CSCI 476 Final Exam Study Guide

Logistics:

- Tuesday December 13th 2:00 – 3:50 PM
- One 8.5 x 11 (standard piece of paper, both sides) note sheet is allowed. Typed and hand-written are both ok
- Final Exam is worth 15% of your final grade
- There won't be a curve unless one is needed

The final exam will consist of <20 short answer questions. You can expect 1-2 questions for each of the major topics that we talked about. For example, the question might ask you define an attack, an attack methodology, the "vulnerability", or about the existing countermeasures.

You won't need to write any code, and you won't need to write any Linux commands. You might need to look at some source code (python or C)

The list of topics that can appear on the final exam can be found on the next page

1. **Threat Modeling**
    a. What is it? Purpose of threat modeling
    b. Threat modeling methods
2. **SET-UID Programs and Environment Variables**
    a. What a Set-UID program is. What makes it special? Why do we need them?
    b. RUID vs EUID. What does a Set-UID program do to these values?
    c. `chmod` and `chown` (you wont need to write these commands, but you should know what these command do, and what `4755` means)
    d. `System()` vs `exec()` family. Why is `system()` considered unsafe?
    e. Attack vectors for SET-UID programs (Environment Variables, Untrusted user input, dynamic linking)
    f. System Calls, Environment Variables, `fork()`
    g. Principle of Isolation
3. **Shellshock Attack**
    a. From a high level, describe what this attack does, the attack methodology, and what the exploitable vulnerability is
    b. What are the two conditions needed to conduct a shellshock attack?
    c. What is a reverse shell? Why won't a normal shell suffice?
4. **Buffer Overflow**
    a. From a high level, describe what this attack does, the attack methodology, and what the exploitable vulnerability is
    b. What is the stack? What data gets put on the stack?
    c. `Esp` vs `ebp`
    d. Why do we need a return address in the stack?
    e. What is the `NOP` operator? Why do we use it in a buffer overflow attack?
    f. What is shellcode? Why did we did we use shellcode rather than injecting our own C program?
    g. Countermeasures: ASLR, Dash, Stack-guard, Non-executable stack. How to bypass those countermeasures?
5. **SQL Injection**
    a. From a high level, describe what this attack does, the attack methodology, and what the exploitable vulnerability is
    b. Basic SQL Query (`SELECT FROM WHERE`)
    c. Given basic code that handles user input for an SQL query, construct a malicious payload
    d. `Prepare()` statements
6. **XSS Attack**
    a. From a high level, describe what this attack does, the attack methodology, and what the exploitable vulnerability is
    b. HTTP Request, HTTP Responses, URL
    c. Static vs Dynamic Web Content
    d. Process of using XSS to steal someone's cookies
    e. From a high level, how we designed a self-propagating worm
    f. Countermeasures: Filtering, Encoding, CSP, CORS

7. **TCP Attacks**
    a. What is packet sniffing? How does an application "sniff" for packets?
    b. What is packet spoofing? What type of packets can we spoof?
    c. TCP Handshake
    d. TCP Flooding: From a high level, describe what this attack does, the attack methodology, and what the exploitable vulnerability is
    e. Countermeasures: SYN Cookies
    f. TCP Reset: From a high level, describe what this attack does, the attack methodology, and what the exploitable vulnerability is
    g. Why did we need the Sequence # in the TCP reset/TCP Hijacking attack?
    h. TCP Hijack: From a high level, describe what this attack does, the attack methodology, and what the exploitable vulnerability is
    i. Using TCP Hijack, what is the process for getting a root shell?
8. **Secret-Key Encryption**
    a. Basics of a crypto system (Plaintext, Cleartext, encryption, decryption, key)
    b. Substitution Cipher. What is it, how did we crack it?
    c. Block Ciphers
    d. Why is ECB considered unsafe? How do the other modes of encryption fix this?
    e. What is padding? When is padding used?
    f. What is an Initialization Vector? What are the three requirements that an IV should have?
9. **Hashing**
    a. What is a cryptographic hash function? What are the three important properties of a cryptographic hash function?
    b. How do you reverse a hash? *(this is a trick question)*
    c. Brute Force Attack and Dictionary Attacks
    d. Applications of hashing (Integrity, passwords, commits)
    e. What is a hash collision?
    f. Birthday Paradox and its relation to hash collisions?
    g. Motivation for generating a hash collision
10. **Asymmetric Crypto**
    a. Difference between Asymmetric Crypto and Symmetric crypto
    b. What a public and private key are, when each one is used.
    c. What the Diffie Hellman Key Exchange is (high level why it is secure, and why it is difficult for a 3rd party to crack the secret)
    d. Limitations of RSA
    e. How Symmetric Crypto, Asymmetric crypto, and hashing work together
    f. Digital Signature (Creating and Verifying)
11. **Secret**
    a. ???

1. What are environment variables? How did we use them to exploit a SET-UID program?

2. One countermeasure to the buffer overflow attack was to use the more secure `/bin/dash` shell instead of `/bin/zsh`. What did `/bin/dash` do that thwarted our buffer overflow attack? How did we bypass this countermeasure?

3. What is a Cross-Site-Scripting (XSS) attack? What is the vulnerability that we exploited for XSS attacks?

4. What is a TCP reset attack? Why did we need to include the sequence number in the spoofed packet for the reset attack?

5. Suppose that we created an efficient tool that can generate two programs that have the same hash (MD5, SHA256, etc) **and** we can control the contents (the source code) of the two programs. How could this tool be used for malicious purposes?

6. In RSA, what is the difference between the public key, and the private key? When is each is key used?