International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 4 517 - 521

Optimum Use of Handheld Device Using Monolithic Kernel Architecture for Security Purpose

Nilesh Dhale
CT Department
YCCE
Nagpur, India
neil.dhale@gmail.com

Prof. Shrikant B. Ardhapurkar
CT Department
YCCE
Nagpur, India
shrikant.999@gmail.com

***Abstract*—** Recompile the kernel to customize as per user need so that unnecessary running application will not be available. This will secure the machine for specific purpose only since we optimize the system and make it reliable. We made camera application and specific module only sustainable in order to achieve our objective security by optimizing the system. Trusted computing based work has been proposed for the system which is necessity of modular monolithic kernel architecture.

Keywords- *Device optimization, kernel opimization, Raspberry Pi*

*****

I. INTRODUCTION

Monolithic Kernel provides the largest possible number of feature as well as maximum number of device driver. In order to cover broadest scale of hardware configuration used. This is why we prefer to compile the kernel in order to only include what they specifically need.[1]

The reason is it may optimize the several things such as ring 0 space complexity as well as optimize the memory consumption since the kernel model. If it is never used and the locally compiled kernel can also limit the security risk problem due to fraction of code is compiled and run.[2].

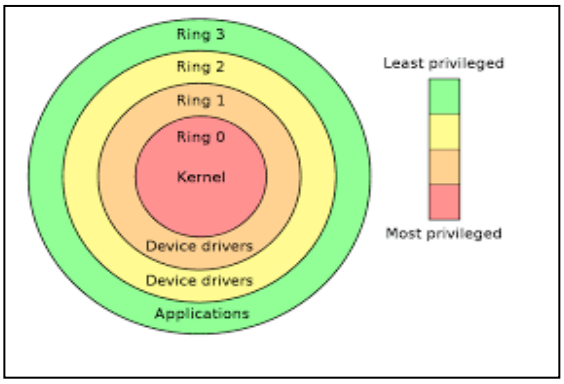

Fig 1. Kernel Ring Architecture

The kernel ring architecture descript that is has 4 ring which limit the security risk i.e. Ring 0 to ring 3.The ring 0 is most kernel level privilege which is only allowed to system call. Ring 1 and ring 2 contains the device driver for hardware enable. Outer most ring is application ring this is also called as user mode. User requested through the applications to the system kernel through the system call. This application could be untrusted and attacker could use this to craft an application that exposed the sensitive information[3]. So the attacker could focus on the any number of the subsystems. Minimizing software access to kernel mode reduces security risks. With less trusted software there is a reduced likelihood of system destabilization. A system of flags is used to associate permission level with specific memory segments. A ring feature enforces these permissions. In order to cross between rings, a context switch occurs. A common design goal is to minimize the number of context switches a subsystem must make in order to perform a common task. Correctly gating access prevents programs with reduced privileges from misusing resources that belong to trusted rings. Security rings correspond with CPU modes in some systems. This provides added hardware protection. The dependability of module must be identified to ignore the untrusted things and recompile these module in the system. Here we are discussing about the raspberry pi camera application so our work is specifically to camera application module. For example, spyware running as a user program in Ring 3 should be prevented from turning on a web camera without informing the user, since hardware access should be a Ring 1 function reserved for device drivers. Programs such as web browsers running in higher numbered rings must request access to the network, a resource restricted to a lower numbered ring [3].

As explaining about the customization of Kernel as per user need we required tiny computer which enable the various sensor. This sensor could be used for various purpose. If we talking about the camera application, this sensor application could be used to take photos of remote areas securely. This device implementation could only know maker so intrusion could not exploit it. This tiny device is Raspberry pi (using B model).

A .Need of the Recompilation of the Kernel

Users want to recompile their kernel, or compile a customized kernel. This might be for several reasons:

- We can install bug-fixes, security updates, or new functionality by rebuilding the kernel from updated sources.
- By removing unused device drivers and kernel sub-systems from your con- figuration, you can dramatically reduce kernel size and, therefore, memory usage.

- By enabling optimizations more specific to your hardware, or tuning the system to match your specific sizing and workload, you can improve performance.
- We can access additional features by enabling kernel options or sub-systems, some of which are experimental or disabled by default.
- We can solve problems of detection/conflicts of peripherals. We can customize some options for example keyboard layout, BIOS clock set. We can get a deeper knowledge of the system.

B .Device Specifiaction :

The Raspberry Pi Model B is the second generation Raspberry Pi[4] credit card size. Device having A 900MHz quad-core ARM Cortex-A7 CPU, 512 MB to 1GB RAM, 4 USB ports,34 to 40 GPIO pins, Full HDMI port, Ethernet port, Combined 3.5mm audio jack and composite video, Camera interface (CSI), Display interface (DSI), Micro SD card slot, VideoCore IV 3D graphics core, camera interface.

As the device provided the CSI the Camera Board is a small PCB that connects to the CSI camera port on the Raspberry Pi using a short ribbon cable. It provides connectivity for a camera capable of capturing still images or video recordings. The camera connects to the Image System Pipeline (ISP) in the Raspberry Pi's SoC, where the incoming camera data is processed and eventually converted to an image or video on the SD card (or other storage).

The above device configuration shows the capability of the system for modern application as usefulness. Although we never used the device capability as whole, something is left out unused. This device is cheaper and having great flexibility capable. Operating system is used Raspbian along with Linux kernel 4.x version.

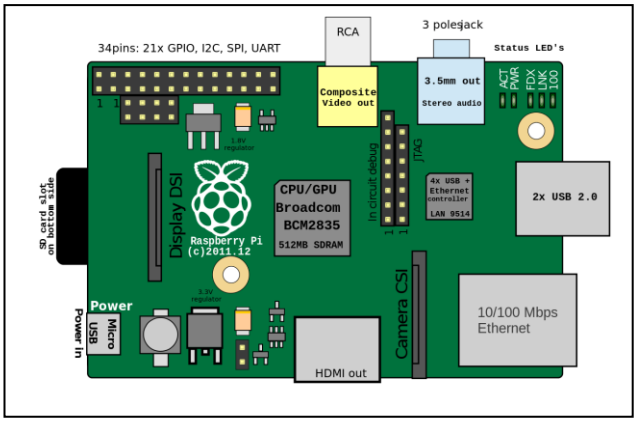

Fig 2. Raspberry Pi Device

II. RELATED WORK :

Dr. Jordan Shropshire[2], analyzed improved hypervisor kernel isolation but increased the vulnerability of subsystem which had migrated out of the hypervisor. This paper explains that Minimizing software access to kernel mode reduces security risks.

JuanMariano de Goyeneche and Elena Apolinario Fernandez de Sousa,1999,This paper gives concept of the flexibility and efficient way to update the kernel and load required module.

Andrew S. Tanenbaum, Jorrit N. Herder, and Herbert Bos[5], Microkernel's long discarded as unacceptable because of their lower performance compared with monolithic kernel might be making a comeback in operating systems due to their potentially higher reliability, which many researchers now regard as more important than performance.

Susmit Bagchi[7], The monolithic kernel based design of distributed IPC architecture can offer high performance and low latency. IPC used as middleware which easy portability kernel-level subsystems offer improved performance and system security.

Kimmo E.E. Raatikainen[10], In order to support reconfigurability, the operating system research must solve several research issues related to self awareness, detection and notifications, system integrity, and power management.

Bo Qu ,Zhaozhi Wu[1], The experiment consists five part: booting, interrupt and exception handling, process schedule, memory management, and signal processing. Studied the kernel duties.

III. IMPLEMENTATION

A. *Work flow of the Project*

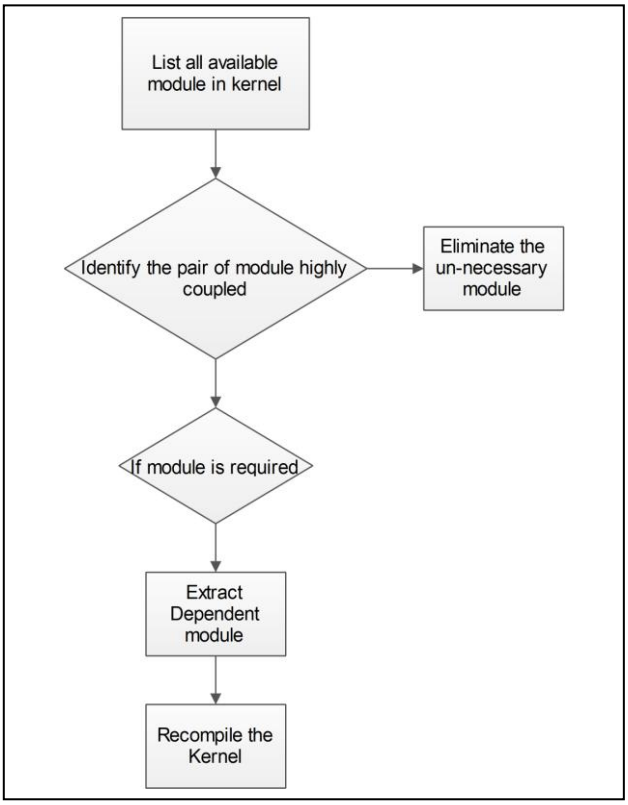

Fig 3. Work Flow of Project

Steps for work flow:

- List all available kernel module List out the Kernel module by using "lsmod" command in the terminal. This command gives the all kernel module.
- Identify pair of highly coupled module After listing the modules there is "used by" column where count number is there. Maximum number of count means

the highly coupled module. This count defines the dependency of each module.

- Required module The highly coupled module and all other dependent module is required to the system. This dependent module extracted for the recompilation.
- Eliminate Unnecessary module If the count is zero in ”used by” column while listing means this module cannot be used by any application and this is unused or unnecessary module. Elimination can be done in rmmod.
- Recompilation of kernel Install the packages needed for Building the Kernel. Install utility for building Linux kernel ,the utility which allow to install the linux kernel module. This module can be install in insmod only and delete the module in rmmod only. Make the changes in kernel. Use the make menuconfig command which is used for pop-up the configuration option window. Clean the source tree and reset the kernel package parameter. Then start recompilation.

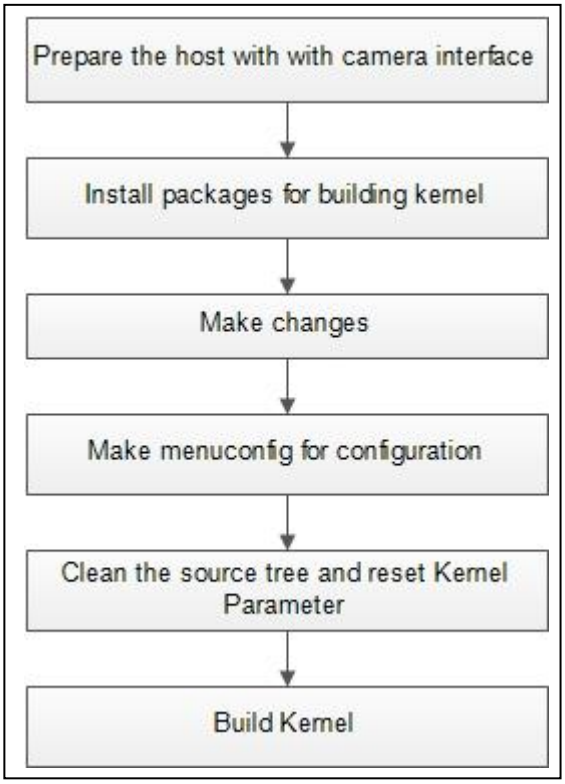

Fig.4 Recompilation Workflow

B. Coupling of Module

The above flow show that List of available module gain from the kernel. This module is currently live and all module. We check the individual module’s used by count value if the value is maximum in all module means this module is used by other module in the kernel. Used by count is the actual number of module which is used the module. We extract this module then check again the module which is less coupled. We do this until all module have only count is 0. This module is dependent on each other which functionally related to each. If any of this is missing in Kernel leads to unreliability in the system.

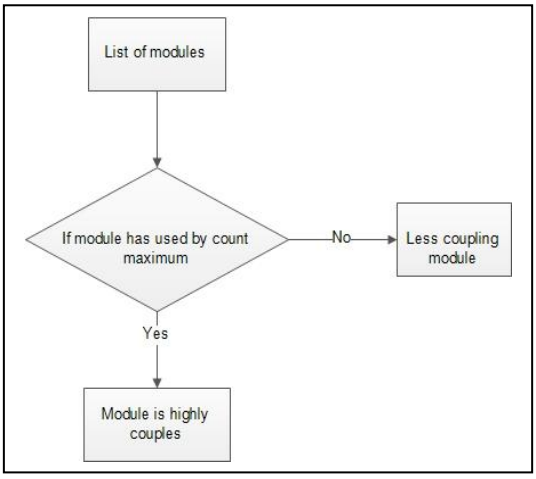

Fig 5. Coupling of Module

C. Required module :

After identifying the kernel coupling module, we recognizing the kernel’s required module. Even if the module used by count is 0 the kernel module could be used by the system. If this Module is used by the system and not dependent on the any other module may require for the device functioning. This module could not allowed the system to be unload from the kernel if we do then kernel may be improper functioning. Other module having count is zero can easily identified the unused module and making overhead to the system. This module is allowed to be unloading from the system.

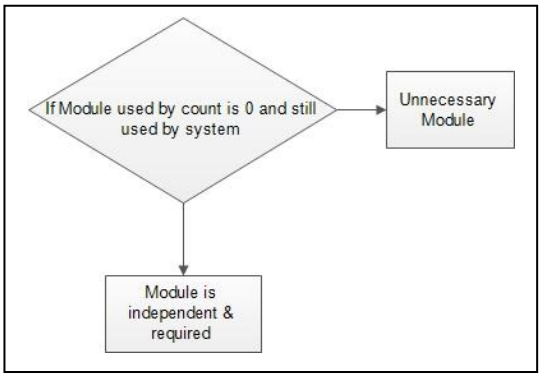

Fig.6 Required Module Identification

All the module which is having count more than zero and used by system even if count zero is functional dependent module. This module is extracted for the recompilation process. conversely we unload the un-necessary kernel module. This works is logically true if we select all the dependent kernel module. This may result in more reliability in the system.

D. Extract dependent Module:

The dependency between kernel modules can be extracted statically from the kernel image and modules. For that, we start from the list of all available kernel modules for a given kernel. We are interested in the relationships between modules, that is, the function call graph that connects them. From this graph, it is possible to identify pairs of modules that are highly coupled and cause frequent execution domain switches when running under module isolation environments. To properly define the level of coupling between two modules, it is not enough to just determine that there is a functional dependency between the two. If a module M1 uses 10 functions from a module M2, while a module M3 only uses 1 function from M2, M1 and M2 are more coupled than M3 and M2.

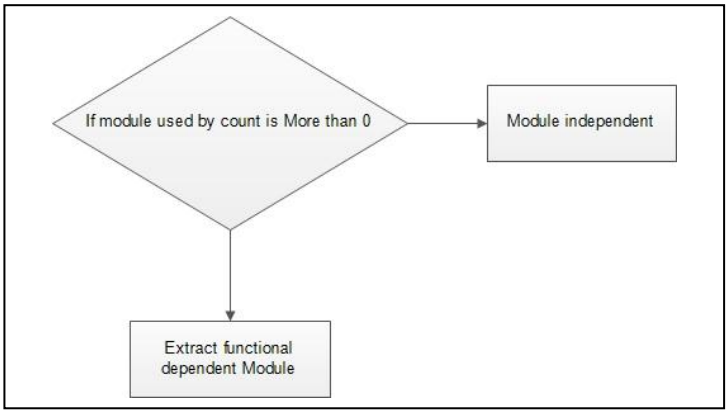

Fig.7 Extract dependent Module

E. Application Development Workflow

1. Configuration: Any computer system requires output device to show the output on the monitor but Raspberry pi has the HDMI port for output device. If supposed don't want to use the HDMI device then we can connect Raspberry pi device to local network and access that device remotely by means of IP Address. Even if you are using in network require certain software to access the graphical environment to remote machine and connection establishment to Raspberry pi device. This both the software very convenient to interact with system and operate it. Local network is created using modem device which gives very easy to create local network. We can use LAN cable or wifi device with Raspberry device. Device has built-in LAN port. We can use wifi with USB port. Here connection established with LAN cable.
2. Software requirement : Xming : Xming is X Window System Server for Microsoft Windows. It is fully featured, lean, fast, simple because it is standalone native Windows, easily made portable. Xming is cross-compiled on Linux for Microsoft Windows. PuTTy: PuTTY is a free and open-source terminal; serial console and network file transfer application. It supports several network protocol, including SCP, SSH, Telnet and raw socket connection. It can also connect to a serial port.
3. Sensor assembly: As objective is to make the camera application, assemble the camera device which is mounted on the Raspberry pi device board. CSI camera serial interface provide to connect the camera with Raspberry pi device.
4. Code for sensor application: sensor code is written in python in the project where the code has been written for camera for capture the images and collected at selected location. This code is to auto capture image in the interval for the monitoring the devices.
5. Test and debug: Run the module written for application in python shell 2.7.9 which started capturing the images automatically

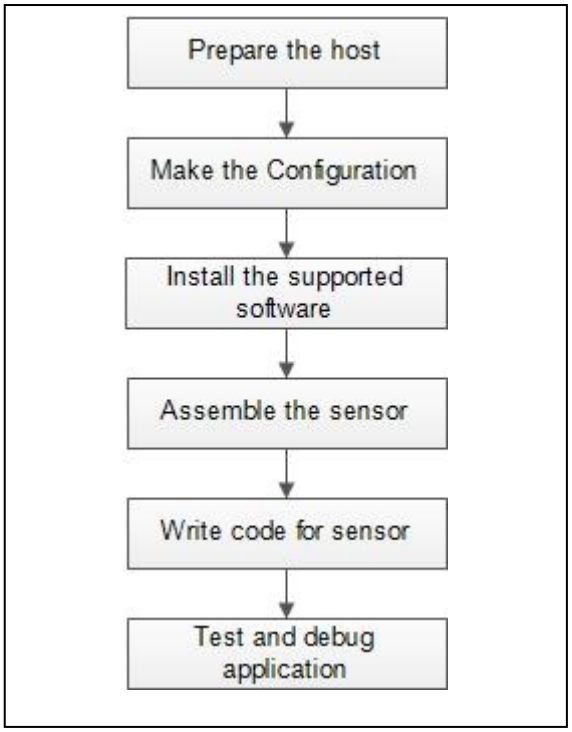

Fig.8 Application Development Workflow

a. Preparing the Host

Preparing the host means install the required software Xming and PuTTy in the system. This xming software helps to retrieve the graphical user interface of the Raspbian OS. This host is Windows machine. Xmimg is x window system server ,Windows OS based Software. puTTy is software that establish the connection through local network between host and Raspberry Pi device.

b. Configuration of the system

1. On the Host Device first.
2. Power the Raspberry Pi device.
3. Connect the Raspberry Pi device to modem By LAN cable.

4. Connect the Host with wifi with modem.
5. Obtain the IP address of Raspberry pi device with help of admin control panel of the modem.
6. Active the xmimg software then receive the IP address of Raspberry pi de- vice. This IP address put up the puTTy software the and enable the xming window by mark check box.
7. Connection is well then automatically xming brings the window of Raspbian OS startup.
8. Enter the user name as 'pi' and password 'raspberry' then device root pre- vilege granted.
9. To start graphical user Interface used command "lxsession".

IV. Conclusion

Trusted computing based research is could be used to secure the hypervisor subsystem of the kernel which is need to be trusted. Some crafted application could exposed the kernel information by gaining the control over the un used kernel module. For those limitation trusted computing based approach is used to Recompile the kernel as per specific need which also gives the performance improvement of the system. As a result we could use this application for monitoring system where the remote areas images could not possible by individual any time. In such a way optimization of the system could be possible for the various application need.

References

[1] Raphael Hertzog and Roland mas,https://debian-handbook.info/browse/stable/sect.kernelcompilation.html, pp.1,05-Apr-16

[2] Shropshire, J., "Analysis of Monolithic and Microkernel Architectures: Towards Secure Hypervisor Design," in System Sciences (HICSS), 2014 47th Hawaii International Conference on , vol., no., pp.5008-5017, 6-9 Jan. 2014 doi: 10.1109/HICSS.2014.615

[3] wikipedia,https://en.wikipedia.org/wiki/Protection_ring,pp.1,08-Apr-16,

[4] RASPBERRYPIFOUNDATION,https://www.raspberrypi.org/products/raspberry- pi-2-model-b

[5] JAndrew S. Tanenbaum, Jorrit N. Herder, and Herbert Bos,"Can We Make Operating Systems Reliable and Secure?", IEEE Computer Society, 2006, pp. 44-51.

[6] Juan-Mariano de Goyeneche and Elena Apolinario Fernandez de Sousa,"Loadable kernel modules" , IEEE, 1999,pp. 66-71.

[7] Susmit Bagchi, "Distributed IPC using Virtual Device Driver in Monolithic Kernel", International Conference on Embedded and Real-Time Computing Systems and Applications, 2012, pp. 51-57.

[8] Gaoshou Zhai, Yaodong Li, "Analysis and Study of Security Mechanisms inside Linux Kernel", International Conference on Security Technology,2008, pp. 58- 61

[9] David Katz ,Antonio Barbalace, Saif Ansary, Akshay Ravichandran and Binoy Ravindran,"Thread Migration in a Replicated-kernel OS",International Conference on Distributed Computing Systems,2015, pp. 278-287.

[10] Kimmo E.E.Raatikainen, "Operating System Issues in Future End-User Sys- tems", International Symposium on Personal, Indoor and Mobile Radio Communications,2015, pp. 2794-2800.

[11] Bo Qu ,Zhaozhi Wu, "Kernel Experiment Series for Operating System Course Teaching", IEEE, 2011,pp.12-16.

[12] A. Brinkmann,D. Eschweiler, "A Microdriver Architecture for Error Correcting Codes inside the Linux Kernel",Nov. 14-20, 2009, Portland, Oregon, 2009, ACM.

[13] Silvana Castano,Maria Grazia, Giancario Martella, Pierangela Samarati ,"De- sign of secure operating system",in DATABASE SECURITY,2nd ed,ACM Press Books,1995,ch 3,sec 10,pp. 218-235