**Question 1 - multiple choice, shuffle**

What is *penetration testing*?

A: A procedure for testing libraries or other program components for vulnerabilities

Feedback: Penetration testing applies to executable components, i.e., programs, not libraries

B: A security-minded form of unit testing that applies early in the development process

Feedback: Unit-testing is component level testing, but penetration testing applies to whole programs/systems

\*C: Whole-system testing for security flaws and bugs

D: All of the above

Feedback: Penetration testing applies to executable components only, i.e., programs, not libraries or units

**Question 2 - checkbox, shuffle, partial credit**

Which of the following are benefits of penetration testing?

\*A: They specifically consider adversarial thinking, which is not usually necessary for normal tests

Feedback: Adversaries will *try* to find weird corner cases to exploit, whereas normal users will focus on common use cases

\*B: Results are often reproducible

Feedback: Penetration test results can be recorded and replayed so they can be reproduced

C: Compositionality of security properties means tested components are secure even if others change

Feedback: Security properties, in general, are not compositional, and security is not guaranteed when components change

D: You can prove a positive: Penetration testing will establish your system is secure

Feedback: Testing cannot produce evidence of security; i.e., the lack of failing tests does not mean there is not faulty functionality that was not tested

**Question 2 - checkbox, variation 1, shuffle, partial credit**

Which of the following are benefits of penetration testing?

\*A: They specifically consider adversarial thinking, which is not usually necessary for normal tests

Feedback: Adversaries will *try* to find weird corner cases to exploit, whereas normal users will focus on common use cases

\*B: Results are often reproducible

Feedback: Penetration test results can be recorded and replayed so they can be reproduced

C: Compositionality of security properties means tested components are secure even if others change

Feedback: Security properties, in general, are not compositional, and security is not guaranteed when components change

D: Full evidence of security: a clean test means a secure system

Feedback: Testing cannot produce evidence of security; i.e., the lack of failing tests does not mean there is not faulty functionality that was not tested

**Question 2 - checkbox, variation 2, shuffle, partial credit**

Which of the following are benefits of penetration testing?

\*A: They specifically consider adversarial thinking, which is not usually necessary for normal tests

Feedback: Adversaries will *try* to find weird corner cases to exploit, whereas normal users will focus on common use cases

\*B: Results are certain and not hypothetical

Feedback: Test results produce actual behavior on the complete system; they are not a conjecture

C: Compositionality of security properties means tested components are secure even if others change

Feedback: Security properties, in general, are not compositional, and security is not guaranteed when components change

D: Full evidence of security: a clean test means a secure system

Feedback: Testing cannot produce evidence of security; i.e., the lack of failing tests does not mean there is not faulty functionality that was not tested

**Question 3 - multiple choice, shuffle**

What does it mean to "be stealthy" during a penetration test?

\*A: Taking care to avoid activities during a penetration test that might attract attention, e.g., by operators or IDS services

B: Performing penetration testing without the target organization knowing

Feedback: *Someone* in the organization should know (i.e., the person or group that contracted the test) even if not everyone does

C: Using encryption during tests to make the source of attacks impossible to determine

Feedback: This might be one means for ensuring the higher-level goal of avoiding detection, but is not the only one.

D: Performing the tests from an undisclosed location

Feedback: This might be one means for ensuring the higher-level goal of avoiding detection, but is not the only one.

**Question 4 - multiple choice, no shuffle**

What is a *web proxy*?

\*A: A piece of software that intercepts and possibly modifies requests (and responses) between a web browser and web server

B: An agent that makes decisions on the client's behalf when interacting with web applications

Feedback: A web proxy intercepts and possibly modifies browser/server messages

C: A piece of software that makes a web application look like a standalone application, making it easier to test

Feedback: A web proxy intercepts and possibly modifies browser/server messages

D: A simulator for the web, for use when off-line

Feedback: A web proxy could potentially implement a simulator, but the reason it can is because it intercepts and modifies/responds to requests.

**Question 5 - multiple choice, shuffle**

What is **Nmap**?

\*A: It is a scanner which works by injecting packets to a range of addresses, and inferring what hosts and services might be at those addresses, based on the responses

B: It is a suite of tools for scripting attacks: probe, construct, encode, inject, wait for response

Feedback: This answer describes Metasploit, not Nmap

C: It is a network fuzz testing tool

Feedback: Nmap does not attempt to find crashing/failing inputs; rather, it tries to find active services

D: It is a map of the Internet

Feedback: Um, try again

**Question 6 - multiple choice, shuffle**

What is *ethical hacking*?

\*A: Hacking systems (e.g., during penetration testing) to expose vulnerabilities so they can be fixed, rather than exploited

B: "Hacking" ethics so they justify unintended selfish behavior

Feedback: Ethics applies to hacking in this case, not the other way around

C: Hacking into systems run by those whose ethics you disagree with

Feedback: Hacking ethically means finding failures so they can be fixed, not breaking into systems (whoever might be running them)

D: A slang term for rapid software development, e.g., as part of hackathons

Feedback: In hackathons, the unmodified term "hacking" refers to programming

**Question 7 - checkbox, shuffle, partial credit**

Which of the following statements describe *fuzz testing* (aka *fuzzing*)?

\*A: It is concerned with finding known-bad behaviors, like crashes and hangs

Feedback: Fuzzing does not use specifications, so it can only look for behaviors it knows are likely to be bad

\*B: It has been used to find security vulnerabilities in many commodity programs

Feedback: Examples include Acrobat, Chrome, and others

C: It focuses on simple testing patterns and does not employ sophisticated analysis techniques

Feedback: Whitebox fuzz testing typically employs symbolic execution, which is a sophisticated analysis technique

D: It is a cost-effective replacement for functional testing

Feedback: Fuzzing *complements* functional testing, but cannot replace it because it doesn't know what a system is supposed to do

**Question 7 - checkbox, variation 1, shuffle, partial credit**

Which of the following statements describe *fuzz testing* (aka *fuzzing*)?

\*A: It has been used to find security vulnerabilities in many commodity programs

Feedback: Examples include Acrobat, Chrome, and others

\*B: It is concerned with finding known-bad behaviors, like crashes and hangs

Feedback: Fuzzing does not use specifications, so it can only look for behaviors it knows are likely to be bad

C: It is always *black-box*, in being indifferent to the software's functionality

Feedback: Fuzzing can be black box, white box, grammar based, etc., which means it can consider what the software does

D: It focuses on simple testing patterns and does not employ sophisticated analysis techniques

Feedback: Whitebox fuzz testing typically employs symbolic execution, which is a sophisticated analysis technique

**Question 7 - checkbox, variation 2, shuffle, partial credit**

Which of the following statements describe *fuzz testing* (aka *fuzzing*)?

\*A: It is a kind of random testing

Feedback: Fuzzing generates inputs randomly

\*B: It is concerned with finding known-bad behaviors, like crashes and hangs

Feedback: Fuzzing does not use specifications, so it can only look for behaviors it knows are likely to be bad

C: It is always *black-box*, in being indifferent to the software's functionality

Feedback: Fuzzing can be black box, white box, grammar based, etc., which means it can consider what the software does

D: It is a cost-effective replacement for functional testing

Feedback: Fuzzing *complements* functional testing, but cannot replace it because it doesn't know what a system is supposed to do

**Question 8 - checkbox, shuffle, partial credit**

Which of the following are true of *whitebox fuzzing*?

\*A: **American Fuzzy Lop** is (at least in part) a whitebox fuzzer

Feedback: American Fuzzy Lop takes into account the lines of code executed by a test in determining the next input, and thus considers the program's internals

B: **Radamsa** is (at least in part) a whitebox fuzzer

Feedback: Radamsa is a blackbox fuzzer, ignoring the program's internals

C: It makes no sense to combine it with grammar-based fuzzing since the latter is just another way to consider the program's semantics

Feedback: A grammar can help focus the search procedure of a whitebox fuzzer when choosing among multiple possible next paths to explore

D: **nmap** is (at least in part) a whitebox fuzzer

Feedback: nmap is a network scanning tool, not a software fuzzer

**Question 8 - checkbox, variation 1, shuffle, partial credit**

Which of the following are true of *whitebox fuzzing*?

\*A: It takes into account the program's internals in some manner when deciding which inputs to choose

B: **Radamsa** is (at least in part) a whitebox fuzzer

Feedback: Radamsa is a blackbox fuzzer, ignoring the program's internals

\*C: **SAGE** is (at least in part) a whitebox fuzzer

Feedback: SAGE uses symbolic execution to find test cases, and is thus guided by the program's code

D: It makes no sense to combine it with grammar-based fuzzing since the latter is just another way to consider the program's semantics

Feedback: A grammar can help focus the search procedure of a whitebox fuzzer when choosing among multiple possible next paths to explore

**Question 8 - checkbox, variation 2, shuffle, partial credit**

Which of the following are true of *whitebox fuzzing*?

A: **Radamsa** is (at least in part) a whitebox fuzzer

Feedback: Radamsa is a blackbox fuzzer, ignoring the program's internals

\*B: It takes into account the program's internals in some manner when deciding which inputs to choose

\*C: **SAGE** is (at least in part) a whitebox fuzzer

Feedback: SAGE uses symbolic execution to find test cases, and is thus guided by the program's code

D: **nmap** is (at least in part) a whitebox fuzzer

Feedback: nmap is a network scanning tool, not a software fuzzer

**Question 9 - multiple choice, shuffle**

Which of the following is true of *mutation-based fuzzing*?

A: It only makes sense for file-based fuzzing, not network-based fuzzing

Feedback: Mutating inputs is a sensible approach wherever the inputs might be coming from (files or network messages)

\*B: It generates each different input by modifying a prior input

Feedback: This is the definition of mutation-based fuzzing

C: Each input is mutation that follows a given grammar

Feedback: The use of a grammar is optional when considering mutations

D: It works by making small mutations to the target program to induce faults

Feedback: Mutations are to the input, not the program under test

**Question 10 - multiple choice, shuffle**

Which of the following styles of fuzzer is more likely to explore paths covering every line of code in the following program?

int main(int argc, char \*\*argv) { char buf[100]; while (fgets(buf,sizeof(buf),stdin) != NULL) { int c = atoi(buf); if (c == 456799) printf("%s\n",(char \*)c); else { int i = 0; for (i=0; i<c; i++) printf("."); printf("\n"); } } return 0; }

\*A: Whitebox

B: Blackbox

Feedback: It is very unlikely that a blackbox fuzzer will generate the input "456799" needed to access the true branch of the $$\color{red}{\verb|if (c == 456799) ...|}$$ code.

C: Mutation-based

Feedback: It is very unlikely that a mutation-based fuzzer, on its own, will generate the input "456799" needed to access the true branch of the $$\color{red}{\verb|if (c == 456799) ...|}$$ code. Mutation, or not, is not the relevant question in generating this input.

D: Generational

Feedback: It is very unlikely that a generational fuzzer, on its own, will generate the input "456799" needed to access the true branch of the $$\color{red}{\verb|if (c == 456799) ...|}$$ code. A grammar is not going to help because all numeric input is legal.

**Question 11 - checkbox, shuffle, partial credit**

Which of the following are functions of a *network-based fuzzer*?

\*A: Acting as a "man in the middle"

Feedback: Network fuzzers may play any role in a network communication, and may intercept messages between legitimate roles

\*B: Acting as a server

Feedback: Network fuzzers may play any role in a network communication

C: Mutating network configuration files

Feedback: Network fuzzers work with communications, not files

**Question 11 - checkbox, variation 1, shuffle, partial credit**

Which of the following are functions of a *network-based fuzzer*?

\*A: Acting as a "man in the middle"

Feedback: Network fuzzers may play any role in a network communication, and may intercept messages between legitimate roles

\*B: Acting as a server

Feedback: Network fuzzers may play any role in a network communication

C: Scanning a network address range

Feedback: Fuzzers don't scan for presence, they attempt to crash/exploit known presences

**Question 11 - checkbox, variation 2, shuffle, partial credit**

Which of the following are functions of a *network-based fuzzer*?

\*A: Acting as a server

Feedback: Network fuzzers may play any role in a network communication

\*B: Acting as a client

Feedback: Network fuzzers may play any role in a network communication

\*C: Acting as a "man in the middle"

Feedback: Network fuzzers may play any role in a network communication, and may intercept messages between legitimate roles

**Question 12 - multiple choice, shuffle**

Suppose you want to use fuzzing on a program to try to find memory errors; which of the following statements is true?

\*A: Compiling the program with address sanitizer (ASAN) will make the source of a *memory error easier to find*

Feedback: This is true because when a fuzzed input causes a buffer to overrun (say) the program will halt immediately, with diagnostic information presented by ASAN

B: Compiling the program with address sanitizer (ASAN) will make errors *harder to reproduce*

Feedback: In fact it's the opposite: When a fuzzed input causes a buffer to overrun (say) the program will halt immediately, with diagnostic information presented by ASAN, making the source of the error more obvious

C: Fuzzing doesn't find memory errors, it finds crashes and hangs

Feedback: Crashes are often due to memory errors

D: You should not use a grammar-based fuzzer, because its adherence to the grammar means it will not find memory errors

Feedback: These issues are orthogonal. For example, a very large, but legal input, could still overrun a buffer.