Certified Degree

in association with National Cyber Security Centre

Lancaster University

2021-2022 ASSESSMENTS

Postgraduate

Masters Degree in Cyber Security –Individual Programming Assessment

Recommended Completion Time [24 Hours]

Assessment Weighting [60%]

SCC.439 NETWORK AND SYSTEMS SECURITY

Academic Honesty and Integrity

Students at Lancaster University are part of an academic community that values trust, fairness and respect and actively encourages students to act with honesty and integrity. It is a University policy that students take responsibility for their work and comply with the University's standards and requirements- found in the Manual of Academic Regulations and Practice. By submitting their answers students will be confirming that the work submitted is completely their own. Academic misconduct regulations are in place for all forms of assessment and students may familiarise themselves with this via the university website:
https://www.lancaster.ac.uk/academic-standards-and-quality/regulations-policies-and-committees/manual-of-academic-regulations-and-procedures/

Plagiarism

Plagiarism involves the unacknowledged use of someone else's work and passing it off as if it were one's own. This covers every form of submitted work, from written essays, video vignettes, and coding exercises. Deliberate plagiarism with the intent to deceive and gain academic benefit is unacceptable. This is a conscious, pre-meditated form of cheating and is regarded as a serious breach of the core values of the University. More information may be found via the plagiarism framework website. All coursework is to be submitted electronically and will be run through our plagiarism detection mechanisms. Please ensure you are familiar with the University's Plagiarism rules and if you are in any doubt please contact your module tutor.
https://www.lancaster.ac.uk/academic-standards-and-quality/regulations-policies-and-committees/principles-policies-and-guidelines/plagiarism-framework/

General Guidance:

Programming should be completed using the Python programming language.

All code should provide suitable comments to explain the operation and function.

You should submit one zip file with python files. You should use the pycryptodomex library for cryptographic functions: https://pycryptodome.readthedocs.io/en/latest/

Your programme must work on the SCC.439 mylab environment as this will be used for marking.

Marking Scheme:

- *Commenting and code structure 10%*
- *Basic but functional network communications 15%*
- *Working message format communication 5%*
- *Basic functional application (user Interface features etc) 15%*
- *Working Mutual Authentication 15%*
- *Working Message integrity and Confidentiality 20%*
- *Suitable security and message validity checks 5%*
- *Secure Message transaction logging 10%*
- *Weakness documentation 5%*

Assessment Details:

You are required to build a secure, asynchronous, peer to peer chat application. Network communications should be conducted via TCP. The flow of the programme should be:

1. Start the programme
2. Ask the user for the following information:
    a. A port on the machine to act for incoming connections
    b. A file which contains the directory of individuals to contact
3. Open Server TCP socket locally on port specified by the user
4. Load up the directory (specification below)
    a. This is a JSON serialised dictionary format.
5. Wait for either an incoming network message or a message to be sent from user
6. If an incoming network message
    a. Connect to the destination using the decided protocol, which must include
        i. Mutual authentication of the outgoing connection
        ii. Establishment of a secure channel using SHA256 HMAC and AES256 CBC Cryptography
        iii. The generation of key material as derived in class

        Note the ordering of steps will be decided by the class during the seminar sessions
    b. Receive Message – Message format below.
    c. Check the control information for consistency
    d. Display received message to the user
    e. Terminate Connection
7. If message to be sent
    a. Wait for the message to be completed – Your choice of end indication (special character, return key, character sequence)
    b. Request destination from user. This should be selected from the directory, but you should provide a manual option to specify IP address, port, and password
    c. Open socket to destination
    d. Connect to the destination using the decided protocol, which must include
        i. Mutual authentication outgoing connection
        ii. Establishment of a secure channel using SHA256 HMAC and AES256 CBC Cryptography
        iii. The generation of key material as derived in class

        Note the order of this will be decided by the class during the seminar sessions
    e. Send Message – Message format below.
    f. Display transmission success or failure message to user
    g. Terminate Connection
8. Go to step 3.

Message Format:

You should use a Python Dictionary message format that can be serialised as follows:

Message = {'header':{'crc': *val*, 'timestamp': *UTC_val*}, 'message': *base64_encoded_text_message,*
'security':{'hmac': {'hmac_type': *val*, 'hmac_val': *val*}, 'enc_type': *val*}}

Directory Format:

You should use a Python List of Dictionaries for the directory format, which can be serialised:

Directory = [{'username': *val*, 'password': *val*, 'port': *val*, 'ip':' *val* '}, {'username': *val*, 'password': *val*, 'port': *val*, 'ip':' *val* '}, …]

Additional Requirements/Information:

- **Before starting to write your application, plan the way you will develop features**
- Use the user name in the CHAP messages to select the incoming information for the key.
- The password should not be used directly for CHAP, HMAC and crypto keys. Utilising a key derivation function will be rewarded with higher marks
- All messages sent and received should be securely logged to protect their confidentiality and integrity
- You may wish to delay the implementation of the user directory functionality to aid simplicity in building the system. Therefore, you may assume only two parties will use this application if this aids and speeds development. However, a user directory is needed for higher marks.
- As a stretch goal, consider implementing confidentiality and integrity in the user directory file, with appropriate crypto material provided on the command line at programme initialisation
- Any messages received out of a time window of 25 seconds should be ignored
- In the Message header use a 32bit Cyclic Redundancy Check calculation from the zlib library
- It is highly recommended that you use an object-oriented approach -- this is not necessary, but higher marks will be awarded for an OO approach.
- You must only use primitive cryptographic libraries and not utilise protocols such as SSL/TLS for example to provide the security functions.
- It is highly recommended that you use some form of FSM for the management of the communications protocol, this is not necessary, but higher marks will be awarded for an FSM approach.
- Please use comments extensively to explain what your code is doing and how it is doing it.
- Please include a short .txt file that explains any security weakness you think are present in the approach – see the marking scheme for the points this is worth.
- Please upload a .zip file containing the working application, any support files, and the .txt file from above. DO NOT INCLUDE LIBRARY FILES. Only include the files you have written.
- You application MUST work on the SCC.439 development virtual machine WITHOUT any additional libraries installed.

Using code supplied to you by someone else, or supplying code to someone else is plagiarism.