International Journal of Cyber-Security and Digital Forensics (IJCSDF) 7(3): 243-247
The Society of Digital Information and Wireless Communications (SDIWC), 2018 ISSN: 2305-001

A Study of the Effects of Heartbleed Vulnerability in Bangladesh

Moniruz Zaman, Delwar Alam, Touhid Bhuiyan
Department of Software Engineering
Daffodil International University
Dhaka, Bangladesh
{m.shojol80, delwaralam, touhidbhuiyan}@gmail.com

Tanjila Farah
Department Electrical and Computer Engineering
North South University
Dhaka Bangladesh
tanjila.farah@northsouth.edu

***Abstract* – One of the most critical and talked about Open Secure Socket Layer (SSL) and Transport Layer Security (TLS) threats is Heartbleed vulnerability. This vulnerability affects the "Heartbeat protocol" of the open SSL library. HeartBleed manipulates the Heartbeat protocol to get access and read the memory of the vulnerable web servers. OpenSSL is used in HTTPS for internet security. As OpenSSL vulnerability, heartbeat has affected websites, web servers, VPN concentrators, client applications and mobile devices. The sensitive information that may be retrieved using this vulnerability includes Primary and secondary key material, and protected content. A patch for this vulnerability exists since 2014 yet there are a good number of web platform vulnerable to this threat. This paper focuses on the attack technique of Heartbleed vulnerability. This papers primary contribution is the detail analysis of affect of the Heartbleed vulnerability on the web platform of Bangladesh. As a newly emerging country in the digital world, Bangladesh needs to be prepared to the existing threats. This paper attempts to guide web developers of Bangladesh to secure the web platform against the Heartbleed vulnerability.**

Index Terms - Heart bleed, Open SSL, TLS, Three Way Handshake, HTTP Keep Alive, Heart beat request.

I. Introduction

Present day Internet runs on database driven web applications used to provide various services (i.e. online banking, transaction, e-commerce). These communications run between servers and a clients using Hyper Text Transfer Protocol (HTTP) [3]. As the client server communication does not require one party knowing the other, this communication operates based on verified trust [13]. This trust is established through encrypted Secure Sockets layer/Transport Layer Security (SSL/TLS) certificates running on HTTP. This is known as HTTPS. With the increase of cyber assaults, the need for secure data transmission has intensified. SSL/TLS uses OpenSSL cryptographic library for generating encryption certificates. As of 2014, two thirds of all web servers used OpenSSL to secure end-to-end communications [4]. SSL/TLS has become a *defacto* secure communication standard [5]. Based on the development, TLS is a successor SSL [9]. There are various versions of SSL, TLS, and OpenSSL available. One of the most important extensions to SSL/TLS is "Heartbeat". This extension keeps the connection between client and server alive.

Although the HTTPS communications are considered secure there are various vulnerabilities in the SSL/TLS protocols. These vulnerabilities are categorized as protocol logic flaw, cryptographic design flaw, implementation flaw, and configuration and infrastructure flaw [8]. This paper focuses on one of the most exalted SSL (more preciously OpenSSL) vulnerability known as "Heartbleed" [1]. This is a vulnerability of the Heartbeat feature of OpenSSL [2]. This vulnerability is categorized as implementation flaw of OpenSSL [10]. This vulnerability aims on data leakage in the Heartbeat protocol implementation and it allows the attackers to read sensitive memory from servers. This vulnerability was disclosed in 7 April 2014 [12]. According to United States Computer Emergency Readiness Team (US-CERT), about half a million widely trusted websites were vulnerable to Heartbleed attack during zero day attack [2]. TheVulnerable website list includes google, facebook, ebay, and amazon [1]. This vulnerability effected OpenSSL versions 1.0.1*a* to 1.0.1*g*. The next version 1.0.2 was published with a patch for Heartbleed vulnerability. This vulnerability existed due to lack of validation in the "Heartbeat" protocol [5]. Though a patch for this vulnerability exists since 2014, a large number of web servers are still vulnerable to this attack.

This paper discusses the technique of Heartbleed vulnerability and its affect on the web servers of Bangladesh after the patch is available. The paper is organized in five sections. In section II and overview of SSL and TLS protocol is presented. This section details about the "Heartbeat" protocol. In section III an analysis of the "Heartbleed" vulnerability is presented. The current conditions of the Bangladeshi servers regarding the "Heartbleed" vulnerability are discussed in Section IV. We conclude in Section 5.

II. Overview of SSL and TLS

Secure Socket Layer (SSL) and Transport Layer security are security protocols used to establish a secure connection between Server and Client. In a regular communication between client and server, data transmitted in plain text. At present the Internet traffic includes data regarding transactions such as online shopping, emails and online banking. Without security protocol anyone using network intrusion software (i.e. wireshark) could access and read that data. To secure such sensitive communication, SSL/TLS has become a *defacto* communication standard [5]. Client Server communication is established using hyper text transfer protocol (HTTP). To implement secure communication most web applications use HTTP secure (HTTPS) [10].

A. *Secure Socket Layer (SSL)*

SSL secures transmission over Transmission control protocol (TCP) [10]. To provide security SSL protocol combines's privacy, authentication, and reliability. SSL protocol ensures the connection is encrypted. It also ensures that the data can be accessed by only authenticated user and the authentication is provided by digital certificate and RSA cryptographic algorithm. Digital certificate is not required for establishing a SSL connection. The SSL protocol is integrated into most web browsers used to access web application.

In server side the web server administrator must acquire a digital certificate from a Certification Authority (CA). The web servers store multiple certificates from multiple web applications and using these certificates authenticates the applications hosted by the server [11]. SSL uses four protocol layers to operate. These are: Record Layer, Change Cipher Spec Protocol, Alert Protocol, and Handshake Protocol. These protocols are used to encapsulate all communication between the client machine and the server [11].

B. *Transport Layer Security (TLS)*

TLS provides cryptographic security, interoperability, extensibility, and relative efficiency. To ensure these services TLS works in two levels: Record protocol and Handshake protocol [6]. The Record protocol uses symmetric cryptography keys, to ensure a private connection between the client and the server. The Handshake protocol initiates the authenticated communication between the client and server. This protocol negotiates the language of communication between client and server, their encryption algorithm and encryption key before the communication starts [13].
There are several minor differences between SSL and TLS. While SSL encryption uses only MD5 and SHA hash functions, the TLS encryption can operate with any hash function. TLS uses the HMAC standard and its pseudorandom function (PRF) output to generates the required key material. On the other hand SSL uses RSA which generates secret information based on the cipher Suite and Parameters selected during session negotiations [13].

C. *SSL/TLS Handshake*

SSL/TLS encryption uses public key private infrastructure to secure the data transmission. The encryption process is uses the public key to encrypt data at source and corresponding private key to decrypt data at destination. These keys are shared during the initial handshake protocol. The authenticity of the public key is verified by a digital certificate issued by a certificate authority (CA) [2].

The secure connection between client and server in a SSL/TLS environment starts with client browser requesting for a secure connection from server as shown in Figure 1. In this request message client browser also sends a list cipher suit and SSL/TLS version it supports. In response the server send its choice of cipher suit and SSL/TLS version based on the list client browser has sent with its request [10]. Server also sends its certificate and public key with this response message. The client uses the certificate to extract public key and use this public key to encrypt "pre-master key" [7]. The Client browser then sends this pre-master key to server. The server then uses its private key to decrypt the "pre-master key". The client and server use the "pre-master key". And the pre decided cipher to generate a shared secret key. This key is used to encrypt and decrypt data in both client and server [11]. In last step the encryption client browser sends an encrypted message and server decrypt the message and verifies and sends a response to the client. After this point all communication between the client and server in encrypted.

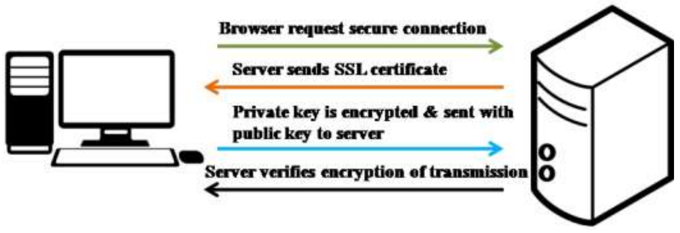
Browser request secure connection
Server sends SSL certificate
Private key is encrypted & sent with public key to server
Server verifies encryption of transmission

Fig. 1 SSL/TLS connection setup.

D. *OpenSSL*

OpenSSL is an open source cryptographic library used with applications and web server. It is the open source implementation of SSL and TLS [10]. OpenSSL is used to validate the server's certificate chain. The library includes tools for generating RSA private keys and Certificate Signing Requests (CSRs), checksums, managing certificates and performing encryption/decryption. OperSSL supports a number of cryptographic algorithms including ciphers (i.e. AES, RC2), cryptographic has function (i.e. MD5, SHA-2), and public key cryptography (i.e. RSA, Diffie-Hellman key exchange). It's estimated that 66% of all Web servers use OpenSSL. The encryption keys used in SSL/TLS is generated using open SSL. The OpenSSL implementation of TLS protocol is known as heartbeat extension (RFC6520).

E. *Heartbeat protocol*

In SSL/TLS protocol has no mechanism to keep the connection alive without continuous data transfer. To overcome this limitation Heartbeat extension was introduced to openSSL implementation of SSL/TLS. Heartbeat protocol runs on top of the Record Layer and maintains the connection between the client and server by requiring them to exchange a "heartbeat". In previous versions of SSL/TLS, a renegotiation technique function to find alive connection was available but that was costly [2].

The "Heartbeat" protocol includes two messages: heartbeat request and heartbeat response. For every "Heartbeat request" message a "Heartbeat response" message should sent between the client and the server. This is called "Keep alive" [2]. If no response message is received the TLS connection is terminated. The structure of a "Heartbeat" message (request and response) is shown in Figure 2. The message type refers to either request of response message. The payload length represents the length of the Heartbeat response message. Payload variable holds the payload content and the padding variable includes random data. The padding field is used to discover the path maximum transfer unit (PMTU).

```
Struct {
HeartbeatMessageType type;
Uin16 payload_length;
Opaque payload [HeartbeatMessage . Payload_length];
Opaque padding [padding_length];
}
HeartbeatMessage;
```

Fig. 2 Heartbeat message structure in code.

In response to the heartbeat request message a copy of the same message is sent as response by the receiver side. This verifies the connection to be still alive [2]. To implement the "Heartbeat response" message, the receiver end first check if the type of the received message "TLS Heartbeat request". Then the receiver extracts the payload length of the received message. Using this value the receiver allocates memory for the heartbeat response. Then the receiver side copies the payload from "Heartbeat request" message to "Heartbeat response" message is send it back to the sender side. The sender side then compares the response message with the original message and if the messages are same, the connection is considered alive.

OpenSSL versions 1.0.1 through 1.0.1g contain a flaw in its implementation of the TLS/DTLS "Heartbeat response" functionality. This flaw allows an attacker to retrieve Primary key material (secret key), Secondary key material (user names and passwords used by vulnerable services), Protected content (sensitive data used by vulnerable services), Collateral (memory addresses and content that can be leveraged to bypass exploit mitigations). This vulnerability is known as "Heartbleed".

III. Analysis of Heart bleed Vulnerability

The "Heartbleed" vulnerability results from improper input validation in the implementation of the TLS heartbeat extension. The "Heartbleed" vulnerability allows an attacker to retrieve private memory of an application that uses the vulnerable OpenSSL library in chunks of 64k at a time. The attacker can repeatedly leverage the vulnerability to retrieve as many 64k chunks of memory as are necessary to retrieve the intended data.

Using Heartbleed vulnerability the attacker modifies the length of the payload of the Heartbeat request message to retrieve information from the server. In the server side there is no validation mechanism for the length of the payload. Thus the server then returns more payload content from the server memory then it was suppose to send through a "Heartbeat response" message. This attack can be implemented user any browser. Any web application using the vulnerable version of HTTPS is exposed to this attack.

The steps of Heartbleed attack are shown in Figure 3. The attack takes place after client side receives a response message for "hello" request. The client server communication initiates with a 3 way handshake (using SSL/TLS).Then the server provides the client with a socket number. As long as the connection is alive the client uses this socket to communicate with server. After the connection is set the attacker/client side sends hello packets. If the hello response is none, it means the server closes connection without any communication.

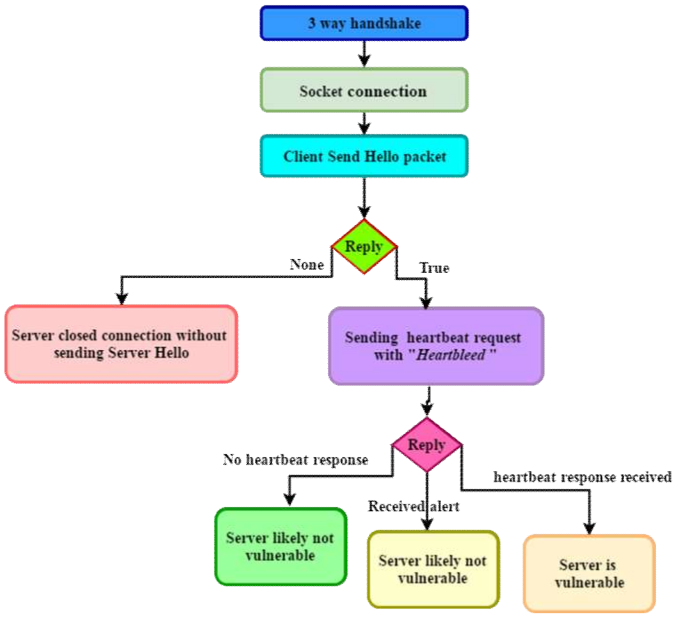
3 way handshake
Socket connection
Client Send Hello packet
Reply
None
True
Server closed connection without sending Server Hello
Sending heartbeat request with "Heartbleed"
Reply
No heartbeat response
heartbeat response received
Received alert
Server likely not vulnerable
Server likely not vulnerable
Server is vulnerable

Fig. 3 Steps of Heatbleed attack

If the hello response is true, the "heartbeat request" packet with "Heartbleed" exploit is sent. If the server responses to the "Heartbeat request", it is identified as vulnerable to "Heartbleed". The code of the "Heartbeat" message with "Heartbleed" exploit is shown in Figure 4. The code is in hexadecimal value. The first line of the code defines the type of the "Heartbeat" message, and packet length. The actual exploit code is written in the second line. The second line indicates the message type and the payload length. The payload length filed id modified to be "FF FF" which is nearly 64kb. This value is larger than the actual "Heartbeat response" message.

```
HeartbleedPacket = (
'18 03 02 00 03'      # Content type = 18 (heartbeat message); Version = 03 02;
                                                          Packet length = 00 03
   '01 FF FF'         # Heartbeat message type = 01 (request); Payload length =
                   FF FF # Missing a message that is supposed to be FF FF bytes long
).replace(' ', '').decode('hex')
```

Fig. 4 Heatbleed exploit code

The older version of OpenSSL "Heartbeat response" does not have the mechanism to check and validate the payload lengths. The OpenSSL runs "memcpy" function to identify the payload length and payload from the "Heartbeat request" message. The mmcpy function call is shown in Figure 5. The 'pl' variable is used to get the payload length from "Heartbeat request" message which in case of "Heartbleed" is the packet shown in Figure 4. The packet in Figure 4 delivers the packet length value to be FFFF.

```
memcpy (bp, pl, payload)
```

Fig. 5 Memory copy function in OpenSSL

To fill up the rest of the data requested by the client, the server will copy the surrounding memory of the server which may include sensitive data. An attacker could continually send malicious heartbeats to a server and keep getting back 64kb of its memory each time. Using this memory leak the Heartbleed attack is implemented from the client end.

IV. DATA ANALYSIS

The "Heartbleed" vulnerability impacted the web servers worldwide. About 24-55% HTTPS web servers were initially vulnerable to this attack [2]. The patch for this vulnerability was implemented as fast as the vulnerability amplified. Yet this vulnerability exists till date. The analysis of world data is out of the scope of this paper. Thus this paper only focuses on the data collected in Bangladeshi domain. Mostly used Bangladeshi domains are ".bd" and ".com". For the analysis purpose we have checked 1298 web applications in both ".bd" and ".com".

Analyzing 1298 web application 538 were found using SSL/TLS protocol. Other 753 were found using no security protocol or other type of security. The Figure 6 indicated that 42% of the web servers use HTTPS protocol.

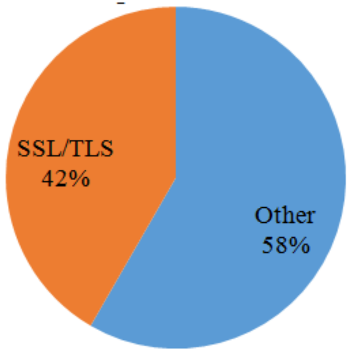

Fig. 6 Percentage of SSL/TLS user

Among the 538 SSL/TLS web applications, 156 uses older version of SSL/TLS and 382 uses updated version of SSL/TLS as shown in Figure 7. In this graph the older version represents the versions published before the "Heartbleed" vulnerability was identified. The updated version represents all the versions published after the "Heartbleed" vulnerability was recognized.

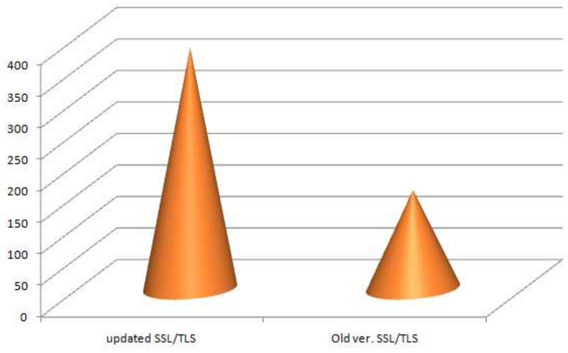

Fig. 7 Usage of various versions of SSL/TLS

Not all older versions of SSL/TLS were vulnerable to "Heartbleed" vulnerability. This vulnerability exists because of an implementation flaw in the versions 1.0.1*a* to 1.0.1*g*. The analyzed dataset, 156 web applications were found using older version of SSL/TLS. Among these 24 web applications uses SSL/TLS version 0.9.1 to 1.0.0 and the other 132 uses SSL/TLS version 1.0.1*a* to10.0.1*g* as shown in Figure 8.

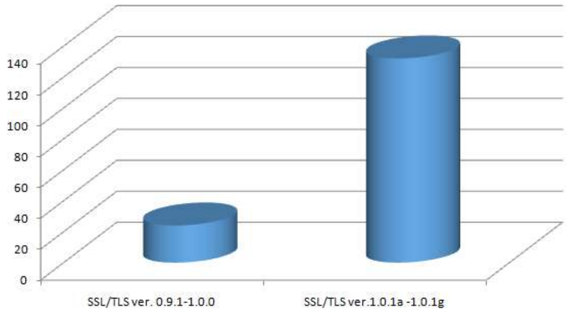

Fig. 8 Statistics of older versions of SSL/TLS

The effects of "Heartbleed" vulnerability are wide spared. Different sectors are vulnerable to attack. The worldwide popularity of OpenSSL for Internet security has made the effect sever. In the dataset analyzed in this paper, the effected sectors found include Education, Financial, Healthcare, and other services as shown in Table I. In the 132 vulnerable web applications 40 were found belonging to various educational organizations. 54 web applications from financial, 25 from healthcare and 37 from other services are found vulnerable to "Heartbleed" attack.

TABLE I
SECTOR WISE VULNERABILITY

Sector	No. of Vulnerable Web applications
Education	40
Financial	54
Healthcare	25
Other Services	37

The effect of vulnerability indicates lack of maintenance of the web applications. We have also analyzed the vulnerability level of these affected web application. This analysis was based on the sensitive information retrieved during testing the vulnerability. According to the sensitive information retrieved through testing we have grouped the dataset in 4 categories as shown in Figure 9. We have found that 29 percent web application after the attack is implemented reveal session cookie and 22% disclose private cookie used for encryption of the data. User name and password could be retrieved from 17% web application. 32% percent of the web application accepted the Heartbleed attack packet but did not reveal any information.

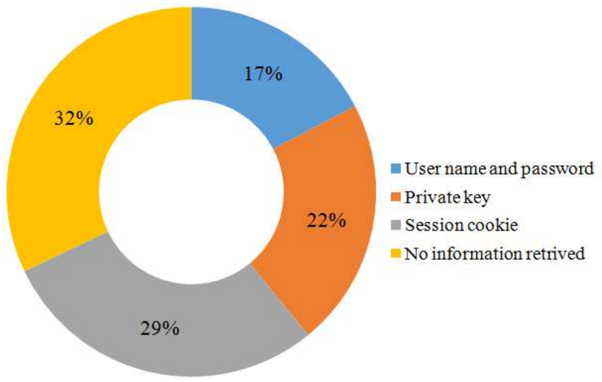

Fig. 9 Statistics of the sensitive information retrieved

For collecting the dataset we have used various online platforms for checking the vulnerability. We have also used few personalized script for retrieving data.

V. Conclusion

Bangladesh is comparatively new in the world of cyber security. The web developers and the leading software development companies are working towards developing a secure web platform. Yet the threats on cyber security are ever growing and it is difficult to keep up. Our analyses of data show just that. Bangladesh started domain registration from year 2012. As Heartbleed was identified in 2014, not a large number of web applications were in developed by then. Thus the effect of the vulnerability is not enormous. Yet this vulnerability exists in the web applications of various sectors. This is due to the lack of knowledge of the web developers and lack of maintenance of the web applications. As the impact of this vulnerability is massive, the web developers should take is vulnerability into consideration. Among the 1200 tested web application only 156 were found vulnerable. Though not a large number yet vulnerability should not exist as the patch for is readily available. As educational and healthcare web application an affected by this attack, the effect is far reaching. All web application should be updated with the latest version of SSL/TLS. In Bangladesh Online users and transaction is increasing every day. The web developers should consider both old and new vulnerabilities while developing the new web applications and upgrading existing web applications.

References

[1] Z. Durumeric et al., "The Matter of Heartbleed". *Proceedings of the 2014 Conference on Internet Measurement Conference- IMC'14*, Vancouver, BC, Canada, 2014, pp. 475-488.

[2] B.A. Chandra, "A Technical View Of The openssl 'Heartbleed' Vulnerability," 1st ed. IBM, 2014. Web. 23 Feb. 2017.

[3] J. Sass, "The Role of Static Analysis in Heartbleed," 1st ed. SANS Institute, 2015. Web. 23 Feb. 2017.

[4] D. A. Wheeler, *How to Prevent the Next Heartbleed*, [Online]. Available: www.dwheeler.com. N.p., 2017. Web. 23 Feb. 2017.

[5] I. Ghafoor, I. Jattala, S. Durrani and C. Muhammad Tahir, "Analysis of OpenSSL Heartbleed vulnerability for embedded systems," *17th IEEE International Multi Topic Conference 2014*, Karachi, 2014, pp. 314-319.

[6] IuPati, T. HEARTBLEED — WHAT'S THE BIG FUSS?. 1st ed. 2017. Web. 23 Feb. 2017.

[7] M. Carvalho, J. DeMott, R. Ford and D. A. Wheeler, "Heartbleed 101," in *IEEE Security & Privacy*, vol. 12, no. 4, pp. 63-67, July-Aug. 2014.

[8] Charette, Robert N. "Heartbleed Bug Bit Before Patches Were Put In Place". IEEE Spectrum: Technology, Engineering, and Science News. N.p., 2017. Web. 23 Feb. 2017.

[9] Burgula, A. M., D.A. Rachana S, and D. S. V. Lakshmi. "Recent Attacks On SSL". *International Journal of Advanced Research in Computer Science and Software Engineerin*, vol. 4, no. 11, pp. 264-266, 2014.

[10] T.P. Mpof, N. Elisa, and N Gati. "The Heartbleed Bug: An Open Secure Sockets Layer Vulnerability". *International Journal of Science and Research (IJSR)*, vol. 4, pp. 1470-1473, 2014.

[11] "HeartbleedBug". [Online]. Available:Heartbleed.com. N.p., 2017. Web. 23 Feb. 2017.

[12] "Heartbleed Vulnerability Scanner Heartbleed Testing Tool". Sslanalyzer.comodoca.com. N.p., 2014. Web. 23 Feb. 2017.

[13] H. L. McKinley (2003). *Ssl-Tls-Beginners-Guide-1029* [Online]. Available: *https://www.sans.org*. Web. 27 Feb. 2017.