The Heartbleed Vulnerability in OpenSSL

Background & Consequences

Stephan Wiesand
Technical Seminar
Zeuthen, 2014-04-15

HELMHOLTZ GEMEINSCHAFT

DESY

Agenda

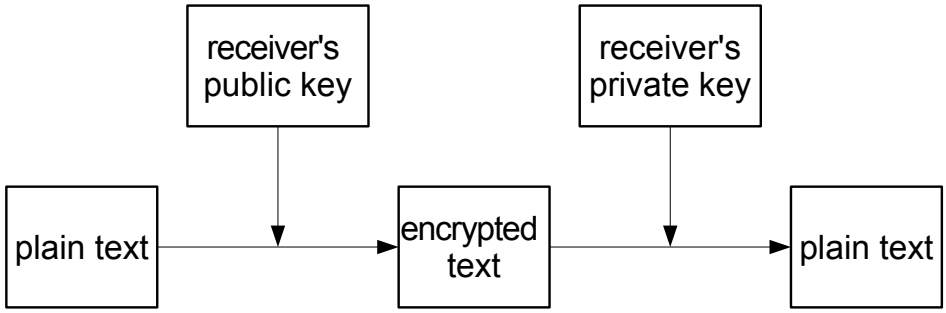
- What's OpenSSL?
  - An Open Source toolkit implementing TLS
- What's TLS?
  - A cryptographic protocol for secure network communication
- What's Heartbeat?
  - A TLS protocol extension
- What's Heartbleed?
  - A name for a bug in OpenSSL's Heartbeat implementation
- What are the consequences?
  - Leakage of random chunks of data from servers & clients
- What now?

DESY

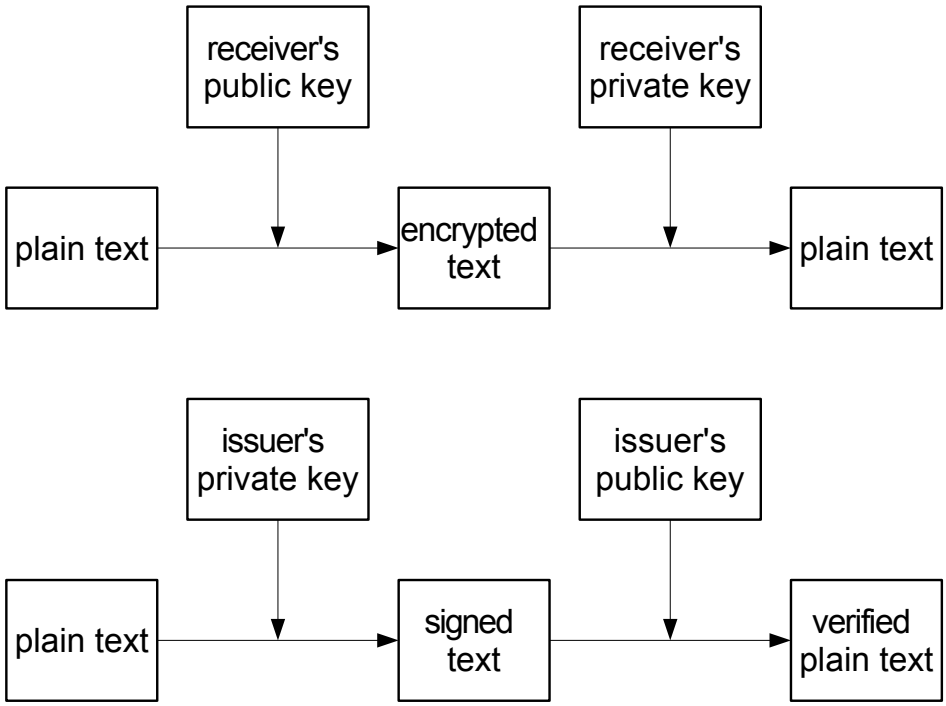
Public Key Cryptography

- based on asymmetric key pairs
  - public key
    - encrypt plain text, can be decrypted with matching private key only
    - verify signatures made with the matching private key
  - private key
    - decrypt messages encrypted by the matching public key
    - sign plain text, to be verified using the matching public key

Sending Data



Sending Data, Issuing & Verifying Signatures



TLS, SSL, OpenSSL

- TLS: Transport Layer Security
  - a protocol defining how to send encrypted data over the wire
    - using X.509 certificates
      - public key + identity information + ...
    - can authenticate servers (& clients) since certificates are signed
      - by (a chain of) Certificate Authorities
        - server.desy.de → DESY CA → DFN CA → Telekom Root CA
- SSL: Secure Sockets Layer
  - the name of older versions of the TLS protocol
- OpenSSL: a toolkit implementing
  - the encryption algorithms (libcrypto)
  - the TLS protocol (libssl)
  - swiss army knife utilities for key/certificate handling and more

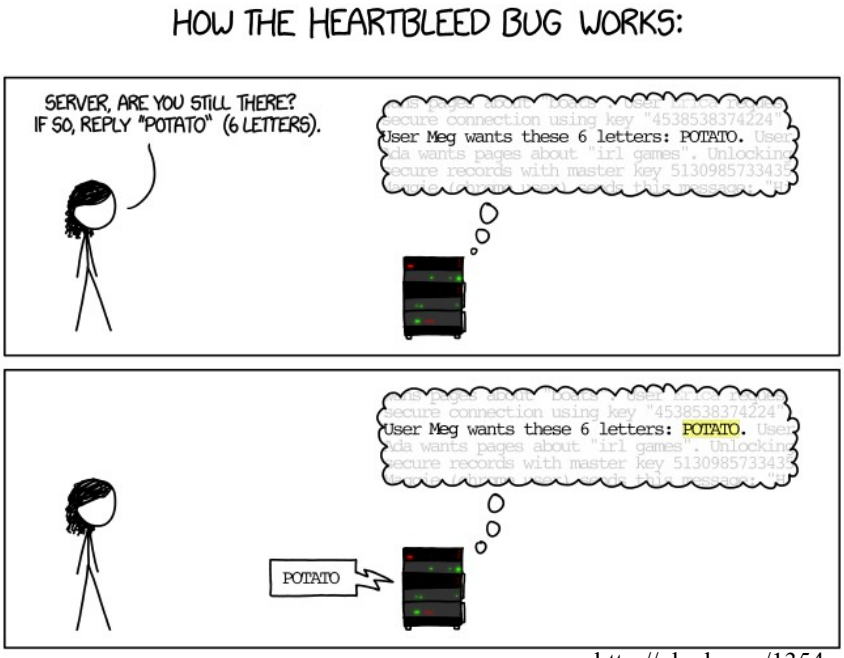
TLS Heartbeat

- Protocol extension to keep alive a connection
  - initially designed for DTLS
    - Datagram TLS, for use over “unreliable” transports like UDP
    - path MTU discovery
    - keep up NAT mappings
  - but implemented for TLS too (which don't necessarily need it)
  - and enabled by default
- What happens:
  - requesting side sends heartbeat request with up to 64 kiB of data
    - “here's nnn bytes of data - please send them back”
  - responding side sends heartbeat response with the same data
- Works in both directions
  - and before a full TLS connection has been set up

Heartbleed: Bug in OpenSSL's Implementation

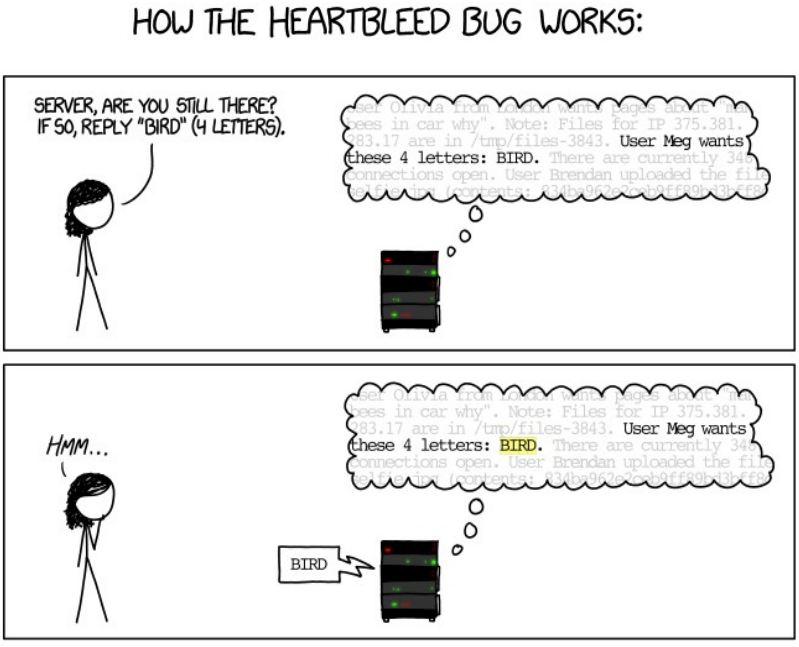
- The responder doesn't check the amount of data actually sent
  - => requestor can trick responder into sending more data than received
- The buffer used for the data isn't cleared before use
  - => it contains whatever was stored there when last used
- OpenSSL implements its own memory management
  - => it's data previously processed by openssl routines
- This data can include, in plain text:
  - payload (document or message contents)
  - private keys
  - session IDs, cookies
  - user/password pairs
- Cheap attack, leaves no traces in the usual logs

Heartbleed Illustrated (1)



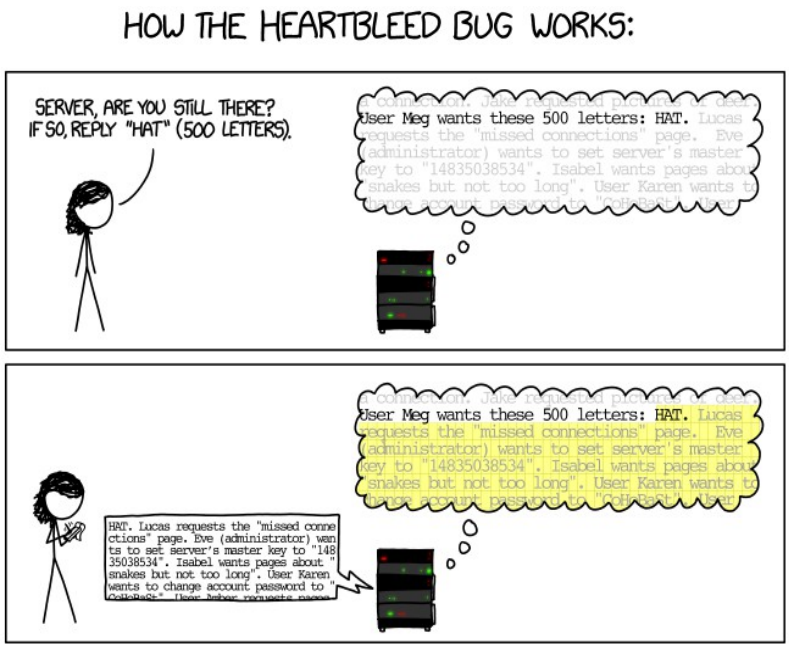
source: http://xkcd.com/1354

Heartbleed Illustrated (2)



source: http://xkcd.com/1354

Heartbleed Illustrated (3)



source: http://xkcd.com/1354

Don't Panic

- OpenSSL is not the only TLS implementation
  - Apple, Microsoft have their own
  - GNUTLS, NSS are other Open Source implementations
    - Firefox & Thunderbird use NSS
  - Java applications usually use JSSE
- Not all OpenSSL versions are vulnerable
  - many services still use pre-heartbeat versions
- Not all software using OpenSSL is vulnerable or easy to attack
  - OpenSSH uses libcrypto, but not libssl - not vulnerable
  - OpenVPN uses libssl, but DTLS - no easy/public exploit yet
- An attacker can't control the location of the buffer in memory
  - leaked data may or may not be valuable

What now? - Servers

- Server software must be patched
- Services must be restarted
- Server certificates should be considered compromised
  - if private key leaked, a successful attacker could
    - impersonate the service
    - decrypt intercepted traffic
- Certificates must be replaced, using a new private key
- Old certificates must be revoked

What now? - Clients

- Clients software must be patched
- Clients must be restarted
- Client certificates could in theory be compromised
  - but only if client connected to a malicious server
    - not known to exist yet

What now? - Users

- User certificates could in theory be compromised
  - but only if user connected to a malicious server
    - not known to exist yet
- Active sessions on a vulnerable service could be compromised
- Passwords used for a vulnerable service could be compromised
- Current recommendation: change passwords
  - if used for a possibly affected service in the past
    - especially if you used the same password for different services
      - and then never do that again!
  - but: do this after the service was secured only
    - changing the password for a vulnerable service now may expose it

Heartbleed Timeline (1)

- March 2012: OpenSSL 1.0.1 released with the bug
  - January 2014: bug lands on SL6 systems
- March 2014: bug discovered at Google and systems patched
- April 1st: bug reported to OpenSSL by Google
  - normal procedure for responsible disclosure starts
    - public announcement deferred to April 9th
- April 7th: bug reported to OpenSSL by Finnish CERT
  - discovered independently by Codenomicon
  - starting 19:30 (GMT+2): bug goes public
    - OpenSSL 1.0.1g release
    - CloudFlare blog post
    - www.heartbeat.com goes online
    - vendors scramble to push out security updates

Heartbleed Timeline (2)

- April 8th
  - 5am updates available for RHEL and CentOS
  - 7am CentOS packages downloaded, resigned and tested for SL6
  - 8:30am contacted D4, IT
  - 8:30am affected services in Zeuthen patched and restarted
  - later press begins buzzing, first exploit scripts circulate
- April 9th
  - client systems patched
- April 11th
  - server certificates replaced, old certificates revoked
  - status report sent to users

Timeline Summary

- The Bug existed for two years
  - but many systems used older versions for a long time
    - many still are (SL5)
- January: SL6 servers could become vulnerable
- Since March: growing number of individuals informed
  - assumption: all wearing white hats
  - some popular services already patched (google)
- Since evening of April 7th: bug public => risk rising sharply
- Since April 8th: vulnerable services may have been exploited

What could have been affected?

- Any service using TLS from recent OpenSSL
  - https
  - imaps
  - smtps
  - instant messaging protocols
  - VPN
  - ...

What was vulnerable

- Zeuthen:
  - Wikis, Confluence, JIRA
    - not using DESY standard passwords
  - OwnCloud test instance
  - SVN
    - when using password authentication and DESY account
    - not when using Kerberos (GSSAPI) authentication
- Others:
  - Google, Facebook, Yahoo, Dropbox, Netflix, Yelp, Flickr, Bing, DuckDuckGo …
  - web.de, gmx.de, Telekom ...

What was not vulnerable

- Zeuthen
  - www-zeuthen, smtp (SL5)
  - imap (dovecot uses OpenSSL, but anticipated such a bug)
- Many services in Hamburg, but no list yet
- Generally
  - SSH
  - Kerberos and services using it for authentication
  - dCache
  - OS X, iOS, Windows (Android: only 4.1.1 is affected)
    - exceptions exist: MacPorts; Apache on Windows uses OpenSSL, …
- Others
  - Apple, Microsoft, Amazon (store, not AWS), PayPal, eBay, ...

Summary

- Many important services were vulnerable, many weren't
- We'll never know what information has leaked
  - but have an idea of how those lists with millions of account/password pairs were possibly generated
- Patch all systems, restart affected software
- Replace & revoke server certificates
- Change passwords ever used for possibly affected services
  - after you have confirmation that the service was patched
    - in doubt, repeat at that point
- Never use a password for more than one service