CROSS OVER ASSIGNMENT

Libraries Used:

1. Paramiko:

Paramiko is a Python (2.7, 3.4+) implementation of the SSHv2 protocol, providing both client and server functionality. While it leverages a Python C extension for low level cryptography , Paramiko itself is a pure Python interface around SSH networking concepts.

2. Sqlite3:

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

3. Smtplib:

The most common use of **SMTP** is to connect to a mail server and send a message. The mail server host name and port can be passed to the constructor, or you can use connect explicitly. Once connected, just call sendmail() with the envelope parameters and body of the message.

4. Crypto.Cipher.AES:

AES (Advanced Standard Encryption) is a symmetric block cipher standardized by NIST . It has a fixed data block size of 16 bytes. Its keys can be 128, 192, or 256 bits long. AES is very fast and secure, and it is the de facto standard for symmetric encryption.

5. PsUtil:

Psutil (process and system utilities) is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors) in Python. It is useful mainly for system monitoring, profiling and limiting process resources and management of running processes.

6. PyWin32:

win32evtlog.pyd is a win32evtlog.pyd belonging to PyWin32 Non-system processes like win32evtlog.pyd originate from software you installed on your system. Since most applications store data in your system's registry, it is likely that over time your registry suffers fragmentation and accumulates invalid entries which can affect your PC's performance.

All of these libraries are predefined and comes with Python2.7 package itself.

Implementation & Working:

This project works on the principal of client-server communication.

1. First the database and the tables are to be created. This could be done by executing the CreateDatabase.py file in the Source Folder.
2. The Server first connects to all of its clients i.e. connects to all the systems within the local network through SSH.
3. All the SSH credentials of the systems are configured in the clients.xml file.
4. The ServerScript.py file executed on the Server.
5. This sever script reads the clients.xml file and connects to all the local nodes through ssh using the credentials predefined in the file.
6. Post connection the server copy's the ClientScript.py file into the Temp Directory of the local system.

   This temp directory varies based on Operating Systems

   For Linux: **/tmp**/

   For Windows: **C:\\Windows\\Temp**

7. This Client Script is executed by all the local systems in their respective temp directories and the output is redirected to the Server.
8. Before the client sending the output data to the server it encrypts the data with AES Encryption.
9. The server receives the encrypted data from all the clients one by one and it decrypts the data and obtains the original system statistical data.

10. This Data has various statistical data like

    System Details

    System Users

    CPU usage

    Memory Usage

    Swap Memory Usage

    Disk Partitions

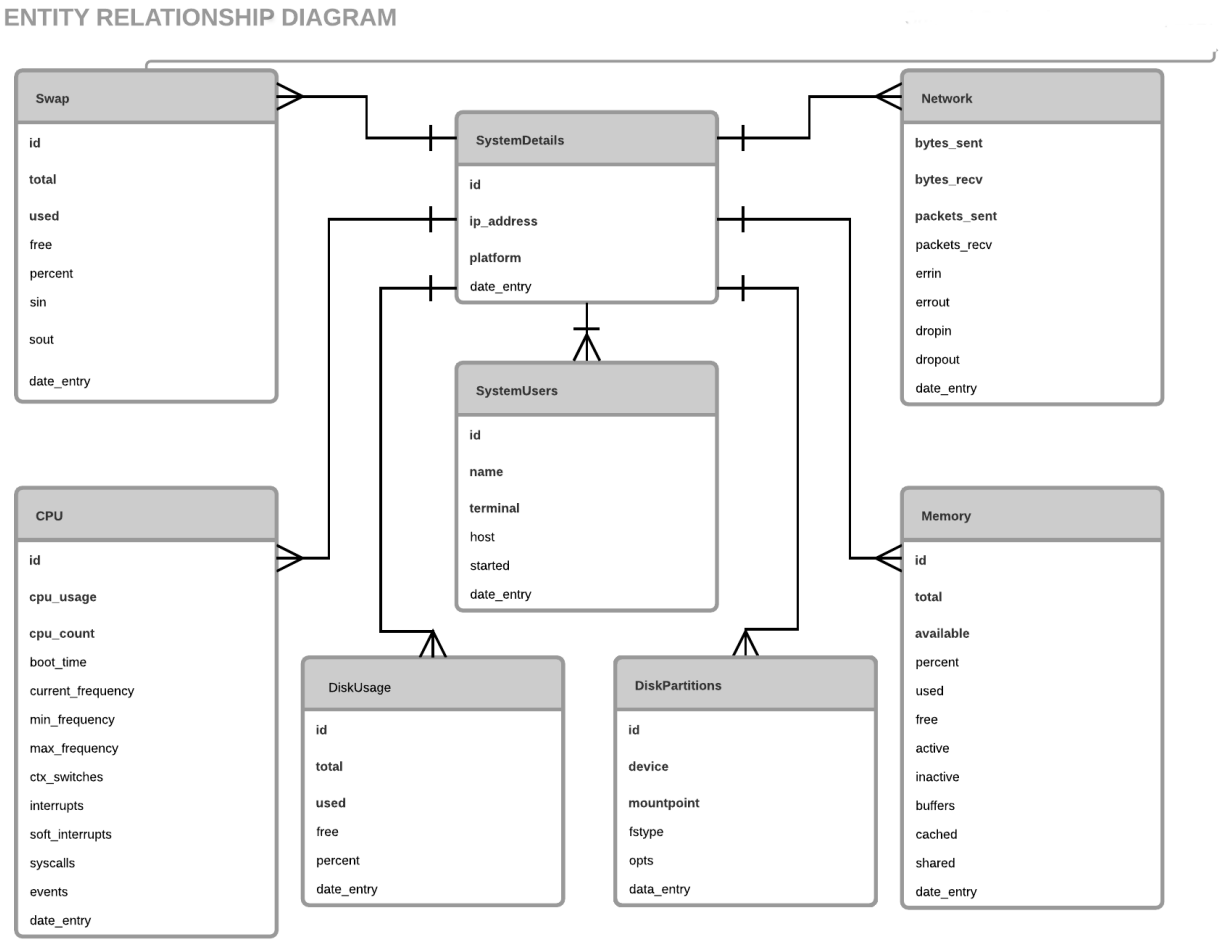
    Disk Usage

    Network Usage

11. All of this data is in json format and is parsed by the server and then Inserted into the their respective tables in the database.

Database Architecture:

We have used the SQlite3 Database.



Unit Tests :

For Unit tests execute the file named UnitTest.py in the terminal.

Steps:

1. Open the Terminal or CMD
2. Change the current directory to the project directory.
3. Execute command : $ python UnitTests.py
4. All unit tests are executed.