CS.201P: Introduction to Computer Security 2/2/2019

Homework 2

Due Thursday 2/14/2019, 11:59pm, by upload to gradescope.com

Solve the following problems from the [GT] textbook:

1 Exercise R-2.16

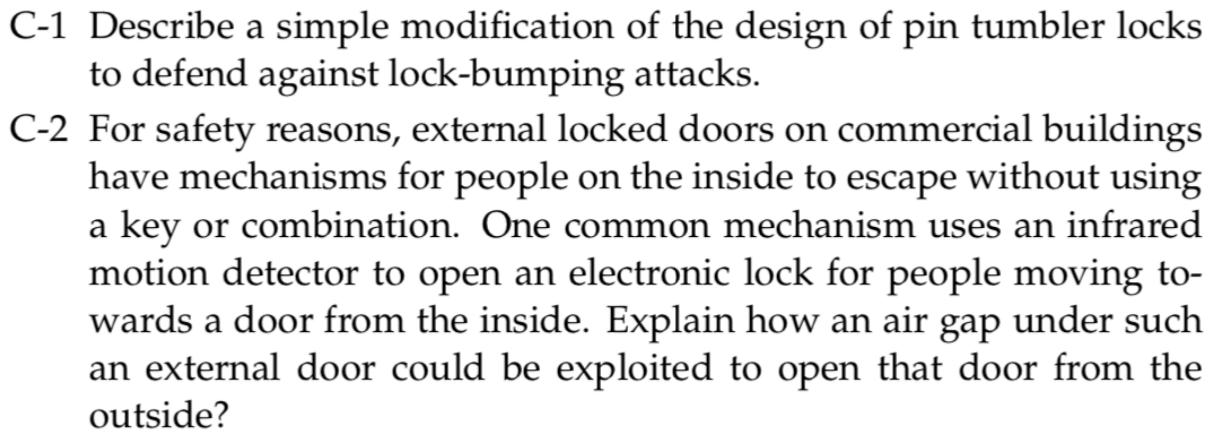
R-16 How can you check if a public computer has a USB keylogger installed?

Solution:

There are two kinds of keylogger, Software and Hardware; for USB keylogger is the latter HKL, which collecting keystrokes coming from the keyboard. It is hard to be detected by conventional Malwarebytes aimed for software keylogger like Anti-Rootkit, and some system command like "lsusb" is hard to show any difference, you can:

1) Firstly, if possible, set up your desk and PC in a way that the connector of your keyboard is always or at least often enough visible, and check by Visual inspection on the connector;
2) When visual inspection is impractical, use a key scrambler software, like KeyScrambler, KeyGhost;
3) To be more specific detail in technology level, we can use additional electronic components / additional power consumption check method, since the HLK will consume more energy; or check by data flow, Time Measurement, Changes to USB Properties, etc. (Fabian Mihailowitsch).
4) Also, since some of the USB keylogger may not be able to work once there's an USB hub in the keyboard (and others claim that their product does), which i.e. Computer → Keylogger → USB Hub → Keyboard, we can thus use an USB extension cable to check whether the keyboard still works in some cases.

2 Exercise C-2.1 and C-2.2


C-1 Describe a simple modification of the design of pin tumbler locks to defend against lock-bumping attacks.

C-2 For safety reasons, external locked doors on commercial buildings have mechanisms for people on the inside to escape without using a key or combination. One common mechanism uses an infrared motion detector to open an electronic lock for people moving towards a door from the inside. Explain how an air gap under such an external door could be exploited to open that door from the outside?

1.1 Solution:

The Pin tumbler locks consist of the following pin mechanism:

1) The mechanism consists of a cylinder lock, a drive pin and a pin. All pins are mounted in a cylinder lock.

2) If the user increases the number of pins in the cylinder lock, the number of system codes may increase.

3) This leads to a large number of system code combinations. But in reality, when the key in the cylinder increases, the possibility of key exchange is reduced, which automatically jeopardizes safety.

4) The possibility of code duplication also increases and highlights the possibility of unauthorized access.

5) When the latch is open, the lever pushes the locking pin. When the latch is closed, the needle stack is prevented.

6) The pins placed here are called drive pins. Hereinafter referred to as a key pin, both are used in conjunction with a spring. Therefore, key copying can be done easily. Design changes can be done in the following ways:

7) If there is no key, the refill is pushed down by the spring so that the driver's pen will cross the plug and the outer casing.

8) When the correct key is inserted, the back of the key pushes the key stack up and they align with the release line.

9) The direction of the plug and play line may cause the pin to rotate.

1.2 Solution:

Items that generate heat also produce infrared radiation, including animals and humans. We can use large warm metal plates. We can slide the paper under the door and move it to trigger the motion detector and unlock the door.

3. Exercise C-2.6.

Compare the size of the effective search space compared to the a-priori search space if all 4-digit codes are equally likely.

C-6 A thief walks up to an electronic lock with a 10-digit keypad and he notices that all but three of the keys are covered in dust while the 2, 4, 6, and 8 keys show considerable wear. He thus can safely assume that the 4-digit code that opens the door must be made up of these numbers in some order. What is the worst case number of combinations he must now test to try to open this lock using a brute-force attack?

Solution:

According to the assumption "an electronic lock with a 10-digit keypad and he notices that all but three of the keys are covered in dust while the 2, 4, 6, and 8 keys show considerable wear.", there should be two cases:

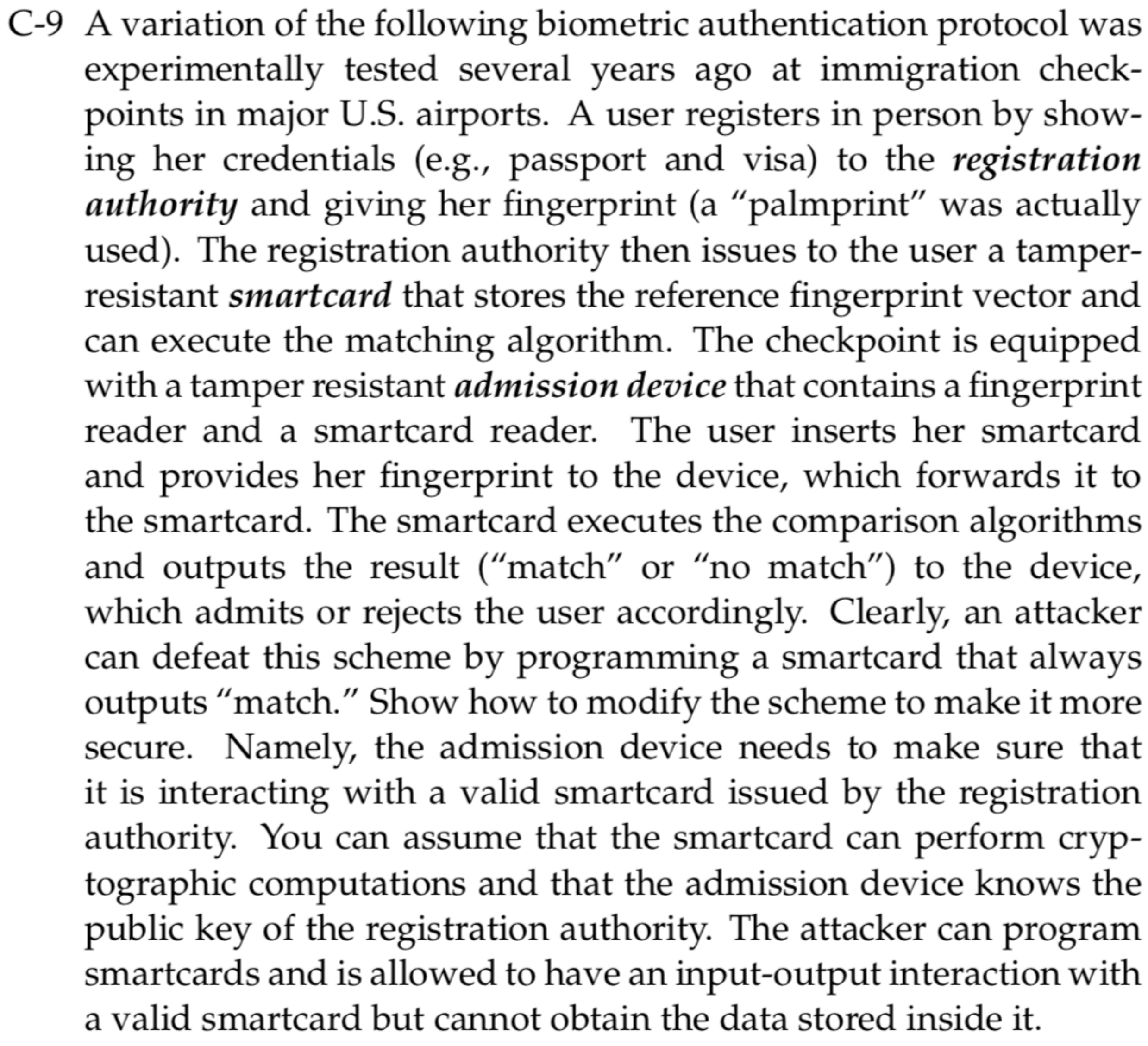
① First we are told that only 3 keys are not covered in dust, which suggests that the PIN is made of only 3 digits corresponding to the 3 keys; but it's strange that the assumption or the thief doesn't see and tell us exactly the clue of which the 3 keys are.

② What's add, then we are told that 4 keys have considerable wear, which suggests that the PIN is made of 4 digits corresponding to these 4 keys.

So we take ② case as the reality. So there are 4*3*2*1 = 24 kinds of permutation of the key code, which means that in the worst case, the thief would only have to test 24 codes.

Compare the size of the effective search space compared to the a-priori search space if all 4-digit codes are equally likely, where there will be 10*10*10*10 = 10000 kinds of search space, in this case the attack space is largely reduced from the clue: have 4 considerable wear + the code is 4 digits.

4. Exercise C-2.9



C-9 A variation of the following biometric authentication protocol was experimentally tested several years ago at immigration checkpoints in major U.S. airports. A user registers in person by showing her credentials (e.g., passport and visa) to the ***registration authority*** and giving her fingerprint (a "palmprint" was actually used). The registration authority then issues to the user a tamper-resistant ***smartcard*** that stores the reference fingerprint vector and can execute the matching algorithm. The checkpoint is equipped with a tamper resistant ***admission device*** that contains a fingerprint reader and a smartcard reader. The user inserts her smartcard and provides her fingerprint to the device, which forwards it to the smartcard. The smartcard executes the comparison algorithms and outputs the result ("match" or "no match") to the device, which admits or rejects the user accordingly. Clearly, an attacker can defeat this scheme by programming a smartcard that always outputs "match." Show how to modify the scheme to make it more secure. Namely, the admission device needs to make sure that it is interacting with a valid smartcard issued by the registration authority. You can assume that the smartcard can perform cryptographic computations and that the admission device knows the public key of the registration authority. The attacker can program smartcards and is allowed to have an input-output interaction with a valid smartcard but cannot obtain the data stored inside it.

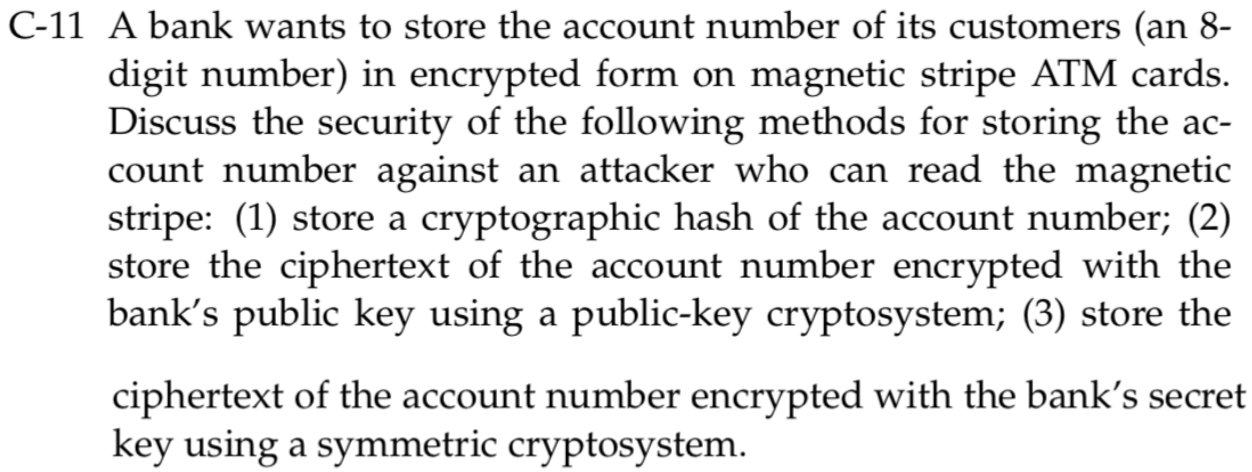
Solution:

There are several valid solutions.

One solution is for the registration authority to provide the smart card with a copy of the reference fingerprint signed by the registration and then perform a fingerprint comparison by the approval device instead of the smart card. In this way, the approval mechanism knows that the reference fingerprint is valid (by checking the RA signature) and that the person's fingerprint matches, the attack thus cannot pretend the "match" result.

Another solution is for the registrar to issue a public / private key pair for each valid smart card and give it a signed copy of the public key. When a smart card is inserted, it provides the registration authority with its (signed) public key, which the admission authority validates. When the registration authority reads the fingerprint, it then encrypts it with the smart card's public key and sends the ciphertext to the smartcard. The smart card then responds with a matching or mismatched response and a decrypted version of the fingerprint ciphertext just sent by the admission mechanism. In this way, the access rights are confirmed, the smart card has a valid public / private key pair and knows the associated private key (since it uses the fingerprint for decryption), so that the access rights can have a higher degree trust the smart card performs an effective test the fingerprint through. With the unique key and bio info, only the true right card with the true user (fingerprint) is able to get the match output.

5. Exercise C-2.11



C-11 A bank wants to store the account number of its customers (an 8-digit number) in encrypted form on magnetic stripe ATM cards. Discuss the security of the following methods for storing the account number against an attacker who can read the magnetic stripe: (1) store a cryptographic hash of the account number; (2) store the ciphertext of the account number encrypted with the bank's public key using a public-key cryptosystem; (3) store the ciphertext of the account number encrypted with the bank's secret key using a symmetric cryptosystem.

Solution:

Since the account number is assumed to be a secret, so:

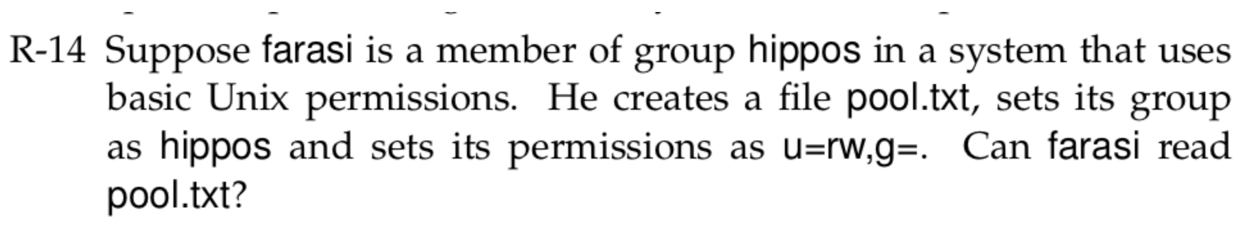
(1) This solution is not very secure, although it is computationally difficult to determine the input of a cryptographic hash function that only returns its output. The problem is that there is not so much plain text. An attacker can hash any possible 8-bit account by brute force and check the hash value stored on the card from this list.

(2) If the public-key cryptography algorithm it uses is deterministic, such as RSA, this solution is not very secure. The problem is that everyone knows the bank's public key and does not have that much plain text, just as the former question, so an attacker can encrypt any 8-bit account by brute force and check the card's encryption against that list.

(3) This solution is very secure because only the key of the bank can encrypt and decrypt the

encrypted text. After decryption, the bank also receives an account, so there is no additional linkage between the card and the account. What's more, any attacker does not know the secret key of the bank and assumes that even if the number of plain texts is relatively small, a sufficiently long ciphertext will be generated.

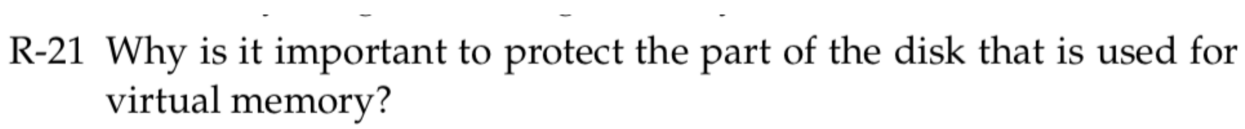
6. Exercise R-3.14


R-14 Suppose farasi is a member of group hippos in a system that uses basic Unix permissions. He creates a file pool.txt, sets its group as hippos and sets its permissions as u=rw,g=. Can farasi read pool.txt?

Solution:

As the file owner, he sets the file permissions to "u = rw. g = "indicates that the user who is the owner of the file gets read and write access to this file, and members of the group hippos cannot access this file. Farasi is both a file owner and a member of the Hippopotamus group, so both rules are theoretically applicable. However, the rules of the file owner take precedence over the rules of the group according to UNIX design. In other words, the rules of a group should not be understood as rules for all group members, but all group members except the file owner whose permissions are set separately by the u= command. So Farasi can not only read this file, but also write it.

7. Exercise R-3.21


R-21 Why is it important to protect the part of the disk that is used for virtual memory?

Solution:

Virtual memory contains the current state of each executing program. Some of the virtual memory is in RAM and some is stored on the hard drive, but in principle all memory may be on the hard drive. Therefore, an attacker who can access this part of the disk can reconstruct all the data (ie, local variables) of the executable program, which may contain sensitive information, for example, the password entered by the user or the key created and used by the user to encrypt the communication, so receiving these an attacker of the key can decrypt this communication. An attacker who can write to the virtual memory space can also change the state of the execution code.

8. Exercise C-3.1

C-1 Bob thinks that generating and storing a random salt value for each userid is a waste. Instead, he is proposing that his system administrators use a SHA-1 hash of the userid as its salt. Describe whether this choice impacts the security of salted passwords and include an analysis of the respective search space sizes.

Solution:

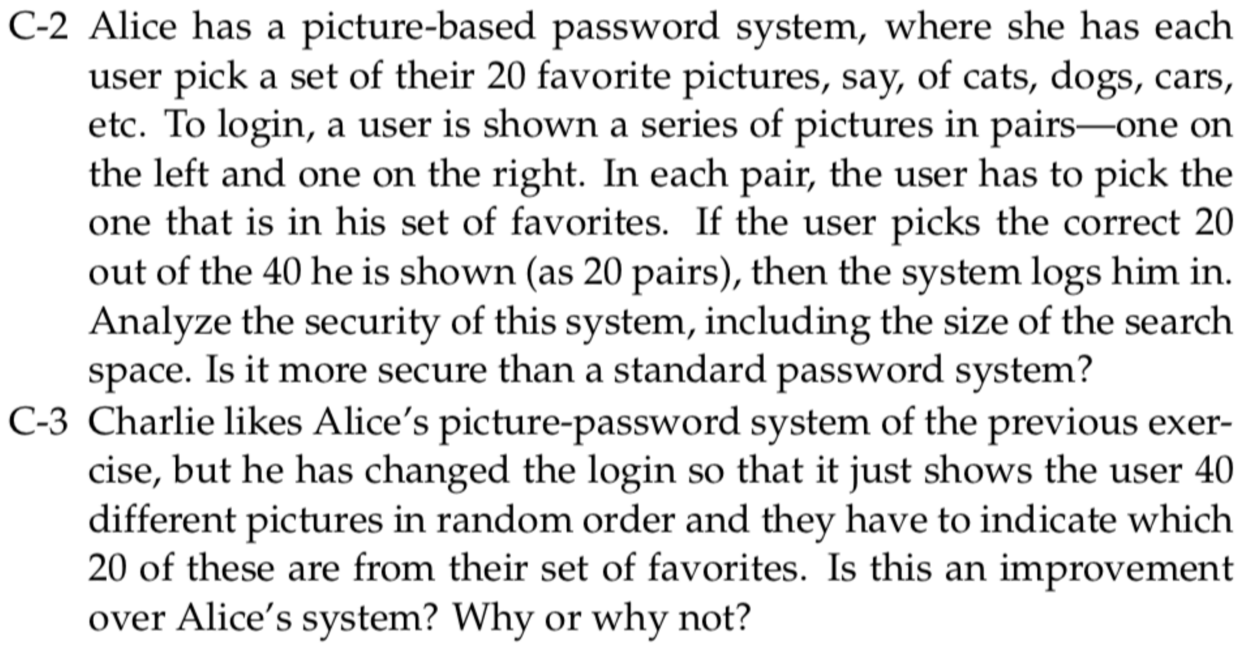
For proving the statement, consider the length of the salt is = 32 bits.

If we use a salt for any password, this technique will increase the address space of the password system to 2^32 times. Since salts with the same user ID may vary at different times due to the random selection of salts, the security is therefore higher.

If the administrator uses the SHA-1 hash of the user ID as salt, the security of the system may be much lower. The hash generated by SHA-1 is the same every time. Thus, if the attacker knows the user ID and cracks the normal password, using the SHA-1 algorithm to generate the hash for the user is not a big question for the attacker.

Therefore, using a SHA-1 hash as a salt instead of a salt with random distribution can reduce the safety of the system. If the attacker knows the user ID, the address space of the cryptosystem is not significantly affected (it does not increase a lot).

9. Exercise C-3.2 and 3.3



C-2 Alice has a picture-based password system, where she has each user pick a set of their 20 favorite pictures, say, of cats, dogs, cars, etc. To login, a user is shown a series of pictures in pairs—one on the left and one on the right. In each pair, the user has to pick the one that is in his set of favorites. If the user picks the correct 20 out of the 40 he is shown (as 20 pairs), then the system logs him in. Analyze the security of this system, including the size of the search space. Is it more secure than a standard password system?

C-3 Charlie likes Alice's picture-password system of the previous exercise, but he has changed the login so that it just shows the user 40 different pictures in random order and they have to indicate which 20 of these are from their set of favorites. Is this an improvement over Alice's system? Why or why not?

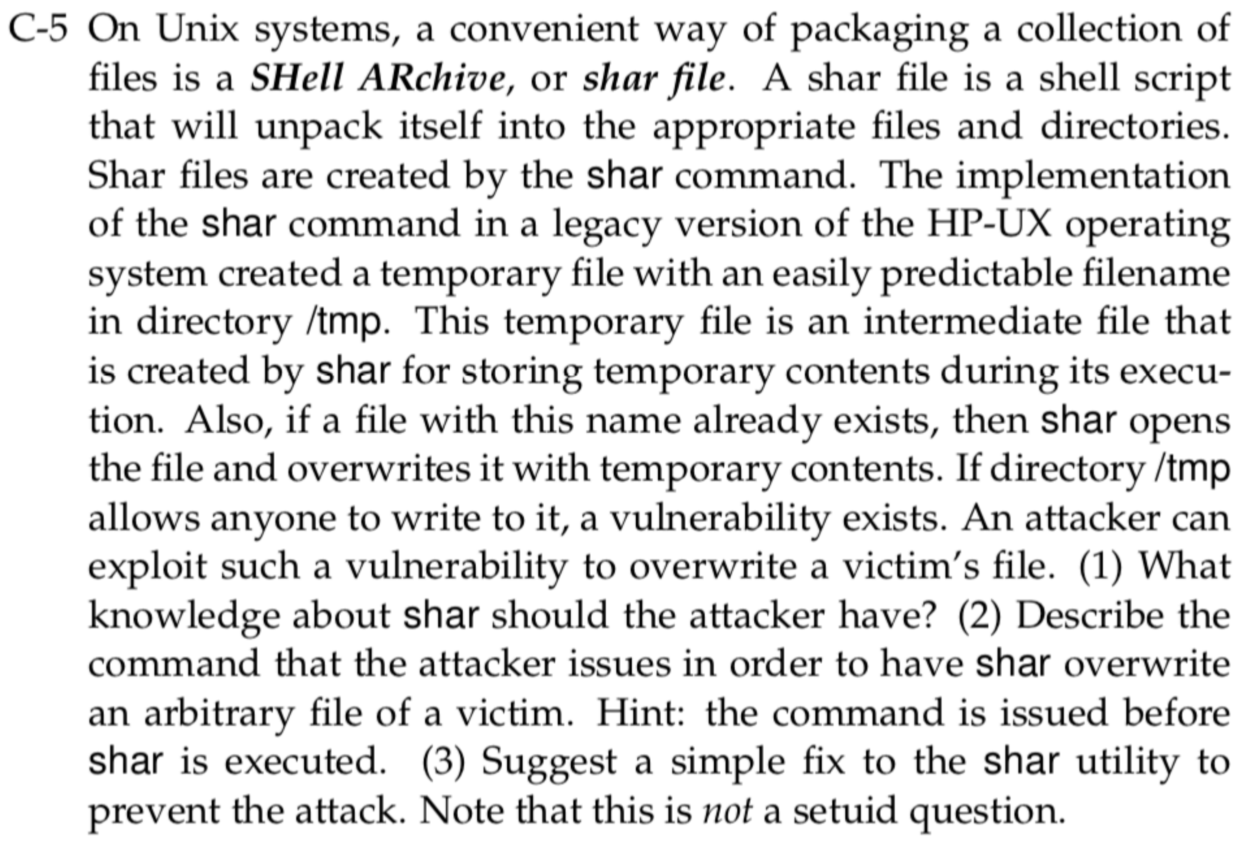
C3.2 Solution:

There are two pictures for each choice, one on the left and one on the right. Thus, with 20 pairs, there is a search space of 2^20, which is roughly 1,000,000. This is roughly as secure than a standard password, for which there are dictionaries with 500,000 passwords that can be used in dictionary attacks. If the number of picture pairs is increased to 40, however, then the security is much higher. Meanwhile, even picking one of 20 picture pairs requires 20 mouse clicks, which would take longer than typing in a traditional password.

C3.3 Solution:

If the search space is defined by picking 20 out of 40, then the size of the search space is the combinatorial function "40 choose 20," which is 40!/(20!)^2, i.e., it is over 137 trillion. So it is largely bigger than the former one.

10. Exercise C-3.5



C-5 On Unix systems, a convenient way of packaging a collection of files is a ***SHell ARchive***, or ***shar file***. A shar file is a shell script that will unpack itself into the appropriate files and directories. Shar files are created by the shar command. The implementation of the shar command in a legacy version of the HP-UX operating system created a temporary file with an easily predictable filename in directory /tmp. This temporary file is an intermediate file that is created by shar for storing temporary contents during its execution. Also, if a file with this name already exists, then shar opens the file and overwrites it with temporary contents. If directory /tmp allows anyone to write to it, a vulnerability exists. An attacker can exploit such a vulnerability to overwrite a victim's file. (1) What knowledge about shar should the attacker have? (2) Describe the command that the attacker issues in order to have shar overwrite an arbitrary file of a victim. Hint: the command is issued before shar is executed. (3) Suggest a simple fix to the shar utility to prevent the attack. Note that this is *not* a setuid question.

Solution:

（1） The attacker should know:

- how to apply the shar command to the targeted file.
- the name and extensions of the targeted file.
- the path of the target file, where is it exists.
- the access permissions of the file.

（2） Commands the attacker should issue to overwrite an arbitrary file of a victim:

- To find the targeted file in system, use list command to view the files in the system.
- use the ls –l command to check all the files in current dir.
- With the targeted file path, name and extension, use shar command on the targeted file.

（3） Fix methods:

- The owner can restrict the permissions for modifying the file to authorized users only.
- Maintain password based security mechanisms for modifying the secured files.
- The owner can have the back-up for important files on some other secured location.

- The owner can hide the secured files from viewing by unauthorized users.

11. Exercise C-3.8

C-8 Consider the following piece of C code:

```
int main(int argc, char *argv[])
{
char continue = 0;
char password[8];
strcpy(password, argv[1]);
if (strcmp(password, "CS166")==0)
continue = 1;
if (continue)
{
∗login();
}
}
```

In the above code, ∗login() is a pointer to the function login() (In C, one can declare pointers to functions which means that the call to the function is actually a memory address that indicates where the executable code of the function lies). (1) Is this code vulnerable to a buffer-overflow attack with reference to the variables password[] and continue? If yes, describe how an attacker can achieve this and give an ideal ordering of the memory cells (assume that the memory addresses increase from left to right) that correspond the variables password[] and continue of the code so that this attack can be avoided. (2) To fix the problem, a security expert suggests to remove the variable continue and simply use the comparison for login. Does this fix the vulnerability? What kind of new buffer overflow attack can be achieved in a multiuser system where the login() function is shared by a lot of users (both malicious and and nonmalicious) and many users can try to log in at the same time? Assume for this question only (regardless of real systems' behavior) that the pointer is on the stack rather than in the data segment, or a shared memory segment. (3) What is the existing vulnerability when login() is not a pointer to the function code but terminates with a return() command? Note that the function strcpy does not check an array's length.

Solution:

(1) Yes, It's vulnerable. The attack is that when strcpy copies the string pointed by argv[1] into the *password* array, if the string is longer than 8 characters, which the size of the array, then the next character in the string will overwrite the variable stored next to the *password* array pointer. If the characters in an array are stored bottom-up (usually the case), from the end to the beginning then the variable which will be overwritten is the one declared just before the *password* variable, i.e. the *continue* variable. In that case the adversary can set the continue variable to 1 just by having a symbol `1` as the 9th character in the arvg[1] string, and this way go to the login procedure regardless of whether the first 8 characters matched the password. However, this attack wouldn't work if the characters in an array are stored in the other direction, for example, define variable password first and then the continue.

(2) The problem is that if the *login* is a pointer whose address is stored in a local variable then this variable will be stored on the stack, just like the *password* variable. Therefore as above, with the buffer overflow attack the adversary could overwrite this pointer so that instead of going to the regular login procedure, the program would jump to a place determined by the value the adversary places in that pointer. If the same procedure instance handles several users, then overwriting this pointer will affect other users as well.

(3) Regardless of the above two problems, the fact that there's no check on the length of the string copied from the argv[1] pointer means that this code is vulnerable to the standard buffer overflow attack, i.e. the adversary can overwrite the variable storing the return address of this procedure, and this way the control after the procedure exits will jump to a location specified by the adversary. This vulnerability aren't solved, the key to solve is to use safely programming, like strncpy().

12. Exercise R-4.15

R-15 What would be the financial advantage for a malware designer to create lots of different malicious code instances that all exploit the same vulnerability yet have different malware signatures?

Solution:

By creating many different instances of malware, designers can set each malware as a separate product to obtain more profits. By setting a different malware signature for each instance, he can advertise that every malware he sell is not yet in anti-malware signature database.

13. Exercise C-4.1

C-1 Explain why any computer worm that operates without human intervention is likely to either be self-defeating or inherently detectable.

Solution:

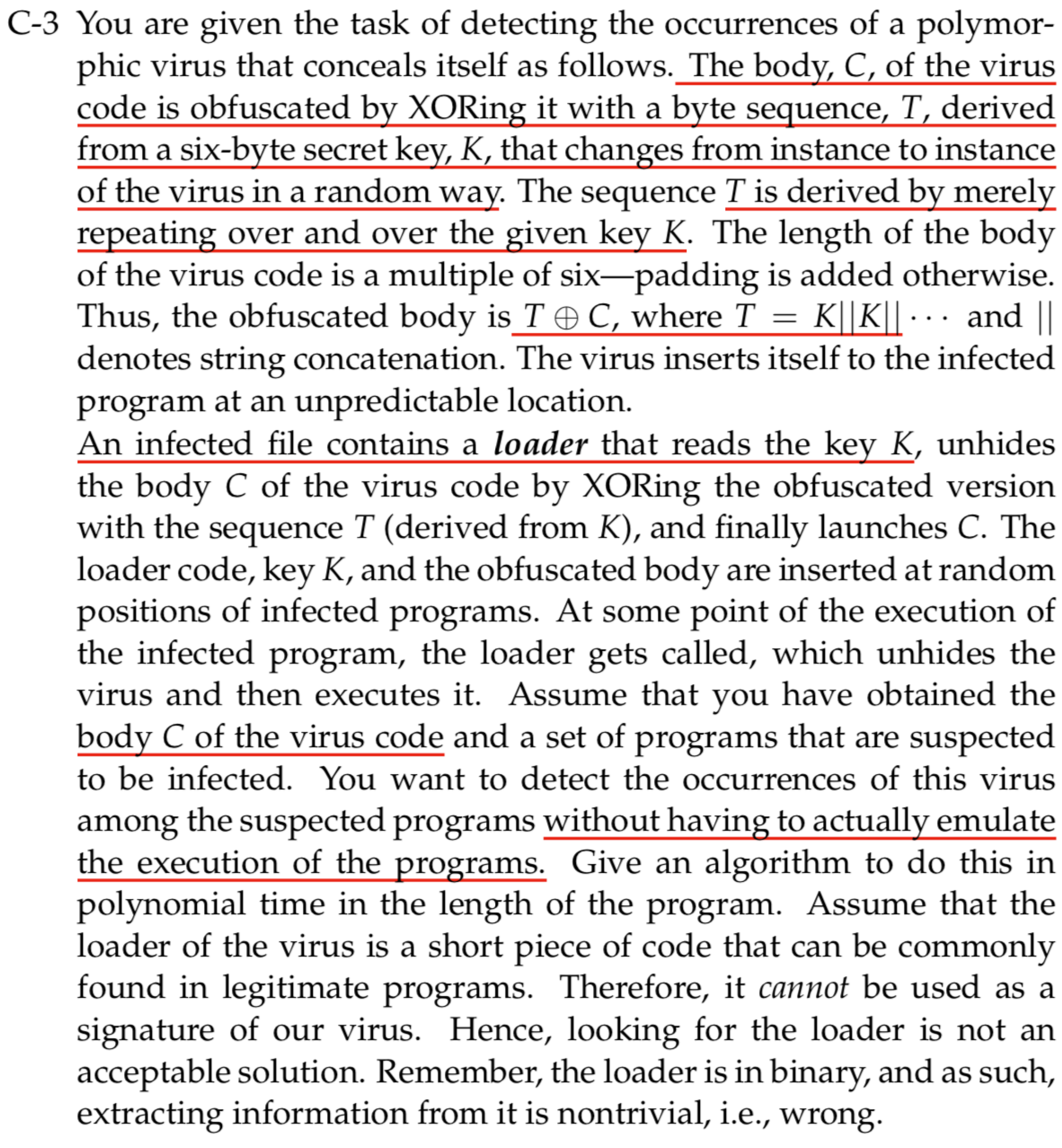
Although a worm may operate without human intervention, it does not mean it will survive self-defeating due to: A worm needs to create entry in system registry to survive reboot, the system registry helps the worms to propagate. However, the malware detection software always checks this entry (and other registry entries specifying programs to run at startup) for suspicious executable names, thus the worm may fail.

For example, in windows there is an entry:

“HKEYLOCALMACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\run”

In these entry windows stores the list of programs that are to be executed on boot. A malware needs to create an entry here in order to replicate itself. Now that most of the worms are designed to create its replica, they create many files that in turn increase the space usage. This change can be easily identified by the user. He may install a tool or manually delete the entries in “RUN” of registry.

14. Exercise C-4.3

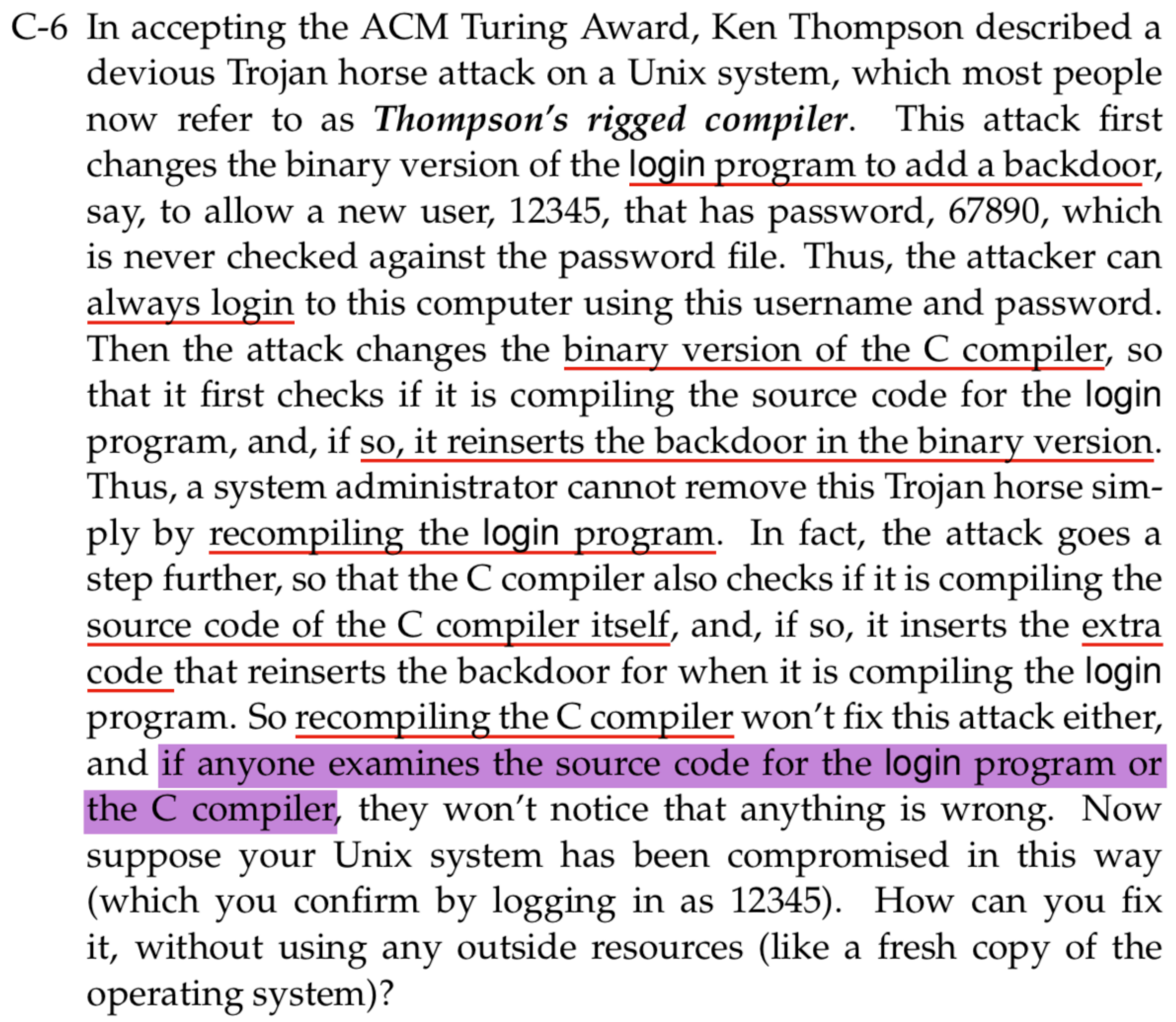


C-3 You are given the task of detecting the occurrences of a polymorphic virus that conceals itself as follows. The body, C, of the virus code is obfuscated by XORing it with a byte sequence, T, derived from a six-byte secret key, K, that changes from instance to instance of the virus in a random way. The sequence T is derived by merely repeating over and over the given key K. The length of the body of the virus code is a multiple of six—padding is added otherwise. Thus, the obfuscated body is $T \oplus C$, where $T = K||K|| \cdots$ and $||$ denotes string concatenation. The virus inserts itself to the infected program at an unpredictable location.
An infected file contains a ***loader*** that reads the key K, unhides the body C of the virus code by XORing the obfuscated version with the sequence T (derived from K), and finally launches C. The loader code, key K, and the obfuscated body are inserted at random positions of infected programs. At some point of the execution of the infected program, the loader gets called, which unhides the virus and then executes it. Assume that you have obtained the body C of the virus code and a set of programs that are suspected to be infected. You want to detect the occurrences of this virus among the suspected programs without having to actually emulate the execution of the programs. Give an algorithm to do this in polynomial time in the length of the program. Assume that the loader of the virus is a short piece of code that can be commonly found in legitimate programs. Therefore, it *cannot* be used as a signature of our virus. Hence, looking for the loader is not an acceptable solution. Remember, the loader is in binary, and as such, extracting information from it is nontrivial, i.e., wrong.

Solution:

Since you know the virus code C, you can try to retrieve T with the file F starting at each offset j (a multiple of 6 bytes) and xor to C, and test whether the result is in the form K|| K || ... for some 6-byte strings K. Note that if j is the correct beginning of the polymorphic virus, the result is the concatenation of the key K with itself, ie K || K || K || ... because of the obfuscated virus body C gets from C xor K || K || K || ...). On the other hand, if j does not point to a obfuscated body, it is unlikely that it will be that form anyway. Therefore, this is a good test that requires a time proportional to the length of the file and is therefore relatively efficient.

15. Exercise C-4.6



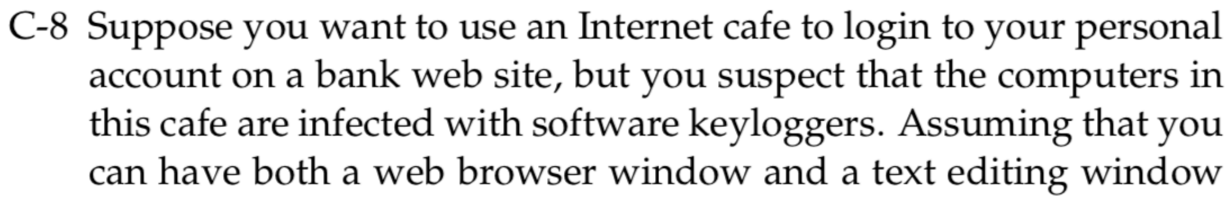
C-6 In accepting the ACM Turing Award, Ken Thompson described a devious Trojan horse attack on a Unix system, which most people now refer to as ***Thompson's rigged compiler***. This attack first changes the binary version of the login program to add a backdoor, say, to allow a new user, 12345, that has password, 67890, which is never checked against the password file. Thus, the attacker can always login to this computer using this username and password. Then the attack changes the binary version of the C compiler, so that it first checks if it is compiling the source code for the login program, and, if so, it reinserts the backdoor in the binary version. Thus, a system administrator cannot remove this Trojan horse simply by recompiling the login program. In fact, the attack goes a step further, so that the C compiler also checks if it is compiling the source code of the C compiler itself, and, if so, it inserts the extra code that reinserts the backdoor for when it is compiling the login program. So recompiling the C compiler won't fix this attack either, and if anyone examines the source code for the login program or the C compiler, they won't notice that anything is wrong. Now suppose your Unix system has been compromised in this way (which you confirm by logging in as 12345). How can you fix it, without using any outside resources (like a fresh copy of the operating system)?

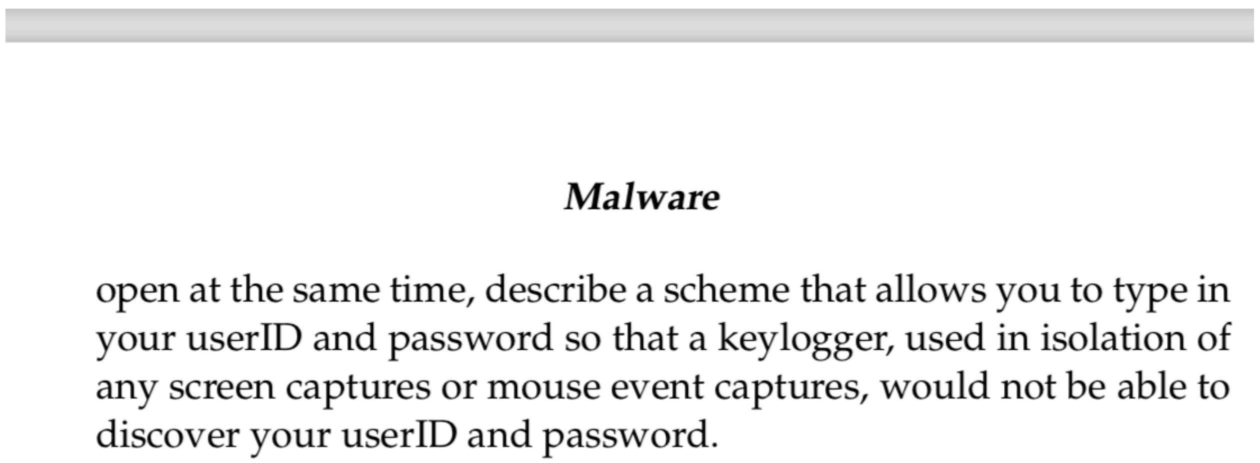
Solution:

One solution could be to edit the source of the C compiler, change the variable names, and insert spurious instructions such as conditional statements that are never executed because their conditions are never met. Hopefully, after rewriting the source code, the virus embedded in the C compiler cannot match this rewrite code with its signature in C compiler code. Ironically, this is a similar approach, and the metamorphic virus tries to avoid being detected by antivirus software by rewriting itself in this way to avoid matching the well-known signature of the code. If the rewrite is successful, the compiled code cannot reinsert the virus into the result. This results in a clean C compiler that can then be used to recompile the login process.

Another possible solution, which is maybe more difficult in practice, is to decompile the C compiler executive and try to find the Trojan code portion that will reinsert this malicious login modifying. If you find it, you can fix it by manually changing the binary C compiler. After cleaning up the C compiler in this way, follow the steps above.

16. Exercise C-4.8


C-8 Suppose you want to use an Internet cafe to login to your personal account on a bank web site, but you suspect that the computers in this cafe are infected with software keyloggers. Assuming that you can have both a web browser window and a text editing window



open at the same time, describe a scheme that allows you to type in your userID and password so that a keylogger, used in isolation of any screen captures or mouse event captures, would not be able to discover your userID and password.

Solution:

The key point of solution is to type random characters on the text editor in between of typing your sensitive information (e.g. password) on the browser.

The keylogger works based on monitoring the interaction of the active window with the keyboard and does not detect mouse events. If we open both the web browser and the text editing window, it's best to bypass the keylogger by tapping quickly between the two windows and randomly press characters and type others when typing username and password. This confuses the keylogger by receiving a long string or unusable characters to the attacker. If there is no text editor on the

computer, we can use the second application, or even the various tabs of the browser, and use them to enter random characters between the two.
It is recommended that you change your password and enable multi-factor authentication immediately after use in a public suspicious area, even if you follow the guidelines above.