Lab 1: Introduction and Shiftciphers

50.020 Security

Hand-out: January 25
Hand-in: February 1, 9pm

1 Objectives

- To understand and modify a simple Python 3 script to read, and parse arguments
- To write a shift cipher for ascii and binary
- If you *did not take* 50.012 Networks, you should start with the instructions at A.

Note: Don't be discouraged by the length of this handout. The majority of content is intended to give you a brief summary of important command line tools in Linux. Read through the command descriptions, and try out the different commands to familiarize yourself.

2 Python warm-up and Shift Cipher

For the following scripts, please take care to validate the arguments provided carefully, which is good practice to prevent bugs and security vulnerabilities. In particular, ensure the following in your handin:

- For each argument, check if it has the correct type (e.g. string, integer)
- If required, ensure that the input comes from a specific set of inputs, e.g. is a number >=0 and <256
- It is good practice to provide a default value for parameters, in case they are not provided by the user

2.1 Part I: Shift Cipher for printable input

- Write a Python script that can be run from the command prompt or shell as the following:

```
$ ./shiftcipher.py -i [input filename] -o [output filename] -k [key]
-m [mode]
```

- For this part of the exercise we will use the `string` module and in particular we will focus on the `string.printable` string as our valid alphabet.

- The key must be between 1 and `len(string.printable) - 1`, otherwise an error message is produced. The key specifies how many characters the original letter should be shifted. The output and input space for characters should be the set specified by `string.printable` in Python. For encryption, the key is added to the original letter. For decryption, the key is subtracted from the original letter. For example:

```
E('A', k=10) = 65 + 10 = 75 = 'K'
D('K', k=10) = 65 = 'A'
```

- The mode must either be 'd' (Decryption), 'e' (Encryption), or their capital letter, otherwise an error message is produced
- Use the text file given 'sherlock.txt' and encrypt it with some key, and then decrypt it. Check whether the text remain the same after the process of encryption and decryption

2.2 Part II: Shift Cipher for binary input

- Based on ex1.py, write a Python script that can be run from the command prompt or shell as the following:

```
$ ./shiftcipher.py -i [input filename] -o [output filename] -k [key]
-m [mode]
```

- The key must be between 0 and 255 (1 Byte integer), otherwise an error message is produced. The input should be interpreted as sequence of Bytes. The key specifies how many characters each original Byte should be shifted. For encryption, the key is added to the original Byte. For decryption, the key is subtracted from the original Byte. For example we can see the result of a binary shift using hexstring representation (`0x0b = 10 = 0b001010`:

```
p = 0x01, k = 0x0b
E(p, k) = c = 0x01 + 0x0b = 0x0c
D(c, k) = 0x01
```

- The mode must either be 'd' (Decryption), 'e' (Encryption), or their capital letter, otherwise an error message is produced
- Use the text file given 'sherlock.txt' and encrypt it with some key, and then decrypt it. Check whether the text remain the same after the process of encryption and decryption.
- Consider the full extended-ASCII range of 256 values in your encryption and decryption implementation
- Hint: `bytearray` from Python3's built-in function is your friend.

2.3 Part III: Break Shift Cipher of flag

- Use your shift cipher script from the last part to find the plaintext corresponding to the provided `flag` file.
- Hint: you can use the `file` command line tool to determine the type of a binary file. Your expected plaintext is a PNG file.

3 What to Hand in

3.1 eDimension submission:

The submission for this exercise consists of the script(s) you wrote

- Please make sure to put your username into the header of the submitted file.
- You are allowed to collaborate with one other student. Please hand in your scripts separately, with both names in the header
- Please provide the Python 3 code for the following two programs and one binary file
  - The printable character shiftcipher script
  - The binary shiftcipher script
  - The decrypted flag image

A First time login

After you start your machine, you will be presented with a login with name `student`. Use the following password to login: `password`.

B Create new user account in Xubuntu

Use the `start menu/settings/users and accounts` to create a new user account with Administrative rights. The initial user account is common to all and, therefore, has the same common password. You are strongly encouraged to create a new user account with your own password. Log out, and back in as your new user.

C How to open shell in a terminal

Linux offers to the user an interactive, text-based interface with the Operating System, from which she can complete (and chain together) many tasks (from simple to complex ones). The program that interfaces the user with the OS is called the *shell* (`bash` on our machine) and the window where you read and type command is called the (virtual) *terminal*.

We will start by opening a terminal (similar to `cmd` in Windows). For this class, we will use the `xfce4-terminal`. You can open it by pressing Alt+F2 and typing `xfce4-terminal`. Or you can use the keyboard shortcut by pressing.

- Ctrl+Alt+T

Opening the terminal will automatically start a *shell*, and interactive environment. By default, we use `bash`, which supports many features, such as command history navigation (using up/down) keys, and auto-completion of commands, file names, and folder names via the `Tab` key.

D Common commands

D.1 Find the usage of commands

If you don't know how to use the shell commands, you can use command `man` (which stands for MANual)to show the usage of them.

```
$ man <command name>
```

Man pages have a well-defined structure and formatting. Each page is divided into sections, each section has an uppercase name. The NAME section is useful to get a short description about the command. The SYNOPSIS section contains a compact representation of the commands arguments, Eg: options, positional parameters and default values. The DESCRIPTION section describes each option in detail. Some man pages even contain EXAMPLES and BUGS section. If you want to quit while reading the manual of the command, press q

D.2 Change the directory

The `cd` (change directory) command allows you to change your current directory to any accessible directory on the system.

```
$ cd <directory>
```

Using the command `cd` without applying any parameter will bring you back to your home directory.

D.3 Listing the contents of a directory

The `ls` (list) command is used to view the contents of the current directory.

```
$ ls [option] <filename>
```

To know more about the field `[option]`, use the command `man ls` . It is frequently used with the options `-al` (all, long) to list hidden files as well (their name start with `.`), and provide more information about each file on a dedicated line. Type `cd /`, `ls`, and `ls -al` in the command line, what is the result?

D.4 locate, less

Command `locate` (find files by name) can be used to search files in Xubuntu. For example, if you want to find all text files, you can use:

```
$ locate *.txt
```

Command `less` (pager program) can be used to display the content of file. Example:

```
$ less myfile.txt
```

Another way to display file is to use command `cat`, example:

```
$ cat myfile.txt
```

D.5 grep

Command `grep` (print lines matching a pattern) can be used to search for strings or patterns in text files. For example, if you want to find all occurrences of "foo" in file A.txt

```
$ grep foo A.txt
```

Command `grep` can be also be used to filter output of other commands using *piping* symbol "|". The following will locate all python scripts on the current machine, but display only the ones in the home directory. Depending on the content of your home directory, the output might be empty.

```
$ locate *.py | grep home
```

D.6 mkdir

`mkdir` (make directories) can be used to create directories. To create `dir1` in the current location, type:

```
$ mkdir dir1
```

E File management Ubuntu commands

E.1 File access control in Linux

Linux uses per-file access control. There are three different types of access to a file: reading (r), writing (w), and execution (x). These three types of access can be set for the file owner, a group of users, and everyone else. As a result, the access rights for each of those three classes of users can be described in a short string, e.g "r-x" for read and execution rights. The rights of all three types of users together yield a longer string, e.g. "rw-r–r–" for a file which can be read and written by the owner, but only read by everyone else.

Try using `ls -al` and `cd` to look around on your system, and see who has what kind of access control to which file. The content of `/etc/` could be especially interesting.

E.2 Move and Copy files and directories

To copy files, you can use the `cp` command, to move (rename) files and directories you can use the `mv` command. The command line

```
$ mv file1 file2
```

allows one to move (rename) `file1` to `file2`. Command `cp` on the other hand copies one file to another.

```
$ cp file1 file2
```

Note that if `file2` does not exist, it will be created; however if it does exist, it will be overwritten. There is NO undo command in Linux. For copying and renaming directories, you can use the `cp` and `mv` commands just like you use them with files. If you want to copy the content of a directory into a new one you need to use `-r` (recursive) because `cp` will recursively copy all the files in the source directory into the newly created one

```
$ cp -r dir1 dir2
```

E.3 Deleting files and directories

The `rm` (remove file and directories) command is used for removing files and directories. To remove a file:

```
$ rm file1
```

You can use `rm` with the `-r` option to delete all the files in the directory first and then the directory itself. NOTE that the following command will remove all the subtree rooted at `dir1` Eg: all subdirectories and their content will be lost. A safe place in your file system to practice file creation and deletion commands is the `/tmp` (temporary) folder

```
$ rm -r dir1
```

E.4 chmod

We introduced Linux file access control in Section E.1. Now we discuss how to change to rights to files. Please note that you have to be either root (Eg: using `sudo`, see below) or the owner of a file to change the file's rights.

`chmod` (change file mode bits) can be used to change the file's permissions. Each file has three associated (group of) user(s): the owner of the file (u), users in the file's group (g), and the rest of the users (o). Typically the file owner is a (single) user who created the file, the file's group is used to group other users who may want to access the file, and the other users are the remaining users. From the terminal, use `ls -l file1` to see the current permission bits state of `file1`, you should get an output such that: `-rw-r-r-`, ignore the first character and focus on the following three triplets of characters. In our example the file's owner has read and write permissions, file's group has read permission, and others user have read permission. You can read the name of the file's owner and file's group from the output of the same command, e.g.: `rob` and `users` There are two ways to set the file's permission bits: symbolic, and octal, in the following examples we will focus on the symbolic one, remember: `r` stands for read permission, `w` stands for write permission, and `x` stands for execute permission.

```
chmod <people><+/-><permissions>
```

Example: `chmod o-w file1` (deny others from writing to `file1`
Example: `chmod u+rwx file1` (give the owner read, write and execute permissions over `file1`)
Example: `chmod +rwx file1` (give everyone read, write and execute permissions over `file1`, NOTE: use with care!)
Example: `chmod +x file1` (allow anyone to execute `file1`)

E.5 User, superuser, sudo and su

By default your system user (type `whoami` to print it) does not have administrator rights (e.g. to install software system wide). In Linux, the administrator account is called "superuser" or "root". Your system is set up to allow you to perform commands as root by using `sudo` in front of the command.