CISC327 - Software Quality Assurance

Lecture “review4”

Review for Mini-Exam #4
Announcements

• Mini-Exam #4 accommodations:
  – Awaiting confirmation of room reservations
Likely topics/questions on mini-exam #4

• From Lecture 24:
  – No specific questions that depend on content from this lecture, but a few parts of questions may be easier or have more obvious answers
Likely topics/questions on mini-exam #4

• From Lecture 25:
  – Basics of measurement
Likely topics/questions on mini-exam #4

• From Lecture 26:
  – difference between faults and failures
  – defect density
    • number of defects found, not the “real” number of defects (which is beyond our mortal knowledge)
    • relationship between faults and failures, e.g. small %age of faults causes large %age of failures
  – metrics of size: what’s wrong with using (S)LOC?
Likely topics/questions on mini-exam #4

• From Lecture 27:
  – COCOMO = Constructive Cost Model
  – Effort = \( a \cdot (\text{Size})^b \)
  – almost meaningless if Size is in LOC
  – different measure of size: function points

• you don’t need to memorize all the specific elements (“logical master files”, etc.) or the coefficients on slide 19, but you should understand the basic idea and why function points are a better measure than LOC
Likely topics/questions on mini-exam #4

• From Lecture 28:
  – general idea of backdoors, including Thompson’s compiler backdoor
  – buffer overruns:
    • what are the necessary elements of an attack?
      – overflowing a buffer
      – knowing or guessing the buffer’s location in memory, so that the return address can be overwritten with a pointer to the “payload”
    • why does address space layout randomization help?
Likely topics/questions on mini-exam #4

• From Lecture 28:
  – Morris worm: depended on buffer overruns, so it needed to know where buffers would be stored in memory
    • requires a relatively *homogeneous* network
  – the worm also exploited the fact that `fingerd` did not follow the Principle of Least Privilege
Likely topics/questions on mini-exam #4

• From Lecture 28:
  – early Macintosh and macro viruses:
    • extremely virulent, thanks to the willingness of the early Mac OS and early Word, Excel, etc. to automatically run whatever code they found
Likely topics/questions on mini-exam #4

• From Lecture 28:
  – Heartbleed and information leaks
    • not a buffer overrun, but related
    • OpenSSL bug that leaked extremely private information
    • potentially addressed through information-flow type systems
Likely topics/questions on mini-exam #4

• From Lecture 29a:
  – Language-based security
    • why is C so vulnerable to buffer overruns?
    • why are Java, Python, Haskell much less vulnerable?
Likely topics/questions on mini-exam #4

• From Lecture 29a:
  – Language-based security
    • why is C so vulnerable to buffer overruns?
    • why are Java, Python, Haskell much less vulnerable?
      – memory safety / type safety
Likely topics/questions on mini-exam #4

• From Lecture 29a:
  – Language-based security
    • why is C so vulnerable to buffer overruns?
      – no array bounds checking
      – casts between pointers and non-pointers
      – casts between different pointer types
    • why are Java, Python, Haskell much less vulnerable?
      – memory safety / type safety
Likely topics/questions on mini-exam #4

• From Lecture 29a:
  – Language-based security
    • why is C so vulnerable to buffer overruns?
      – no array bounds checking:
        breaks memory & type safety
      – casts between pointers and non-pointers:
        breaks memory & type safety
      – casts between different pointer types:
        breaks memory & type safety
    • why are Java, Python, Haskell much less vulnerable?
      – memory safety / type safety
Likely topics/questions on mini-exam #4

• From Lecture 29a:
  – more *generally*, however, there are few clear connections between implementation language and quality
Likely topics/questions on mini-exam #4

• From Lecture 29b:
  – SQL code injection
    • how it works
    • how to stop it
  – URL manipulation
    • how it works
    • how to stop it
Bonus question

• involves Bogosys and security