Software Engineering: Tutorial 13

David Voigt

February 3rd, 2023

Agenda

Fuzzing

Heartbleed

- Critical security bug (buffer over-read) in the OpenSSL cryptography library (2012 - 2014) [a]
- Used in the TLS protocol (most prominently used in the HTTPS protocol)
- Even though many cryptography and other security experts audited the *open source* code, the bug remained for *years*

[a] Relevant xkdc

Figure 1: Source: Wikipedia

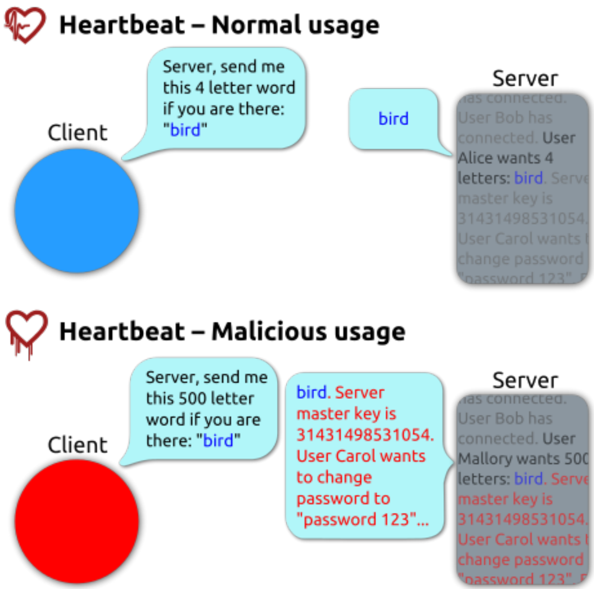
Heartbeat – Normal usage
Client
Server, send me this 4 letter word if you are there: "bird"
bird
Server
has connected. User Bob has connected. User Alice wants 4 letters: bird. Serve master key is 31431498531054. User Carol wants t change password "password 123". F
Heartbeat – Malicious usage
Client
Server, send me this 500 letter word if you are there: "bird"
bird. Server master key is 31431498531054. User Carol wants to change password to "password 123"...
Server
has connected. User Bob has connected. User Mallory wants 500 letters: bird. Serve master key is 31431498531054. User Carol wants t change password "password 123". F

Figure 2: Source: https://en.wikipedia.org/wiki/Heartbleed

Consequences

- A malicious attacker could in theory retrieve up to 64 kB of information from the servers memory
  - e.g. passwords, session cookies (for impersonation), private keys, . . .
- While the bug was fixed the same day it was publicised, it is unkown if this bug was exploited previously
- Even publicly available (open source), audited and high-profile libraries are not immune to bugs

Conclusion

We need to ~~expect~~ test the unexpected!

$\Rightarrow$ Use *fuzzing* for generating random, unexpected and/or malformed data as input for software as tests

Recap: Fuzzing

- (Coverage-guided) *fuzzing* **automatically** generates unexpected, malformed and/or random data
- This data is provided as input for program under test
- The program's behavior is monitored for crashes or other *undefined behavior*
- **Goal**: Validate that a program is robust against all kinds of different input and does not reveal *undefined behavior*

Exercises

Now it is **your** turn to write a fuzzer!

https://github.com/se-tuebingen-exercises/tut7-exercise13