IJCSI International Journal of Computer Science Issues, Volume 13, Issue 5, September 2016
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org https://doi.org/10.20943/01201605.1119

Web Server Vulnerability Analysis in the context of Transport Layer Security (TLS)

Md Samsul Haque

School of Agricultural, Computational and Environmental Sciences, University of Southern Queensland Toowoomba, Queensland 4350, Australia

Abstract

Mission critical business information of an organization needs to be secured at all times, both in data transmission and while at rest in a server. Web servers hosts the critical static and dynamic contents of an organization, and provides capabilities of access control, authorization, authentication, integrity, and confidentiality during electronic transfers. Securing web servers and transmitting data securely over a public network is a challenging job. Understanding threats to web server and being able to identify appropriate preventative measures permits to anticipate many attacks and thwart in the ever-growing numbers of vulnerabilities. Reviewing the general security threats on a web server and the mechanism to deal extensively with building a diligent wall against the newest vulnerabilities present in today's web technology or programming which are exploitable by attackers are the key points of focus in this paper.

Keywords*: Web Server; Secure Socket Layer (SSL); Transport Layer Security (TLS); Heartbleed.*

1. Introduction

Network security has become a major concern to organizations and researchers with the number of growing attacks to web servers. Knowing the attacking methods, allows for the appropriate security models to emerge.[1] Web server security is as important as securing the website or web application itself and the network around it. If the web application is secure but not the web server, or vice versa, the organization will be in a huge risk. A secure web server provides a protected foundation for hosted web applications, and properly constructed web server configuration plays a critical role in web application's security.[2] When clients make an online transaction, they require a secure connection to ensure that their data are not intercepted and they are connecting to the genuine website; and finally, that their data are not tampered with. These requirements can respectively be restated as confidentiality, authenticity and integrity.[3] To secure a web server effort needed in both hardware and software level. The physical security of computer systems can be enhanced by ensuring that servers are kept in secure rooms so that they are not easily accessible by others. Attackers usually target the exploits in the software which builds the web server to gain authorized entry to the server. Some of the common vulnerabilities of web servers[4], that attackers take advantage of are:

a) Server settings; such as default user id and passwords can be easily guessed by the attackers and allow to run commands on the server which can be exploited,

b) Misconfiguration and bugs; of operating systems and networks setups allows users to execute commands on the server that can lead to gain unauthorized access to the system,

c) Lack of security policy and procedures; such as updating antivirus software, patching the operating system and web server software can create security loop holes for attackers.

Information Security Timelines and Statistics on July 2016 of cyber-attacks shows account hijackings rank on top of the known attack vectors with 14.3% compare to previous months' statistics of 14.5%.[5]. The common[4] attacks against web servers can be categorised as:

1) Directory traversal attacks; which exploits bugs in the web server to gain unauthorized access to files and folders allowing attacker to download sensitive information, execute commands on the server or install malicious software.

2) Denial of Service (DoS); attackers may crash the web server or make unavailable to the legitimate users. The recent DoS attack on the Australian Bureau of Statistics' collapsed the website in August 2016 leaving millions of users unable to complete their census form.

3) Sniffing; is a kind of attack where unencrypted data sent over the network may be intercepted and used to gain unauthorized access to the web server.

4) Phishing; the invaders impersonates the websites and directs traffic to the fake website. Unsuspecting users may be tricked into submitting sensitive data such as login details, credit card numbers etc.

5) Defacement; the attacker replaces the organization's website with a different page that

contains the hacker's name, images and may include background music and messages.

6) Arbitrary Code Execution; an attacker runs malicious code on a server either to compromise server resources or to mount additional attacks against downstream systems.

7) Profiling; also called host enumeration, is an exploratory process used to gather information about the website. An aggressor uses this information to attack known weak points, such as unnecessary protocols, open ports and misconfigurations of web servers[6].

The process to construct a secure web server includes too many aspects to pay attention, however this paper presents a theoretical discussion of the processes and analysed different vulnerabilities. The rest of the paper is organized as follows. Section 2, provides a brief understanding of technologies and the background on the SSL/TLS protocol, cryptosystem and Public Key Infrastructure (PKI). In section 3, summarized the way to secure web servers. Section 4, briefly describes and analysed two high severity threats which took place in recent years and their countermeasure in the perspective of two most popular open source application Apache and OpenSSL. Section 5 concludes the paper with future direction.

2. Background and Related Technologies

A web server can be exploited by using as the launching pad for an organization's entire networking system[7]. If the web server is destabilized an intruder can get access to the server computer as well as to the whole site which can then be exploited to gain access to sensitive information and perform other malicious actions. A relatively common approach to provide web server security is to implement security using Secure Sockets Layer (SSL)\Transport Layer Security (TLS) protocol and digital signature.

2.1 Overview of Cryptosystem

Cryptosystem is a more general term that includes the cryptographic scheme and the sets of possible keys, plaintexts, and cipher texts. Cryptography is the science of protecting information from unwanted person and converting it into a form that is undistinguishable by its attackers. Formally speaking, cryptography is the study of mathematical techniques related to aspects of information security [7] such as:

1) Authentication; is the property of ensuring the identity of an entity, which may be a human, machine, or other asset. The identity is not of the user himself, but of the cryptographic key of the user [8]. A typical example is the SSL certificate of a web server providing proof to the user that he or she is connected to the correct server.

2) Confidentiality; this is the property that protects the content of information from all users other than the ones intended by the legal owner of the information. Other terms for confidentiality have been used synonymously are privacy or secrecy.

3) Integrity; cryptography can provide a means to ensure data is not viewed or altered during storage or transmission. Cryptographic hashes for example, can safeguard data by providing a secure checksum.

4) Non-repudiation; is a cryptographic method to prove that a unique entity has committed an action and must not be able to refute that actions at a later time. For example, a customer may request a transfer of money from her account to be paid to another account. Later, she claims never to have made the request and demands the money be refunded to the account. Non-repudiation, through digitally signing the transaction request, we can prove that the user authorized the transaction[9].

There are two main families of cryptosystems namely symmetric and asymmetric cryptography.

2.1.1 Symmetric Cryptography

Symmetric Cryptography is the most traditional form of cryptography. In symmetric key encryption, same key is used for both encryption and decryption process. Symmetric algorithms have the advantage of not consuming too much of computing power[10]. A symmetric-key cryptosystem consists of a set of encryption and decryption functions E and D, which use the same secret key for both encryption (the enciphering key) and decryption (the deciphering key). Thus, if e denotes the enciphering key and d the deciphering key, in a symmetric cryptosystem $e = d = k$, where, for simplicity, k is called the secret or private key of the cryptosystem. The encryption function E takes as input the plaintext message m and the secret key k and outputs the cipher text c. The decryption function D takes as input the cipher text c and the secret key k and outputs the original plaintext m. Figure 1, describes the above process[8].

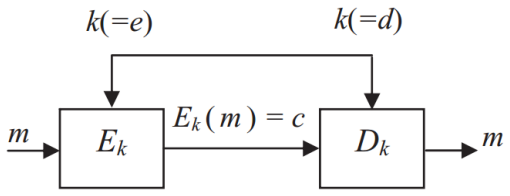

Figure 1: Symmetric Cryptographic Process

Common examples of symmetric algorithms are:

1) DES & Triple DES; Data Encryption Standard (DES) has a key length of 64 bits, however 8 bits are

used for parity, therefore the effective key length is 56 bits. The 56-bit keys used in DES are short enough to be easily brute-forced by modern computer systems and DES should no longer be used. Triple DES is same as the DES operation. It uses three 64-bit keys and overall key length of 192 bits. The procedure for encryption is exactly the same as DES, but this process is repeated three times.

2) Advanced Encryption Standard (AES); National Institute of Standards and Technology (NIST) introduced Advanced Encryption Standard. AES is a symmetric block cipher with block size 128 bits, and cipher keys 128, 192 and 256 bits [7]. For most applications 3DES is currently acceptable, but for most new applications it is advisable to use AES.

3) Blowfish Algorithm; is a type of symmetric key encryption that has a 64-bit block size and a variable key length from 32 bits to 448 bits in general. It is based on 16 round fiestel cipher network that uses the large key size[11].

2.2.2 Asymmetric Cryptography

Asymmetric key encryption is the technique, in which the two different keys are used for the encryption and the decryption process. One key is public and published freely and second key is kept private. It is often referred to as Public/Private key cryptography[7]. Public key methods are important because they can be used for transmitting encryption keys or other data securely even when the both the users have no opportunity to agree on a secret key algorithm. The keys used in public-key encryption algorithms are usually much longer that improves the security of the data being transmitted[10]. An asymmetric-key cryptosystem consists of a set of encryption and decryption functions E and D which use two different but mathematically bounded keys, the enciphering key e and the deciphering key d, where $e \neq d$. In an asymmetric cryptosystem, the enciphering key e is publicly known and only the deciphering key d is kept secret. For this reason the encryption key is also known as the public key and the decryption key is also known as the secret or private key. The encryption function E takes as input the plain text message m and the public encryption key e and outputs the cipher text c. The decryption function D takes as input the cipher text c and the secret decryption key d and outputs the original plaintext m, as shown in Figure 2.[8]

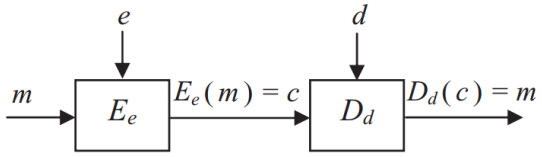

Figure 2: Asymmetric Cryptographic Process

Transport Layer Security (TLS) and Pretty Good Privacy (PGP) protocols are comprehensively utilized and brings out by the asymmetric cryptography. The most popular public key cryptographic algorithm is Rivest-Shamir-Adleman (RSA) Cryptosystem. It can be used for both encryption and digital signatures. The RSA cryptosystem is based on the difficulty of factoring a composite integer n that is a product of two sufficiently large primes' p and q. This prime numbers are random and independent to each other. The prime factors may be numbers each 500 bits long. The RSA cryptosystem is comprised of a key generation, an encryption, and a decryption algorithm [8] which are as follows:

(a) RSA Key Generation; choose two large primes p, q of about equal length and compute $n = p * q$. The two primes are kept secret. Choose a random number e such that e and $(p - 1)(q - 1)$ are relatively prime, that is, the greatest common divisor of e and $(p - 1)(q - 1)$ is equal to 1. The product $(p - 1)(q - 1) = \varphi(n)$, is the Euler *phi* function. Compute d such that $e * d = 1 \bmod (p - 1)(q - 1) = 1 \bmod \varphi(n)$. Thus, d is the inverse of $e \bmod \varphi(n)$. The public encryption key is (e, n) and the secret decryption key is (d, n).

(b) RSA Encryption; Let m be the plaintext message, where m is bounded by n. The RSA encryption of m with the public encryption key (e, n) is $E_e(m) = m^e \bmod n = c$. Since the encryption key is public, anyone can perform the encryption.

(c) RSA Decryption; Let c be the RSA cipher text of m with the encryption key (e, n). The RSA decryption of c with the secret decryption key (d, n) is $D_d(c) = c^d \bmod n = m$. Since the decryption key is secret, only the owner of the decryption key can decrypt cipher texts produced with the corresponding encryption key.

2.3 Primer to SSL/TLS Protocol

Secure Sockets Layer (SSL) technology is a security protocol that was developed by Netscape and is the standard internet protocol for secure communications today across the Internet. The newest version of the SSL was standardized by The Internet Engineering Task Force (IETF) and is now called TLS (Transport Layer Security)[12]. SSL/TLS protocol provide services like authentication of client and servers, to ensure that data is sent to the correct client and server; encrypt data to prevent data filch during transmission; maintaining data integrity during transmission to ensure data is not changed[13].

2.3.1 SSL protocol architecture

SSL protocol is located in between the TCP/IP protocol model of the network layer and application layer. The Transmission Control Protocol/Internet

Protocol (TCP/IP) governs the transport and routing of data over the Internet. The SSL protocol includes two sub-protocols: the SSL record protocol and the SSL handshake protocol. Figure 3, shows SSL runs above TCP/IP and below high-level application protocols.

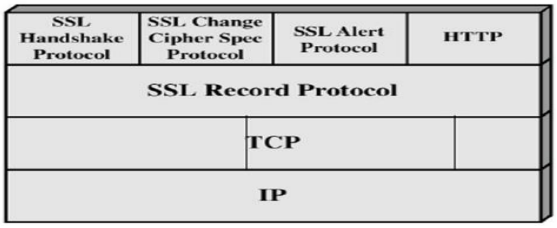

Figure 3: SSL/TLS protocol architecture.

The SSL record protocol defines the format used to transmit data. The handshake protocol is used for the client and the server for authenticating each other before the application data is sent. The client and the server settle various issues including the encryption algorithm to be used, the MAC algorithm and the cryptographic keys. If there is a problem with the connection, an alert is sent. In case of a serious problem, considered fatal, the connection is immediately terminated[14, 15]. The SSL protocol supports the use of a variety of different cryptographic algorithms, or ciphers, to create a secure, confidential communications "pipe" between two entities. Data transmitted over an SSL connection cannot be tampered with or forged without the two parties becoming immediately aware of the tampering.

Most of the security attacks on SSL/TLS protocols are implementation weaknesses of the protocol. SSL/TLS is a complex protocol that supports many cipher suites with cryptographic blocks defined in different specifications. SSL/TLS supports five protocol but not all of them are secure. SSL 3.0 is now being replaced by its successors TLS 1.2. However, TLS implementations still remain backward compatible with SSL 3.0. The best practice is to use TLS v1.0 as the main protocol and TLS v1.1 and v1.2 if they are supported by the server platform. That way, the clients that support newer protocols will select them, and those that don't will fall back to TLS v1.0.

2.3.2 Datagram Transport Layer Security

Datagram Transport Layer Security (DTLS) is a communication protocol which implements TLS over unreliable transport protocol i.e. Datagram Congestion Control Protocol (DCCP) or User Datagram Protocol (UDP).

2.4 Overview of PKI infrastructure

Public Key Infrastructure (PKI) refers to the mechanisms, procedures and policies that collectively provide a framework for addressing the fundamentals of security – authentication confidentiality, integrity, non-repudiation and access control. PKI enables people and businesses to utilize a number of secure Internet applications. For example, secure and legally binding emails and Internet based transactions, and services delivery can all be achieved through the use of PKI[16].

The implementation of public key cryptography requires several supporting components to handle key creation, distribution, and revocation. A PKI provides the means to bind public keys to their owners and helps in the distribution of reliable public keys in large heterogeneous networks. Public keys are bound to their owners by public key certificates. These certificates contain information such as the owner's name and the associated public key and are issued by a reliable certification authority. In PKI, senders apply to a certificate authority (CA) for a digital certificate. Upon verifying the sender's identity, the CA—which charges for its services—issues a certificate to be attached to their electronic communications. It typically includes the sender's name; the certificate's serial number and expiration date; a copy of the certificate holder's publicly available, CA-signed encryption key; and the CA's digital signature.

2.4.1 Digital Signatures

Digital Signatures apply the same functionality to an e-mail message or data file that a handwritten signature does for a paper-based document. The Digital Signature vouches for the origin and integrity of a message, document or other data file. Digital signature is a mechanism by which a message is authenticated i.e. proving that a message is effectively coming from a given sender, much like a signature on a paper document. For Digital signature, another technique used is called hashing. Hashing produces a message digest that is a small and unique representation of the complete message. Hashing algorithms are a one-way encryption, i.e. it is impossible to derive the message from the digest[17].

2.4.2 Digital Certificate

A Digital Certificate is a digital file used to cryptographically bind an entity's Public Key to specific attributes relating to its identity. The entity may be a person, organization, web entity or software application. Digital Certificates are issued under the technical recommendations of the x.509 Digital Certificate format as published by the International Telecommunication Union-Telecommunications Standardization Sector (ITU-T). Based on the degree of sender validation there are three types of digital

CrossMark click for updates

certificates: a) Domain-validated (DV) certificates; validate only the sender's name. This doesn't provide much security unless the recipient knows and trusts the domain-name owner, b) Organization-validated (OV) certificates; require validation of an organization's formal and DNS names. CAs validate the formal name by asking for copies of paperwork, such as articles of incorporation, and c) Extended-validation (EV) certificates; CAs must meet high minimum validation criteria as required by the Certification Authority Browser Forum, an organization of leading CAs, browser makers, and application vendors. CAs have no discretion in implementing the standardized security policies, as they do with OV certificates[18].

2.4.3 X.509 Certificate

An X.509 certificate is a digital certificate that uses the widely accepted international X.509 public key infrastructure (PKI) standard to verify that a public key belongs to the user, computer or service identity contained within the certificate. As public key is included in the X.509 certificate, anyone who gets a certificate can both verify the identity (via a Certificate Authority) and encrypt data with the included public key. X.509 was first published in 1998, as part of the X.500 Directory Services standard by International Telecommunications Union's (ITU). When X.509 was revised in 1993, two more fields were added resulting in the Version 2 format. These two additional fields support directory access control. X.509 Version 3 defines the format for certificate extensions used to store additional information regarding the certificate holder and to define certificate usage. Collectively, the term X.509 refers to the latest published version.

2.4.4 Certification Authorities

Digital Certificates are issued by Certification Authorities (CA). Like a central trusted body is used to issue driving licenses or passports, a CA fulfils the role of the Trusted Third Party by accepting certificate applications from entities, authenticating applications, issuing certificates and maintaining status information about the certificates issued. The incorporation of a CA into PKI ensures that people cannot masquerade on the Internet as people they are not by issuing their own fake Digital Certificates for illegitimate use. The Trusted Third Party CAs will verify the identity of the Certificate applicant before attesting to their identity by Digitally Signing the applicant's Certificate. Because the Digital Certificate itself is now a signed data file, its authenticity can be ascertained by verifying its Digital Signature. Therefore, in the same way we verify the Digital Signature of a signed message, we can verify the authenticity of a Digital Certificate by verifying its signature. The CA provides a Certification Practice Statement (CPS) that clearly states its policies and practices regarding the issuance and maintenance of Certificates within the PKI. The CPS contains operational information and legal information on the roles and responsibilities of all entities involved in the certificate lifecycle.

2.4.5 Certificate Chain

All SSL sites use certificates as their digital IDs. However, in many cases a chain of certificates is needed to create a trust link between the user and a trust anchor. A common mistake is that the certificate chain is incomplete, which often results with certificate warnings on sites that are otherwise well configured. A recent survey on August 2016 conducted by Trustworthy Internet Movement shows 3.6% of the sites (4987 sites) surveyed with incomplete certificate chain[19].

2.4.6 Certificate Revocation

In addition to issuing certificates, CAs are also responsible for making available a list of certificates it has issued that have been revoked, after which clients should no longer consider those certificates valid. If a CA's intermediate or root certificate is revoked, all leaf certificates signed by that CA will fail to validate.

2.4.7 Certificate Reissues

When a site ceases to use a certificate for instance because they found that the certificate has been compromised, or because the certificate expired they must use a new certificate instead. This process is referred to as reissuing the certificate. To do so, the system administrator must contact the CA who signed their certificate and request a new signature; this is typically done by sending the CA a Certificate Signing Request (CSR).

3. Securing Web Server

Securing web server data involves protecting data in both in transit and at rest.

3.1 Securing Data in Transit

There are two fundamentally different approaches to securing data in transit. In the network-layer approach, the encryption and authentication is added directly into the networking stack so that traffic is protected without requiring the application to incorporate it. Traffic reaching the remote system is automatically decrypted and verified by the remote system's networking stack before the operating system passes it to the server application. In the application-level approach, the application itself is modified so that traffic is encrypted before it is submitted to the operating system and network layer.

It is then decrypted by the receiving server application.

Cryptographic primitives, algorithms, and protocols are widely used and implemented in network security mechanisms, such as SSL/TLS protocol suites, Internet Protocol Security (IPSec), the secure shell (SSH), virtual private networks (VPNs), and remote authentication services[18]. Secure server certificates, in conjunction with a public key encryption system allow Web servers to establish SSL sessions with Web clients for electronic data transmission. There are three modes of SSL web server authentication process[20]. Suppose requesting client "A", and "R" as server, "PK" as public key, "E_{PK_R}" as the server's public key. The tree patterns can be shown as follows:

Pattern (1):

$A \rightarrow R : E_{PKR}[K_S] \,||\, E_{K_S}[M]$

In this pattern, the requestor A knows the public key of the server. "A" encrypted the unique session key K_S and message M by session key K_S. Thus in this way, the sensitive plaintext M could be received and deciphered by " R ".

Pattern (2):

$A \rightarrow R : M \,||\, E_{SKA}[H(M)] \,||\, E_{SKAU}[T \,||\, ID_A \,||\, PK_A]$

In pattern (1), "R" knows that the text is transmitted to him but is not sure that the real sender is "A", since there can be another one pretend to be "A". Pattern (1) needs authentication and non-repudiation. By decipher $E_{SKAU}[T \,||\, ID_A \,||\, PK_A]$, "R" can be sure that the sender is "A". The non-repudiation is implemented by $E_{SKA}[H(M)]$ since there's no one who has E_{SKA} besides "A". However in this pattern M is transacted in plaintext and can be accessed unauthorized user.

Pattern (3):

$A \rightarrow R : E_{PKR}[M||E_{SKA}[H(M)]]||E_{SKAU}[T_S||ID_A||PK_A]$

Compared with pattern (2), the text is encrypted by public key E_{PKR} and then only can be accessed by "R" with E_{SKR}. From the perspective of security, pattern (3) provides the best services with authentication, non-repudiation and confidentiality. For simply use of confidentiality, pattern (1) is used widely in World Wide Web. In this case, message M is encrypted and could not be accessed by others than the server. The session key is used to prevent the intruder. A simple HTTPS connection to the Web server is all that is needed to access the services. For more complex purpose pattern (2) or (3) would be adopted. [20]

3.2 Securing data at rest

Securing data while at rest focuses on the Web Server's host system configuration and operational practices and provides a foundation for server security. Challenges in host system security may include complexity, access control, and accountability. Securing access control can be physical or logical. Beyond locks are the most fundamental physical control mechanisms. Other sophisticated systems like biometric authentication should be included for controlling physical access. Password complexity requirements and encryption of passwords provide a far greater level of protection than simple passwords and passwords transmitted in plain text. Firewalls can be used to limit external users from unauthorized access of a web server. [15]. Apache web server has the module mod_access to provide access control. The mod_access module as its name clearly implies, provides access control for documents. It allows one to restrict or allow access to resources based on the client's host name, IP address, or network address. Study shows that most incidents result from server misconfiguration. It is important to keep a current understanding of server configurations to adapt against possible vulnerabilities in a timely manner. For example a running proxy service that allows anyone to use it without restriction represents a big configuration error[21]. Another frequent configuration problem is the unrestricted availability of web server access logs. The logs if unprotected, can reveal links to server resources. Apache provides a number of modules to support logging. Some of them are[22]: a) mod_log_forensic, this module provides for forensic logging of client requests. Logging is done before and after processing a request, so the forensic log contains two log lines for each request, b) mod_logio, this module provides the logging of input and output number of bytes sent and received per request, c) mod_log_config, this module provides for flexible logging of client requests. Logs are written in a customizable format, and may be written directly to a file, or to an external program.

4. Web Server Vulnerability

Based on SSL/TLS Protocol's security architecture and security vulnerability of PKI system this section of this paper focuses on 2 most recent and severe security issues on Web Servers.

4.1 Heartbleed attack

The Heartbleed Bug is one of the most impactful SSL/TLS Protocol vulnerability. Heartbleed is a buffer over-load memory leak issue and the name was taken from RFC6520[23], TLS and DTLS Heartbeat Extension. The bug is in a piece of logic; a simple missing bounds check that relates to the popular open source OpenSSL cryptographic software library that implements the TLS heartbeat mechanism. The bug is present in OpenSSL versions

1.0.1 through to 1.0.1f. By mistreating this mechanism, attacker can request that a running TLS Web server provide a relatively large slice with up to 2^{16} bytes (~64KB) of its private memory space. Since this is the same memory space where OpenSSL also stores the server's private key material, an attacker can potentially obtain web server's long-term private keys, to decrypt past sessions, or impersonate the server going forward leaving no traces, TLS session keys, Confidential data like passwords, and Session ticket keys. Neel Mehta of Google Security, first reported the bug to the OpenSSL team in March 2014. Even after one year of first discovery of the bug a scan report of Forbes Global 2000 companies by security firm Venafi, on April 2015, shows that 74% of these organizations web servers were still vulnerable to Heartbleed attacks.

4.1.1 TLS Heartbeat Protocol

The TLS Heartbeat mechanism is designed to keep connections alive even when no data is being transmitted. The Heartbeat Extension allows either end-point of a TLS connection to detect whether its peer is still present, and was motivated by the need for session management in DTLS. Standard implementations of TLS do not require the extension as they can rely on TCP for equivalent session management. The Heartbeat protocol is a new protocol on top of the Record Layer. The protocol itself consists of two message types: HeartbeatRequest and HeartbeatResponse. HeartbeatRequest message can arrive at any time during the lifetime of a connection but there must not be more than one HeartbeatRequest message in flight at a time. Following negotiation, either end-point can send a HeartbeatRequest message to verify connectivity. HeartbeatRequest messages consist of a one-byte type field, a two-byte payload length field, a payload, and at least 16 bytes of random padding. Upon receipt of the request, the receiving endpoint responds with a similar HeartbeatResponse message, in which it echoes back the HeartbeatRequest payload and its own random padding [24], Figure 4, shows the heartbeat protocol.

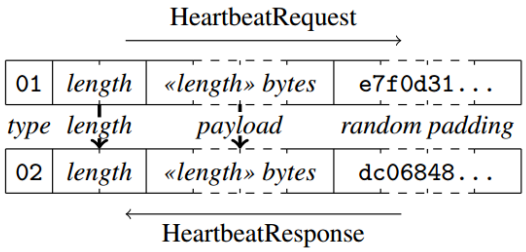

Figure 4: TLS Heartbeat Protocol.

Heartbeat requests include user data and random padding. The receiving peer responds by echoing back the data in the initial request along with its own padding.

4.1.2 OpenSSL Coding error

OpenSSL's incorrect memcpy function that build the HeartbeatResponsemessage is

memcpy(bp, pl, payload);

The problem here is that the OpenSSL heartbeat response code does not check, to make sure that the payload length field in the HeartbeatRequest message matches the actual length of the payload. If the HeartbeatRequest payload length field is set to a value larger than the actual payload, the memcpy code will copy the payload from the heartbeat message and whatever is in memory beyond the end of the payload. A HeartbeatRequest payload length can be set to a maximum value of 65535 bytes. Therefore the bug in the OpenSSL HeartbeatResponse code could copy as much as 65535 bytes from the machine's memory and send it to the requestor.

The OpenSSL code fix for the Heartbleed bug for version 1.0.1g shows the change in OpenSSL's file t1_lib.c. This code fix has two tasks to perform[25]: Firstly, it checks to determine if the length of the payload is zero or not. It simply discards the message if the payload length is 0, as shown below

```
if (1 + 2 + 16 > s->s3->rrec.length)
return 0;
```

The second task performed by the bug fix makes sure that the heartbeat payload length field value matches the actual length of the request payload data. If not, it discards the message as below

```
if (1 + 2 + payload + 16 > s->s3->rrec.length)
return 0;
```

4.1.3 Countermeasures

The potential impact of the bug is severe: it allows an attacker to read private memory, potentially including information transferred over the secure channel and cryptographic secrets. Exploitation of this bug does not leave any trace of anything abnormal happening to the logs. The scariest part of the OpenSSL Heartbleed bug is that, even after taking these measures, no one can completely relax. This vulnerability has existed for more than 2 years before it was first discovered. No one knows if their application has been exploited because the attack leaves no traces of it. There is a possibility that attackers might have been reading passwords, secret keys and other encrypted data. This theft cannot be known unless the misuse of the data is observed or the attacker discloses it. The remedies are as follows [26], a) upgrade to OpenSSL 1.0.1g or rebuild current version of OpenSSL from source without TLS Heartbeat support by adding compile switch: -DOPENSSL_NO_HEARTBEATS. The switch ensures that the defected code never executed, b)

generate new keys and certificates, c) issue and install new certificates, and d) Revoke old keys and certificates.

4.2 Bypassing SSL/TLS certificate checks

Open-source software, is widely used in two thirds of all web servers in the world. The vulnerability relates to the popular opens source software OpenSSL's certificate verification process. The vulnerability was reported to the developers of the SSL/TLS toolkit on June 2015 by Google's Adam Langley and David Benjamin. This vulnerability was described as high severity issue by OpenSSL as an alternative chain certificate forgery flaw (CVE-2015-1793)[27], which was introduced with OpenSSL versions 1.0.1n and 1.0.2b. SSL certificates are issued in chains, moving from the root certificate authority (CA) through a number of intermediate CAs down to the end user certificate, known as the leaf certificate. Generally, a client has a list of trusted certificate authorities to validate an X.509 certificate and possibly also a list of helper certificates that can be used to establish a chain of trust to one of the trusted CA. Bypassing certificate checks allows remote attackers to spoof a Certification Authority role and trigger unintended certificate verifications via a valid leaf certificate.

Web servers use Digital Certificates to prove ownership of the Web Server or domain name and establish SSL / TLS encrypted sessions between the server and client. Certificate validation is a twostep process where at first, a path from the certificate to a trusted certificate authority is established, then this path is verified. There might be more than one path, and should verification fail, another path might be determined and verification is tried again. If it doesn't, it will return an error message and a secure connection will be denied[28]. In essence, the vulnerability could enable a man-in-the-middle attack to occur, since applications would mistakenly view untrusted SSL certificates as valid.

For example, an attacker can forge or spoof the authentication between an end-user and their bank website. In such a "man-in-the-middle" scenario, all personal data communicated in the browser session can be intercepted and/or compromised. Both integrity and confidentiality of the data exchanged in that session are at risk. Figure 5, illustrates this attack against SSL/TLS digital certificates.

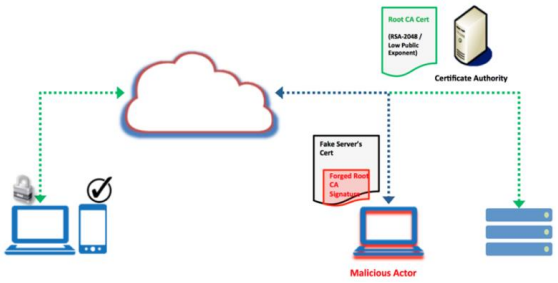

Figure 5: Bypassing SSL/TLS certificate attack.

4.2.1 Countermeasures

To fix the bug, Web server administrators need to update OpenSSL libraries and Verify certificates by CA flag when creating new certificates.

5. Conclusion

Web Servers are very attractive targets for attackers and that's why security is an essential topic for both internet-facing and intranet servers. Simply providing encryption and message integrity gives little security in a client-server environment, if we do not know the legitimacy of the other party. Web server security is a continuous process, while PKI is a useful component of that process but it is dangerous to think it provides security in and of itself. Even after many years researchers have been still finding some weaknesses of the popular security protocols which are vulnerable for Web Servers. Recent attacks on web servers like Heartbleed and alternative chain certificate forgery are all harsh reminders that no protocol in practical use is ever likely to be invincible, and that the best we can do is try and find the holes before the vulnerability happens. In order to understand the current network security threats being evolving in today's world, background knowledge of the security protocols, its vulnerabilities, attacking methods, and security technology is very crucial and therefore further research is always necessary.

References

[1] B. Daya, "Network security: History, importance, and future," University of Florida Department of Electrical and Computer Engineering, 2013.

[2] S. B. Almin, "Web Server Security and Survey on Web Application Security."

[3] E. S. Alashwali, "Cryptographic vulnerabilities in real-life web servers." pp. 6-11.

[4] Guru99. "How to hack a Web Server," 2016; http://www.guru99.com/how-to-hack-web-server.html.

[5] P. Passeri. "July 2016 Cyber Attacks Statistics," August 18, 2016; http://www.hackmageddon.com/2016/08/18/july-2016-cyber-attacks-statistics/.

[6] J. Meier *et al.*, "Improving web application security: threats and countermeasures," Microsoft Corporation, vol. 3, 2003.

[7] W. Stallings, Cryptography and network security: principles and practices: 5th edn, Pearson Education, 2011.
[8] P. Kotzanikolaou, and C. Douligeris, "Appendix A: Cryptography Primer: Introduction to Cryptographic Principles and Algorithms," Network Security: Current Status and Future Directions, pp. 459-479, 2007.
[9] OWASP. "Guide to Cryptography," September 12, 2015; https://www.owasp.org/index.php/Guide_to_Cryptography.
[10] J. Thakur, and N. Kumar, "DES, AES and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis," International journal of emerging technology and advanced engineering, vol. 1, no. 2, pp. 6-12, 2011.
[11] B. Schneier, "The Blowfish encryption algorithm," Dr Dobb's Journal-Software Tools for the Professional Programmer, vol. 19, no. 4, pp. 38-43, 1994.
[12] T. Dierks, and E. Rescorla, "IETF RFC 4346,"," The transport layer security (tls) protocol version, vol. 1, 2006.
[13] J. Du, X. Li, and H. Huang, "A study of man-in-the-middle attack based on SSL certificate interaction." pp. 445-448.
[14] M. S. Bhiogade, "Secure socket layer." pp. 85-90.
[15] S. Kesh, and S. Ramanujan, "A Model for Web Server Security," The Journal of American Academy of Business, Cambridge, pp. 354-359, 2004.
[16] N. Komninos, "PKI systems," Network Security: Current Status and Future Directions, pp. 409-418, 2007.
[17] CGI. "Public Key Encryption and Digital Signature: How do they work?," http://www.cgi.com/files/white-papers/cgi_whpr_35_pki_e.pdf.
[18] N. Leavitt, "Internet security under attack: The undermining of digital certificates," Computer, vol. 44, no. 12, pp. 17-20, 2011.
[19] T. I. Movement. "SSL Pulse: Survey of the SSL Implementation of the Most Popular Web Sites," August 03, 2016; https://www.trustworthyinternet.org/ssl-pulse/.
[20] D. Zhang *et al.*, "Application Research of SSL Encryption on Linux Web Server." pp. 1-3.
[21] B. Fogel, "A Survey of Web Vulnerabilities," Auburn University, 2015.
[22] S. Jahid, I. Hoque, and M. Hafiz, "Security Issues of Web Server."
[23] R. Seggelmann, M. Tuexen, and M. Williams, "RFC6520-Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension," IETF, tools. ietf. org/html/rfc6520, 2012.
[24] Z. Durumeric *et al.*, "The matter of heartbleed." pp. 475-488.
[25] B. Chandra, "A technical view of the OpenSSL Heartbleed vulnerability," White paper, IBM developerWorks, 2014.
[26] G. Hill, "Still Bleeding One Year Later—Heartbleed 2015 Research," 2015.
[27] NIST. "National Cyber Awareness System. Vulnerability Summary for CVE-2015-1793," August 22, 2016; https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-1793.
[28] Acmetek. "OpenSSL: Alternative Chains Certificate Forgery Vulnerability (CVE-2015-1793)," July 10, 2015; https://www.acmetek.com/Support/the-new-alternative-chains-certificate-forgery-vulnerability-cve-2015-1793.

Md Samsul Haque, is a Maters of Computing Technology, Student. With interest in Computer Programming, Computer Security, and Machine Learning and Data Science.