**Cryptography HW1**

**Name: Spandan Patel**

**SUID: 247455437**

**Part 2**

1. (10 points) Given the algorithm (see the posting on NIST link https://www.itl.nist.gov/div898/handbook/eda/section3/eda35d.htm) determine whether the given sequence of coin tosses is random or not. HHHHHTTHHHHHTTTTTHHTTHTHTHHHTT

Text, letter

Description automatically generated

1. (40 points) From the enclosed PDF copy of the SEED Labs–Secret-Key Encryption Lab (prepared by Kevin Du - used with permission), do all the tasks (see https://seedsecuritylabs.org/Labs\_16.04/Crypto/Crypto\_Random\_Number/). Upload a zipped file containing the report.

**Task 1:** **Generate Encryption Key in a Wrong Way**

The following is the random\_number.c file for random number generation. It starts with the random number to see the variation in the output.

**Graphical user interface, text

Description automatically generated**

The time() function is used to display the time. We will learn the srand() function purpose by commenting first and then uncomment it.

Result after running the c program multiple times with commenting srand().

**Text

Description automatically generated**

We see that the time seconds are changing, but the random number generated is not changing, since no seed is set.

The following image is of the c program run after uncommenting srand().

**Text

Description automatically generated**

We see that along with the seconds of time, the pseudo random number generated is also changing with every new run, since srand() function used to seed the random number generated.

**Task 2: Guessing the Key**

We will calculate the 2 different seconds of the 2 hours timestamp of the encrypted file

**Text

Description automatically generated**

Keys.c file to generate all possible keys

**Text

Description automatically generated**

Run the file on the terminal

**Text

Description automatically generated**

Keys.txt file is generated as per the c program with all the possible keys between the timestamp period we got earlier.

**Text

Description automatically generated**

Guesskey.py file to guess the key from the possible keys for the provided message, ciphertext and IV using Crypto library

**Text

Description automatically generated**

Run the python code on terminal and get the key

**Text

Description automatically generated with medium confidence**

Time as the seed value is not a real random number, so it is not advisable to generate random numbers with time.

**Task 3: Measure the Entropy of Kernel**

To check the current entropy of the kernel using ‘watch -n .1 cat /proc/sys/kernel/random/entropy\_avail’ on the terminal

**Text

Description automatically generated**

We see that every time there is any kind of action like move mouse or click on keyboard the entropy changes.

**Task 4: Get Pseudo Random Numbers from /dev/random**

Run command on terminal to read pseudo-random numbers from / dev /. We print the output tube beautifully to the exdump tube

**Graphical user interface, text

Description automatically generated**

The entropy does not change every time. The entropy only changes if the mouse is moved to a certain extend.

**Question:** If a server uses /dev/random to generate the random session key with a client. Please describe how you can launch a Denial-Of-Service (DOS) attack on such a server.

Aside from the fact that DoS attack methods will work against the system, we can separately launch a DoS attack by requesting a very large number of new ID’s.

**Task 5: Get Random Numbers from** /dev/urandom

Run ent on the file. The ent instruction found locally cannot be used on seed12, thus we use kali again.

**Text

Description automatically generated**

Real\_random\_task5.c file to generate real random numbers. Linux provides another way to access a random pool through the / dev/urandom device, except that the device does not block/ Both dev/random and / dev/urandom use the random data in the pool to generate pseudo-random numbers.

Modify code for 256 bits key

**Graphical user interface, text, application, email

Description automatically generated**

Run c file on the terminal

**Text

Description automatically generated**

The above output is the real number because it uses /dev/urandom.