Priority 1 (Core Functionality):

The following requirements entail the core functionality of the system and are top priority as the project doesn't exist if they are incomplete. This software project does not have any safety critical requirements which would take top priority if they existed.

R1)The tool shall generate network traffic streams in the form of a pcap file containing up to 10,000 packets with the following customisable attributes: Protocol stack implemented by packets, Data payload of packets (size and content), Size of stream (in packets, bytes or seconds), Packet timing (as packet rate, inter-packet gap or byte rate)

R1 completeness criteria:

- There is an implementation of a python function which generates pcaps
- All the configuration options detailed partially in R1 and completely in the project specification can be chosen by the user through function arguments
- Every possible configuration option is fulfilled as expected in the pcap file output

R1 testing limitations and considerations: There are a lot of configuration options, many of which have infinite possible values. Great care will have to be taken to select a representative set test values large enough to create confidence in correctness but small enough that the tests can be carried out efficiently.

R1 test strategy: Test for correct construction of input objects first then test for correct pcap generation. Use a mixture of combinatorial techniques for test-case input selection exhaustive for limited value data-items and do catalogue selection for an input value set for data-items with infinite/large possible values. The output files will have to be checked manually to verify that they are in accordance with the configuration parameters.

R1 scaffolding and instrumentation:

- A python unittest test environment
- Either a database with test case input configurations or a script which generates input configuration for each case
- A complete pcap generation unit
- A catalogue for each elementary input item
- Python logging library function

R1 testing timeline: After development of generation unit (the first development task) and to be run as regression tests after each update to the unit

R2)The tool shall read, save and summarise pcap file contents

R2 completeness criteria:

- Given a pcap file, store the bytes contained in each packet,the timestamp associated with each packet and packet metadata present in the pcap such as protocol header types
- Use this data to calculate all the summary statistics listed in the project specification

R2 testing limitations and considerations:

- Similarly to R1 an infinite number of possible inputs, so I must select a sufficiently large and representative set of pcap files for test inputs
- Different OSes have different file management systems so successful reading of a file on one doesn't guarantee the same on others.

R2 test strategy: Since the functional units described by R1 and R2 are meant to be used together in the same context (as input generation and output analysis for testing another company product). If R1 test cases are representative, the outputs of these test cases should also be representative of the inputs to the pcap analysis unit. Furthermore, it is difficult to acquire very specific pcap files from public sources otherwise the development of the generation unit would be futile . With this in mind, I believe the most appropriate testing strategy for R2 is to first test R1 and gain sufficient confidence in its correctness, then we can use the outputs from the R1 test cases as inputs to the R2 test cases and check that the summary statistics match the configuration parameters. This leaves us with two very tightly coupled unit-tests so there should be some additional R2 test cases with pcap inputs sourced on the internet to loosen the correctness dependence slightly

R2 scaffolding and instrumentation:

- A python unittest test environment
- The output pcaps from a successful pcap generation unit test labelled with pcap configuration values
- Some externally sourced pcaps with manually calculated summary statistics
- A complete pcap analysis unit
- Python logging library function

R2 testing timeline: The test code should be run as soon as the initial development of the pcap analysis unit is complete and the pcap generation unit tests have passed. Manual checking of test results will be time intensive so to be done in tandem with development of CLI interface, ensure to leave enough time for two or three test-debug cycles.

Priority 2 (Crucial for Project Success):

The following requirements are not core functions of the system but the system will not be usable for its intended purpose if they are not met.

R3)The generated pcaps shall be compatible with the the SNE traffic replay feature

R3 completeness criteria: Any pcap file generated by the system can be uploaded to a SNE unit and replayed with the SNE traffic replay feature.

R3 testing limitations and considerations:

- The SNE software is frequently updated so current success doesn't guarantee future success
- Need access to at least one, ideally multiple SNE units for testing which are a shared company resource

R3 test strategy: generate many pcaps varied in size, packet protocols and packet validity. Upload these files to one or more SNEs and replay them through a map, ensure there are no errors.

R3 scaffolding and instrumentation:

- Access to at least one SNE unit
- A completed pcap generation unit that has passed unit tests

R3 testing timeline: To be tested after both unit tests have passed

R4)The analysis component shall be compatible with pcaps generated by the SNE traffic capture

R4 completeness criteria: Any pcap file generated by the traffic capture feature of the SNE can be successfully and correctly used as input to the pcap analysis unit of the system.

R4 testing limitations and considerations:

- The SNE software is frequently updated so current success doesn't guarantee future success
- Need access to at least one, ideally multiple SNE units for testing which are a shared company resource

R4 test strategy: capture a range of different pcap files with the SNE traffic capture using either the pcap generator and traffic replay or live traffic from an STC and feed them into the analysis tool ensuring the saved contents match that of the pcap input.

R4 scaffolding and instrumentation:

- Access to at least one SNE unit
- A completed pcap generation unit that has passed unit tests or access to an STC
- A completed pcap analysis unit that has passed unit tests

R4 testing timeline: To be tested after both unit tests have passed

Priority 3 (Desirable qualities)

Without the following requirements, a finished product is possible but it would not be ideal in terms of performance or usability.

R5)Both traffic generation and analysis should execute in two minutes or less for up to 10,000 packets.

R5 completeness criteria: execution of both function in under two minutes for pcaps containing 10,000 of the estimated largest packets that the system will deal with.

R5 testing limitations and considerations: Performance time will vary depending on the computer it is run on.

R5 test strategy: TIme the execution of both functions for 10000 packet pcaps on multiple machines of varying performance.

R5 scaffolding and instrumentation:

- A pcap generation unit that has passed unit tests
- A pcap analysis unit that has passed unit tests
- At least two computers of varying performance capabilities

R5 testing timeline:

- After unit tests have passed and as a regression test after any updates

R6) There shall be a command-line interface to execute pcap generation

R6 completeness criteria:

- There is a CLI tool which calls the pcap generation function
- This tool has flag options to configure the pcap in every way the python function can
- The CLI flag inputs correspond to the correct argument inputs to the function

R6 testing limitations and considerations: Python CLI tool implementation varies slightly across OSes

R6 test strategy: Design test case inputs to the CLI tool to obtain 100% MC/DC coverage of the CLI interface code. Label each case with the expected corresponding arguments to the function call and assert that the function call that is made matches.

R6 scaffolding and instrumentation:

- Completed CLI interface code
- Completed pcap generation and analysis units
- Some record of each condition in the code
- A dataset of test inputs that will execute every option for each condition in the code or a script to generate the input for each case
- A python unittest test environment
- Access to machine/s that run Linux and Windows OSes

R6 testing timeline:

- At the end of the development process after tests for R1 and R2 after or in parallel with tests for R3, R4 and R5

R7) The CLI interface should be intuitively usable for other company engineers (easier than calling the python API function)

R7 completeness criteria: a sample of Calnex software and test engineers are able to learn how to use all the features of the CLI tool in an hour or less by reading the documentation.

R7 limitations and considerations: Every engineer will have a different knowledge base and capacity to learn things

R7 test strategy: once all unit and integration tests are passed, deploy a user acceptance test to a representative sample of end users and ask them to verify this quality.

R7 scaffolding and instrumentation:

- A completed system that has passed all other tests
- Time in the schedule company engineers

R7 testing timeline:

- After all other development and testing activities