expõe as vulnerabilidades dos protocolos a certo tipo de ataques, agrupados por área de negócio.

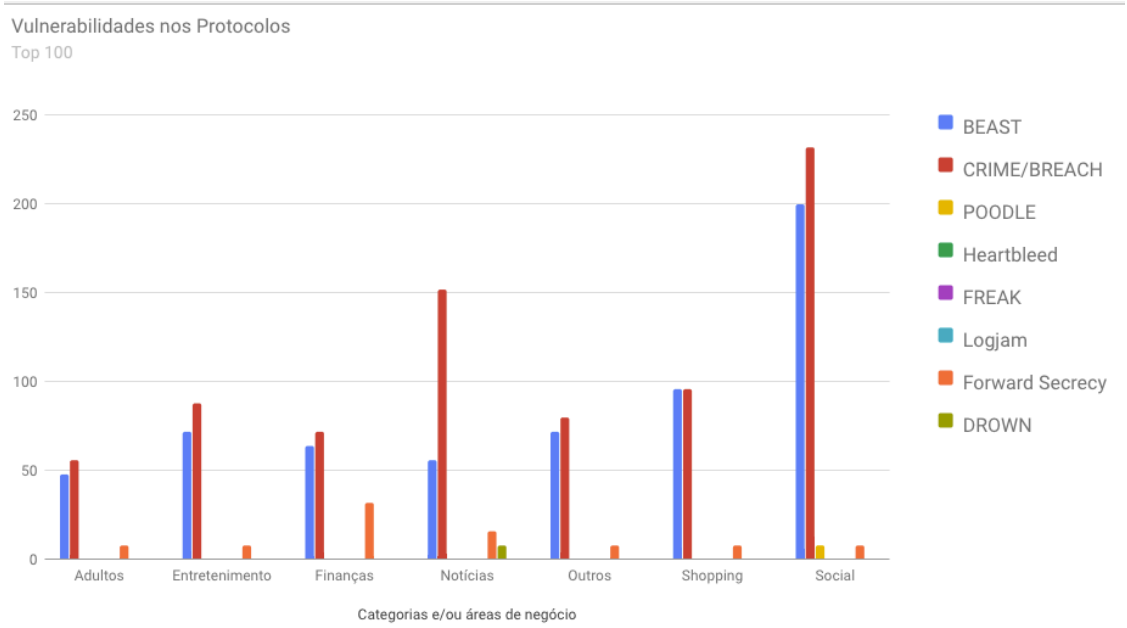


Figura 73 – Vulnerabilidades dos protocolos a ataques agrupados por área de negócio – top 100

Para uma apreciação com mais rigor era necessário definir uma proporção dos respetivos domínios web e áreas de negócio. Esta figura pretende apenas dar uma ideia ao leitor da distribuição das ocorrências pelas respetivas áreas de negócio.

Destaca-se claramente a suscetibilidade a ataques do tipo CRIME/BREACH em qualquer área de negócio. Isto quer dizer que a maior parte dos domínios web não têm desativada a compressão (secção 3.10) o que torna propícios os ataques do tipo MITM. Também os ataques do tipo BEAST estão presentes em grande escala, o que quer dizer que 40,5% das ocorrências encontram-se

em versões antigas do protocolo, nomeadamente versões inferiores a *TLS* 1.0, inclusive (subsecção 2.3.1).

De seguida apresenta-se uma breve comparação entre o top 100 já analisado e os domínios web nacionais.

6.4.2 Comparação entre o Top 100 e os Domínios Web Nacionais

Esta subsecção tem o propósito de comparar o estudo já realizado na subsecção anterior e os domínios web portugueses (".pt") da amostra já mencionada na secção 6.1.

Da amostra total de 300 domínios web foram filtrados 88 domínios ".pt".

A Figura 74 compara as vulnerabilidades encontradas na totalidade dos *hosts* analisados.

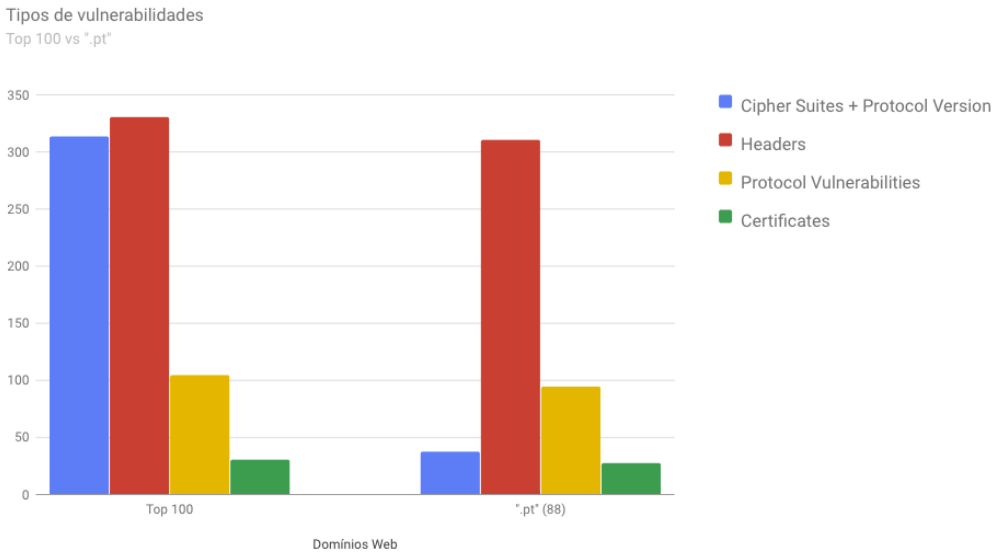


Figura 74 – Comparação das vulnerabilidades encontradas

Para uma apreciação com mais rigor era necessário definir uma proporção dos respetivos domínios web e áreas de negócio. Esta figura pretende apenas dar uma ideia ao leitor da distribuição das ocorrências pelos respetivos domínios web.

É de realçar que tanto no top 100 como nos domínios web portugueses, os certificados (*Certificates*), as vulnerabilidades ao nível do protocolo (*Protocol Vulnerabilities*) e os cabeçalhos HTTP (*Headers*) estão equivalentes. Existe uma discrepância positiva no suporte das versões do protocolo bem como a *cipher suite* associada.

De forma a tirar melhores ilações a Figura 75 representa o suporte às versões do protocolo *TLS*.

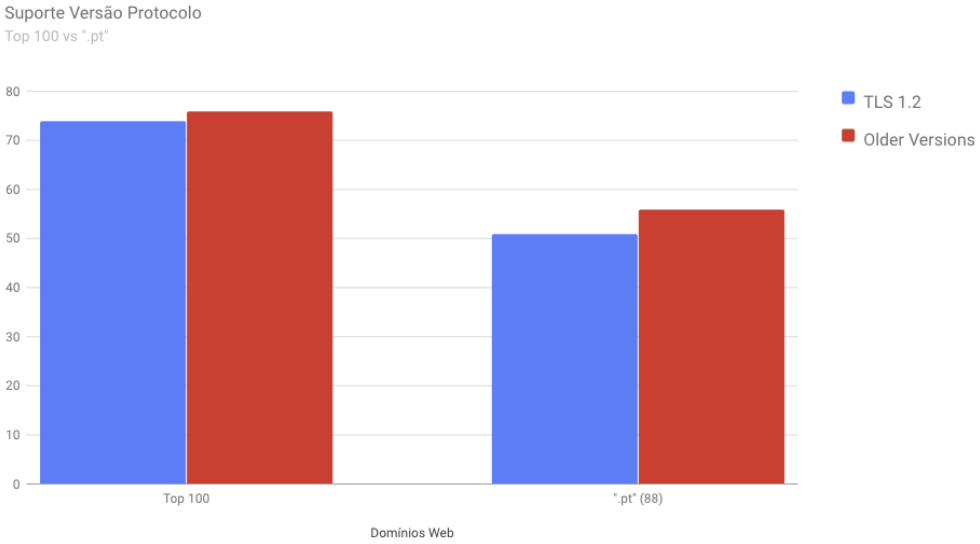


Figura 75 – Comparação do suporte de versões do protocolo TLS

Como não existe nenhum domínio web que suporte apenas a versão TLS 1.2, a figura anterior representa também o suporte à versão TLS 1.2, ou seja, pode haver domínios web que estejam expostos com a cor azul e com a cor vermelha.

Explicada esta apresentação pode-se verificar que os grupos dos domínios web mantêm-se equivalentes no suporte das versões, ou seja, ainda existem muitos domínios web que suportam versões antigas dos protocolos. O ideal seria aumentar a barra azul e diminuir a barra vermelha, para ambos os casos.

De frisar que ao nível dos certificados, os *hosts* portugueses têm mais vulnerabilidades que os *hosts* do top 100, como ilustra a Figura 76

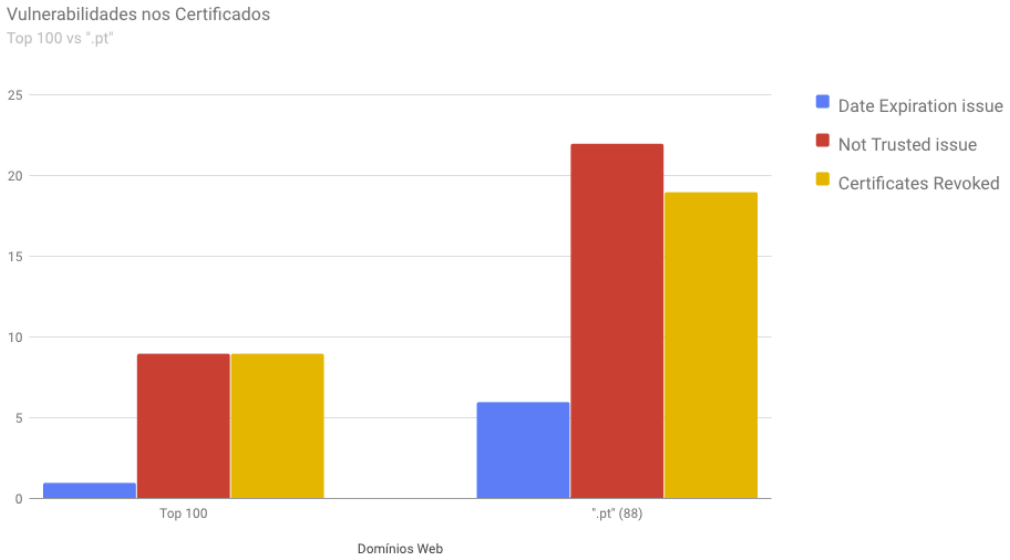


Figura 76 – Comparação das vulnerabilidades nos certificados

Pode-se atestar através desta figura que do universo analisado, existem poucos certificados expirados (*Date Expiration issue*) e grande parte dos problemas residem nos certificados não fidedignos (*Not trusted issue*) e nos certificados revogados (*Certificates Revoked*).

Para finalizar, destaca-se o suporte *HTTPS* através da Figura 77.

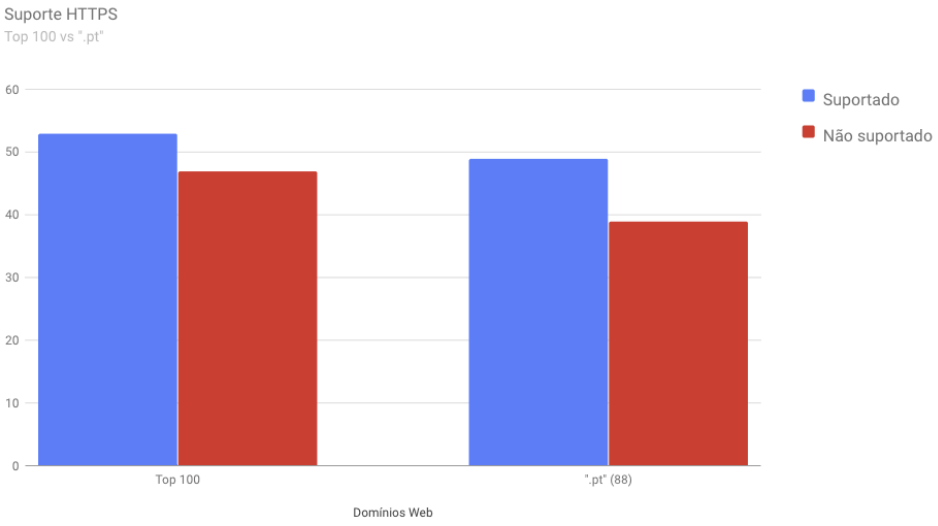


Figura 77 – Comparação do suporte HTTPS

De notar que tanto no top 100 como nos domínios web portugueses, o suporte *HTTPS* fica muito aquém do esperado.

6.5 Conclusões

Em resumo pode-se concluir que os domínios web que os portugueses navegam assiduamente encontram-se com bastantes vulnerabilidades e não se consegue garantir que a partilha de informação seja efetuada de forma segura entre as duas partes envolvidas.

É possível que não haja consciência de todos os administradores de sistemas para esta situação, mas deve-se dar mais visibilidade para este problema. Os profissionais devem-se manter atualizados por forma a fornecer um suporte e qualidade de serviço acima da média.

7 Conclusões

Neste capítulo é apresentada uma breve conclusão do projeto, fazendo um ponto de situação final e identificando se os objetivos inicialmente definidos foram alcançados ou não. Serão apresentadas as limitações e o trabalho futuro, acabando por frisar a preocupação e sugestões do autor relativamente a este tema, tanto para quem configura e disponibiliza os serviços via TLS quer para quem os utiliza.

7.1 Ponto de situação final

Estamos numa era em que a comunicação através da Internet prevalece e estabelece-se com predominância a cada dia que passa e ao longo dos últimos anos têm sido reportadas vulnerabilidades de segurança informática relacionada com o protocolo TLS, que provocam desconforto e insegurança aos utilizadores que utilizam os serviços disponíveis, principalmente quando há troca de informação sensível como é o caso de troca de credenciais nas transações bancárias ou quando se acede ao correio eletrónico.

Embora existam boas práticas recomendadas para a configuração segura destes protocolos e soluções conhecidas para diagnosticar configurações, por vezes não estão implementadas nos serviços oferecidos aos utilizadores. Há pouca divulgação nos meios sociais sobre o assunto e a falta de visibilidade leva a um desleixo/ignorância por parte de quem disponibiliza os serviços ao cliente.

Desta forma a presente dissertação procurou investigar o estado da arte à volta dos protocolos TLS e tentar perceber como funciona a comunicação numa ligação "segura". Identificou-se possíveis boas práticas de configuração a implementar nos serviços oferecidos aos utilizadores e expôs-se algumas das vulnerabilidades que o autor considerou importantes de forma a dar a entender ao leitor que para além das 11 vulnerabilidades/ataques descritos de forma breve, existem muitos mais e com a evolução e lançamento de novas tecnologias, também são aprimorados os ataques e novas oportunidades para os atacantes são reveladas.

Para investigar o quão expostos estão os portugueses relativamente às vulnerabilidades descritas ao longo desta dissertação, foram estudadas ferramentas que ajudassem a identificar os problemas já mencionados e recolher informações para executar uma análise e apresentar os resultados obtidos.

Para facilitar a recolha e análise dessa informação o autor arquitetou e implementou uma solução modular que pudesse integrar outras ferramentas já examinadas com a mais valia da criação de um relatório para um ou mais domínios web a examinar.

Propôs-se um método de seleção, recolha e análise para avaliar a exposição dos portugueses aos domínios web que frequentam assiduamente e chegou-se à conclusão que não é possível garantir a partilha de informação de forma segura entre as partes envolvidas.

É importante reforçar e alertar não só os portugueses como a população em geral para tomarem consciência e mais cuidado quando navegam na Internet.

Os administradores de sistemas ou quem configura e disponibiliza os serviços via TLS, devem manter-se em constante atualização com o intuito de fornecer aos seus clientes um serviço de qualidade e sobretudo mais seguro na troca de informação sensível. Tal como o autor dissertou na secção 4.1, existem ferramentas que ajudam a despistar eventuais más configurações nos protocolos TLS e a solução que o autor concebeu é uma mais valia para todos os interessados, uma vez que reúne informações sobre as outras ferramentas, analisa os dados, expõe os resultados obtidos e ainda apresenta boas práticas nas configurações destes protocolos.

Existem também boas práticas para os utilizadores quando navegam na Internet, para além do cuidado e especial atenção ao que é apresentado nos domínios web. Há diversas extensões e *plugins* para o *browser* que forçam, por exemplo, a se um determinado domínio web suportar a ligação HTTPS, comunicar-se sempre desta forma e não por HTTP. São comportamentos simples que fazem a diferença entre dar as credenciais de acesso bancárias aos atacantes, ou manter a informação sensível alheia a pessoas mal-intencionadas.

7.2 Objetivos

Esta dissertação tinha como objetivos:

- Visar o estado da arte dos protocolos TLS;
- Propor e desenvolver métodos para a coleta de informações, que possibilitam a análise dos dados recolhidos considerando as características de segurança avaliadas;
- Responder à questão: “Quão expostos estão os domínios web que os portugueses utilizam assiduamente?”.

É possível assegurar que todos os objetivos propostos foram alcançados com sucesso, como se pode constatar ao longo da dissertação e também descrito no ponto de situação final de forma mais sintetizada.

7.3 Limitações e trabalho futuro

Ao longo do desenvolvimento da dissertação, quer na escrita do documento, como na implementação da solução, o autor deparou-se com algumas limitações e dificuldades no desenrolar do percurso.

A vida profissional teve um impacto (negativo) colossal e consumiu tempo que o autor gostaria de despender no desenvolvimento no global da presente dissertação.

No desenvolvimento da solução o autor procurou usar várias tecnologias em *JavaScript* para evoluir o seu conhecimento e aprender uma nova linguagem de programação, mas a curva de aprendizagem era demasiado grande e devido à vida profissional, o autor optou por uma abordagem mais eficiente e eficaz.

Para trabalho futuro:

- Divulgar e alertar sobre este tema nas redes sociais e se possível, nos meios de comunicação tradicionais;

- Efetuar uma recolha pontual (em cada mês, por exemplo) ao longo de 6 meses ou 1 ano e executar uma comparação temporal, de modo a investigar se surge uma adoção de boas práticas na configuração dos serviços;

- Desenvolver uma extensão/*plug-in* para os *browsers* em que seja possível aos utilizadores com pouco conhecimento sobre o tema, verificar se o domínio web que estão a navegar está vulnerável a certos tipos de ataques. Esta extensão pode e deve utilizar a solução preconizada através dos *Web Services* disponíveis na arquitetura proposta;

- Adicionar o teste de simulação de *Handshake* por *User Agent* à solução implementada.

Referências

Apple Inc., 2015a. Model-View-Controller [WWW Document]. URL https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html (accessed 10.7.16).

Apple Inc., 2015b. Protocols [WWW Document]. URL https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Protocol.html#//apple_ref/doc/uid/TP40008195-CH45-SW1 (accessed 10.7.16).

Apple Inc., 2015c. Singleton [WWW Document]. URL https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/Singleton.html (accessed 10.7.16).

Aviram, N., Schinzel, S., Somorovsky, J., 2016. DROWN: Breaking TLS using SSLv2.

Barnes, R., Thomson, M., Pironti, A., Langley, A., 2015. Deprecating Secure Sockets Layer Version 3.0 (No. RFC7568). RFC Editor.

Bass, L., Clements, P., Kazman, R., 2013. Software architecture in practice, 3rd ed. ed, SEI series in software engineering. Addison-Wesley, Upper Saddle River, NJ.

Benchmarks for the Top Server-Side Swift Frameworks vs. Node.js [WWW Document], 2016. . Medium. URL https://medium.com/@rymcol/benchmarks-for-the-top-server-side-swift-frameworks-vs-node-js-24460cfe0beb#.99z4n3ena (accessed 10.7.16).

Berners-Lee, T., Fischetti, M., 1999. Weaving the Web: the original design and ultimate destiny of the World Wide Web by its inventor, 1st ed. ed. HarperSanFrancisco, San Francisco.

Blaze, M., Ioannidis, J., Keromytis, A., 2000. DSA and RSA Key and Signature Encoding for the KeyNote Trust Management System (No. RFC2792). RFC Editor.

Carnevale, P.J., Pruitt, D.G., 1992. Negotiation and Mediation. Annu. Rev. Psychol. 43, 531–582. doi:10.1146/annurev.ps.43.020192.002531

Claessens, J., Dem, V., De Cock, D., Preneel, B., Vandewalle, J., 2002. On the Security of Today's Online Electronic Banking Systems. Comput. Secur. 21, 253–265. doi:10.1016/S0167-4048(02)00312-7

CMU Software Engineering Institute, 2008. Vulnerability Note VU#836068 - MD5 vulnerable to collision attacks [WWW Document]. URL https://www.kb.cert.org/vuls/id/836068 (accessed 10.19.16).

Codenomicon, 2014. Heartbleed Bug [WWW Document]. URL http://heartbleed.com/ (accessed 2.14.16).

Eastlake, D., Jones, P., 2001. US Secure Hash Algorithm 1 (SHA1) (No. RFC3174). RFC Editor.

Edward Jiang, 2016. Swift on the Server - Where Are We Today [WWW Document]. URL https://stormpath.com/blog/swift-on-the-server-today (accessed 10.7.16).

Eronen, P., Tschofenig, H., 2005. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS) (No. RFC4279). RFC Editor.

Evans, C., Palmer, C., Sleevi, R., 2015. Public Key Pinning Extension for HTTP (No. RFC7469). RFC Editor.

Feather, J., 2013. The information society: a study of continuity and change, 6. ed. ed. Facet Publ, London.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., 1999. Hypertext Transfer Protocol -- HTTP/1.1 (No. RFC2616). RFC Editor.

Filzmoser, M., Vetschera, R., 2008. A Classification of Bargaining Steps and their Impact on Negotiation Outcomes. Group Decis. Negot. 17, 421–443. doi:10.1007/s10726-008-9106-1

Freed, N., Borenstein, N., 1996. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies (No. RFC2045). RFC Editor.

Fu, D., Solinas, J., 2007. IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA) (No. RFC4754). RFC Editor.

Grigorik, I., 2013. High-performance browser networking: [what every web developer should know about networking and browser performance], 1. ed. ed. O'Reilly, Beijing.

Hodges, J., Jackson, C., Barth, A., 2012. HTTP Strict Transport Security (HSTS) (No. RFC6797). RFC Editor.

International Organization for Standardization, 1994. Information Technology. Open Systems Interconnection. Basic Reference Model. Conventions For The Definition Of OSI Services.

Jobber, D., Lancaster, G., 2000. Selling & sales management, 5th ed. ed. Prentice Hall, Harlow.

Kaliski, B., Staddon, J., 1998. PKCS #1: RSA Cryptography Specifications Version 2.0 (No. RFC2437). RFC Editor.

Kaufman, J., 2012. The personal MBA master the art of business. Portfolio/Penguin, New York, N.Y.

Martin, R.C., 2003. Agile software development: principles, patterns, and practices, Alan Apt series. Prentice Hall, Upper Saddle River, N.J.

McGrew, D., Igoe, K., Salter, M., 2011. Fundamental Elliptic Curve Cryptography Algorithms (No. RFC6090). RFC Editor.

Merkle, R.C., 1978. Secure communications over insecure channels. Commun. ACM 21, 294–299. doi:10.1145/359460.359473

Mike West, 2016. Content Security Policy Level 3 [WWW Document]. URL

https://www.w3.org/TR/CSP3/ (accessed 10.9.16).

Nicola, S., Ferreira, E., J. J. Pinto Ferreira, 2014. A Quantitative Model for Decomposing & Assessing the Value for the Customer.

Nicola, S., Pinto Ferreira, E., Pinto Ferreira, J.J., 2012. A Novel Framework For Modelling Value For The Customer, An Essay On Negotiation. Int. J. Inf. Technol. Decis. Mak.

Osterwalder, A., Pigneur, Y., Bernarda, G., Smith, A., 2014. Value proposition design: how to create products and services customers want, Strategyzer series. John Wiley & Sons, Hoboken.

Osterwalder, A., Pigneur, Y., Clark, T., 2010. Business model generation: a handbook for visionaries, game changers, and challengers. Wiley, Hoboken, NJ.

OWASP Secure Headers Project - OWASP [WWW Document], 2016. URL https://www.owasp.org/index.php/OWASP_Secure_Headers_Project (accessed 10.8.16).

PORDATA, 2016. Assinantes do acesso à Internet [WWW Document]. URL http://www.pordata.pt/DB/Portugal/Ambiente+de+Consulta/Gr%C3%A1fico (accessed 2.14.16).

Quotes, 2016. Forward Secrecy [WWW Document]. URL https://www.forwardsecrecy.com/

Rackham, N., DeVincentis, J.R., 1999. Rethinking the sales force: redefining selling to create and capture customer value. McGraw-Hill, New York.

Rescorla, E., 1999. Diffie-Hellman Key Agreement Method (No. RFC2631). RFC Editor.

Rescorla, E., Ray, M., Dispensa, S., Oskov, N., 2010. Transport Layer Security (TLS) Renegotiation Indication Extension (No. RFC5746). RFC Editor.

RFC 7457 - Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS) [WWW Document], n.d. URL https://tools.ietf.org/html/rfc7457 (accessed 6.5.16).

Ritter, T., 2012. Details on the “Crime” Attack [WWW Document]. URL https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2012/september/details-on-the-crime-attack/

Rivest, R., 1992. The MD5 Message-Digest Algorithm (No. RFC1321). RFC Editor.

Rivest, R.L., Shamir, A., Adleman, L., 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 120–126. doi:10.1145/359340.359342

Ross, D., Gondrom, T., 2013. HTTP Header Field X-Frame-Options (No. RFC7034). RFC Editor.

Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C., 2013. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP (No. RFC6960). RFC Editor.

Schaad, J., Housley, R., 2002. Advanced Encryption Standard (AES) Key Wrap Algorithm (No. RFC3394). RFC Editor.

Scott Helme, 2015. Hardening your HTTP response headers [WWW Document]. URL https://scotthelme.co.uk/hardening-your-http-response-headers/ (accessed 10.9.16).

Sheffer, Y., Holz, R., Saint-Andre, P., 2015. Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS) (No. RFC7457). RFC Editor.

SSL and TLS Deployment Best Practices [WWW Document], 2016. URL https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices (accessed 10.22.16).

Stack Overflow Developer Survey 2016 Results [WWW Document], 2016. URL https://stackoverflow.com/research/developer-survey-2016 (accessed 10.7.16).

StarForce Technologies, 2014. What is a man-in-the-middle attack? [WWW Document]. URL https://sfletter.com/?_lng=en&_action=blog-08-12-2014 (accessed 2.14.16).

Symantec, 2016. What is SSL, TLS and HTTPS? | Symantec [WWW Document]. URL https://www.symantec.com/page.jsp?id=ssl-information-center (accessed 2.2.16).

Symantec, 2014. How does SSL work? What is an SSL handshake? [WWW Document]. URL http://www.symantec.com/connect/blogs/how-does-ssl-work-what-ssl-handshake (accessed 2.1.16).

Syme, M., Goldie, P., 2004. Optimizing network performance with content switching: server, firewall, and cache load balancing, Prentice Hall series in computer networking and distributed systems. Prentice Hall, Upper Saddle River, NJ.

Turner, S., Polk, T., 2011. Prohibiting Secure Sockets Layer (SSL) Version 2.0 (No. RFC6176). RFC Editor.

Ulaga, W., Eggert, A., 2006. Value-Based Differentiation in Business Relationships: Gaining and Sustaining Key Supplier Status. J. Mark. 70, 119–136. doi:10.1509/jmkg.2006.70.1.119

Von Mises, L., Batson, H.E., 2009. The theory of money and credit. Signalman Pub., Orlando.

Vranken, G., 2015. HTTPS Bicycle Attack [WWW Document]. URL https://guidovranken.wordpress.com/2015/12/30/https-bicycle-attack/ (accessed 1.9.16).

Web Scraping Explained [WWW Document], n.d. URL https://www.webharvy.com/articles/what-is-web-scraping.html (accessed 10.5.16).

Weigand, E., Dascal, M., International Conference on Pragmatics and Negotiation (Eds.), 2001. Negotiation and power in dialogic interaction: [papers presented at the International Conference on Pragmatics and Negotiation at Tel Aviv University and the Hebrew University of Jerusalem in June, 1999], Amsterdam studies in the theory and history of linguistic science Series 4, Current issues in linguistic theory. Benjamins, Amsterdam.

APÊNDICES

APÊNDICE A - Análise de valor

É muito importante concretizar uma análise de valor por forma a enaltecer o produto que estamos a idealizar.

Eis uma citação de Von Mises (Von Mises and Batson, 2009):

Value it is the way in wich man reacts to the conditions of his environment.

As próximas secções pretendem explicar a importância de uma proposta de valor, definição, valor percebido e possíveis cenários de negócio.

Necessidade de uma proposta de valor

Qualquer negócio de sucesso cria valor (Kaufman, 2012). O mundo está cheio de oportunidades para tornar a qualidade de vida das pessoas melhor, em alguma maneira. É necessário ser-se perspicaz para entender o que as pessoas precisam e fornecer algo que preencha essa lacuna.

A criação de valor é um conceito difícil de alcançar, perceber e conceptualizar. Alguns autores consideram que a criação de valor é uma troca entre benefícios e sacrifícios percebidos pelos clientes durante a oferta de um fornecedor (Jobber and Lancaster, 2000).

Numa perspetiva de negócio, eis alguns exemplos de valor para vender ao cliente:

Segue-se também alguns exemplos de valor que é vendido como negócio:

Claro que com isto, para o mesmo produto os clientes podem ter diversas perceções quanto ao valor. Além disso, as organizações envolvidas no processo de compra podem ter diferentes perceções de oferta de valor para o cliente (Ulaga and Eggert, 2006).

O valor percebido varia entre benefícios e sacrifícios ao nível do produto, serviço e relacionamento.

Ao nível do produto temos por exemplo, a qualidade (benefício). Já no serviço, espera-se confiança (benefício), sendo o preço um sacrifício. No relacionamento, temos a imagem (benefício) e tempo/esforço como sacrifícios.

Uma proposta de valor é uma visão global, uma apresentação, do leque de produtos e serviços que são o valor para o cliente (Osterwalder et al., 2014). Uma proposta de valor bem definida responde às seguintes questões:

1. Qual é o produto?
2. Que valor fornece?
3. Para quem é que se vai fornecer valor?
4. Porquê que é único?

Neil Rackman (Rackham and DeVincentis, 1999), acredita que uma proposta de valor deve consistir essencialmente em quatro partes: capacidade, impacto, resistência e custo.

Resume-se, portanto, que é necessário um modelo de negócio sofisticado para desenvolver uma proposta de valor bem definida.

Valor (benefícios/sacrifícios) que esta dissertação vai acrescentar

Por forma a descrever o valor desta dissertação, é essencial identificar e definir uma proposta de valor.

A presente dissertação pretende fornecer um método, um processo de recolha, tratamento e análise sobre o estado da arte dos domínios *web* (portugueses) que operam com os protocolos *SSL/TLS* para disponibilizar os seus serviços. Esta solução irá auxiliar fundamentalmente os Administradores de Sistemas para elucidar eventuais ameaças com as configurações atuais e pontos a melhorar. Tem a mais valia da capacidade de agregar outras ferramentas e validar os dados recolhidos.

A próxima tabela apresenta o valor traduzido em alguns benefícios e sacrifícios da solução preconizada.

Benefícios	**Sacrifícios**
Confiança (resultados obtidos)	Custo
Flexibilidade (adicionar outras ferramentas)	
Melhor segurança (serviços prestados)	
Credibilidade (negócio)	

Cenários de negócio

Qualquer atividade de negócio é intrinsecamente sobre negociação. É sobre a entrega de um bem ou serviço tangível e intangível em que o seu valor é recompensado.

Weigand (Weigand et al., 2001) define negociação partindo do princípio em que existem dois ou mais participantes numa interação, com objetivos individuais, o que pode ser parcialmente incompatível. É recorrente no processo de negociação, investigar alternativas de modo a que seja mutualmente acordado um resultado aceitável entre os intervenientes.

Eis uma citação de Filzmoser e Vetschera (Filzmoser and Vetschera, 2008):

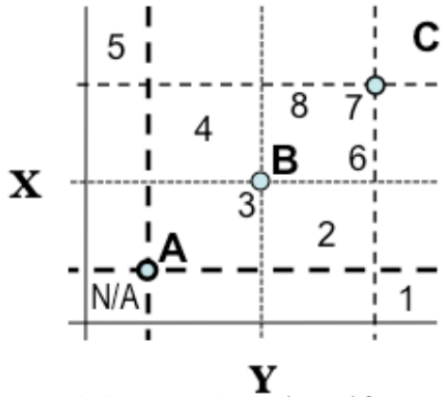
Negotiations are dynamic processes in wich the parties involved communicate to exchange offers, make concessions, raise threats, or otherwise influence each other in order to reach na agreement.

Existem fundamentalmente dois tipos de negociação: distributiva e integrativa.

Numa negociação distributiva as partes competem pela distribuição de uma soma fixa de valor, como por exemplo, a compra de produtos ou serviços. Normalmente associa-se a uma situação "*win-lose*", em que uma das partes ganha e a outra perde.

Já uma negociação integrativa pressupõe a existência de uma ou mais soluções para o problema em que ambas as partes cooperam entre si para obter o máximo possível de benefícios, originando uma situação "*win-win*".

Para *Carnevale* (Carnevale and Pruitt, 1992), existem quatro modos para atingir um acordo:



(Carnevale and Pruitt, 1992)

1. Não chegar a acordo, representado na figura por "N/A";
2. O acordo favorece apenas umas das partes sem compensação para as restantes, representado por "1" ou "5"; (negociação distributiva)
3. Apenas um compromisso, representado por "2", "3" ou "4";
4. O acordo favorece ambas as partes (negociação integrativa), representado por "6", "7" e "8".

As letras representadas por "A", "B" ou "C" transmitem os limites tangíveis ou intangíveis das partes envolvidas.

Modelo de Canvas

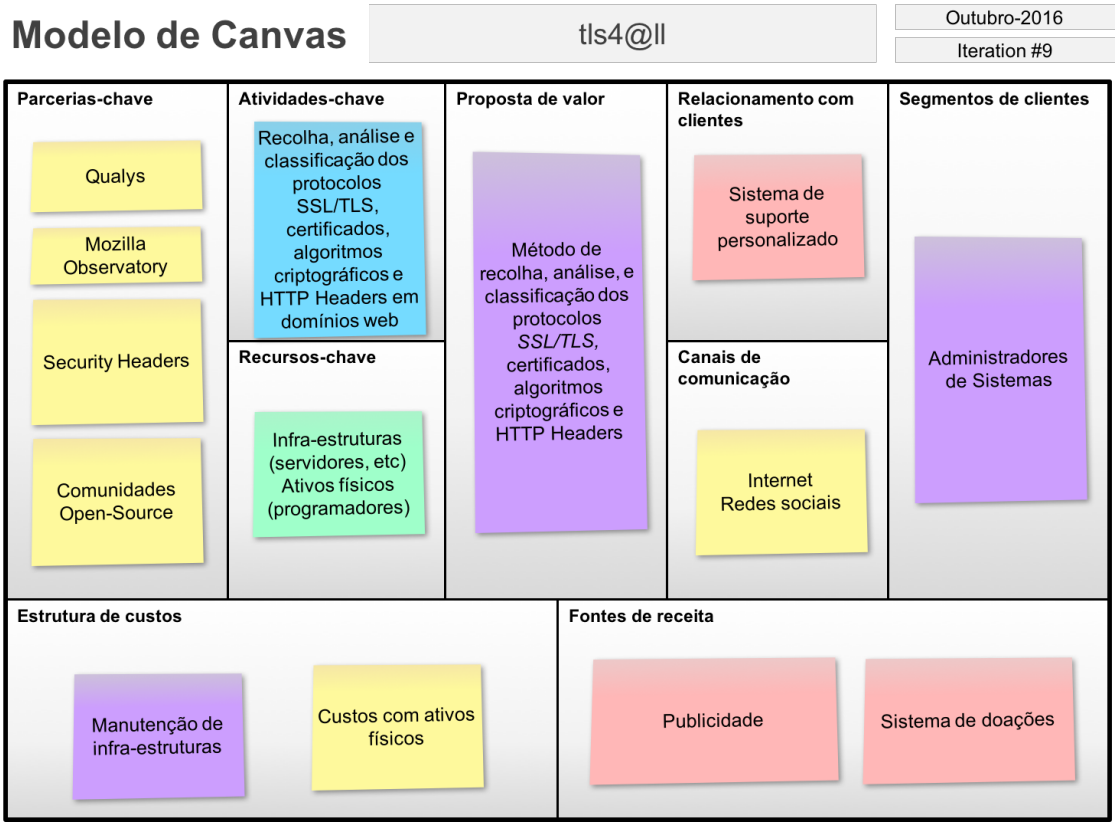
Um modelo de negócio descreve a lógica de como uma organização cria, proporciona e obtém valor (Osterwalder et al., 2010).

O Modelo de Canvas pode assemelhar-se a um modelo de negócio onde envolve nove componentes:

1. Segmentos de clientes (para quem estamos a criar valor);
2. Proposta de valor (produtos/serviços que geram valor para os clientes);

3. Canais de comunicação (como se faz chegar a proposta de valor);
4. Relacionamento com clientes;
5. Fontes de receita;
6. Recursos-chave (ativos físicos, intelectuais, recursos humanos e financeiros, etc.);
7. Atividades-chave (o que se deve fazer para que o modelo de negócio funcione);
8. Parcerias-chave (fornecedores e parceiros que garantem o funcionamento do modelo de negócio);
9. Estrutura de custos (custos inerentes ao modelo de negócio).

Apresenta-se na próxima imagem uma abordagem do Modelo de *Canvas* aplicado a esta dissertação:



Modelos conceptuais

Por forma a analisar/modelar a criação de valor, estudou-se um modelo conceptual (Nicola et al., 2014) e (Nicola et al., 2012) que se baseia na combinação dos seguintes conceitos:

1. The concept of Forms of value and Value temporal positions (WoodDall, 2003);

2. The concept of Value Network ando n the network Exchange of tangible and intangible deliveries among the network roles, building on both tangible and intangible enterprise assets (Allee, 2000);

3. The concept of Enterprise Endogenous and Exogenous assets, extracted from the Reference Model for Collaborative Network Organizations (ARCON) (Camarinha-Matos and Afasarmanesh, 2008);

4. On the Concept of Perceived Beneficts (PBi) / Sacrifices (PSi) (Lapierre, 2000, Woodall, 2003).

Cada conceito e dimensão dessas perspetivas na proposta de Modelo conceptual têm como finalidade compreender o valor percebido em componentes mais simples.

APÊNDICE B – Relatório gerado pelo TLS4@ll (Excerto)

tls4@ll

Assessement

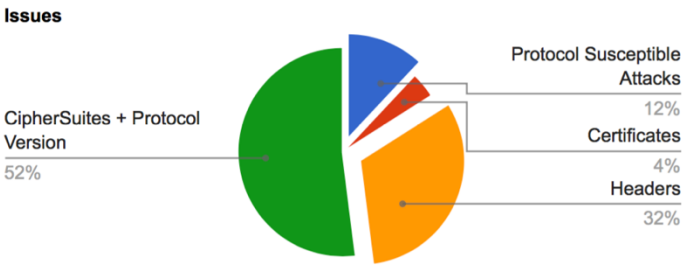
Start Date: out 16, 22:14 PM

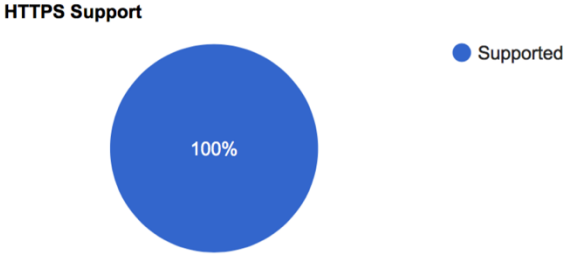
Duration: 3m 51s

URL's (3)

www.[redacted].pt | www.[redacted].com | www.[redacted].com

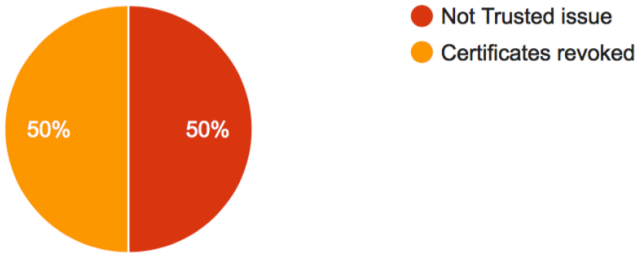
Overall - Issues



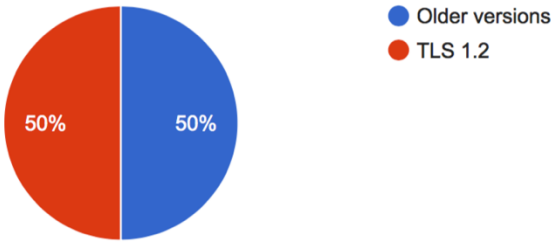


Headers

Certificates



Protocol version support



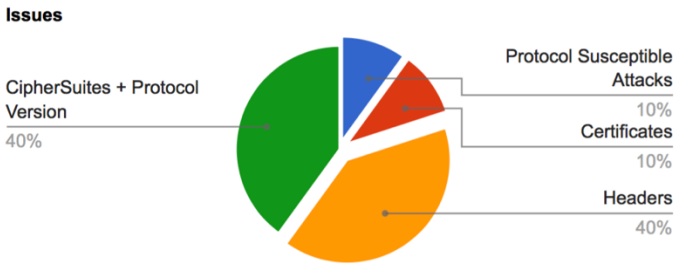
Protocol Vulnerabilities

URL [www[redacted]pt]

ID: *s3FAZoKdhaYucLZhLtqPzSEMtLFUEr*

(IPv4) [redacted] (IPv6) [redacted]

HTTPS: ✅



Test Headers

Headers (Implementation)

Name	Status	Explanations
Strict-Transport-Security	❌	Header not implemented! HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to enforce the use of HTTPS. Recommended value "strict-transport-security: max-age=31536000; includeSubdomains".
Content-Security-Policy	❌	Header not implemented! Content Security Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets.
Public-Key-Pins	❌	Header not implemented! HTTP Public Key Pinning protects your site from MiTM attacks using rogue X.509 certificates. By whitelisting only the identities that the browser should trust, your users are protected in the event a certificate authority is compromised.
X-Frame-Options	✅	Ok.
X-XSS-Protection	✅	Ok.
X-Content-Type-Options	❌	Header not implemented! X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".

Additional Info

Name	Description
Server	Server value has been changed. Typically you will see values like "Microsoft-IIS/8.0" or "nginx 1.7.2".
X-XSS-Protection	X-XSS-Protection sets the configuration for the cross-site scripting filters built into most browsers. The best configuration is "X-XSS-Protection: 1; mode=block".
X-Frame-Options	X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking.

Test Certificates

1

Name	Description
Subject	
Common Names	
Alternative Names	
Valid From	2016-10-06 12:29:50 +0000
Valid until	2016-12-29 12:28:00 +0000 (expires in 73d 14h 9m 51s)
Key	EC 256
Issuer	
Signature Algorithm	SHA256withRSA
Trusted	Trust

2

Name	Description
Subject	
Common Names	
Alternative Names	
Valid From	2015-04-01 00:00:00 +0000
Valid until	2017-12-31 23:59:59 +0000 (expires in 441d 1h 41m 50s)
Key	RSA 2048
Issuer	GeoTrust Global CA
Signature Algorithm	SHA256withRSA
Trusted	Trust

Test Protocol Versions & Cipher Suites

Protocols

Protocol Version	Supported
TLS 1.2	true
TLS 1.1	true
TLS 1.0	true
SSL 3	false
SSL 2	false

Cipher Suites

Cipher name	Cipher Strength
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	128
OLD_TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	128
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	128
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	128
OLD_TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	256
TLS_CECPQ1_RSA_WITH_CHACHA20_POLY1305_SHA256	256
TLS_CECPQ1_RSA_WITH_AES_256_GCM_SHA384	256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	128
TLS_RSA_WITH_AES_128_GCM_SHA256	128
TLS_RSA_WITH_AES_128_CBC_SHA	128
TLS_RSA_WITH_AES_128_CBC_SHA256	128
TLS_RSA_WITH_3DES_EDE_CBC_SHA	168
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	256
TLS_RSA_WITH_AES_256_GCM_SHA384	256
TLS_RSA_WITH_AES_256_CBC_SHA	256
TLS_RSA_WITH_AES_256_CBC_SHA256	256

Recommendations

remove ciphersuites ECDHE-RSA-CHACHA20-POLY1305-OLD, ECDHE-RSA-AES128-SHA, AES128-GCM-SHA256, AES128-SHA, AES128-SHA256, DES-CBC3-SHA, ECDHE-RSA-AES256-SHA, AES256-GCM-SHA384, AES256-SHA, AES256-SHA256
consider adding ciphers ECDHE-ECDSA-AES256-GCM-SHA384, ECDHE-ECDSA-CHACHA20-POLY1305, ECDHE-RSA-CHACHA20-POLY1305, ECDHE-ECDSA-AES128-GCM-SHA256, ECDHE-ECDSA-AES256-SHA384, ECDHE-ECDSA-AES128-SHA256
remove protocols TLSv1, TLSv1.1
consider enabling OCSP stapling

Test Protocol susceptible attacks

Name	Vulnerable	Explanations
BEAST	YES	❌ Insecure! (More info)
CRIME / BREACH	NO	✅ Ok
POODLE	NO	✅ Ok
Heartbleed	NO	✅ Ok
Freak	NO	✅ Ok
Logjam	NO	✅ Ok
Forward Secrecy	NO	Forward Secrecy is achieved with modern clients. (The server supports ECDHE suites, but not DHE)
DROWN	NO	✅ Ok