# CSC/ECE 574 - Project 1*

Assigned August 29, 2018; Due 11:59pm on September 18, 2018

Prof. William Enck

## 1 Secure Netcat {100 points}

Netcat (the Unix `nc` command) is a powerful command line utility that is commonly used by systems administrators and adversaries alike. The netcat manpage has several examples, and you are encouraged to do a Web search for "netcat fun" to discover more.

While netcat has many command line options and lots of interesting functionality, this assignment focuses on the functionality described by its name: "net" and "cat." The Unix `cat` command takes a file and prints it to STDOUT (or wherever STDOUT is redirected). For our purposes, think of netcat performing `cat` across a network. A client instance will receive input from STDIN and send it to a server instance. The server instance will receive network bytes and send it to STDOUT. As you might imagine, netcat provides no protection for the information as it is transmitted across the network.

The goal of this assignment is to provide *confidentiality* and *integrity* to this simple version of netcat. The standard way to add confidentiality and integrity to messages is through the use of cryptography. Encryption functions (e.g., AES) provide confidentiality. MAC functions (HMAC) provide integrity. As discussed in course lectures, there are different ways to combined encryption and MAC functions to provide both confidentiality and integrity: encrypt-and-MAC (E&M), encrypt-then-MAC (EtM), and MAC-then-encrypt (MtE). Out of these methods, E&M is insecure (sending the MAC of plaintext in the clear leaks information). Both EtM and MtE are used in a variety of cryptographic protocols, but only EtM reaches the highest definition of security. In addition to these ways of combining encryption and integrity, there are cipher modes that provide Authenticated Encryption (AE). These modes security combine encryption and MAC into the construction itself. The most notable of these modes is Galios Couter Mode (GCM). We will use AES-GCM for this assignment.

**Program description.** Create the `snc` (secure netcat) command line program that has the following command line options:

```
snc [-l] [--key KEY] [destination] [port]
```

*Last revised on September 12, 2018.

The instance running on the client will read from STDIN and send AES-GCM protected data to the instance running on the server. The instance running on the server will read from the network, decrypt and authenticate the data, and write it to STDOUT. Both instances should terminate an EOF character is encountered, or if a keyboard interrupt (i.e., control-c) occurs. If the data fails to authenticate (e.g., the data was manipulated, or the keys were different), the server should terminate, reporting an error to STDERR.

The following is a sample execution:

```
[client]$ ./snc --key CSC574ISAWESOME server.add.ress 9999 < some-file.txt

[server]$ ./snc --key CSC574ISAWESOME -l 9999 > some-file.txt
```

To be equivalent to the `nc` command, information entered in STDIN on the server should make its way to STDOUT on the client. To do this, you will need to use the *select* call to block and wait for input *either* on standard input or the network socket. As an alternative to using *select*, you may use multiple threads, although this is far less efficient.

An sample execution with bi-directional flow is as follows:

Please make sure that your program conforms to the following:

```
[client]$ ./snc --key CSC574ISAWESOME server.add.ress 9999 < file1-in.txt > file2-out.txt

[server]$ ./snc --key CSC574ISAWESOME -l 9999 > file1-out.txt < file2-in.txt
```

**Additional requirements and hints.**

- You must write your program in Python. Please see the teaching staff if you would like to use another programming language.
- You must use the PyCryptodome crypto library. See the installation page for installation instructions (take note that PyCryptodome replaces the previous PyCrypto package, which may be installed on your machine, but does not support GCM mode). The examples are also helpful. The API pages describe how to use GCM mode.
- You must use AES-256 in GCM mode. To compute the key from the command line argument, use PBKDF2 (Password-Based Key Derivation Function). PyCryptodome provides an API for this.
- The random IV (nonce) must be securely created and transmitted between the program instances.
- You may not collaborate on this homework. This project should be done individually. You may search the Internet for help, but you may not copy (either via copy-and-paste or manual typing) code from another source. You **may** use code from the textbook, or from the instructor or TAs.

- Your program should not take in any additional command-line options other than the options described above.
- A single snc program can act as a server or client depending on supplied command-line arguments.
- Both server and client will close the network connection and terminate when:
  - User has entered CTRL-C in a console window of either server or client.
  - The server and client have finished sending and recieving their piped files.
- The Python Socket Programming HOWTO can help you bootstrap your socket code.
- You might try PyLint to keep your code neat.

## Grading

Project 1 is worth 100 points. A non-comprehensive list of deductions for the programming portion of this assignment is provided in Table 1.

We will award partial credit when possible and appropriate. To maximize opportunities for partial credit, please rigorously comment your code. If we cannot understand what you intended, we cannot award partial credit.

| **Description** | **Deduction** |
|---|---|
| Interpreter errors | 80 |
| Runs, but data is neither successfully transmitted nor received | 60 |
| Data is not encrypted | 50 |
| Data is encrypted, but not successfully decrypted | 40 |
| Lack of integrity verification | 30 |
| Incorrect integrity verification | 20 |
| Only works from client to server (and not server to client) or visa-versa | 20 |
| Fails to send in both direction simultaneously | 10 |
| General instability (e.g., occasional segfaults) | 10 |
| Run-time error (e.g., crash) on large input | 10 |
| Non-conformant command-line options (hinders automated testing) | 10 |
| Does not follow prescribed command line args (hinders automated testing) | 10 |

Table 1: Grading rubric. Note that this grading rubric is not intended to be comprehensive.

## Submission Instructions

Submit your solution using WolfWare (one `.py`). To upload your assignment, navigate to the CSC574 course. Use the "Project 1" assignment under "Mini-Projects."

Please post questions (especially requests for clarification) about this homework to Piazza.