

Homework 1

Due: February 11, 2009, 10:00 PM EST

Please hand in your solutions as a pdf document

Problem 1 As soon as Barack took office, he decided to embrace modern technology by communicating with cabinet members over the Internet using a device that supports cryptographic protocols.

1. In a first attempt, Barack exchanges with Tim brief text messages, encrypted with public key cryptography, to decide the exact amounts of bailout money to give to the largest 10 banks in the country. Let p_B and p_T be the public keys of Barack and Tim, respectively. A message m sent by Barack to Tim is transmitted as $E_{p_T}(m)$ and the reply r from Tim to Barack is transmitted as $E_{p_B}(r)$. The attacker can eavesdrop the communication and knows the following information:

- public keys p_B and p_T and the encryption algorithm;
- the total amount of bailout money authorized by congress is \$900B;
- the names of the largest 10 banks;
- the amount each bank will get is a multiple of \$1B;
- messages and replies are terse exchanges of the following form:

Barack: How much to Citibank?

Tim: \$144B.

Barack: How much to Bank of America?

Tim: \$201B.

...

Describe how the attacker can learn the bailout amount for each bank even if he cannot derive the private keys.

2. As a result of the above attack, Barack decides to modify the protocol for exchanging messages. Describe two simple modifications of the protocol that are not subject to the above attack. The first one should use random numbers and the second one should use symmetric encryption.
3. Barack often sends funny jokes to Hillary. He does not care about confidentiality of these messages but wants to get credit for the jokes and prevent Bill from claiming authorship of or modifying them. How can this be achieved using public key cryptography?
4. As public-key cryptography is computationally intensive and drains the battery of Barack's device, he comes up with an alternative approach. First, he shares a secret key k with Hillary but not with Bill. Next, together with a joke x , he sends over the value $d = h(k||x)$, where h is a cryptographic hash function. Does value d provide assurance to Hillary that Barack is the author of x and that x was not modified by Bill? Justify your answer.

5. Barack periodically comes up with brilliant ideas to stop the financial crisis, provide health care to every citizen, and save the polar bears. He wants to share these ideas with all the cabinet members but also get credit for the ideas. Extending the above approach, he shares a secret key k with all the cabinet members. Next, he broadcasts each idea z followed by value $h(k||z)$. Does this approach work or can Tim claim that he came up with the ideas instead of Barack? Justify your answer.

Problem 2 Write a program in pseudocode that acts as a *guardian* for a file, allowing anyone to append to the file, but to make no other changes to it. This may be useful, e.g., to add information to a log file. Your program, to be named `append`, should take two strings `file1` and `file2` as arguments, denoting the paths to two files. Operation `append(String file1, String file2)` copies the contents of `file1` to the end of `file2`, provided that the user performing the operation has read permission for `file1` and `file2`. If the operation succeeds, 0 is returned. On error, -1 is returned.

Assume that the operating system supports the `setuid` mechanism and that `append` is a `setuid` program owned by a user called `guardian`. The file to which other files get appended (`file2`) also owned by `guardian`. Anyone can read its contents. However, it can be written only by `guardian`.

Write your program in pseudocode using the following Java-style system calls:

- `int open(String path_to_file, String mode)` opens a file in a given mode and returns a positive integer that is the descriptor of the opened file. String `mode` is one of `READ_ONLY` or `WRITE_ONLY`.
- `void close(int file_descriptor)` closes a file given its descriptor.
- `byte[] read(int file_descriptor)` reads the content of the given file into an array of bytes and returns the array.
- `void write(int file_descriptor, byte[] source_buffer)` stores a byte array into a file, replacing the previous content of the file.
- `int getUid()` gets the real user ID of the current process.
- `int getEuid()` gets the effective user ID of the current process.
- `void setEuid(int uid)` sets the effective user ID of the current process, where `uid` is either the real user ID or the saved effective user ID of the process.

Error conditions that occur in the execution of the above system calls (e.g., trying to open a file without having access right to it or using a nonexistent descriptor) trigger exception `SystemCallFailed`, which should be handled by your program. Note that you do not need to worry about buffer overflow in this question.

Problem 3 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 existed, 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.

Problem 4 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 $T = K||K||\dots$. The virus inserts itself to the infected program at an unpredictable location.

An infected file contains a *loader* that reads 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. The figure below shows a schematic drawing of two infected programs.

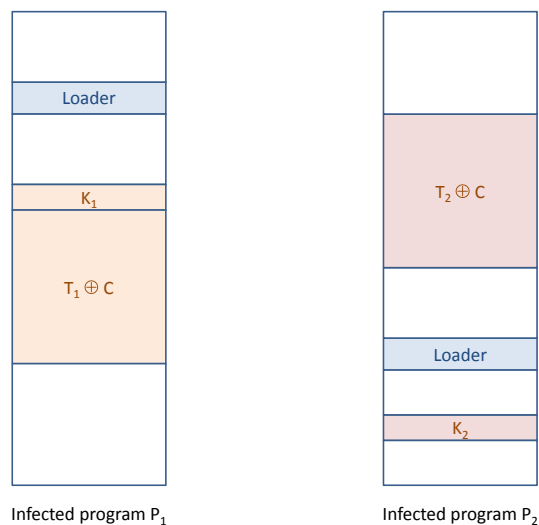


Figure 1: Two infected programs P_1 and P_2 . Note that the obfuscated virus may be inserted at different positions of a program.

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. How would you approach this problem? 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.