Introduction to Computer Security (ECE458) Mid-term Exam

ABOUT THIS MID-TERM

This midterm exam consists of questions derived from material taught in class on control hijack attacks, basic cryptography and program analysis techniques for automatic detection of security vulnerabilities/malware. In order to get a good grade, please provide detailed, complete and correct answers to the questions below.

INSTRUCTIONS

You must work on your exam strictly individually. Questions are in order of the course material, and not in order of difficulty level.

No materials allowed:

- No books, no course notes or printouts of any kind. No calculators, no cellphones/smartphones, computers, or electronic devices of any kind. You must turn off any electronic devices and store them under your desk simply having any device (even if turned off) with you during the exam constitutes a violation and will be reported. If you need to borrow a pencil, sharpener, eraser, etc., you must ask a proctor. You're not allowed to directly ask any of your neighbours

Before, during, and at the end of the exam:

- You're not allowed to leave during the examination period, except washroom visits.
- Do not stand up or talk until all exams have been picked up.
- You must stop writing when any of the proctors or the instructor announces that the examination is finished. NO EXCEPTIONS.
- Only ask meaningful questions to the proctor. If you harass the proctor for answers or other students in the exam room, you are liable to be reported to appropriate University officials for violation of exam policy.
- If you are found cheating, involved in discussions, talking to other students or causing any kind of disturbance during the exam, then you will be reported to appropriate University officials for violation of exam policy.
- Answers must be properly marked in the answer book with the corresponding question number. Only answers in the answer book will be marked and graded. (This exam book is also your answer book.)
- Return both the answer/question books back to the proctor before leaving the exam hall.

FULL NAME:

UW ID:

Question 1: Short answer questions (30 points)

Each of the following sub-questions is worth 3 points. Please provide a short, complete and correct answers to get full points.

1. Explain the purpose of the NX bit (not execute bit) in modern OSes.

   Answer: If the NX bit is set to true the memory content of the corresponding memory pages are deemed not executable by the OS. Such a featuer is useful in the context of stack-based control-hijack attack. In such attacks the attacker send a payload to a program with a buffer overflow vulnerability, causes an overflow, and overwrites the return address such that control jumps to his payload on stack. By making the memory pages where the call-stack is located non-executable using the NX bit, such attacks can thwarted.

2. Is the following assertion true or false: In modern systems, buffer overflows on the heap cannot be exploited since contents of heap are by definition not executable irrespective of the status of the NX bit in the corresponding memory pages.

   Answer: False. Contents of the heap are routinely treated as executable by JIT compilers like Javac as long as the NX bit is set to false.

3. In class we discussed how randomized stack canaries are used prevent (with high probability) a class of stack-based control-hijack attacks. This mechanism inserts a random cookie at the appropriate place on stack such that any buffer overflow on stack that causes the canary to be overwritten is detected in the function epilogue. Discuss one meaningful way in which the protection provided by stack canaries can be overcome by an attacker whose resources are bounded by some polynomial over the length of memory addresses (32 bits). State your assumptions clearly.

   Answer: The attacker can launch the following attack under the assumption that the canary doesn't change between executions of the same program: He overwrites only one byte of the canary at a time without disturbing the remaining bytes, on a byte addressable machine. Modern systems shut down the program when an attack is discovered in the epilogue of a function call, and otherwise allow the program to continue normal execution. In one out of 256 attempts the attacker will notice that the system doesn't crash, concluding that the corresponding value is the correct value of the byte of the canary. Thus in 1024 attempts in the worst-case the attacker will discover the canary value.

4. In modern implementations of address-space layout randomization schemes, the *entire* layout of memory is randomized. No exceptions.

   Answer: False. The code and data segment are often not randomized. Often even the heap is not completely randomized. Only the libc location is randomized.

5. Most defense mechanism against control-hijack attacks are implemented at the OS or compiler level that forces recompilation of the program-under-protection. In order to be implemented and deployed correctly, these defense mechanism require major modification to the OS (e.g., NX bits), compilers (e.g., Canaries) or both (e.g., ASLR). Can you discuss a method that protects against a class of control hijack attacks that doesn't require recompilation or modification to the OS? (Hint: you are allowed to modify the C library.)

   Answer: Libsafe is a dynamically linked library whose advantage is that it doesn't require any changes to the program-under-protection or the OS or recompilation. LibSafe is a safe library which intercepts all calls to unsafe library libc functions by C/C++ programs, and performs appropriate bounds checks on buffers to protect against buffer overflow attacks at runtime.

6. Explain why the public key (pk) and secret key (sk) in the RSA encyrption scheme are inverses in group mod $\phi(n)$ and not inverses in group mod n, where n is the product of two distinct large prime numbers?

   Answer: In the RSA encryption scheme, the public key pk and the modulus over which all operations are performed during encryption and decryption are know to the attacker, i.e., the attacker knows n. Given attacker's knowledge of pk and n, he can easily compute sk, and the scheme would be broken. However, it turns out that given n and pk there is no known polynomial time method to compute the totient $\phi(n)$ of n. Hence, the pk and sk are set to be inverses mod $\phi(n)$.

7. Argue as to why RSA is secure by appealing to the hardness of factorization of the product of two large distinct primes. Ignore side-channel methods to break RSA.

   Answer: To break the RSA scheme the man-in-the-middle attacker has to either guess the secret key sk of the receiver or the message being encrypted or somehow use the information available to him to learn sk. Guessing sk and message in polynomial time seem impossible over sufficiently large space of message and key space. The other option is to learn sk given the public key pk and the modulus n. The modulus n is chosen to be the product of two large primes. The only known (non side-channel) way to break sophisticated implementations of RSA is factoring n into primes p and q. If the attacker can factor n, then he can compute the totient $\phi(n) = (p-1)(q-1)$. However, as far as we know there are no known ways to perform factorization in polynomial time. (except perhaps through the use of quantum computers which are still no where close to being realized practically.)

8. A central authority is assigned the task of key generation for the RSA scheme. This authority decided to use the same n (i.e., the modulus) for generating keys for Alice and Bob. (To be very precise the central authority is using the same totient of n to generate the key pairs for Alice and Bob.) What are the potential problems with this approach?

   Answer: If the same n (or more precisely the totient same $\phi(n)$) is used to generate the key pairs for both Alice and Bob, then Bob can easily learn Alice's secret key (sk_A) by using Alice's public key (pk_A) and the totient $\phi(n)$ which is common to both.

9. Argue informally but carefully as to why one-time pad (OTP) is perfectly secure. Provide an appropriate notion of security. State your assumptions clearly. Make your answer brief, correct and complete.

   Answer: Let M represent the space of messages, K represent the space of keys and C represent the space of ciphertexts. Assume they are all same-length bit-vectors. We define an encryption scheme to be perfectly secure if the attacker's probability of guessing a message with and without the ciphertext is the same (The indistinguishability argument). In OTP, the key is used only once and the encryption scheme is the XOR function. Observe that the encryption function simply maps a message from M to random element in $c \in C$, assuming the key is chosen at random from K. The attacker's a priori (before analyzing c) probability distribution over the message space is uniform at random. This probability remains the same after analyzing c (a posteriori probability) because given a truly random key the ciphertext may map to any message, and all are equally likely.

10. Many arguments in cryptography are of the form: Informally, "if an attacker with her time budget bounded by a polynomial in the size of a security parameter can break xyz encryption scheme, then attacker can also solve some hard mathematical problem within the same polynomial-time resource limit. However given that we believe this mathematical problem cannot be solved in time polynomial in the security parameter, it implies that the scheme is secure." Breaking an encryption scheme means

that the attacker can reliably learn the secret information (e.g., secret key and totient) from public information. Such arguments are called cryptographic reductions.

Consider the RSA encryption scheme. Technically, there is no such air-tight cryptographic reduction argument for RSA. I.e., you could break RSA without having to factor large composite numbers (Fortunately, as far as we can tell no one knows such a method, and all known methods to break RSA require factoring large composites into large distinct primes.) Suppose you had such a reduction argument for RSA. How would you use your technique to break RSA to factor large composites n into large distinct primes p and q?

Answer: Let A denote the attacker's technique, which given the public key pk and the modulus n, outputs the totient of n. First, observe that totient of n ($\phi(n)$) is:

$$\phi(n) = (p-1)(q-1) = pq - p - q + 1 = n - p - q + 1$$

Hence,

$$p + q = n + 1 - \phi(n)$$

Given that A's technique gives us $\phi(n)$ and n is public knowledge, the attacker knows p+q. Given p+q, simply cycle through all pairs of primes less than p+q and check if their product is equal to n. The pair p, q such that its product matches n are the factors of n. Note that from fundamental theorem of algebra all numbers can be factored into primes. Given that p and q are large primes, it follows that p, q will be unique.

Question 2: Control-hijack Attacks (20 points)

Question 2.1 (10 points)

In the following questions assume we are trying to control-hijack a program `vulnerable` through a stack-smahsing attack. Assume that for this question the OS is Linux and the memory addresses are 4 bytes long. All programs are assumed to be written in the C programming language. Also assume that `vulnerable` has the following method called from its main(), whose inputs can be controlled by the attacker. In the function below, the programmer intends to copy a substring of the string str to the buffer buf. The substring starts at position pos1 in str and ends at pos2. The programmer carefully checks the length substring before using malloc to allocate memory for the buffer. Assumes the string is indexed as follows: the leftmost character is at position 0, and so on.

```
int parse_string(char *str, int pos1, int pos2) {
  ⋮
  char buf[1024];

  if(pos2-pos1 > 1024)
    exit(-1);
  strcpy(buf, str);
  ⋮
  return 1;
}
```

Explain how there can be a buffer overflow in the above program.

Answer: The answer is YES. First observe that nowhere is it stated in the description or the code that there is any relation between the length of the str on the one hand, and pos2/pos1 on the other. The attacker simply supplies values to pos2 and pos1 such that $pos2 - pos1 \leq 1024$, and str is sufficiently longer than buf such that a buffer overflow can be caused.

Question 2.2 (10 points)

How would you fix the above code such that there is no possibility of a buffer overflow.

Answer: First compute the substring of str (using a appropriate library function) and assign it to, say, a new variable substr1. Make sure to null-terminate substr1. Write a lengthof function (or use an appropriate library function) to compute the length of substr1. If the length is above 1024, then exit(-1). Else copy substr1 using strcpy.

Answer:

Question 3: Limits of Concolic Testing and Static Analysis in Detecting Malware (20 points)

Consider the following piece malware written in C.

```
int innocent(char * input) {
  if (input == 'key')
     Execute_Malware();
}
```

Malware such as the one sketched above are trojans that attackers can install on your machine, to be activated at their convenience later on. The malware is executed only when a secret input 'key' is sent to it. Let us assume that you are able to execute the malware in a sandbox (an isolated system whose sole purpose is to run malware and understand its properties). If you run it symbolically, then you can study its properties and learn that it is malware. What are the techniques the attacker can use to hide both the key and malware from both symbolic and static analysis?

Answer: The attacker can hash the 'key' value (to proetct against symbolic analysis) and encrypt the malware using the 'key' as follows (to protect against static analysis). In the following, 'key' is the encryption key (you can use any symmetric scheme). This will prevent any static analysis from knowing the code, and symbolic analysis cannot be used to execute the malware in a sandbox. The way the attacker would modify the above code is as follows:

```
int innocent(char * input) {
  if (hash(input) == pre-computed-hash-of-key)
     Execute(Decrypt(Encrypt(Execute_Malware(), pre-computed-hash-of-key), input));
}
```

Question 4: Basic Cryptography (30 points)

In this question you will be asked 2 sub-questions each worth a maximum of 15 points. Please provide complete, correct and pithy answers to get full points.

Question 4.1: Naive RSA based Digital Signatures

The following is a well-known attack, called a existential forgery attack, on RSA-based digital signatures, where we assume that RSA is used naively. The attacked party Alice's public key is e, secret key is d and the modulus is n. The attacker's goal is construct a pair of message and signature which the receiver Bob is fooled into thinking that it came from Alice.

The steps followed by the attacker are:

1. The attacker chooses a random number r

2. Using Alice's public key, he computes

$$v = r^e \ mod \ n$$

3. He then constructs a message, signature pair to perform an existential forgery attack.

Your job is to specify what the message and the signature are in an existential forgery attack. Mathematically justify your answer.

Answer: The message/signature pair constructed by the attacker is $\langle v, r \rangle$, i.e., v is marked as the message and r as the signature. The mathematical explanation is that when Bob receives this pair he raises r to e mod n, i.e., he computes $r^e \ mod \ n$. (Recall that in digital signatures you raise the signature to the public key, to obtain a value. During the signature verification step, this value is checked against the message. If they match you know that the message was signed by Alice.) The value thus computed is checked against v, which obviously matches. Thus a signature existential forgery has been performed by the attacker, assuming Alice used RSA naively to sign her messages.

Question 4.2: Universal Forgery Attack

Suppose the attacker has been logging the traffic between Alice and Bob. It is assumed that the attacker can reliably distinguish between messages and their signatures. In the process he found two messages x and y signed by Alice, i.e., he has pairs $< x, sign(x) >$ and $< y, sign(y) >$. Further assume that x and y have the property that the attacker can construct a meaningful message m with the property that

$$m = x \times y$$

. Using only the mathematical properties of RSA, explain how he can fool Bob into believing that the message m was signed by Alice.

Answer: Given the two pairs $< x, sign(x)$ and $< y, sign(y) >$. Using modulus arithmetic it follows that

$$sign(m) = sign(xy) = (xy)^d = (x^d).y(^d) = sign(x).sign(y) \ mod \ n$$