Journal of Theoretical and Applied Information Technology
30th April 2012. Vol. 38 No.2
© 2005 - 2012 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A MICROKERNEL BASED SECURE OPERATING SYSTEM USING GENODE FRAMEWORK

[1]B.PRUTHIVIRAJ, [2]G.S.MADHUSUTHUN, [3]S.VIJAYASARATHY, [4]K.CHAKRAPANI.

[1]M.Tech., Embedded Systems, School of Computing (SOC), SASTRA UNIVERSITY, Thanjavur.

[2] Senior Technical Consultant, [3] Senior Research Associate, Society for Electronic Transactions and Security (SETS), Chennai-600021, India.

[4] AssistantProfessor, School of Computing (SOC), SASTRA UNIVERSITY, Thanjavur-613401,India.

E-mail: pruthivi2010@gmail.com,madhu@setsindia.net,vijayasarathy@setsindia.net,kcp@core.sastra.edu

ABSTRACT

In a period when financial transactions have become substantially online, it is important from both security and functional perspective to have a secure and robust operating environment where these transactions can be performed. Although many operating systems can be configured to be reasonably secure, the problem behind the approach is in the design of the operating system. For example, most Linux distributions (including the acclaimed SELinux) execute a monolithic kernel approach, which essentially do not differentiate between the basic kernel operations and the operating system functionality built over them. This work discusses the advantages of a microkernel based approach to Operating System Design as given by the Genode operating system framework, and proposes two test security applications – a) An OTS Threat Management application which we build into the Genode framework; and b) A client-server based security application using the PolarSSL library, which we build into the framework.

Keywords: *Genode Operating System framework, Microkernel, Integrated Thread Management Appliances (ITMA), PolarSSL.*

1. INTRODUCTION

The modern world has predominantly moved by conducting financial transactions online. Example includes banking transactions, ticket reservation and credit card payments etc. Unfortunately, the applications which enable these transactions are executed in an environment which is susceptible to security leaks or failure for some reason. For example, browsers may store information in some form (cookies for example), which may be stolen by a malicious process running beside it. Another weird situation is one when an application crashes due to another application, which may cause inconsistencies and hassles. It is important from both security and functional perspective to have a secure and robust operating environment where these transactions can be performed. In essence, a Trusted Computing Base (TCB) is required for every application. Both these aspects are related to the OS design approach.

Most Linux distributions follow a monolithic kernel approach – where the kernel and the operating system are seen together as an integrated component. This brings forth the problem of maintainability – the kernel operations become unmanageable because of the large number of lines of kernel code that have to be managed. The extensibility support in such kernels is also very limited due to the fact that bug fixes in critical pieces of the kernel calls for kernel recompilation, which is resource and time consuming. Hence, the operating system design should be modular in order to improve maintainability and security.

Operating System has two parts, viz. kernel space and user space. In monolithic kernel based Operating System, each and every system service (Process Manager, File system, Input Driver, etc.,) runs in a kernel space. So all the process runs in a single address space.

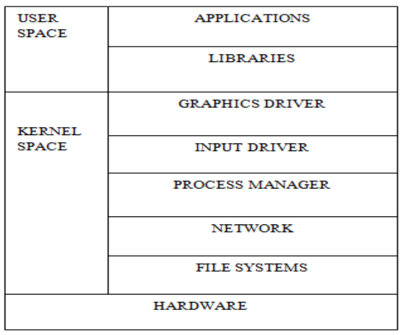


Figure 1-A Typical Monolithic Kernel Based Design

Another perspective of looking into the problem is to look at kernel certification. In critical applications, many organizations consider it important to have the underlying kernel certified for use with the applications. A modular design will improve modeling the interactions of entities with the kernel, which will help in mathematically verifying if a kernel is secure. The Operating System must maintain strong code and data separation on a per-page level. All code sections should be verifiable on a per-page level. Usually this means some signing or hashing scheme should be implemented. Operating System must allow safely reading physical memory by a third person application.

In summary, in order to maintain a trusted and a reliable application environment, the operating system design must be a) Modular; and b) Verifiable.

2. MICROKERNEL BASED APPROACH TO OS DESIGN

The solution to the above problems lies in a micro-kernel based approach to OS design. A microkernel is a bare minimum version of the kernel which supports only essential kernel functions. In Micro kernels, the kernel is broken into separate processes .Some process runs in kernel space and others in user-space.

All processes are kept separately and run with different address spaces. The size of the kernel is kept at a minimum and we can extend our services, thereby making it easy to maintain our system. If any bug is fixed, that process alone gets affected, rest of the processes of the system should work. It doesn't affect the whole system.

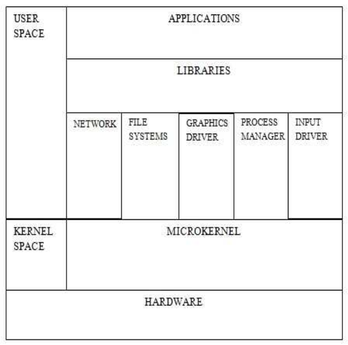


Figure2-A Typical Micro Kernel Based Design

By using micro kernels instead of monolithic kernel to our operating system, we can ensure that the critical applications running over the system are provided access to a secure Operating System environment.

2.1 Genode Operating System Framework

The Genode operating system framework means to provide decoupling the kernel and the operating system and make OS design modular. This framework proposes a microkernel based 3-tier approach. At the lowest tier is the microkernel, which consists of bare minimum kernel operations. On top of the microkernel is the OS, which act as an intermediary between microkernel and the Para virtualized Linux.

OS is the collection of programs that manage hardware resources and it provides services for application. While it is desirable to have all or most applications running directly over the operating system, it is desirable to have a virtualized Linux environment to facilitate backward compatibility. This is provided on the top of the OS in the form of a Para-virtualized Linux layer. Terminals, Audio player, PDF viewer which are currently running are the native Linux application. Although Running in a Para virtualized Linux environment, they act like a native Genode applications.

Genode is Secure Operating System architecture. This framework supports various Micro kernels like fiasco, pistachio and it has a Para Virtualized Linux Layer (Tiny Core Linux) itself. Genode is able to reduce the system complexity for each security sensitive applications individually.To minimize the system complexity it uses a strict organizational structure for all software components and applications.

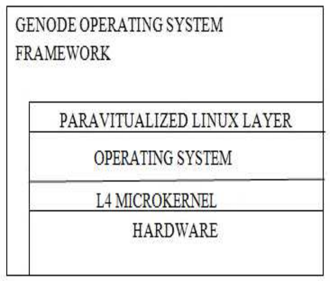


Figure 3-Genode Operating System Framework Stack

Genode maintains a tree of processes. The arrow denotes child processes. The parent process defines the virtual environment whenever the new child process was executed. It creates the arbitrarily structured subsystem. When the child wants to create children from its assigned resources and it shouldalso defines their inter relationship.It gets isolated when a new process is not allowed to communicate with its immediate parent. The parent processprovides the rights to communicate their child subsystems. The attacker's job is very easy whenever they are having visible names or ids. But Genode protects names, process identification number etc., L4Linux kernel and hypervisor is ported into the Genode Operating system framework.

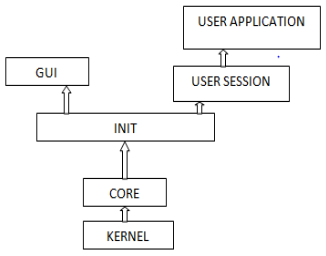


Figure 4- Organizational Structure of Genode

Each application only depends on the TCB (Trusted Computing Base). TCB is minimumwhen TCB of applications gets executed completely, isolated and independent to each other. We can use Genode framework as an application over OS by using the Genode tool chain and Genode source. Genode allows to build an application directly communicate with underlying OS and kernel, instead of using the virtual Linux layer.

3. CONTRIBUTION OF THE WORK

Secure Operating System approach consists of two modules.

a. Integration of standard applications.
b. Write a test application using PolarSSL library.

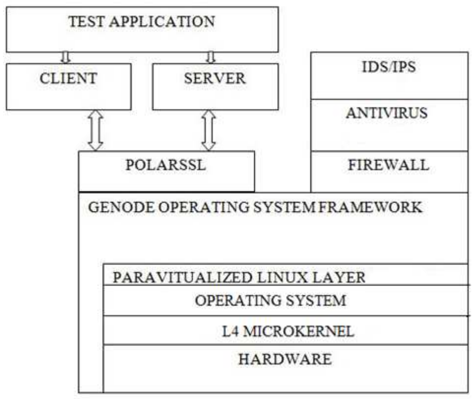


Figure 5-Architecture of Microkernel Based Secure Operating System

3.1.1. Integration ofStandard Applications

The first attempt was to build over the Genode framework, a collection of network monitoring and traffic handling tools, together known as a Threat Management Appliance. Before integrating these ITMA components, we need to boot the Genode ISO into the Virtual box and configure all the settings in the virtual box.

ITMA Components

ITMA components are Firewall, IDS/IPS and Antivirus. These security components are the open sources which provide confidentiality, integrity, availability to the operating system environment. Get these sources and build it over the Para-Virtualized Linux Layer.

Intrusion Detection System (IDS)-Snort

Snort is an intrusion Detection system that is used for performing real-time traffic analysis and packet logging. Snort depends on m4, Flex, Bison, Libpcap, PCRE, Libdnet, zlib, Barnyard2, DAQ. We need to get these dependencies built it over a Para-virtualized Linux Layer before building a snort.

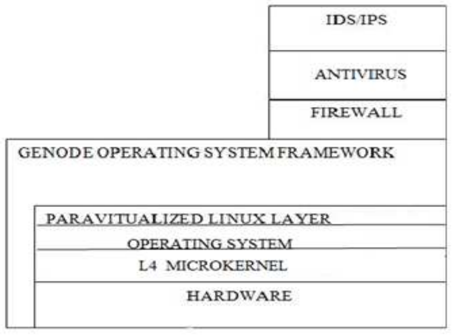
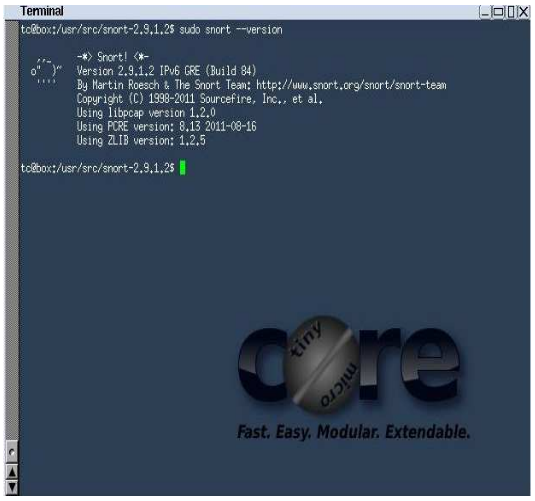


Figure 6-Integration of Standard Applications

The following figure shows building of a snort over a Para-virtualized Linux Layer.



```
tc@box:/usr/src/snort-2.9.1.2$ sudo snort --version

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.1.2 IPv6 GRE (Build 84)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.2.0
           Using PCRE version: 8.13 2011-08-16
           Using ZLIB version: 1.2.5

tc@box:/usr/src/snort-2.9.1.2$
```

Figure 7-Startup of a Snort

.

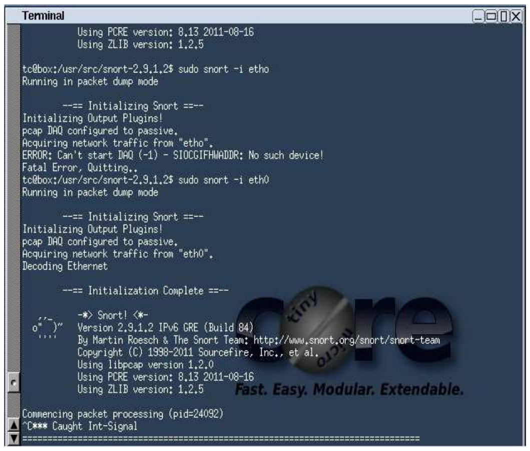
Similarly, firewall and antivirus were built.

Firewall

Firewall is mainly designed to prevent the unauthorized access from a private network. By using iPtables, the Firewall should be implemented in open source environment. It should be built it over a Para-virtualized Linux Layerby getting the shoreline firewall source.

Antivirus

ClamAV is an antivirus engine used for detecting the Viruses, Trojans and other malicious threats. Get a ClamAV source and built it over aPara-virtualized Linux Layer. Here the Para virtualized Linux Layer is tiny core Linux.It is running at the top of the Genode Operating system Framework.



```
           Using PCRE version: 8.13 2011-08-16
           Using ZLIB version: 1.2.5

tc@box:/usr/src/snort-2.9.1.2$ sudo snort -i etho
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "etho".
ERROR: Can't start DAQ (-1) - SIOCGIFHWADDR: No such device!
Fatal Error, Quitting..
tc@box:/usr/src/snort-2.9.1.2$ sudo snort -i eth0
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
pcap DAQ configured to passive.
Acquiring network traffic from "eth0".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.1.2 IPv6 GRE (Build 84)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-team
           Copyright (C) 1998-2011 Sourcefire, Inc., et al.
           Using libpcap version 1.2.0
           Using PCRE version: 8.13 2011-08-16
           Using ZLIB version: 1.2.5

Commencing packet processing (pid=24092)
^C*** Caught Int-Signal
===============================================================================
```

Figure 8-Running of a Snort

Based on the target hardware and target environment, open source components were customized and built by modifying the configuration files.

3.1.2. Security Analysis

Security of our proposed platform is managed by the several aspects.

i. Antivirus (clam AV), Intrusion Detection System (Snort), Firewall (Shore wall) is integrated over the Para virtualized Linux Layer. Hence this Operating System has these security components as default.
ii. The platform provides facility to run critical applications, completely isolated from the OS environment and independent from each other. So even in the case of application gets compromised it won't affect the underlying system.
iii. The platform offers to write a device driver with high performance.
iv. One of the requirements to be a secure Operating System its size of the microkernel should be minimal. The code size of the microkernel used is 20KLOC.This provides easy code auditing.
v. The platform provides an encrypted secure boot, which means it boots with several encryption schemes.

3.2. Write A Test Application Using PolarSSL Library

Writing a security test application on the Genode framework can be achieved by using a standard PolarSSL library. PolarSSL is a crypto library which is easily integrates with the embedded applications. PolarSSL is written by c language. Normally we are using TCP/IP Stack. The PolarSSL SSL/TLS library is building on the network-interface and it provides an abstraction layer. This layer is used for secure communication between client and server. Client must use this PolarSSL and abstract a secure communication from itself.

Now our goal is to secure our client-server communication on the Genode using PolarSSLlibrary. To achieve this objective first we have to get an open source PolarSSL library and port it into the Genode operating system framework which acts as a server for our communication. Server decides in which cipher suite (MD5, RSA, and SHA-1 etc.) is used for communication. We can implement this model using socket programming or any other programming techniques. It can be done by the following steps.

i. Server runs an arbitrary port number, then client starts with the hostname and server's port number. After which client can send data to the server.

ii. After that the server performs cryptographic functions like MD5 for a input data's or message by linking PolarSSL libraries and it sends the result to the client.

iii. Client receives the data and it performs MD5 function by linking PolarSSL libraries in order to verify with the original data.

So the client-server communication is done securely by using PolarSSL library. By porting the PolarSSL library in the Genode Operating System Framework, we can perform various cryptography schemes likeAES, DES, Triple DES,DSA,RSA,MD2,MD4,SHA-1,SHA-2 etc.

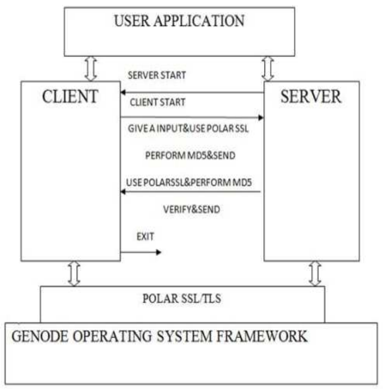


Figure 9-Write a Test Application on the Genode

4. CONCLUSION AND FUTUREWORK

Working in a Secure Operating System Environment is achieved by integrating ITMA components and built a test application on a Genode Operating System framework. Process communication is performed using a socket programming .So the Operating System can be used in any security conscious industries. As a future work, we plan to run different security components such as Firewall, IDS, and Antivirus in different isolated Para virtualized Linux.. We can also use it as a stand-alone Operating System. And also, this operating system can be ported into any supporting mobile devices.

REFERENCES

[1] 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 Systems*, 80(5), 687699, May 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] Christian Helmuth, Norman Feske, "Design of the Genode OS Architecture", *TU*

Dresden technical report TUD-FI06-07, Dresden, Germany,December 2006.

[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] M.Gasser, “Building a Secure Computer System”, *Van Nostrand Reinhold Co*., 1988.

[6] T.Garfinkel, B.Pfaff, J.Chow, M. Rosenblum, D.Boneh, “Terra: A Virtual Machine-based Platform for Trusted Computing”, *In Proceedings of the nineteenth ACM Symposium on Operating systems principles, pages 193–206,ACM Press*, 2003.

[7] M.Atighetchi, P.P.PaL, C.C. Jones, P. Rubel, R.E.Schantz, J.P.Loyall, J.A Zinky, “Building Auto-adaptive Distributed Applications: the QuO-APOD Experience”, *Distributed Computing Systems Workshops, Proceedings, 23rd International Conference* on 19-22 May .2003.

[8] J.Li, M.Krohn, D.Mazières, D.Shasha, “Secure Untrusted Data Repository (SUNDR)”, *In Proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 121–136, San Francisco, CA,Dec. 2004.

[9] Bernhard Kauer, “OSLO: Improving the Security of Trusted Computing”, *Proceedings of 16th USENIX security symposium*, 2007.

[10] F.Al-Masalha, R.Hasimoto, A.Khokhar, “Performance Evaluation of Different Encryption Schemes on Portable and Mobile Platforms”, *Green Computing Conference l5-18* Aug. 2010.