Lecture 28: December 4, 2019

Reading: *text*, §24.4.2–24.5 **Assignments:** Homework 5, due on December 6, 2019 at 11:59pm
Lab 3, due on December 6, 2019 at 11:59pm

1. Greetings and felicitations!
2. Some common vulnerabilities
   (a) Catalogues: CVE (Common Vulnerabilities and Exposures), CWE (Common Weakness Enumeration)
   (b) 2011 MITRE/SANS Top 25 Most Dangerous Software Errors
   (c) OWASP Top 10 – 2017 The Ten Most Critical Web Application Security Risks
3. MITRE/SANS list
   (a) Insecure interactions among components
      i. SQL injection
      ii. OS command injection
      iii. Cross-site scripting
      iv. Unrestricted upload of file with dangerous type
      v. Cross-site request forgery
      vi. URL redirect to untrusted site
   (b) Risky resource management
      i. Buffer copy without checking size of input
      ii. Improper limitation of a pathname to a restricted directory
      iii. Download of code without integrity check
      iv. Inclusion of functionality from untrusted control sphere
      v. Use of potentially dangerous function
      vi. Incorrect calculation of buffer size
      vii. Uncontrolled format string
      viii. Integer overflow or wraparound
   (c) Porous defenses
      i. Missing authentication for critical function
      ii. Missing authorization
      iii. Use of hard-coded credentials
      iv. Missing encryption of sensitive data
      v. Reliance on untrusted inputs in a security decision
      vi. Execution with unnecessary privileges
      vii. Incorrect authorization
      viii. Incorrect permission assignment for critical resource
      ix. Use of a broken or risky cryptographic algorithm
      x. Improper restriction of excessive authentication attempts
      xi. Use of a one-way hash without a salt
4. OWASP list
   (a) Injection

   (b) Broken authentication and session management
   (c) Sensitive data exposure
   (d) XML external entities
   (e) Broken access cointrol
   (f) Security misconfiguration
   (g) Cross-site scripting
   (h) Insecure deserialization
   (i) Using components with known vulnerabilities
   (j) Insufficient logging and monitoring

5. Comparison
   (a) Everything on the OWASP list is also on the MITRE/SANS list
   (b) Injection is #1 on both lists
   (c) The MITRE/SANS list covers vulnerabilities generally; OWASP covers only web vulnerabilities
6. Penetration Studies
   (a) Why? Why not direct analysis?
   (b) Effectiveness
   (c) Interpretation
7. Flaw Hypothesis Methodology
   (a) System analysis
   (b) Hypothesis generation
   (c) Hypothesis testing
   (d) Generalization
8. System Analysis
   (a) Learn everything you can about the system
   (b) Learn everything you can about operational procedures
   (c) Compare to other systems
9. Hypothesis Generation
   (a) Study the system, look for inconsistencies in interfaces
   (b) Compare to other systems' flaws
   (c) Compare to vulnerabilities models
10. Hypothesis testing
    (a) Look at system code, see if it would work (live experiment may be unneeded)
    (b) If live experiment needed, observe usual protocols
11. Generalization
    (a) See if other programs, interfaces, or subjects/objects suffer from the same problem
    (b) See if this suggests a more generic type of flaw
12. Elimination
13. Where to start
    (a) Unknown system

   (b) Known system, no authorized access
   (c) Known system, authorized access
14. Examples
   (a) Michigan Terminal System