Graham Dungan
April 11, 2025
UIN: 332001764

CSCE 413: Software Security
Self-Study: Heartbleed

What is the Vulnerability?

Heartbleed was a bug in the OpenSSL cryptography library released in February of 2012, discovered on April 1, 2014, and patched six days later. Heartbleed exploited an overlooked bug in TLS (implemented on OpenSSL) by falsifying the size of heartbeat messages sent to servers to maintain TLS connections. Because of this, malicious users were able to view the contents of the server's RAM, possibly becoming privy to sensitive information that the server was processing. Due to the internet's reliance on this open-source project, an overwhelmingly large portion of the internet was affected by the bug.

What was the Root Cause?

TLS connections are expensive to maintain, so it is in the best interest of servers to terminate connections when possible. Because of this, the TLS protocol had a "heartbeat" extension added in February of 2012. The user (or server) can send heartbeat messages to each other to notify that the connection is still alive and should not be terminated. Otherwise, if no heartbeat is received after a long period, the connection would be closed to save resources.
On April 14, 2014, Neel Mehta of Google's security team disclosed that there was a bug in OpenSSL's code concerning the heartbeat extension. The heartbeat itself can be many sizes, from 1kb to 64kb. As is convention, if the client sends a 64kb heartbeat, the server will respond with the same 64kb heartbeat. While seemingly harmless, any client can forge the payload length field compared to the payload. This is where the bug comes into play- OpenSSL did not verify that the size of the payload was equal to the payload size. Because of this, a malicious client can send a 1b payload disguised as (the payload size is set to) a 64kb payload. The server would then fill in the missing bytes with somewhat random information currently in the RAM. What follows is the vulnerable code snippet,

```
1 hbtype = *p++;
2 n2s(p, payload);
3 if (1 + 2 + payload + 16 > s->s3->rrec.length)
4     return 0; /* silently discard per RFC 6520 sec. 4 */
5 pl = p;
```

As seen here, line 3 only offers a brief "sanity check" verifying that the data received is no greater than the data obtained from the payload. This code does not check that the data received is equal to the data expected.
Because of this discrepancy, a malicious client can continuously send heartbeats to a server and will receive portions of its memory. This could then be used to access sensitive information such as passwords, keys, and other private server attributes (so long as it is in RAM).

What is the Extent of its Impact?

Researchers had found that there were reports of attackers using this flaw at least five months before its discovery. Once Heartbleed was discovered, there were some notable instances of it being used- the Canadian Revenue Agency reported a theft of social insurance numbers, a UK parenting site named Mumsnet was hijacked, and security keys were stolen from Community Health Systems. Given the nature of Heartbleed, it also served as a cultural phenomenon, bolstering support towards securing critical open-source projects.

How to Patch it?

The patch was rather simple- validating that the length of the payload matched the length of the payload size before copying memory. Additional fixes included disabling the heartbeat extension.

How Could it Have Been Prevented?

While this is an instance of a lack of review for critical code, understanding why this code went unreviewed is important. Given that OpenSSL is open-source, only underpaid volunteers have contributed to the codebase. Due to these constraints, bugs and vulnerabilities can be very hard to detect and fix. Following the hyper-dependence of OpenSSL, despite it being a potentially vulnerable library due to these constraints, it put OpenSSL in a position to be critical infrastructure for the internet. Overall, this could have been prevented via more thorough code inspection, an implicit 0-trust policy and scrutiny for all data sent over the internet, and support of open source projects.

References

1. The Heartbleed Bug
2. Heartbleed - Wikipedia
3. Heartbleed: 10 Years of Heartache - John Robustelli
4. Explaining Dirty Cow - Computerphile
5. Heartbleed explained in under 2 minutes - Keith Rozario
6. Heartbleed Vulnerability and Remediation - CyberArk, Machine Identity Security Hub
7. Heartbleed - What Happened? A Bug That Nearly Broke the Internet - Pro Tech Show