“Heartbleed”: A Misuse Pattern for the OpenSSL Implementation of the SSL/TLS Protocol

ALI ALKAZIMI, Florida Atlantic University
EDUARDO B. FERNANDEZ, Florida Atlantic University

Transport Layer Security (TLS) is the successor of the Secure Sockets Layer (SSL) protocol, a cryptographic protocol that provides a secure communication channel between a client and a server. Its secure communication prevents an attacker from eavesdropping an established client-server connection and it is used in most Internet communications for enabling secure web browsing. OpenSSL is an open-source implementation of the SSL/TLS protocol; it has several security issues and Heartbleed is one of its most serious threats. This vulnerability results from improper input validation (lack of bounds check) in the implementation of the TLS heartbeat extension. Heartbeats are used to check if nodes are working. We present here the Heartbleed misuse pattern for OpenSSL, which allows the attacker to obtain sensitive data from servers by modifying the length of the heartbeat message body.

Categories and Subject Descriptors: D.2.11 [Software Engineering] Software Architectures–Patterns; D.5.1 D.4.6 [Security and Protection] Authentication

General Terms: Design

Keywords: OpenSSL, security, misuse patterns, security patterns.

1. INTRODUCTION

Transport Layer Security (TLS) is the successor of the Secure Sockets Layer (SSL) protocol, a cryptographic protocol that provides a secure communication channel between a client and a server. TLS establishes a point to point communication between client and server that uses encryption to provide message confidentiality and authentication. The encrypted messages may be object of attacks that include message reading, modification, and interruption. In spite of the fact that TLS Handshake protocol has been verified (Diaz et al., 2004), many vulnerabilities have been found in actual implementations.

OpenSSL is an open-source cryptographic library for the implementation of SSL/TLS. The OpenSSL main library was written in the C programming language and has had many released versions. As of 2014, two thirds of all web servers used OpenSSL to secure end-to-end communications (Goodin 2014). OpenSSL has had several notable vulnerabilities and the Heartbleed vulnerability has had the most serious impact (Durumeric et al. 2014). Heartbleed is a vulnerability of the heartbeat extension of OpenSSL, where a message is sent to another node that is echoed back to assure the sender that the receiver is working (Carvalho et al. 2014). A Heartbeat is a well-known reliability mechanism, described by a pattern in (Buckley and Fernandez 2009). This vulnerability results from improper input validation (lack of bounds check) in the implementation of the TLS heartbeat extension (OpenSSL, Wikipedia). Heartbleed is registered in the Common Vulnerabilities and Exposures system as CVE-2014-0160 (CVE). While this extension is an implementation feature of OpenSSL, it is possible that other protocols may use heartbeats in the future, in which case designers should know how to avoid a potential vulnerability. This possibility makes this attack a pattern as opposed to just an incident.

Author's address: Ali Alkazimi, email: aalkazi@fau.edu; Eduardo B. Fernandez, ed@cse.fau.edu, Dept. of Computer and Elect. Eng. and Computer Science Florida Atlantic University, Boca Raton, FL 33431, USA.

In order to design a secure system, we first need to understand the possible threats to the system. For that reason, we introduced the concept of misuse pattern (Fernandez 2007) which describes from the attacker's point of view how an information misuse is performed. Misuse patterns define which architectural units are used by the attack, how to stop the attack by providing countermeasures, and provide forensic information in order to trace the attack once it happens. Misuse patterns are useful for system designers and network administrators to know how vulnerabilities in a system can be exploited. We present here a misuse pattern to describe the Heartbleed attack. Our audience includes architects, system designers, web developers, network administrators, testers, and researchers

Section 2 presents the template we use to describe misuse patterns (Fernandez 2013); which is based on the POSA template (Buschmann et al., 1996), with tailoring of some sections. In Section 3, we present a threat pattern for the Heartbleed. Section 4 presents some conclusions and possible future work.

2. TEMPLATE FOR MISUSE PATTERNS

Since misuse patterns are not well known we present first the template that we use to describe them (Fernandez et al. 2007), which is based on the template used by (Buschmann et al. 1996).

2.1 Name

The name of the pattern should correspond to the generic name given to the specific type of threat in standard attack repositories, such as OWASP, CVE, or Bugzilla.

2.2 Intent or thumbnail description

A short description of the intended purpose of the pattern (the goals of the attacker).

2.3 Context

It describes the generic environment including the conditions under which the attack may occur. This may include minimal defenses present in the system as well as standard vulnerabilities of the system.

2.4 Problem for the attacker

From an attacker's perspective, the problem is how to find a way to attack the system. The forces indicate what factors may be required in order to accomplish the attack and in what way; for example, which vulnerabilities can be exploited.

2.5 Solution

This section describes the solution to the attacker's problem, i.e., how the attack can reach its objectives and the expected results of the attack. UML class diagrams show the system units involved in the attack. Sequence or collaboration diagrams show the exchange of messages needed to accomplish the attack.

2.6 Affected system components (Where to look for evidence)

The pattern should indicate in the class diagram the role of all components that are involved in the attack. From a forensic viewpoint, this section describes what information can be obtained at each stage tracing back the attack and what can be deduced from this data.

2.7 Known uses

Specific incidents where this attack occurred are preferred but for new vulnerabilities, where an attack may have not occurred, specific scenarios for the potential attack are enough.

2.8 Consequences for the attacker

Discusses the benefits and drawbacks of the pattern from the attacker's viewpoint. This list should match the forces.

2.9 Countermeasures

Describes the security measures necessary in order to stop, mitigate, or trace this type of attack. This implies an enumeration of which security patterns or other practical measures that are effective against this attack.

2.10 Related Patterns

Discusses other threat patterns with different objectives but performed in a similar way or with similar objectives but performed in a different way.

3. HEARTBLEED

3.1 Intent

In the Heartbleed attack, the attacker modifies the length of the payload of the heartbeat to receive a much longer amount of information from the server (instead of just a short message). The attacker will be able to read protected memory from web servers running one of the OpenSSL vulnerable versions and thus maybe collect some sensitive information.

3.2 Context

In an OpenSSL secure connection, the person's device that visits a website sends a "heartbeat request" message periodically during the lifetime of the connection. The message contains an arbitrary field that defines the payload length "payload_length". The corresponding "hearbeat_response" message is sent from the destination website on the other end that has the same copy of the "payload_length" to make sure that the connection is still active (Carvalho et al. 2014). Figure 1 shows a class diagram for the basic architecture of OpenSSL secure communication after establishing the connection with the client and the server to set a Secure Channel. The client initiates a Heartbeat Request and sends it to the Server. OpenSSL takes the request through the Secure Channel that was established during the connection then sends back the Heartbeat Response to the Client.

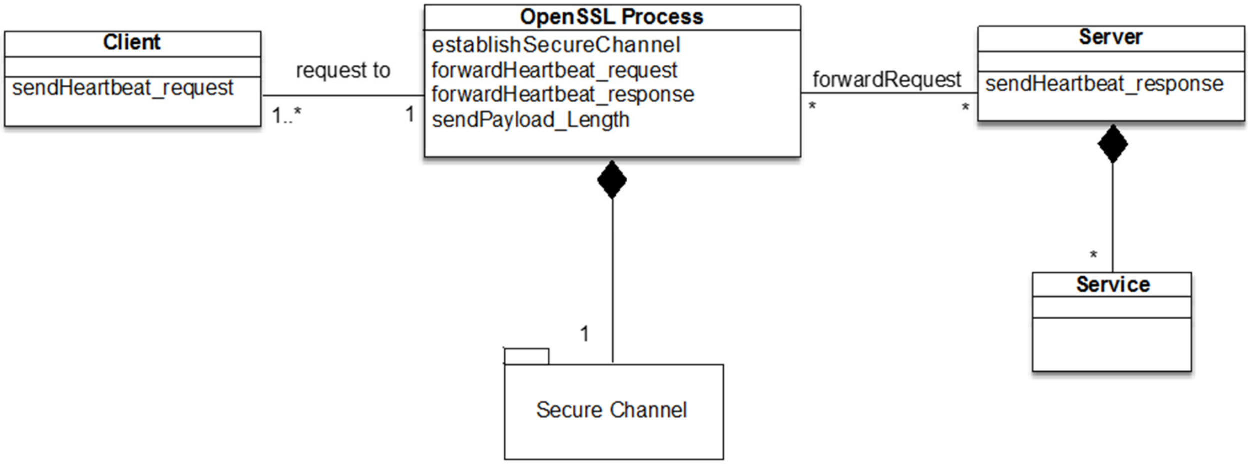

Fig. 1. Class diagram for Heartbeat connection in OpenSSL

3.3 Problem for the attacker

From the attacker's perspective, the problem is getting to execute a script in a victim's browser to modify the payload message between the client and the server which gives the attacker access to possible sensitive confidential information without leaving a trace of the attack.

The attack can be performed by taking advantage of the following vulnerabilities:

- The client's request can be intercepted and modified.
- Some data in the server's memory can be read. That means the attacker can get some sensitive information from the server.
- The system is using an outdated security policy for OpenSSL which makes it vulnerable to this attack.

3.4 Solution

As mentioned previously, the OpenSSL library has vulnerabilities that were inherited from previous versions of the SSL/TLS protocols. These vulnerabilities allow the attacker to read the memory content of the server by intercepting and modifying the exchanged heartbeat message between the client and the server. The problem lays in the vulnerability of the "Payload_Length" message that the client sends to the server. This message can be modified by the attacker that sends the "Payload_length" message which leads to controlling the length of data in the "Heartbeat_request" message and the actual parent length, located in the SSL3 record field that was implemented in the OpenSSL library and never checked (Williams 2014). For example, the attacker sends a four-byte actual parent length of "Heartbeat_request" including a single byte "Payload_length". The attacker gives a false number in the "Payload_length" field and claim it is 65535 bytes in size. The victim reads the 65535 bytes from its own memory, starting from "Heartbeat_request" payload and copies it into a suitably sized buffer to send back to the attacker in the "hearbeat_response" message (Williams 2014).

3.4.1 Structure (Affected System Components)

Figure 2 shows a class diagram for compromising a server with Heartbleed vulnerability. The Attacker, who impersonates the Client, sends requests to the SSL Process which links it to the Server to access Services. The SSL Process sets up a Secure Channel between the Client and the Server.

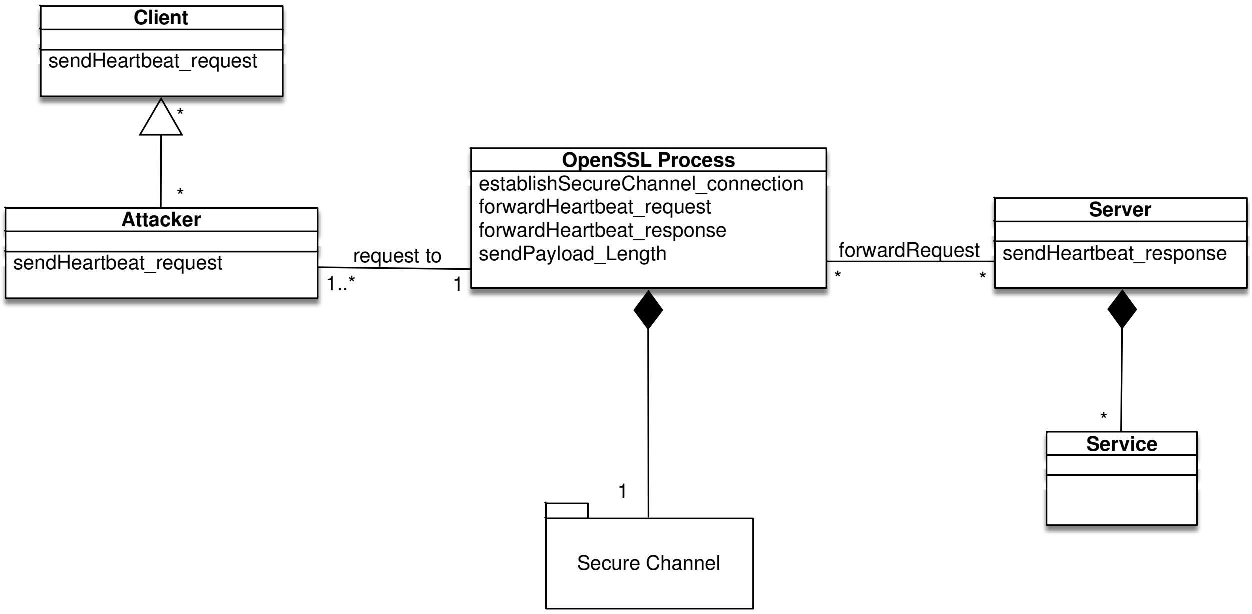

Fig. 2. Class diagram showing the affected units.

3.4.2 Dynamics

The following use case describes the sequence of the attack.

Use Case: Heartbleed attack (Figure 3)

<u>Summary:</u> The Attacker, who impersonates the client, sends a Heartbeat request with a "Modified_Payload_Length" that includes a false length in order to get more information from the attacked server. The OpenSSL process receives the input that includes the false length and sends back information located in the server's memory which may include sensitive information.

Actor: Attacker

Description:

- The Client sends a Heartbeat request including a payload and its length to the server.
- The Attacker intercepts the Heartbeat message and modifies the Payload Length (to its maximum value, 64K) and waits for the server's response. The server does not know whether the client on the other side of the connection is the actual client or an impostor and responds sending an amount of information defined by the Modified Payload Length (long response).

Postcondition:
The attacker may have obtained some sensitive information for the attacker.

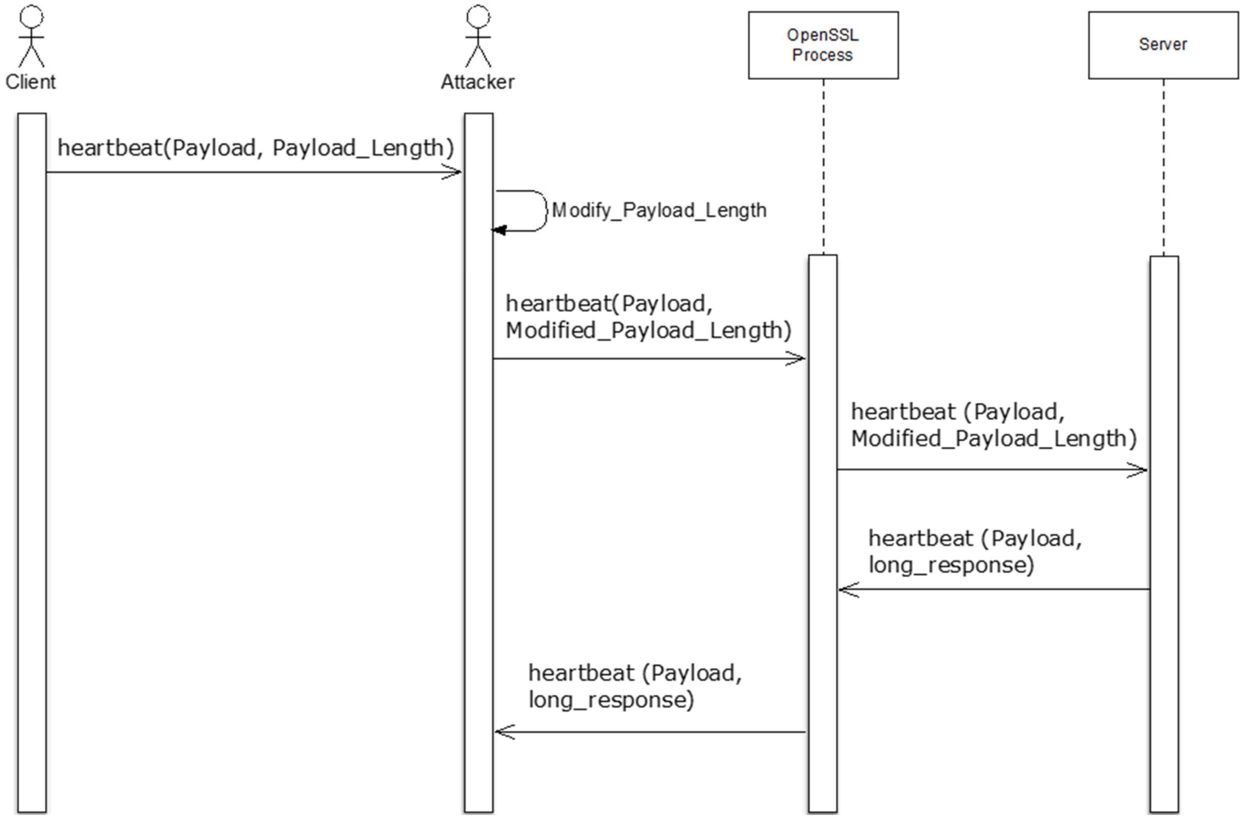
Client
Attacker
OpenSSL Process
Server
heartbeat(Payload, Payload_Length)
Modify_Payload_Length
heartbeat(Payload, Modified_Payload_Length)
heartbeat (Payload, Modified_Payload_Length)
heartbeat (Payload, long_response)
heartbeat (Payload, long_response)

Fig. 3. Sequence diagram for use case to get server information

3.5 Affected system components (Where can we find evidence of this attack?)

- This attack does not leave any trace even if we check the attacked server's log.

3.6 Known Uses

Some potential places where this pattern could have been used are the following:

- Web servers like Apache and NGINX are the most used open source web servers using OpenSSL.
- Around 66% of HTPS sites were vulnerable (Durumeric et al. 2014; Heartbleed 2014).
- (Al-Bassam 2014) completed a vulnerability scan of the Alexa Top 10,000 domains on April 8, 2014 at 16:00 UTC and found 630 vulnerable sites, 3,687 supporting HTTPS but not vulnerable, and 5,683 not supporting

HTTPS. The sites include Yahoo, Stack Overflow, Flickr, and some other websites that were found vulnerable (Durumeric et al. 2014)

Specific incidents include (Wikipedia):

- Security keys were stolen from Community Health Systems in August 2014, the second-biggest for-profit U.S. hospital chain in the United States, compromising the confidentiality of 4.5 million patient records
- The Canada Revenue Agency reported a theft of Social Insurance Numbers belonging to 900 taxpayers, and said that they were accessed through an exploit of the bug during a 6-hour period on April 8, 2014

3.7 Consequences

The success of this attack results in:

- The attacker can impersonate a client or even be a legitimate client, access the server and view sensitive information.
- Once the attack is successful, the attacker may get the user credentials and can use those in attacking other systems.

Disadvantages include:

- If the attacker does not know that the server is not using a vulnerable OpenSSL version, he might be identified by using the logging system.

Failure of the attack

The implementations of OpenSSL from version 1.0.1 until before version1.0.1g do not handle Heartbeat extension packets.

Affected Components

When the attack is completed, several components will be affected:

- Server Certificate: When the attack is successful, the attacker can identify which certificate the server is using and use this information for future attacks.
- The data retrieved by the attacker may be used to change the configuration of the server and make it vulnerable to other attacks.

3.8 Countermeasures

There are several possible recommended mitigations:

- Inspect client Heartbeat request and reject it when the response length is longer than the one in the request.
- Version 1.0.1g of OpenSSL adds some bounds checks to prevent the buffer over-read (OpenSSL, Wikipedia).
- Regenerate new private keys for the client and the server during the active connection.
- Revoke old certificates to avoid attackers from using them.
- Message Authentication to prevent the messages from being modified.
- Sender Authentication to insure the non-repudiation of the messages.
- Reject client request to connect through OpenSSL vulnerable versions.
- Consequences of Heartbleed may remain even after the vulnerability was fixed. The system’s integrity must be carefully checked as well as certificates or keys that might have been compromised.

(Wheeler 2014) describes several measures to detect or mitigate the effects of Heartbleed, including negative testing, using a safer language, and others. He also mentions formal methods to specify and verify the protocol;

however, although the basic protocol was formally verified (Diaz et al. 2014) extensions like Heartbeats are added later and require a new versification.

3.9 Related Patterns

- Transport Layer Security (TLS) Protocol: This pattern presents the security measures of the TLS protocol (Kumar and Fernandez 2012)
- The Authenticator pattern (Brown et al. 1999), which describes the procedure of client server authentication in a distributed setting.
- This attack is a variety of Man-in-the-Middle attack.

4. DISCUSSION AND CONCLUSIONS

The misuse pattern presented in the paper can give a better understanding of the Heartbleed OpenSSL library vulnerability and how attacks can be performed on it. This attack is not possible anymore since almost all the implementations of OpenSSL have been patched (Durumeric et al. 2014). However, it is paradigmatic in that it is a variety of Man-in-the-Middle attack and the idea could be applied to other situations.

Future work will include writing patterns for other attacks on the SSL/TLS handshake protocol such as Triple Handshake Attack (Bhargavan et al 2014; Green 2014). In fact, Heartbleed and Triple Handshake are based on a common type of vulnerability: in both cases more information than needed is sent back by the server.

ACKNOWLEDGEMENTS

Our shepherd, Antonio Maña, provided valuable suggestions that improved this paper. The participants in the PLoP workshop also provided valuable suggestions.

REFERENCES

Ajoy Kumar and Eduardo B. Fernandez. 2012. A Security Pattern for the Transport Layer Security (TLS) Protocol. In *Proceedings of* the 19th. International Conference on Pattern Languages of Programs (PLoP2012).

Chris Williams. 2014. Anatomy of OpenSSL's Heartbleed: Just four bytes trigger horror bug. Retrieved February 25, 2016 from http://www.theregister.co.uk/2014/04/09/heartbleed_explained/

CVE, CVE-2014-0160. Retrieved June 2, 2016 from http://CVE Mitre.org

Dan Goodin. 2014. Critical crypto bug in OpenSSL opens two-thirds of the Web to eavesdropping. Retrieved February 22, 2016 from http://arstechnica.com/security/2014/04/critical-crypto-bug-in-openssl-opens-two-thirds-of-the-web-to-eavesdropping/

David A. Wheeler. 2014. Preventing Heartbleed, Computer, vol. 47, no. 8, pp. 80-83, August 2014, IEEE. DOI: 10.1109/MC.2014.217

Eduardo B. Fernandez, Juan Pelaez, and Maria Larrondo-Petrie. 2007. Attack patterns: A new forensic and design tool, Procs. of the Third Annual IFIP WG 11.9 Int. Conf. on Digital Forensics, Orlando, FL, Jan. 29-31, 2007. www.cis.utulsa.edu/ifip119. Chapter 24 in Advances in Digital Forensics III, P. Craiger and S. Shenoi (Eds.), Springer/IFIP, 2007, 345-357

F. L. Brown, James DiVietri, Graziella Diaz de Villegas, and Eduardo B. Fernandez. 1999. The Authenticator pattern. In *Proceedings of* the International Conference on Pattern Languages of Programs (PLoP99).

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. 1996. *Pattern-Oriented Software Architecture - Volume 1: A System of Patterns*. Wiley Publishing.

Gregorio Diaz, Fernando Cuartero, Valentin Valero, and Fernando Pelayo. 2004. Automatic verification of the TLS handshake protocol. In *Proceedings of the 2004 ACM symposium on Applied computing* (SAC '04). ACM, New York, NY, USA, 789-794. DOI:http://dx.doi.org/10.1145/967900.968063

Heartbleed. (2014). Heartbleed Bug. Retrieved February 23, 2016 from http://heartbleed.com/

Ingrid Buckley, and Eduardo B. Fernandez. 2009. Three patterns for fault tolerance. In *Proceedings of* the OOPSLA MiniPLoP, Orlando, FL October 26, 2009.

Karthikeyan Bhargavan, Antoine Delignat-lavaud, Alfredo Pironti and Pierre-yves Strub. 2014. Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. In Proceedings - IEEE Symposium on Security and Privacy (pp. 98–113). http://doi.org/10.1109/SP.2014.14

Marco Carvalho, Jared Demott, Richard Ford, David A. Wheeler. 2014. Heartbleed 101. IEEE Security and Privacy, vol. 12, no. 4, pp. 63–67. DOI:http://doi.org/10.1109/MSP.2014.66

Matthew Green. 2014. Attack of the Week: Triple Handshakes (3Shake). Retrieved May 11, 2016 from http://blog.cryptographyengineering.com/2014/04/attack-of-week-triple-handshakes-3shake.html

Mustafa Al-Bassam. 2014. Top Alexa 10,000 Heartbleed Scan. Retrieved April 14, 2014 from https://github.com/musalbas/heartbleed-masstest/blob/ 94cd9b6426311f0d20539e696496ed3d7bdd2a94/ top1000.txt.

OpenSSL: Cryptography and SSL/TLS toolkit, https://www.openssl.org/

Wikipedia. 2016. Heartbleed. Retrieved July 24, 2016 from https://en.wikipedia.org/wiki/Heartbleed

Zakir Durumeric, James Kasten, David Adrian, J. Alex Halderman, Michael Bailey, Frank Li, Nicolas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, and Vern Paxson. 2014. The Matter of Heartbleed. In Proceedings of the 2014 Conference on Internet Measurement Conference (IMC '14). ACM, New York, NY, USA, 475-488. DOI:http://dx.doi.org/10.1145/2663716.2663755