Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Failure of ensuring the safety of web infrastructure components could cause confidential information leakage and fatal damages. In April 2014, a simple careless mistake causes a security nightmare for the Internet, the Heartbleed Bug[1]. It is a serious vulnerability in the OpenSSL cryptographic library[2], an extensively used Transport Layer Security (TLS) protocol implementation, that is disclosed by Google's Security team and Codenomicon separately. This vulnerability allows attacker to steal sensitive information from HTTPS services, and even impersonate services and users. About half a million (17.5%) of trusted HTTPS websites was influenced, including banking services, email services, cloud storage, and any web services using the vulnerable versions of OpenSSL, according to the Secure Sockets Layer (SSL)[3] survey conducted by the Netcraft [1].

This security weakness occurred in the TLS "heartbeat" extension, which is a keep-alive feature intended to ensure the connection by sending an arbitrary payload to and exacting to receive the exact same copy from the other end. However, the implementation wrongly trust the non-sanitised user input and simply use it as the length parameter of `memcpy()` without any bounds check [3]. This missing of a proper input validation makes the reading of the payload beyond the end of the buffer. Hence, attackers are able to extract the memory contents, protected by the vulnerable versions of OpenSSL in the server, including identifications of service providers, secret keys for encrypting communications and users login information. In other words, they can steal data from communications directly and impersonate the services and users.

[1]The Heartbleed Bug: http://heartbleed.com/
[2]The OpenSSL github repository: https://github.com/openssl/openssl
[3]Secure Sockets Layer: the predecessor of TLS, often be called TLS/SSL

However, such a serious security vulnerability caused by buffer overruns can be easily detected using formal software verification techniques, for example, Bounded Model Checking, which is demonstrated by the study of M. Voelter et al. [2]. Therefore, not only the approach itself must be formally proven secure, but also the corresponding implementations are necessary to be verified.

Consider that the data integrity and privacy of Internet communications for the online applications and virtual private networks (VPNs) are provided by TLS/SSL. Therefore, ensuring the safety of the implementation of TLS/SSL is particular importance to securing the web. This project is motivated by the necessity of insuring such critical software components and the success of software verification, hence, we aim at verifying memory safety of the Amazon's s2n, which is an light-weight implementation of the TLS protocol, by using a bounded model checking tool, CMBC.

1.2 Objectives

The main objective for this project is to verify the memory safety of the Amazon's s2n, which is an light-weight implementation of the TLS protocol, by using a bounded model checking tool, CMBC.

The research questions we would ask as follows:

1. Is the implementation memory safe?
2. Does the component based on any standard approach, such as encryption, protocol?
3. If the component based on some standard approaches, has it been implemented correctly?
4. Is there any undiscovered bugs?
5. Has the component been code reviewed or security audited?

1.3 Main Achievements

Main achievements in this project include:

1.4 Dissertation Structure

This dissertation is composed of ?? chapters, fully describing the background theory and demonstrating all my work for this project.

Chapter 1 is ...

Bibliography

[1] Netcraft. Half a million widely trusted websites vulnerable to heartbleed bug. http://news.netcraft.com/archives/2014/04/08/half-a-million-widely-trusted-websites-vulnerable-to-heartbleed-bug.html, April 2014.

[2] Markus Voelter, Zaur Molotnikov, and Bernd Kolb. Towards improving software security using language engineering and mbeddr c. In *Proceedings of the Workshop on Domain-Specific Modeling*, DSM 2015, pages 55–62, New York, NY, USA, 2015. ACM.

[3] C. Williams. Anatomy of openssl's heartbleed: Just four bytes trigger horror bug. http://www.theregister.co.uk/2014/04/09/heartbleed_explained/l.