A virus detection framework based on SPMOS

Tianzhou Chen, Jijun Ma, Nan Zhang, Qingsong Shi
Zhejiang University-Intel Technology Center
College of Computer Science
Zhejiang University, Hangzhou, Zhejiang, 310027, P. R. China
{tzchen,ijjm,andyzn,zjsqs@zju.edu.cn}

Abstract

Embedded systems have been used in many different areas in which sensitive information communication and storage are needed. This makes security a serious concern in embedded system design, especially in operating system design. At the same time computer virus has been mutating and developing as fast as the upgrading speed of embedded operating system. Even it is possible for some intelligent virus to destroy the anti-virus software process in the memory. The system-on-a-chip technology provides Scratch-Pad Memory (SPM) which is physically isolated with main memory and more efficient than other kind of memories. We construct a demilitarized zone (DMZ) on SPM and design a small OS named SPMOS in the DMZ. A watchdog is contained in GPOS to monitor the events occurred. If an abnormal event is detected, GPOS will trap itself to SPMOS which will invoke anti-virus program. It is a big challenge to switch the two OSes without any virtual layer support. The way to protect SPM showed that the anti-virus detection platform based on SPMOS is secure. Then the experiment results show that the platform is efficient while switching between OSes.

Key Words: Embedded Security, Virus Detection, Scratch-Pad Memory

1 Introduction

With the development of electronic technology, more and more modern electronic systems are produced recent years, such as personal computers, PDAs, cell phones, network routers, smart cards, and networked sensors. All of these systems need to access, store, or communicate sensitive information, making security a serious concern in their design. Today, an increasing number of embedded systems need to deal with security in one form or another – from low-end systems such as wireless handsets, networked sensors, and smart cards, to high-end systems such as network routers, gateways, firewalls, and storage and web servers.[1].

In the early time, computer viruses infected a lot of personal computers, but now, they also infect embedded devices such as PDAs, cell phones and network routers. The early viruses were file-infecting viruses, which were simply machine language programs and had the ability to attach and spread identical copies of themselves. This type of virus is called a parasitic computer virus, since it does not kill its host and the host acts as a carrier. Because the parasitic computer virus just spreads identical copies of themselves. The virus would take control of the computer and infect other files. Thus it was easy for anti-virus products to detect and remove this type of viruses.

But the most complex class of virus to date is the polymorphic computer virus, which avoids detection by mutating itself each time it infects a new program. The mutated virus can perform the same tasks as its ancestor but it may look entirely different. These clever viruses cannot be detected effectively using traditional anti-virus scanning methods. Though many anti-virus programs start to use generic decryption technology to detect these complex polymorphic viruses, generic decryption technology uses an simulator to simulate the execution environment when you want to check whether a binary executable is infected by the polymorphic computer virus. The simulator will record the activities of the mutation engine and will search for the virus signature to detect the virus.

With the increase of embedded devices, computer viruses start to infect mobile phones. The Cabir worm successfully infected mobile phones through bluetooth connection. Cabir worm virus disguises itself as a Symbian utility called ‘Caribe Security Manager’ and is sent in Symbian’s .sis file format. If the user accepts the file, Cabir activates. It will search for nearby Bluetooth-enabled devices to pass itself along when the handset is turned on[2]. The study of Carettoni, Luca Merloni, Claudio Zanero, Stefano.[3] also shows that Bluetooth-enabled devices are easily becoming
As the integrated circuit technology is growing faster and faster, the latency between electronic units becomes the bottleneck of the improvement of system performance. SoC(System-on-a-Chip) technology introduced by Motorola in 1994 brings a solution to improve the system performance overall. With SoC technology, processor, memory units and peripheral devices are connected by a high-speed bus in chip. The main merit of SoC technology is that the whole system is integrated in a single chip. The SoC contains an on-chip memory named Scratch Pad Memory(SPM) which provides fast memory access speed.[4]

SoC design provides an integrated solution to challenge the security problems on embedded systems. This paper takes advantage of the SPM on chip to construct a security zone to contain anti-virus software. This paper introduces two operating system running on an SoC. A small operating system named SPMOS is running in SPM with a mechanism that the GPOS can not access the SPM address. The SPMOS provides an execution platform for anti-virus program. The GPOS is a general purpose operating system which is running on the chip at the same time with the SPMOS. A basic virus monitor detection routine is placed in GPOS to monitor the system. When it detects a virus, it will freeze itself and the SPMOS controls the machine. The SPMOS will invoke the anti-virus program.

The memory access mechanism is indicated in figure 2. As it shows, the memory access is always checked by the checker. Memory access from software in DMZ is dispatched to the on-chip memory or off-chip memory while memory access from GPOS to on-chip memory is denied. The program in SPMOS which is located in DMZ performs virus detection and cleaning while the GPOS can not modify the program in DMZ.

The memory size of SPM is extremely small, so the software located in DMZ should as simple as possible and its functionality could not be complex. However, network communication, virus detection program, task management and memory management is also required in DMZ. So we designed a small operating system, SPMOS, which provides these necessary modules.

2 Architecture

2.1 SoC DMZ Architecture

A normal SoC architecture with reconfigurable logic and multiple processing elements sharing a common memory hierarchy such L2 cache and memory, is quite common recently[5]. The on-chip memory named Scratch Pad SRAM(SPM) can be accessed by these processing elements. Scratch-Pad SRAM is a kind of Scratch-Pad memory equipment and it has the characteristic of high reading-writing speed and low power consumption. Recently, lots of advanced micro-processors have this kind of on-chip memory integrated. Generally, size of SPM is 256-768KB.

The SPM is usually divided into several banks and each bank has an access queue. Each memory bank has a dedicated single-entry queue and data storage. If a memory bank is in standby mode, the access requests are stored in the queue while the memory bank is placed in run mode. The access is completed when the memory bank has entered run mode. If a memory bank is in run mode and the queue does not contain any pending access requests, the queue is bypassed and the memory is accessed normally.

The on-chip memory provides an access queue for pending access from the outer layer. This property provides the possibility to implement fast and secure zone in chip. In computer networks, a Demilitarized Zone(DMZ) is a computer host or small network inserted as a "neutral zone" between a company’s private network and the outside public network. This paper describes a DMZ constructed on SoC, as figure 1 shows. In our design the SPM-based OS is put into the DMZ and so hidden from the general purpose operating system(GPOS). This neutral zone is composed with only one entrance point from general purpose operating system. The general purpose operating system can trap itself into SoC only through special interrupt switch. In normal time GPOS is running, the access to memory is checked by the memory checker.

The rest of this paper is organized as follows. Section 2 is the overview of the SPMOS based embedded system security architecture. Section 3 presents the implementation of the virus detection framework. The framework components are discussed in detail in this section too. Finally, in Section 4 we give the result of our implementation and describe the future work.

2.2 SPMOS Frameworks

The SPMOS contains five modules: task management, memory management, virus detection software, network
communication and interrupt routines. The task management is the main work of the task management module. It is designed to be able to schedule tasks by different policies, such as FCFS (first come, first serve) or SJF (shortest job first). The memory management splits the memory into three type of regions – kernel region, real-time region and user region – for different type of processes. An anti-virus detection software is contained in SPMOS to do virus detection and cleaning jobs. A network communication module is used to update virus signature database and an interrupt routine is used to switch execution back from SPMOS to GPOS.

In addition to SPMOS in SoC, the general purpose operating system is located in off-chip memory. The architecture of the general purpose operating system and SPMOS is shown in figure 1. It is indicated in figure 1 that SPMOS is located in the on-chip memory in DMZ constructed on SoC and the GPOS is in the off-chip memory. The memory accesses from CPU are all be checked by the access checker. If it is accessing the off-chip memory, there will be no barriers while if it is accessing the on-chip memory from GPOS, it will be rejected.

It is important to mention how is the two operating system loaded into the board. Figure 3 shows that the SPMOS is firstly loaded into the on-chip memory and then the SPMOS tries to load the GPOS into off-chip memory, that is main memory. Finally, after the load process, the SPMOS turns the memory access checker on. From then on, the DMZ on SoC is established. It means that the access checker can only be used or controlled by SPMOS and can not be accessed by GPOS. With this limitation, the virus located in GPOS can not change the access permission to memory access checker. It is also possible to turn the memory access checker on within boot-loader. After the boot-loader successfully loads the SPMOS and the memory access checker is turned on.

### 2.3 The Watchdog Design

Watchdog is responsible for detecting abnormal activity incurred by virus or malicious code while GPOS is running. This paper designs a watchdog running on GPOS to monitor the operating system’s behavior. The watchdog is read-only if it is running in the system. If an abnormal event occurred, it would trap itself into the SPMOS and start anti-virus program. The abnormal event can be categorized by the types of viruses. The watchdog monitors the following events:

- The CPU usage increases to 100% and it lasts for specific time period.
- The file system usage suddenly increases as the read/write action keep busy for a period of time.
- The network transfer suddenly increases and the traffic block lasts for a period of time specified by watchdog.

If the watchdog came across with the situation listed above, GPOS would trap itself to SPMOS by software interrupt. In our Intel Xscale platform, instruction “swi” is used. Then GPOS and the applications are “frozen” in the off-chip memory because it has no chance to run any program. Before switch, GPOS copy the suspicious file to a specific address which could be seen by both GPOS and SPMOS. Then the anti-virus software start to scan and kill the suspicious files. If the cleaning process finishes successfully, the SPMOS will trap itself back to the GPOS by interrupt switch. If the cleaning process fails, there are a lot of choices. For example, SPMOS could try to reload the GPOS and restart it over.

It is necessary to correctly identify the critical resources to be monitored for an effective watchdog. These resources include files, communication ports, processes and threads information. The watchdog will monitor actions take on

<table>
<thead>
<tr>
<th>Resources</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>create, read, write, rename, copy, execute</td>
</tr>
<tr>
<td>Communication ports</td>
<td>create, send, receive</td>
</tr>
<tr>
<td>Processes and threads</td>
<td>create, stop, kill, modify, allocate memory</td>
</tr>
</tbody>
</table>
these resources. The action can be taken on critical resources is shown in table 1. The watchdog monitors all the changes taking on the critical resources and compares the action with the security policy stored in database. If the action is abnormal according to the security policy, GPOS will trap itself to SPMOS for suspicious file scanning and cleaning.

Both operating systems’ interrupt related code have to be modified to fit for the switch between GPOS and SPMOS. The technical detail is listed in section 3.1.

2.4 Virus Cleaning Process

The computer virus cleaning process is also shown in figure 3. When the first interrupt triggered the switch from general purpose operating system, the SPMOS in on-chip memory starts to execute the anti-virus software. When the SPMOS finishes the execution, it will yield the execution ability to the general purpose operating system.

The virus intruders try to compromise or destroy computer system in various ways. They can destroy the file system of the general purpose operating system by copies of garbage files. They can remove the important executable or data files in the system. The modification of some data files is also performed by some computer virus such as DBASE virus. SRI848 virus can destroy the file allocation table and the DIR virus would change the information of file storage. Some virus will destroy the applications in memory, then the application will become slow and run into errors. The anti-virus application also can be destroyed by some well-designed virus. In one word, computer virus is also a software program, and it can perform everything that a normal program can do. Because the computer virus is a software program, it can not perform the tasks that a normal program can not perform. For example, it can not intrude into the RAM when machine is not active, or it can not write to a write-protected disk.

With the ability and limit of destruction of computer virus in mind, we designed the DMZ in chip to protect the SPMOS from the attacks of computer viruses. The SPMOS is waken up when watchdog in GPOS monitors an abnormal situation according the security policy. The anti-virus program in SPMOS checks the suspicious files and tries to clean the virus. If it successfully cleans the virus, it triggers an interrupt and jumps back to GPOS. If it fails, it may notify GPOS and then try to get support from network. Even more, it could restart the whole system.

3 Implementation

3.1 Switch between the two OSes

The most common way to implement multiple oses within a single platform is to provide a virtual layer under commodity OSes. The layer transfers request of OSes to underlying devices and record run-time information of each os so that they are logically isolated. We implement multiple OSes with a different method. In our implementation, each OS is considered as a task and there is no master. Switch is triggered by abnormal behavior of system or root’s command.

It may lead to error if we just change PC to jump to SPMOS and then jump back because the two OSes are not logically isolated. In our switch scheme, we define a context of operating system which includes registers, program counter address and mode information. First we save the previous context of commodity os and then run SPMOS. After virus detection process is finished, the saved context is restored. The detail is described below.

The ARM processor has seven mode. USR(usr) mode is the least privileged mode in which user program executes. In the implementation of the multi-os, we used three mode: usr, irq and svc. We use Linux version 2.6.11 kernel as the general purpose operating system and developed a small SPMOS to cooperate with it. Linux initialize the interrupt vector table in function trap_init() in arch/arm/kernel/traps.c file. Main work it should do is copy interrupt vectors in entry-armv.S into memory at address 0xffff0000.

We use the following interrupt item to switch between the general purpose operating system and the SPMOS.
The whole process of switch involves two interrupts. The first interrupt related mode is denoted by MODE(1) and the second interrupt related mode is denoted by MODE(2). All of the modes can be IRQ, SVC and USR. For example, the first interrupt switch related mode IRQ is denoted as IRQ(1). The interrupt switch occurs in two steps:

The first interrupt based switch contains three steps: switch from USR to IRQ(1); switch from IRQ(1) to SVC(1); switch from SVC(1) to IRQ(2). Because we have not designed file system in SPMOS. So when the first interrupt comes, the general purpose operating system transfers suspicious file into specified address in SPM and then give the execution right to the SPMOS. The SPMOS then have the ability to see and perform virus detection routine.

This is simple but useful for switching between the two operating systems. The way to just change the PC is error prone and requires additional complex work to deal with it. So we make use of interrupt scheme which handles all of these complex things automatically. It is a simple task to modify the return address which restores the general purpose operating system to its original status.

The second interrupt based switch also contains three steps: switch from IRQ(2) to SVC(2); switch from SVC(2) to IRQ(1); switch from IRQ(1) to USR(1).

At this time, the interrupt switch is triggered in SRAM, so the return address and SPSR (Saved Program Status Register) belong to SPMOS. In order to switch back to the general purpose operating system, the return address lr_irq and spsr_irq should be changed to the value saved before the first interrupt occurred. With these efforts, the general purpose operating system can run continually and don’t know that the SPMOS has something done.

### 3.2 Protection of SPM

Memory protection is an important problem while multiple OSes are running on a single machine. As Figure 1 shown, in our architecture, GPOS are running in SDRAM while SPMOS in SPM. If GPOS was hacked by some virus, SPMOS may be compromised along with the IDS being undependable. So any access from process in GPOS should be blocked. The protection is implemented by checking bits in descriptor of page table entry related to memory space of SPM.

On our platform, a collection of sections, large pages or small pages are called a domain. The first-level descriptor got in the process of address translation contains a four bits field, which selects one of sixteen domains controlled by a 2-bit field in the Domain Access Control Register (32 bits). A program could be a client or a manager of the domain.

<table>
<thead>
<tr>
<th>Value</th>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b00</td>
<td>No access</td>
<td>Any access generates a domain fault</td>
</tr>
<tr>
<td>0b01</td>
<td>Client</td>
<td>Accesses are checked</td>
</tr>
<tr>
<td>0b10</td>
<td>Reserved</td>
<td>UNPREDICTABLE</td>
</tr>
<tr>
<td>0b11</td>
<td>Manager</td>
<td>Accesses are not checked</td>
</tr>
</tbody>
</table>

Each field in the Domain Access Control Register determines the process’s role. If it is a manager, there will not be access checking. Otherwise, every memory access will be checked against AP field in the first-level descriptor. If access permission is violated, a MMU fault is raised. This method allows the access to a domain to be enabled and disabled very quickly.[6]

The protection mechanism could be depicted in Figure 4. Memory access consists of accessing on-chip memory or off-chip memory. Off-chip memory access, that is SPM access from GPOS, is denied. So IDS in SPMOS can perform network package examining and filing without any interference from GPOS. Table 2 lists different roles for a certain domain the application could play.

 Obviously, we should choose the whole SPM as a domain and select general purpose operating system and other application as client of the domain. What we should do is to set domain field to "client" and set AP field without write access while mapping SPM into process page tables.

### 3.3 Virus Detection Platform

What the SPMOS can do with the general operating system is to do virus detection or virus cleaning jobs. Our research provides a framework for the computer virus detection platform, but does not provide the anti-virus software. It means that the virus detection software is provided by third party companies, and the new update of virus feature is also provided by anti-virus software companies.
In the virus detection platform constructed in SPMOS, anti-virus software is protected by the SPMOS and can efficiently check virus in the general purpose operating system. As the general purpose operating system is transparent to the SPMOS but the latter is sheltered from the former, the anti-virus program is well protected from disturbance by the computer virus or other malicious program.

The anti-virus software is provided by third party companies and the updates of the virus feature library is also provided by these companies. The recent update methods usually depends on network connections. New virus features are downloaded from the server of anti-virus software providers. Our virus detection platform also support network connections in its design. Thus, it is easy to make the anti-virus feature library up-to-date.

4 Result and Future Work

Intel PXA27x processor is the product of Intel Company and it is an integrated system-on-a-chip microprocessor designed for mobile devices. The frequency of PXA27x development board is 520 M-HZ. PXA27x provides extra 256KB cache which is considered as internal memory. This cache is internal memory-mapped SRAM which consists of four banks with the capacity of 64KB. The SRAM array module consists of four banks of 8KB x 64bit memory arrays. Each memory bank has a dedicated single-entry queue and 8KB x 64bits for data storage. If a memory bank is in standby mode, the access request is stored in the queue while the memory bank is placed in run mode. The access is completed when the memory bank has entered run mode. If a memory bank is in run mode and the queue does not contain any pending access requests, the queue is bypassed and the memory is accessed normally.

We constructed the DMZ with the Intel PXA27x processor. The Intel PXA27x processor family provides industry-leading multimedia performance, low-power capabilities, rich peripheral integration, and second-generation memory stacking. These excellent features are especially designed for wireless clients such as smart phone. Our SPMOS is developed on Intel PXA27x processor for this reason.

With the access checker of SPM, the DMZ on the chip cannot be accessed from the general purpose operating system. Thus, it is safe for SPMOS located in the chip.

As it is shown in table 3, the size of SPMOS image is 8K; the load time of SPMOS is less then 3 ms and the average task switch time is 4.307 μs and the code size of the switch routine is 390 lines of arm assembly language. As the switch procedure involves registers modification and memory accesses, there are some degree of latency. It proves that the virus detection framework based on SPMOS is a good try to solve security problem on embedded systems. Our approach is based on the SoC architecture. This shows great performance increment. The VLSI technology is developing fast recent years. It will be a good try to make the SPMOS and anti-virus program into hardware.

Acknowledgements: The research was supported by the National Natural Science Foundation of China under Grant No. 60673149 and the National High-Tech Research and Development Plan of China ‘Hardware-software Codesign of virtual FPGA’.

References