Building Client Server Based Application in Genode Operating System Framework

Bulusu Karteek Pradyumna*
Department of Computer Science and Engineering
Meenakshi Sundararajan Engineering College
Chennai, India- 600024
bkartik92@gmail.com

Harish K**
Department of Computer Science and Engineering
Meenakshi Sundararajan Engineering College
Chennai, India- 600024
harishkrish93@gmail.com

S.S.Naresh Kumar***
Department of Computer Science and Engineering
Meenakshi Sundararajan Engineering College
Chennai, India- 600024
nareshkumar7593@gmail.com

M.K.Sandhya****
Senior Assistant Professor
Department of Computer Science and Engineering
Meenakshi Sundararajan Engineering College
Chennai, India- 600024
mksans@gmail.com

Abstract—This paper briefs about new free and open source microkernel based Operating system framework, Genode. It shows the building of a SMTP (Simple Mail Transfer Transfer protocol) based Mail client application and further porting the same inside Genode Operating system framework. The application is developed using Qt user interface design application framework and further built and ported inside Genode using make and run files. By this application, user can add his email account, view, send and manage the account. Since the operating system is still in developing stage, this application would help in its improvement.

Keywords- Genode, SMTP, Qt UI design framework, microkernel, GCC.

I. INTRODUCTION

In this present world, security has become a prime issue concerning every common man. Each person, institute or company demands security at each level. At this time of demand, came up Genode operating system framework. Genode OS architecture is well known for its improvised security issues. It uses microkernel approach in its operating system architecture. Technically, kernel is computer program that manages input/output requests from software and translates them into data processing instructions for the central processing unit and other electronic components of a computer. Microkernel is approach in which minimum software support can be provided for the operating system to build the files. There are three types of microkernel approaches: address space management, thread management, inter-process management. Genode utilizes address space management. By this method, operating system allocates individual address to each process and thus no two processes are dependent on each other and parallel processing can be done. By this technique, security issues are improvised since upon fail of a process, other processes active are not affected and thus avoids data loss.

The heart of the Genode OS architecture is an organizational structure that loosely resembles the structure of a hierarchy organized structure. Each root will have its sub-root. Any damage or process delay to the sub-root will manage only that particular root and not whole processing unit hence avoiding the loss of data. Microkernel- based systems use these techniques not only for user applications but also for, file systems, device drivers and other typical kernel-level services. Therefore, the effect of a bug-prone component is locally restricted. [1]The microkernel withdraws all non-needed privileges from each component and thereby shrinks the overall complexity of code running in privileged mode by an order of magnitude compared to a monolithic kernel.

Figure 1 shows structured layered architecture of Genode operating system framework. Genode organizes processes as a tree. The blue arrows symbolize that child processes are created out of the resources of their respective parents. When creating a child process, a parent should provide the virtual environment in which the new process can get executed. [2] The child, in turn can further create children from root, and create an arbitrarily structured subsystem. Each parent maintains full control over the subsystems it created and defines relationship between root and subsystems, for example by selectively permitting communication between them or by assigning physical resources. The parent- child interface is the same as at each hierarchy level, which makes this structured approach recursively applicable. Using the recursive property as a tool clears the way for separating policy and mechanism in very flexible ways and facilitates the strict separation of duties.

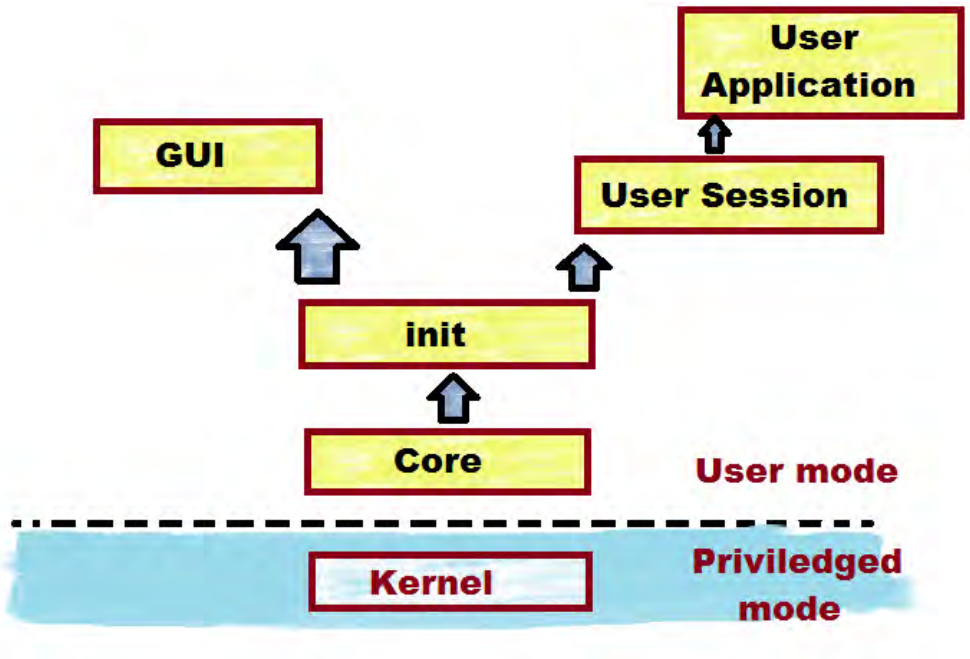

Figure 1: Structured Genode OS framework Architecture

For Genode to build or compile a file, it needs a compiler and set of patch files. A regular GCC CC++ compiler can be used for this task but Genode has provided two specific tool-chains for themselves. A tool-chain is a combination of compiler and patch files used to build a file. Genode tool-chains consist of GCC compiler and compiles make and run files and processes the application. There are two tool-chains available: Genode tool-chain-11.11 and Genode tool-chain-12.11.

II. MAIL CLIENT APPLICATION IN QT IDE

Development done inside Genode is to build one secure mail client application that would send and receive mails after the user adds his/her email account in this application. Qt is a cross-platform application framework that is widely used for developing application software with a graphical user interface and also used for developing non-GUI programs such as command-line tools and consoles for servers. It is very simple and user friendly application user interface designing framework. The mail client application is designed using Qt creator and compiled and built using Qt debugger. The reason for using Qt cross platform IDE is since firstly it uses C++ programming and secondly, Qt user interface designing IDE is been ported inside Genode by the developer. So, the application being developed in Qt IDE will work efficiently inside Genode.

The mail client application developed is based on SMTP connection. SMTP is Simple Mail Transfer Protocol. It is a standard for electronic mail transmission. All webmail applications and websites use SMTP connection to transfer mail and receive mail.

The application's user interface was designed in Qt creator where the email address, subject body and default buttons are provided and saved as .ui file. One cpp programming file is written to execute the ui file. Another cpp file is written to implement SMTP connection in the mail application.

Figure 2 shows the programming window in the qt creator where the C program is used to develop the user interface of the application.

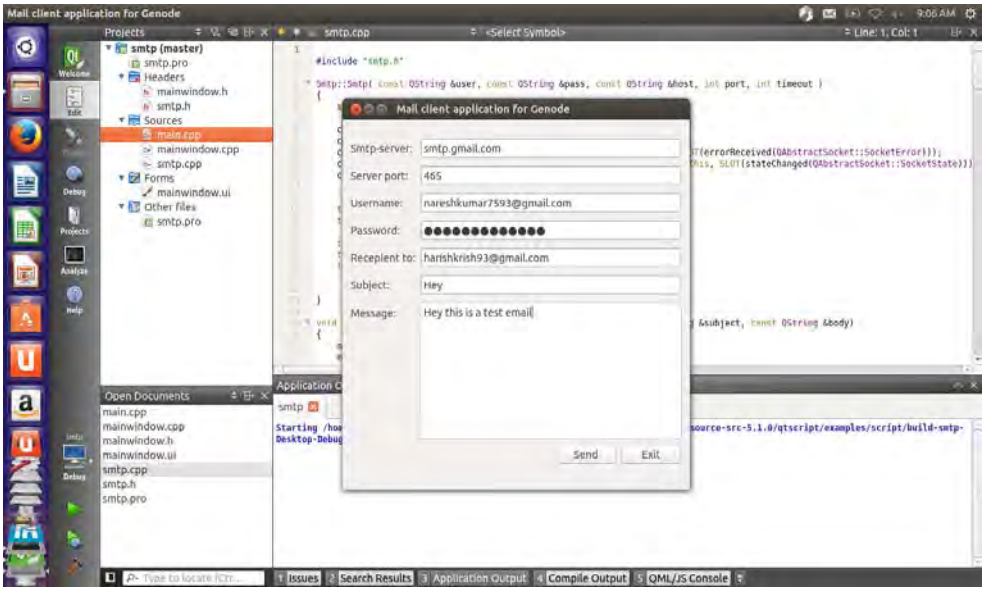

Figure 2: Mail Client Application built on Qt creator

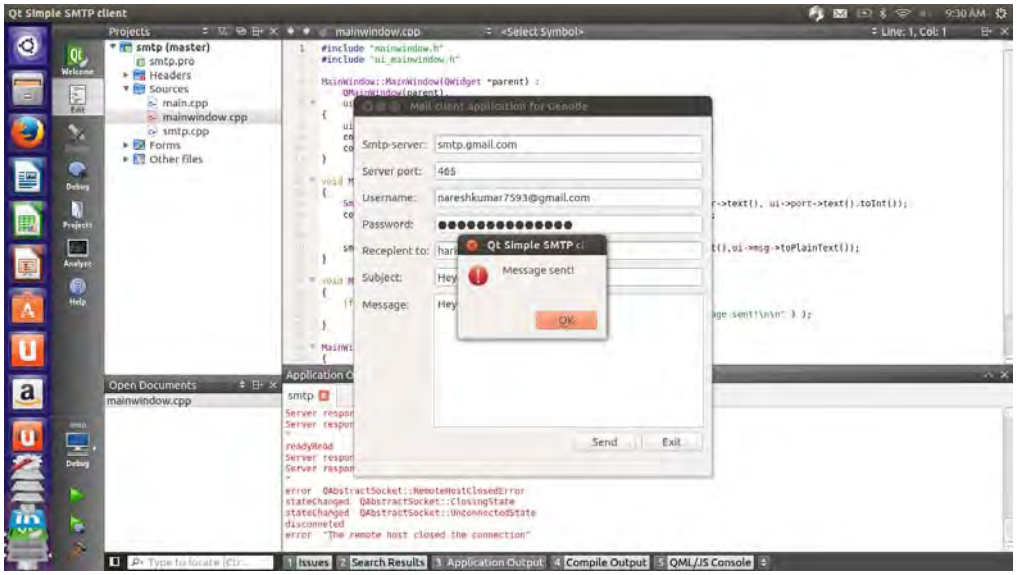

Figure 3: Mail sent from Mail client Application

Figure 3 shows the application built in Qt creator sending the mail to the receiver mail address successfully. This application is successfully built in Ubuntu 12.04 operating system. Next task is to port the same application inside Genode with its dependencies and simulate the application inside Genode.

III. PORTING MAIL CLIENT APPLICATION TO GENODE

After the application is been designed, in the final stage, the mail client application is to be ported to Genode operating system framework. Porting technically means that the application should be moved to Genode and should work efficiently in it. Porting of an existing program or library depends on how complex the application or the code is. There are various stages in porting a file. Initially, user should determine the library files and dependencies. The application can be ported once all dependencies are fulfilled.

In the second step, for the application, Makefile should be created. Source code for the application should be prepared inside Genode. Genode uses files called as “port files” for fetching the rules and commands. These port-files describe how the source is obtained. [3] This patches are later applied to the source code and address it the original code is stored and configure.

Thirdly, dependent code is checked for the application. It requires changes to the original source code of the application to be compilable to Genode. Later, after Makefiles are prepared, these are built. Within these rules we also declare all dependencies such as library files that are needed by it. The address of these commands depends on the type of the application been developed. Usually it is saved in .mk form inside Genode source code directory.

After building Makefile, run scripts are been created. Run scripts are used for implementing test scenario for the application. This run program developed is used to automatically build all components of the Genode OS framework that are needed to run the application as well as the simulator itself.

After testing the application, it is been compiled. The ported application is compiled from within the respective build directory like any other application or component of Genode [4]. The build system of Genode uses the build rules which are already been created. Post compilation, the application executed. If there is any fault in the programming or compilation, debugging of the issue should be performed. Bugs in implementation, building dependencies and library files should be debugged in this stage so that it would run in Genode.

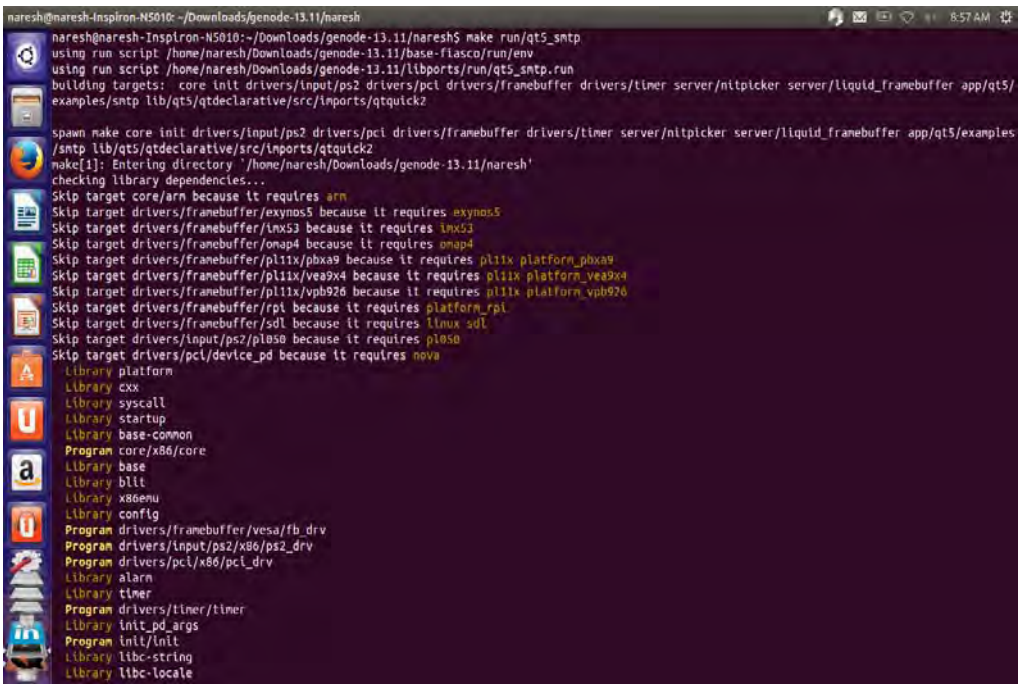

Figure 4: Compilation of all library files prior to building the application

Figure 4 shows the terminal side, where all the library files and dependency files are been compiled. The application is saved as “qt5_smtp” as specified in the terminal. For the application to run, that is to compile all the library and dependency files, we give following command: make run/qt5_smtp. This would execute run file (qt5_smtp.run). [5]Run file contains extension of all the library files which would direct the compiler to those files and are then executed. Once all the library files are compiled, program and object files are been built. Objects files are saved as .o format contains all required elements pertaining to the build object.

Figure 5 is the output been obtained in Qemu simulator after the application is been ported. Post building of the application, since dependencies include Qemu simulator, it is been compiled and later is opened and the application is displayed in launcher. Qemu launcher acts as a Genode user interface displaying the application. Here the user should specify the SMTP connection and port number pertaining to that address. Later users email address through which the mail to the receiver to be sent followed by his/her password for validating the account. On clicking send, receive would get the mail from the user. This is simple, secure and easy to use mail client application for very secure operating system framework, Genode.

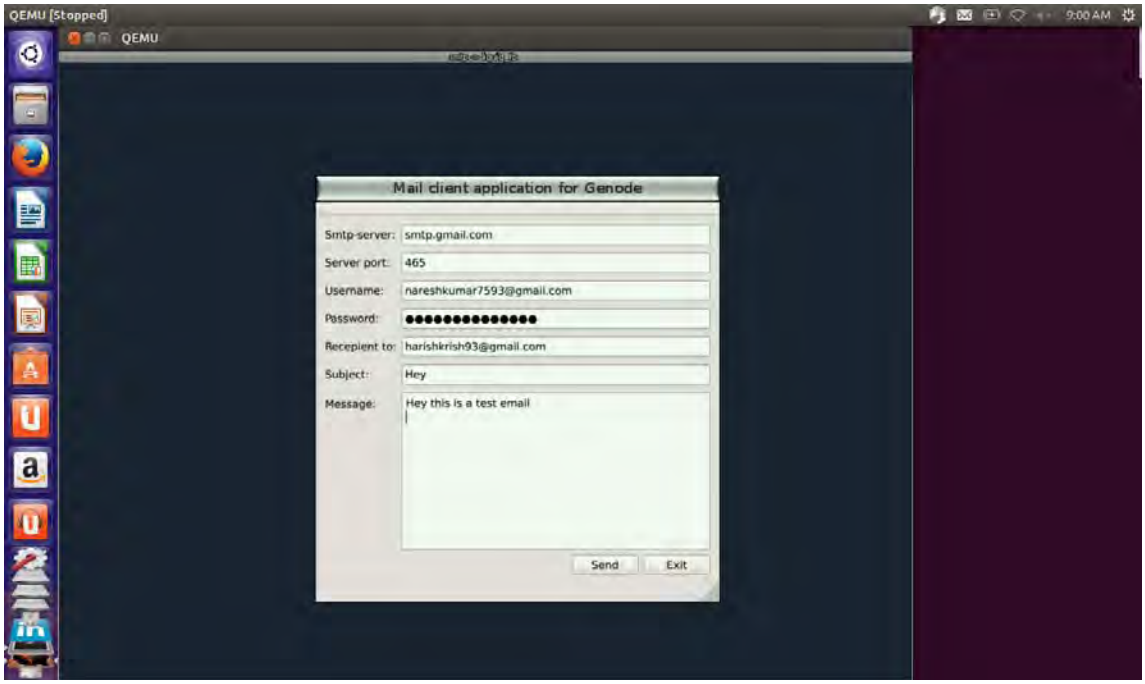

Figure 5: Mail Client Application in Qemu simulator

IV CONCLUSION

With the explosive growth in network connected devices, the need for security assurance has grown considerably. The Genode OS architecture focuses on the root of the problem by providing means to minimize the underlying system complexity for each security-sensitive application individually. On the other hand, user can enable multiple applications to execute on the system at the same time whereas each application may have different functional requirements from the operating system. The Operating system provides a higher level of security since it uses a Micro kernel approach. Applications built on this framework will hence be highly secured and can be trusted. This Mail client application built in qt creator uses SMTP connection in order to receive and send mail to particular mail address. Porting this application in Genode would prove useful to the Genode users to explore the new, simpler and easier application for managing mails.

V REFERENCES

[1] Bernhard Kauer, “OSLO: Improving the Security of Trusted Computing”, Proceedings of 16th USENIX security symposium, 2007.
[2] Hermann Härtig, Michael Hohmuth Norman Feske, Christian Helmuth, Adam Lackorzynski, Frank Mehnert, Michael Peter, “The Nizza Secure-System Architecture”, Collaborative Computing:Networking, Applications and Worksharing ,1-4244-0030-9 , December 2005.
[3] Ihor Kuz, Yan Liu, Ian Gorton, GernotHeiser, “CAmkES:A Component Model for Secure Microkernel-based Embedded Systems, ”Journal of Systems and Software Special Edition on Component-Based Software Engineering of Trustworthy Embedded Syste,m s80(5), 687699, May 2007.
[4] N. Feske, H. Härtig, “DOpE — a Window Server for Real-Time and Embedded Systems”, Technical Report TUDFI03- 10-September-2003, TU Dresden, 2003.
[5] Christian Helmuth, Norman Feske, "Design of the Genode OS ArchitecturTeU", Dresden technical report TUD-FI06-07