Mittuniversitetet
MID SWEDEN UNIVERSITY

Final exam

DT145G Computer Security

Daniel Bosk
daniel.bosk@miun.se
Phone: 060 - 14 8709

2014-06-03

Instructions

Carefully read the questions before you start answering them. Note the time limit of the exam and plan your answers accordingly. Only answer the question, do not write about subjects remotely related to the question.

Write your answers on separate sheets, not on the exam paper. Only write on one side of the sheets. Start each question on a new sheet. Do not forget to *motivate your answers.*

Make sure you write your answers clearly, if I cannot read an answer the answer will be awarded no points – even if the answer is correct. The questions are *not* sorted by difficulty.

Time 5 hours.

Aids Dictionary.

Maximum points 29

Questions 5

Preliminary grades

The following grading criteria applies: E $\geq$ 50%, D $\geq$ 60%, C $\geq$ 70%, B $\geq$ 80%, A $\geq$ 90%.

Questions

The questions are given below. They are not given in any particular order.

1. Explain the following terms:

(1p) (a) Confidentiality

(1p) (b) Integrity

(1p) (c) Availability

(1p) (d) Accountability

(1p) (e) Non-Repudiation

2. A user has set the access control list of a file in such a way that only she has access to this file.

(3p) (a) If you would like to read this file you can, explain how you would do and why it works (also what assumptions you need).

(3p) (b) Suggest how the user must do to guarantee that no one but herself can read this file, also motivate why this protection works.

3. The University password composition policy is as follows[1]: The password must be at least eight (8) characters long, further, the first eight characters must contain two uppercase and two lowercase letters as well as two numbers.

(3p) (a) Compute the information gained by an attacker from knowing this policy.

(3p) (b) Compare this to the information gained if the policy would have been "the password must be at least 16 characters long".

4. The DRM manager in a company claims to have devised a secure DRM for the company's software product.

   It works as follow: To use the software the user has to enter a authentication code provided with the software. If valid the software will connect to the servers and use some functionality stored there.

   For the connection the software computes a MAC of itself (the executable file) using HMAC-SHA256 and a secret key k shared with the company servers, this is to prevent modification attempts of the executable file. It then connects to one of these servers, it verifies the server's certificate to establish a secure connection, then sends the MAC which is verified by the server, the server verifies and performs the software's request if the MAC is valid.

   There are several flaws in this DRM scheme.

(3p) (a) Explain what can be done to work around this DRM. (A solution which doesn't need a copy of a valid authentication code is of course better than a solution which does.)

(3p) (b) Suggest some improvements of this DRM and motivate what vulnerabilities each improvement removes. Also motivate whether your improved version is totally secure or not.

5. Look at the C code in Listing 1 on the next page.

(3p) (a) Identify all vulnerabilities in that code and motivate by stating how they can be exploited.

(3p) (b) Suggest improvements to remedy these vulnerabilities, you must motivate why they work.

[1] It's slightly more advanced than this, but it's simplified here for reasons of convenience. Also, the effect might actually be the same anyway, depending on if users actually use the extension or not.

```c
#include <stdio.h>

int
get_some_input( void )
{
  char buffer[128];

  printf( "Please enter the key: " );
  scanf( "%s", buffer );

  /* process input */

  return 0;
}

void
make_full_name( char *dst, int dstlen,
                const char *src, int srclen,
                int maxsize )
{
  if ( dstlen + srclen + 1 >= maxsize )
    return -1;

  strncat( dst, " ", 1 );
  return strncat( dst, src, srclen );
}

int
main( int argc, char **argv )
{
  char first[256];
  char last[256];

  printf( "Please enter your first name: " );
  scanf( "%s", first );
  printf( "Please enter you last name: " );
  scanf( "%s", last );

  make_full_name( first, strlen( first ),
                  last, strlen( last ), 256 );

  if ( get_some_input() < 0 )
    return -1;

  return 0;
}
```

Listing 1: Some vulnerable C code.