CS 4440 Introduction to Computer Security

Practice Exam

This practice exam is intended to help you prepare for the final exam. It does **not** cover all material that will appear on the final. We recommend that you use this practice exam to supplement your preparation, in addition to going over your lecture notes, quizzes, and programming projects.

This practice exam has no deadline and will not be graded. However, you will get the maximum benefit out of this exam review by treating it **as if it were the real exam:** you may refer to your two-sided 8.5"×11" cheat sheet, but allow yourself only 2 hours to complete the exam.

The final lecture will serve as an in-class review session covering the solutions to this practice exam. **Solutions to this practice exam will be discussed in-class only—do not skip this lecture!**

---

1. **Cryptography.** Alice and Bob, two CS 4440 alumni, have been stranded on a desert island for several weeks. Alice has built a hut on the beach, while Bob lives high in the forest branches. They plan to communicate silently by tossing coconuts over the treeline.

   Compounding Alice and Bob's misfortune, on this island there also lives an intelligent, literate, and man-eating panther named Mallory. The pair can cooperate to warn each other when they see the animal approaching each others' shelters, but they fear that Mallory will intercept or tamper with their messages in order to make them her next meal. Fortunately, Alice and Bob each have an RSA key pair, and each knows the other's public key.

   (a) Design two protocols that leverage RSA, such that Alice can securely transmit a message to Bob whilst upholding (1) message *confidentiality* and (2) message *integrity*.

   (b) After arriving at a shared secret, Alice and Bob plan to use symmetric cryptography to protect their messages, but they disagree about how to apply it. Alice believes it is best to encrypt their plaintext then add a MAC to the ciphertext, while Bob wants to MAC first then encrypt. Explain whose approach is faster, and why.

   (c) Alice and Bob's protocol allows them to communicate securely, but it still leaks to Mallory the fact that they *are* communicating. Describe an approach that the two could use to obscure this (naming the approach is sufficient).

2. **Authentication.** Many organizations (including The U) deploy two-factor authentication through the use of key-fob-sized devices that display pseudorandom codes at a fixed time interval. These codes are generated based on a built-in clock and a device-specific secret s that is also stored on a central authentication server tied to the user's account. Here is one way such a device might work: Let n be the number of minutes that have elapsed since the UNIX epoch; output the first 20 bits of $\text{HMAC}_s(n)$. Successful authentication requires the user's username and password and the current pseudorandom code from the user's device.

   (a) Name three common attacks against authentication that are *mitigated* by these devices.

   (b) Name one common attack against this authentication that is *not* mitigated.

   Some devices use a counter instead of a clock and generate a single-use code each time the user presses a button on the device. One way this might work is as above, letting n be a register that is initially zero; upon each button press, display the current code for one minute and increment n on the device; on each successful authentication increment n on the server.

   (c) Describe one *security* advantage of single-use codes compared to time-based codes.

   (d) Describe one *usability* advantage of single-use codes compared to time-based codes.

   (e) A major usability problem with single-use codes in practice is that the counter on the device sometimes gets out of sync with the counter on the server (often as a result of inadvertent button presses in the user's pocket). Explain how we might extend the server to mitigate this without significantly reducing security.

3. **Ciphers.** Penguin Pat wants to keep his picture secret. Answer the following:

(a) Before he determines a method of encryption, Pat needs criteria on which to judge the security of his cipher output. Please define and explain the criteria Pat should use.

(b) Consider the input image and output encrypted image below. Which block cipher mode is Penguin Pat using, and why does it offer poor security guarantees?

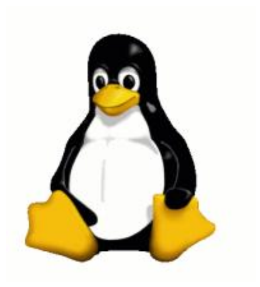


Figure 1: Input

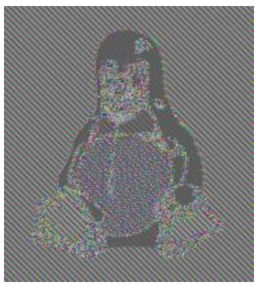
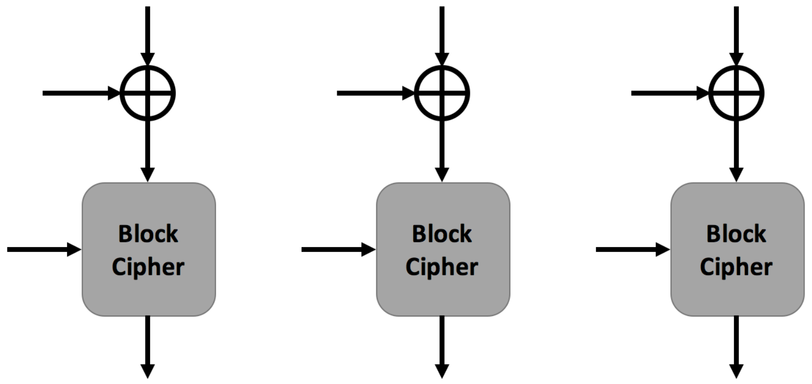


Figure 2: Output

(c) Pat realizes the danger of his current approach. He now adopts a more secure block cipher mode in which the output of one block cipher influences the next. Complete the following diagram to depict the *encryption* phase of this block cipher:



4. **Network Security.** WiFi Wallace is a friend of yours using CoffeeCult's free WiFi internet.

   (a) Wallace is using FTP to download some potentially useful files from his web server. Not long after, Wallace can no longer access his account—his password was changed! Wallace suspects that his account was *hacked*. Can you explain to him why?

   (b) Wallace believes there may be a malicious entity on the network, searching for open ports on devices. Explain how Wallace can locate the malicious entity's IP address.

   (c) Wallace cannot access his favorite sites—they redirect elsewhere! What happened?

5. **Certificates.** A *self-signed certificate* makes the claim that a public key belongs to a particular server, without any trusted Certificate Authority (CA) to verify it. Browsers show warning messages when sites present such certificates, yet users often override them. Some websites use self-signed certs to avoid the trouble of obtaining a cert from a trusted CA.

   (a) How could a man-in-the-middle attacker compromise connections to a website that uses a self-signed certificate, assuming that the site's users are already accustomed to manually ignoring their browsers' warnings about self-signed certificates?

   (b) Briefly compare the security of the following website design choices:

      i. HTTPS via self-signed certificate for all pages.
      ii. HTTPS via CA-signed certificate for all pages.
      iii. HTTPS via CA-signed certificate only for login pages, and HTTP for all others.

6. **Web Security.** An imaginary social media site FacePalm is full of security-ignorant users. Answer the following questions about potential attacks against FacePalm's users.

   (a) FacePalm's homepage has a "`Delete Account`" link leading to the following page:

   ```
   <p>Are you sure you want to delete your account?</p>
   <form action="/deleteuser" method="post">
     <input type="hidden" name="user" value=request['user']></input>
     <input type="submit" value="Yes, please delete my account."></input>
   </form>
   ```

   The implementation of `/deleteuser` is given by the following pseudocode:

   ```
   if account_exists(request['user']):
       delete_account(request['user'])
       return '<p>Thanks for trying FacePalm!</p>'
   else:
       return '<p>Sorry, ' + request['user'] + ', an error occurred.</p>'
   ```

   Assume that the attacker knows the username of an intended victim. What's a simple way that the attacker can exploit this design to delete the victim's account without any direct contact with the victim or the victim's browser?

   (b) Suppose that `/deleteuser` is modified as follows:

   ```
   if validate_login(request['user'], request['cookie']):
       delete_account(request['user'])
       return '<p>Thanks for trying FacePalm!</p>'
   else:
       return '<p>Sorry, ' + request['user'] + ', an error occurred.</p>'
   ```

   ... where `validate_login()` checks that the cookie sent by the browser is authentic and was issued to the specified username. Assume that `cookie` is tied to the user's account and difficult to guess. How can the attacker still delete the victim's account?

   (c) FacePalm has decided to deploy more aggressive *per-request* token validation—CSRF tokens—mitigating the above vulnerability. However, the site has a messaging form that fails to sanitize user input. How can this flaw be exploited to log-in as the victim?

7. **Application Security.** Answer the following questions related to app attacks and defenses:

(a) Below is the beginning disassembly of a function in **AT&T** syntax (“`op src, dst`”). Note that all offsets and immediates are in integer form (e.g., $8 = 8).

```
0x0804a014 <+00>:  push %ebp
0x0804a015 <+01>:  mov %esp, %ebp
0x0804a017 <+03>:  sub $8, %esp
0x0804a01a <+06>:  mov 16(%ebp), %eax
0x0804a01c <+08>:  mov $3, 4(%esp)
```

Suppose this function has just been called: its return address `0x0804a000` has been pushed to the stack; the instruction pointer now points to `0x0804a014`; and the `ebp` (base pointer) and `esp` (stack pointer) registers hold `0xbffff440` and `0xbffff400`, respectively. At this point in time, this function's stack frame resembles the following (note that memory contents are actually stored as *little-endian* format, as shown below):

```
↑ higher memory

0xbffff400 | 0x00  0xa0  0x04  0x08 ← Return address
0xbffff3fc | 0x    0x    0x    0x
0xbffff3f8 | 0x    0x    0x    0x
0xbffff3f4 | 0x    0x    0x    0x

↓ lower memory
```

Complete the stack diagram above as you work through the disassembly one instruction at a time, and leave unusued cells blank. Immediately after the instruction at address `0x0804a01c` has executed, to what addresses will registers `ebp` and `esp` point to?

(b) StackGuard is a compiler-based technique for defending against stack-based buffer overflows. It works by inserting a *canary*—a known value stored in each function's stack frame immediately below the return address—and verifying that this canary value hasn't changed when the function returns; if it has, the program halts. Explain why this prevents the basic form of stack buffer overflow attacks you performed in Project 2.

(c) You are writing a basic overflow-to-shellcode exploit. After trial and error, you finally find an input that succeeds—but then then you try again with exactly the same bytes and it doesn't seem to work anymore! What is going on? How can you bypass it?

8. **Ethics.** Consider the following scenario: a worm is infecting systems by exploiting a bug in a popular program. It spreads rapidly, and systems where it is deleted quickly become reinfected. A security researcher decides to launch a counterattack via a *defensive* worm: whenever a break-in attempt comes from a remote host, the defensive worm detects it, thwarts the break-in, and exploits the same bug to spread to the attacking host; on that host, it deletes the original worm; it then waits until *that* system is attacked, and the cycle repeats.

   (a) Many people would claim that launching such a counterattack in this scenario is ethically unacceptable. Briefly argue in support of this view.

   (b) Are there circumstances or conditions under which an active security counterattack would be ethically justified? Briefly explain your reasoning.