The Heartbleed Vulnerability

Authors: Adina-Maria VAMAN, Dragoș IOANA

Table of Contents

OpenSSL

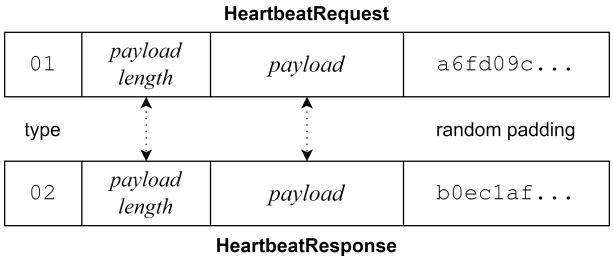
- a popular open-source cryptographic library that implements the SSL and TLS protocols
- it is widely used by server software to facilitate secure connections for web, email, VPN, and messaging services
- it was integrated in popular server products such as *Apache* and *Nginx*

Datagram Transport Layer Security

- usually, TLS must run over a reliable transport channel - typically TCP
- RFC6347 introduced *Datagram Transport Layer Security Version 1.2*, which aims to secure unreliable datagram traffic for application layer protocols that are designed to use UDP transport
- standard implementations of TLS rely on TCP for session management which UDP lacks - the need for *TLS Heartbeat Extension*

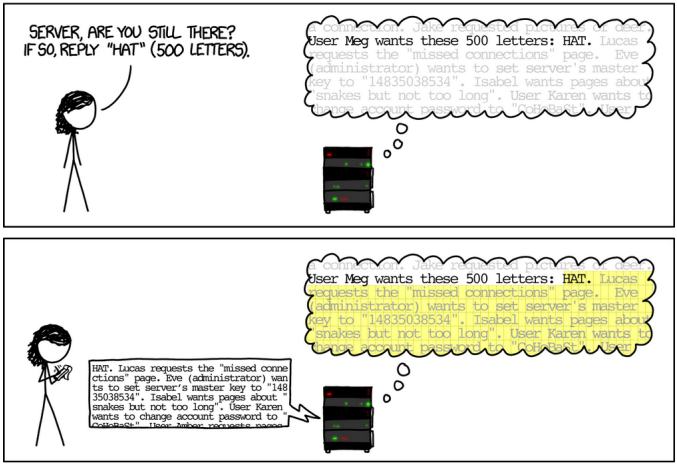
TLS Heartbeat Extension

Figure: Heartbeat package



- allows either endpoint of a TLS connection to detect whether its peer is still present
- either endpoint can send a HeartbeatRequest message to verify connectivity

The Heartbleed vulnerability



The Heartbleed vulnerability

- it allows either end-point to read data following the payload message in its peer's memory by specifying a payload length larger than the amount of data in the *HeartbeatRequest* message

- Because the payload length field is two bytes, the peer responds with up to 216 bytes (around 64 KB) of memory

Timeline

- 03/21 Neel Mehta of Google discovers Heartbleed
- 03/21 Google patches OpenSSL on their servers
- 03/31 CloudFlare is privately notified and patches
- 04/01 Google notifies the OpenSSL core team
- 04/02 Codenomicon independently discovers Heartbleed
- 04/03 Codenomicon informs NCSC-FI
- 04/04 Akamai is privately notified and patches
- 04/05 Codenomicon purchases the heartbleed.com domain
- 04/07 NCSC-FI notifies OpenSSL core team
- 04/07 OpenSSL releases version 1.0.1g and a security advisory

Impact

- it may have affected any service that used OpenSSL to facilitate TLS connections
- any type of confidential data could have been exposed:
  - session cookies
  - private keys
  - server master keys
  - authentication material

Mitigation

- *patching OpenSSL to a version $\geq$1.0.1g*
- changing passwords
- regenerating key pairs
- revoking old certificates
- deleting cookies
- compiling OpenSSL sources with *-DOPENSSL_NO_HEARTBEATS* flag

PoC description and implementation

Exploit architecture:

- vulnerable Apache server using OpenSSL v1.0.1e which exposes an HTML page that saves login data as cookies
- exploit script as source code in C, which does the TLS Handshake and then requests Heartbeat responses to look for server echoing back sensitive information

PoC description and implementation

Live demo