A Snort-based Approach for Heartbleed Bug Detection
Yu Zhang¹, Qingzhong Liu², Yanling Liu¹

¹Department of Computer Science, Hainan Normal University, Haikou, China
²Department of Computer Science, Sam Houston State University, Houston, USA
bullzhangyu@gmail.com

Keywords: Heartbleed, Bug detection, Snort, OpenSSL

Abstract. Heartbleed is a catastrophic bug in the OpenSSL which is a widely used implementation of the Transport Layer Security (TLS) protocol on the Internet. The Heartbleed bug which allows attackers to get sensitive information such as the secret keys, the user names, passwords and the actual content is a serious security threat to E-commerce. We proposed a Snort rule-based approach to detect the heartbleed bug. Experiment results show that the presented approach is suit for detecting the Heartbleed bug.

1 Introduction
A newly discovered bug in widely used Web encryption technology has made data on many of the world’s major websites vulnerable to theft by hackers in what experts say is one of the most serious security flaws uncovered in recent years. This catastrophic Internet bug is called Heartbleed [1]. The Heartbleed bug [2] allows attackers on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop communications, steal data directly from the services and users and to impersonate services and users. So this bug is catastrophic to E-Commerce applications in which the OpenSSL is widely used [3].

The Heartbleed bug is the result of a mundane coding error in OpenSSL, the world's most popular code library for implementing HTTPS encryption in websites, e-mail servers, and applications [4]. The result of a missing bounds check in the source code, Heartbleed allows attackers to recover large chunks of private computer memory that handle OpenSSL processes. The leak is the digital equivalent of a grab bag that hackers can blindly reach into over and over simply by sending a series of commands to vulnerable servers. The returned contents could include something as banal as a time stamp, or it could return far more valuable assets such as authentication credentials or even the private key at the heart of a website's entire cryptographic certificate.

After carefully study the source codes of Heartbleed, we proposed a Snort rule-based approach to detect the Heartbleed bug by checking the network flows. The experiment results show that the proposed method is easy and efficient in Heartbleed bug detection.

In the remaining sections of the paper, we first describe the Heartbleed bug including the OpenSSL hierarchy in section 2. Then we discuss the proposed Snort-based approach to detect the Heartbleed bug in section 3. Finally, we state our conclusions in section 4.

2 Heartbleed Bug
2.1 OpenSSL
The OpenSSL Project [5] is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. The project is managed by a worldwide community of volunteers that use the Internet to communicate, plan, and develop the OpenSSL toolkit and its related documentation.
OpenSSL is based on the excellent SSL'ey library developed by Eric A. Young and Tim J. Hudson. The OpenSSL toolkit is licensed under an Apache-style licence, which basically means that you are free to get and use it for commercial and non-commercial purposes subject to some simple license conditions [6].

OpenSSL is a popular cryptographic library used to digitally scramble sensitive data as it passes to and from computer servers so that only the service provider and the intended recipients can make sense of it. The hierarchy of OpenSSL is shown in Figure 1.

Fig. 1 The hierarchy of OpenSSL

The interactive process between the server and the client that use OpenSSL protocol is shown in Figure 2.

2.2 Heartbleed Bug

An information disclosure bug (CVE-2014-0160) [7] has been discovered in 1.0.1 and 1.0.2-beta releases of OpenSSL versions including 1.0.1f and 1.0.2-beta1. This bug is due to a missing bounds check in the handling of the Transport Layer Security (TLS) heartbeat extension. The bug may allow an attacker to retrieve memory in chunks of 64 kilobytes from a connected client or server. This means, an attacker can access sensitive information (read: private keys, login credentials for Internet Banking, Emails, Social Networking sites etc. or contents of encrypted traffic) from a connected client or server by sending specially-crafted TLS “heartbeat” requests [8].

The TLS Heartbeat request module is composed of four data structures including Type, Payload Length, Payload, and Padding (Figure 3).

Fig. 3 The structure of TLS Heartbeat module

As shown in Figure 3, the TLS heartbeat consists of a request packet including a payload; the other side reads and sends a response containing the same payload (plus some other padding).

In the code that handles TLS heartbeat requests, the payload size is read from the packet controlled by the attacker:

```c
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;
```
Here, p is a pointer to the request packet, and payload is the expected length of the payload (read as a 16-bit short integer; this is the origin of the 64K limit per request). pl is the pointer to the actual payload in the request packet.

Then the response packet is constructed:

```c
/* Enter response type, length and copy payload */
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, pl, payload);
```

The payload length is stored into the destination packet, and then the payload is copied from the source packet pl to the destination packet bp.

The bug is that the payload length is never actually checked against the size of the request packet. Therefore, the memcpy() can read arbitrary data beyond the storage location of the request by sending an arbitrary payload length (up to 64K) and an undersized payload.

3 Snort-based Detection

3.1 Snort

Snort [9] is an open source network intrusion prevention system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more.

Snort is an open source network intrusion prevention and detection system (IDS/IPS) developed by Sourcefire. Combining the benefits of signature, protocol, and anomaly-based inspection, Snort is the most widely deployed IDS/IPS technology worldwide. With millions of downloads and nearly 400,000 registered users, Snort has become the de facto standard for IPS [10].

Snort can be configured in three main modes: sniffer, packet logger, and network intrusion detection. In sniffer mode, the program will read network packets and display them on the console. In packet logger mode, the program will log packets to the disk. In intrusion detection mode, the program will monitor network traffic and analyze it against a rule set defined by the user. The program will then perform a specific action based on what has been identified [11].

3.2 Detection Method

As shown in Section 2.2, we know that the Heartbleed bug comes from a missing bounds check in the handling of the Transport Layer Security (TLS) heartbeat extension. So the first important thing to do is to fix the bug, and then to detect the bug.
To fix the Heartbleed bug, we can do as follows (Figure 5).

```
/* read type and payload length first */
if (1 + 2 + 16 > s->s3->rec.length)
    return 0; /* silently discard */
htype = *p+;
n2sp(p, payload);
if (1 + 2 + 16 > s->s3->rec.length)
    return 0; /* silently discard per RFC 6520 sec. 4 */
pl = p;
```

Fig. 5 The fix code of the Heartbleed bug

We can see from above codes that do two things: the first check stops zero-length heartbeats, and the second check checks to make sure that the actual record length is sufficiently long.

To detect the Heartbleed bug, we proposed a Snort-based detection method by inspecting the network traffic with two sets of Snort signatures. The first rule has a higher detection rate but also quite a few false positives (Figure 6).

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 443 (msg:"OpenSSL Heartbleed attack");flowto_server,established; content:"[18 031]; depth: 5; byte_test2: >, 200, 3, big; byte_test2, <, 16385, 3, big; threshold: type limit, track by src, count 1, seconds 600; reference: CVE-2014-0160; class: type: bad-unknown; sid: 21001313; rev:2;)
```

Fig. 6 The rule 1 to detect the Heartbleed bug

The second rule has a lower false positive ratio but might also have a slightly lower detection rate (Figure 7).

```
alert tcp any any -> any (msg:"OpenSSL Heartbeat Response");flow:established;
flowbits:isset,foxxis:session; content:"[18 031]; depth: 2; byte_test1, <=, 3, 2; byte_test1, !=, 2, 1; byte_test2, >, 200, 3; threshold: type limit, track by src, count 1, seconds 600; reference: CVE-2014-0160; class: type: bad-unknown; sid: 21001313; rev:5;)
```

Fig. 7 The rule 2 to detect the Heartbleed bug

### 3.3 Experiment Results

The experiments environment is with Apache version 2.4.7, OpenSSL version 1.0.1f, MySQL version 5.5, and Snort version 2.9.5.5. The experiments are carried on as follows. Firstly, we send the packets with Backtrack5, a famous bug scanner, to our experimental 10 websites without Snort. Secondly, we do the same things but with our proposed Snort detection rules. At last, we analyze the results of above two experiments.

We can get the sensitive information such as phone number, username, applets version. The experiment results are shown in Figure 8 and Figure 9.

```
%.
```

Fig. 8 The experiment result

The comparison experiment results between rule 1 and rule 2 are shown in Table 1. As shown in Table 1, both proposed Snort rules are high detection rate in detecting the Heartbleed bug, though rule 1 is better than rule 2.

Fig. 9 The experiment result
4 Conclusion

The Heartbleed bug allows attackers on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. Our proposed Snort-based detection method can work well to detect the Heartbleed bug.

Furthermore, the Heartbleed is the result of missing bound check. Some popular computer languages like C without memory checks are unsuitable for writing secure code. They need to be restricted to writing a small core system, preferably small enough that it can be checked using formal methods. We cannot end all bugs in software, but we can write using managed code such as C# and Java.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (61262077, 61462025); the Natural Science Foundation of Hainan (613161); the National Students' Innovation and Entrepreneurship Training Program (201211658036).

References


