**Word Count:** 1275

**A technical view of theOpenSSL ‘Heartbleed’ vulnerability**

Jiashan Li

University of Calgary

**ABSTRACT**

In 2014, a “Heartbleed” vulnerability was identified in the open source cryptographic library OpenSSL which implements Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols. The vulnerability allowed attackers to read protected memory, including cryptographic keys, login information, and other private data. In this work, I propose a review on cause, fixing method, impact and also the prevention of this serious vulnerability.

**Keywords**

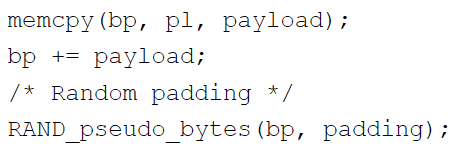
Heatbeat response, bound check, memory access, payload.

**1. BACKGROUND**

The cause of the serious Heartbleed vulnerability is the Heartbeat Extension, which is used to detect whether the connection between client and server is still present, and it was motivated by the need for session management in Datagram TLS (DTLS). It consists of two message types: HeartbeatRequest message and HeartbeatResponse message. 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. When a receiver receives a HeartbeatRequest message, the receiver should send back an exact copy of the received message in the HeartbeatResponse message in which the payload should be the same as data in HeartbeatRequest. The sender verifies that the HeartbeatResponse message is same as what was originally sent. If it is same, the connection is kept alive. If the response does not contain the same message, the HeartbeatRequest message is retransmitted for a specified number of retransmissions.

**2. CAUSE OF VULNERALBILITY**

There is a bug in the memcpy() function in the implementation of the Heartbeat response after receiving Heartbeat request message. Heartbeat response copies the received payload to the Heartbeat response message to verify that the secured connection is still active, without checking if the payload length is same as the length of the request payload data. The line of OpenSSL code with the bug is shown below.



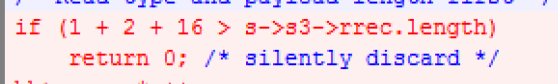
Here “pl” is the information requestor wants to get back from heartbeat response, and “payload” is a number that says how big “pl” is. It works correctly unless “payload” provides wrong length parameter. If the heartbeat request payload length field is set to a value larger than the actual payload, the memcpy code will copy the payload from the heartbeat message and whatever is in memory beyond the end of the payload until the length reaches the request length. A heartbeat request the payload length can be set to a maximum value of 65535 bytes. Therefor the bug in the OpenSSL heartbeat response code could copy as much as 65535 bytes from the memory and send it to the requestor. For example, if the actual payload length is only 4 bytes, but the attacker set the payload length in the Heartbeat response function to 65535 bytes, there will be 65531 bytes customers’ private information includes in the data back to the attacker. The contents of the retreived memory depends on what's in memory in the server at the time, but could potentially contain usernames, passwords, session IDs or secret private keys or other sensitive information. The attackers could try time after time until they get useful data. This serious memory leak is now known as Heartbleed bug.

**3. IMPACT**

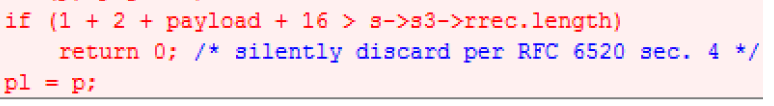
This “Heartbleed” vulnerability may lead to a large quantity of impacts. It affects any service that used OpenSSL to facilitate TLS connections, including popular web, mail, software, applications, and database servers. The HTTPS protocol used to encrypt Internet traffic was weak during this period, such as Yahoo, Google, Facebook, online banking, and the thousands of other vulnerable targets on the web. Attackers get into the memory of security systems and gather up whatever personal information was there, including usernames, passwords, and the actual content of accounts such as credit card data or other sensitive personal information. Sometimes it will be your phone number and hackers will send you a fack message contains a fishing link and cheater will make a phone call to seduce you to pay for something which actually doesn’t exist.

**4. CODE FIXING**

The fixing code is adding a bounds check for the payload length. There are two parts for the fixing of the bug. First, it checks if the length of the payload is zero or not. It will discard the message if the payload length is 0. The source code is as shown below.



Second, 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. The code excerpt that performs this task is shown below.



**5. HEARTBLEED PREVENTIONS**

Although the bug is fixed, it is still dangerous for the users providing their private information. An attacker having gained user information may pretend the material's owner after the victim has patched Heartbleed, as long as the material is accepted, for example, until the password is changed or the private key revoked. Heartbleed is still a critical threat to confidentiality. Attacker impersonating a victim can also change data such as user name and password and unauthorized use the credit card until you notice it. Indirectly, Heartbleed's consequences may thus go far beyond a confidentiality breach for many systems. So it is really important to prevent the loss caused by the vulnerability.

Here are some methods to prevent and avoid the vulnerability. All Heartbleed vulnerable systems should upgrade to OpenSSL 1.0.1g as soon as possible. If you are not sure whether an application you want to access is Heartbleed vulnerable or not, try some Heartbleed detector tools such as Tripwire SecureScan and SSL Diagnos. Do not log into accounts from afflictedsites until you're sure the company has patched the problem. Typically, Once you’ve got confirmation of a security patch, change your password on a regular basis and always use strong password, try not to use the same password in all sites to cut down the loss of the vulnerability. What’s more, you should email or call a customer service of a small company who keeps your private data to make sure the information are safe there. The high-profile company like Google and Yahoo certainly know the problem, but small businesses may not be aware of that. Keep your eyes open on such news of security vulnerabilities and on the financial statement in the next few days. Even after taking all these steps, Heartbleed could also affect browser cookies, which could track users’ activities online, including which websites they are visiting. So even just browsing a vulnerable web page without logging in is unsafe. We might have to stay away from the internet entirely for sometime while things settle.

**6. CONCLUSION**

In conclusion, the Heartbleed has a serious effect to most of the website and our life. It has shaken the Internet community on its dependency on the open source software. Even though OpenSSL is a very popular library, it was not properly scrutinized. We can not trust anything via websites or software applications. It will be hard to close this trust gap as Heartbleed will always remind us. But meanwhile it provides the developers great experience to avoid the similar vulnerabilities in the future development. As the rapid development of technology and network, more and more people choose to keep their information in this virtual world. So property safety and life security in the real world will be a tendency for hackers to attack online. It is more important to test and check the code carefully than pursuing the speed of developing and implementing the software. We have to care more about the details not the quantity. It’s our responsibility towards creating more secured system by industry, organizations, developers and the open source community.

# 7. REFERENCES

1. Zakir Durumeric , James Kasten, David Adrian, J. Alex Halderman, Michael Bailey, Frank Li, Nicholas Weaver, Johanna Amann, Jethro Beekman, Mathias Paye, Vern Paxson. “The Matter of Heartbleed”. University of Michigan, University of Illinois, Urbana Champaign.
2. Bipin Chandra, “A technical view of theOpenSSL ‘Heartbleed’ vulnerability”, A look at the memory leak in the OpenSSL Heartbeat Implementation, Version 1.2.1, May 13, 2014.
3. Wikipedia, <https://en.wikipedia.org/wiki/Heartbleed>
4. Eric Limer, “How Heartbleed Works: The Code Behind the Internet's Security Nightmare”, September 9, 2014, <http://gizmodo.com/how-heartbleed-works-the-code-behind-the-internets-se-1561341209>
5. Chris Williams,Anatomy of OpenSSL's Heartbleed: Just four bytes trigger horror bug,9 Apr 2014 at 15:21. <http://www.theregister.co.uk/2014/04/09/heartbleed_explained/>