CS 3501: ICS Exam 2, spring 2019

Name _____

You MUST write your e-mail ID on **EACH** page and bubble in your userid at the bottom of this first page. And put your name on the top of this page, too.

If you are still writing when "pens down" is called, your exam will be ripped up and not graded – even if you are still writing to fill in the bubble form. So please do that first. Sorry to have to be strict on this!

Other than bubbling in your userid at the bottom of this page, please do not write in the footer section of this page.

There are 6 pages to this exam. Once the exam starts, please make sure you have all the pages. Questions are worth different amounts of points.

If you do not bubble in this first page properly, you will not receive credit for the exam!

Answers for the short-answer questions should not exceed about 20 words; if your answer is too long (say, more than 30 words), you will get a zero for that question!

This exam is CLOSED text book, closed-notes, closed-calculator, closed-cell phone, closed-computer, closed-neighbor, etc. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly. Please sign the honor pledge below.

_____

_____

_____

_____

In theory, there is no difference between theory and practice.
But, in practice, there is.

(the bubble footer is automatically inserted into this space)

Page 2: Exam 1 material

1. [3 points] What is password *salting*?

2. [3 points] Describe the steps to generate RSA keys.

3. [3 points] When an executable is loaded into memory, why does it expand in the process?

4. [3 points] What is the most challenging part of writing a virus?

Page 3: Binary exploits

5. [3 points] Write a *single* `printf()` statement that writes the value 32,769 (0x8001) to pointer address 0x12345678. Your answer obviously can not be 32 thousand characters or so.

6. [3 points] List the best thing you can do to prevent a format string attacks.

7. [3 points] Describe how an off-by-one vulnerability can be exploited.

8. [3 points] A heap buffer overflow attacks a variable on the heap. List three ways that one can create a variable on the heap in C/C++.

Page 4: Buffer overflow

9. [4 points] Consider a buffer overflow attack against a binary executable that uses stack canaries. How would one enter the canary value into data used to perform the overflow?

10. [4 points] Consider performing a buffer overflow against an executable compiled with `clang++ -fomit-frame-pointer vulnerable.cpp`. A local subroutine with 2 parameters has three variables, defined in this order: `char x[100]` (the buffer you are attempting to overflow), `int y`, and `float z[100]`. Draw (or list in order) what the activation record in the stack would look like in this routine (assume no register backups). If you have to make *valid* assumptions, state them as well.

11. [4 points] Consider the stack that you described in the last problem. Assume that your shell code is 40 bytes in length. Write what the data to perform the overflow attack would be. We are looking for a high level description here ("40 bytes of shell code followed by x bytes of Y").

Page 5: Networks

12. [3 points] What is the difference between a router and a switch?

13. [3 points] Briefly descripe the three types of firewalls?

14. [3 points] Other than a SYN flood, list and *briefly* describe three other types of denial-of-service attacks.

15. [3 points] Briefly describe how a SYN flood denial-of-service attack works.

Page 6: SQL, XSS, CSRF

16. [3 points] To defend against a SQL attack, you can escape your input string. What characters must you escape?

17. [3 points] Briefly describe the difference(s) between a GET and a POST attack in relation to XSS.

18. [3 points] Describe a *real-world* XSS attack (what you would attack, what you would steal, etc. – not the technical details).

19. [3 points] Assume you have a program that presents a web form, and you want to defend against CSRF attacks. You can assume it does *NOT* use a web framework. List the steps you would take to implement a CSRF defense. We are looking for the half-dozen or so high-level steps, not implementation details.