International Journal of Trend in Scientific Research and Development, Volume 1(2), ISSN: 2456-6470
www.ijtsrd.com

Capability Memory Protection for Embedded System

Anitha M. T.
Lecturer in Electronics, Government Polytechnic College, Kothamangalam, Kerala, India

ABSTRACT
Embedded systems need memory protection measures to be effective. This is a big change for embedded systems, which previously didn't require memory protection techniques. Modern embedded systems, on the other hand, necessitate the inclusion of memory safety methods. Such systems, such as car navigation and high-end cell phones, are unreliable because of software corruption caused by buggy programmes that are found in big embedded systems. The OS in embedded systems distributes resources to be safeguarded through middleware and applications in a different way than in a general-purpose computer system (GPC).

Keywords*: capability protection, embedded systems, temporal memory safety, RTOS, password; protection; revocation*

Introduction
A new era of fully autonomous environments is on the horizon, one in which tasks will be completed in milliseconds and with little human intervention. An increase in the utilisation of embedded systems has made this possible. Embedded computers are becoming more prevalent in our daily lives, from automobiles to mp3 players to dishwashers, and even thermostats [1]. It is these systems that drive technological advancement in all aspects of our lives. To begin with, embedded systems were found in consumer electronics such as the iPod, mp3 player, Bluetooth headphone, and PlayStation, but today they are used in a wide range of applications, from washing machines to smart phones to self-driving cars to banks to the military to space exploration. Concerns regarding embedded systems' security and privacy have exploded in recent months as the technology has gained in prominence. [2]

Memory protection methods are essential even in embedded systems. Since programmes are larger and more sophisticated in embedded systems, a single failure might have a cascading effect on the rest of the system and cause it to shut down. If one application fails, memory protection is needed to keep the problem isolated from other programmes. When utilising a memory protection programme, it is easier to debug defective programmes since illegal behaviour can be spotted. As a last step, memory protection is also necessary to safeguard embedded systems against software faults and malicious software downloads[3].

There are three types of attacks on embedded systems based on their targets.

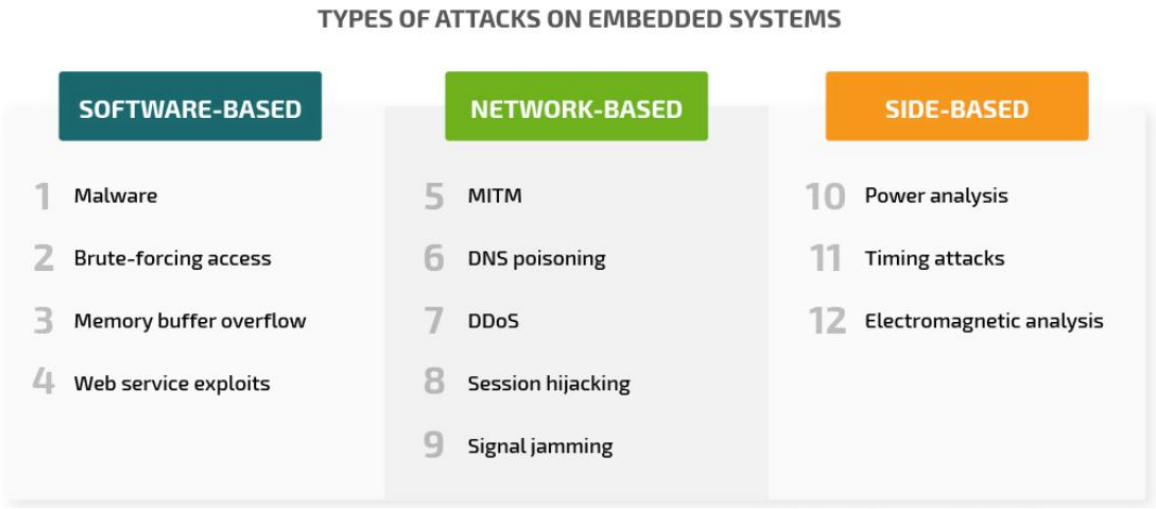
- **Software-based attacks**

The programme that controls the devices is the target of software-based assaults. Embedded systems can be taken over or their data accessed when an attack on software is successful.

The most common kind of attack is to look for flaws in software design and code, which can be carried out remotely. Hackers need not have specialised knowledge to carry out a software-based assault because they can employ common techniques like malware deployment and brute-forcing.

Attacks based on software are most common, and include:

- Malware
- Brute-forcing access
- Overflowing a memory buffer
- It was initially published at https://www.apriorit.com/[4] on exploiting web application security issues



A typical attack against embedded devices is depicted in Fig. 1.

- Network-based attacks

This type of attack takes use of flaws in the network infrastructure and can even be carried out remotely. Embedded systems are vulnerable to hacking attacks that take use of these flaws to listen in on, intercept, and modify data traffic.

Consider the following examples of network-based attacks:

- Man in the middle (MITM)
- Domain name system (DNS) poisoning
- Distributed denial of service (DDoS)
- Session hijacking
- Signal jamming
- Side-channel attacks

The goal of side-channel attacks is to exploit embedded system security weaknesses to get access to them. The most difficult and expensive sort of assault is a side-channel attack, which necessitates extensive knowledge of the target system's hardware design and physical availability. Hackers acquire information about system power consumption, electromagnetic leakage, operation time, etc. in order to carry out a side-channel attack. Thus, they may be able to figure out how a system and its connected devices work from the inside, steal cryptographic keys, or even take over the system.

Some of the more frequent side-channel assaults are listed below. [5]

- Power analysis
- Timing attacks
- Electromagnetic analysis

Review of Literature

With embedded systems in mind, attacks to alter the flow of control are discussed in [6]. Data and control flow can be separated from each other using a hardware solution. The return addresses are stored in the control flow stack, which is located in a separate memory module from the data stack. The control flow stack is protected by hardware safeguards from being accidentally or maliciously modified. As a result, return addresses on the stack are protected against being overwritten with random data because access to this stack is restricted to the call and return instructions.

Low power microcontroller memory layout is expected to be basic and static in [7]. This suggests a trade-off between hardware expense and usability to set the maximum number of unique software activities. A memory protection unit that implements a segmented view of the address space is utilised in a protection system. If an access violation is detected, the MPU acts on the enable signal of the memory and input-output devices to physically halt a data transfer. For example, segmentation is used to safeguard the small memory portions

associated with an input-output device's internal registers in a memory-mapped view. MPU hardware, in particular, has to keep an associative segment lookup table for segment descriptors, which adds complexity.

Objectives

- To learn more about how the memory protects memory
- To learn about MMU and MPU
- An investigation of the attack on embedded systems
- To examine embedded systems' need for memory protection.

Research Methodology

In the discipline of academics, methodology refers to the systematic and theoretical investigation of the procedures used to conduct research. Theoretical analysis of the methods and principles linked with a particular field of study is included. Paradigm, theoretical model and phases as well as quantitative or qualitative methodologies are common concepts included. It is necessary to do extensive study into secondary sources in order to use analytical and descriptive methods. To fully develop the textual analysis, it is necessary to consult secondary sources and do close reading analyses of a few of their writings.

Result and Discussion

In many microcontrollers, a Memory Protection Unit (MPU) provides simple hardware memory segregation. There is no page-granular translation between virtual and physical addresses in MPUs, unlike in memory management units (MMUs). MPUs, on the other hand, let you restrict access to certain areas of a physical address. Systems requiring low power consumption and predictability, as well as affordability and simplicity, are common places for MPUs. [8-10]

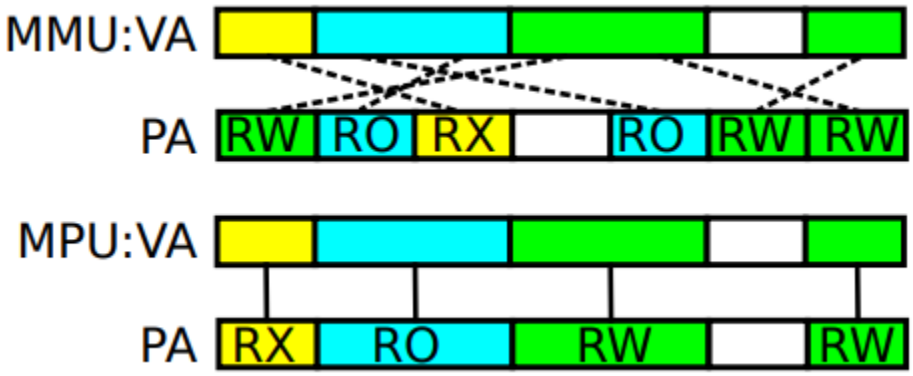
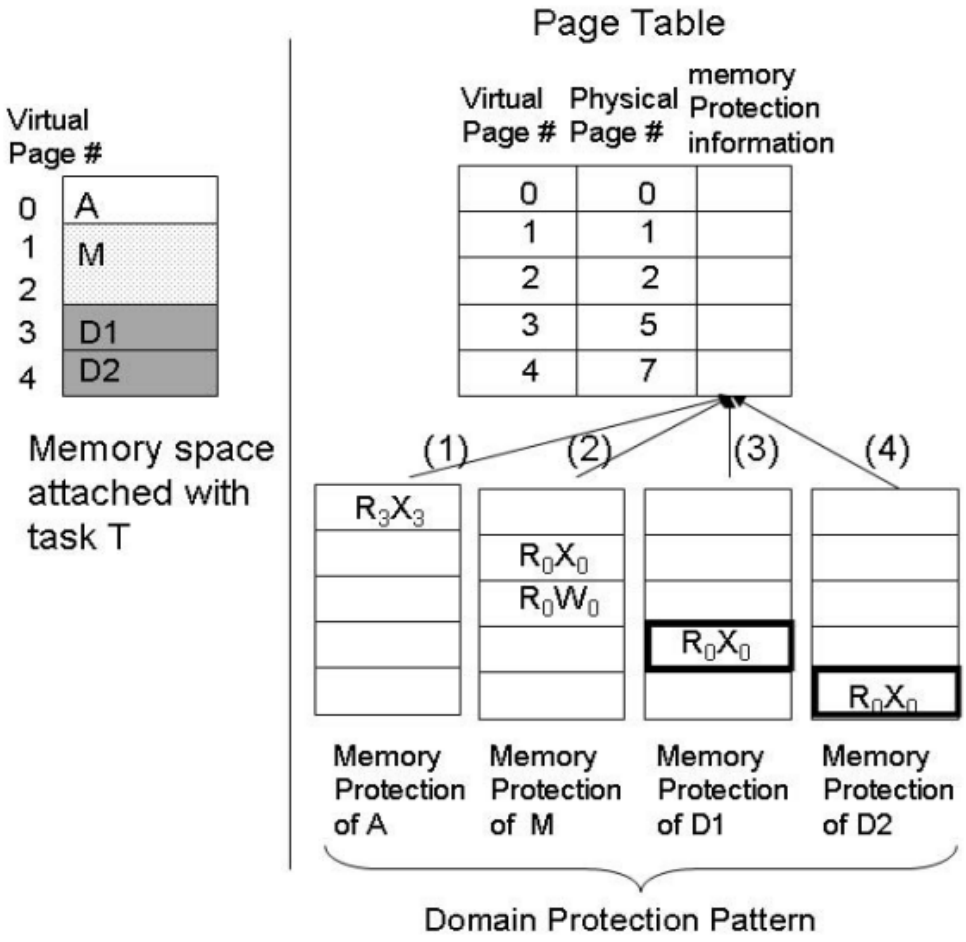


Fig. 2 shows the differences between the MMU and MPU. A dotted line indicates that MMUs provide page-based translation, whereas a vertical solid line indicates that MPUs do region-based protection. RW: read-write; RO: read-only; RX: read-only and executable.

The MPU access control state is maintained directly in registers that are programmable solely at the kernel level and have more flexible granularity control. Due to the fact that MPUs do not allow for address virtualization, all memory must be non-overlapping in order to create an Single Address Space Operating System. [11-13]

A Memory Management Unit (MMU) is responsible for implementing the above-described memory protection features (MMU). Memory access is monitored using a page table by the MMU, which performs the translation of virtual addresses to physical addresses. The virtual address and memory protection information are both stored in the page table for each page entry. [14]



Changes in memory protection mechanisms are depicted in Fig. 3

The memory protection information in Fig. 4 indicates that $P_i \in \{R, W, X\}, i = 0, 1$ with a domain denotes access in the execution mode $j(0 \leq j \leq i)$ is granted to the domain, where i=0 and i=1 indicate the kernel mode and user mode, respectively.[15]

Conclusion

In addition to making our lives easier and more comfortable, embedded devices can pose a threat to our safety. The commercial viability of new goods or the proper operation of current ones can be jeopardised by an expanding number of security threats against embedded systems and other hacker attacks. As 100% security does not exist, an attacker having enough time, resources and motivation could always break into any system. For this reason, manufacturers must protect their products against specific dangers in order to establish a balance between the expense of security implementation and the benefits gained. Multiple memory regions, nested memory areas, and the ability to guard against memory area changes are all features of the memory protection system. Because of their flexibility and generic nature, these properties are well-suited to a wide range of embedded system uses. The integration of the embedded component system and the access control mechanism leads to a more secure and safe embedded programme runtime environment. Next, we plan to create the general memory protection mechanism and test it in a variety of component usage scenarios.

References

[1] P. Koopman, "Embedded system security," IEEE, pp. 95-97, 12 July 2004.

[2] https://www.ioxtalliance.org/the-pledge

[3] https://ogi-cdn.s3.us-east-2.amazonaws.com/csis/firmware-security-best-practices-v1.1.pdf

[4] https://www.nccgroup.com/uk/our-research/security-of-things-an-implementers-guide-to-cyber-security-for-internet-of-things-devices-and-beyond/

[5] https://www.nccgroup.com/uk/our-services/cyber-security/specialist-practices/secure-development-cycle/

[6] A. Francillon, D. Perito, C. Castelluccia, "Defending embedded systems against control flow attacks," Proceedings of the First ACM Workshop on Secure Execution of Untrusted Code, Chicago, Illinois, USA, November 2009, pp. 19–26.

[7] O. Stecklina, P. Langendörfer, H. Menzel, "Design of a tailor-made memory protection unit for low power microcontrollers," Proceedings of the 8th IEEE International Symposium on Industrial Embedded Systems, Porto, Portugal, June 2013

[8] T. Azumi, M. Yamamoto, Y. Kominami, N. Takagi,H. Oyama, and H. Takada. A New Specification of Software Components for Embedded Systems. In Proc. 10th IEEE In-ternational Symposium on Object and Component-Oriented Real-Time Distributed Computing, pages 46–50, May 2007.

[9] Ken Sakamura. ITRON4.0 Specification Ver.4.00.00.TRON Association, 2002.

[10] Y. Nakamoto. Software TLB Management for Embedded Systems. Transaction on Information and Systems, Special Issues on Advanced Computer Systems, E86-D (10):2034–2039, Oct. 2003.

[11] Renesas Technology. SH7727 group Hardware Manual, Rev.5.00, Dec. 2005.

[12] T. Shinagawa, K. Kono, M. Takahashi, and T. Masuda. Ker-nel support of fine-grained proctection domains for extention components. Journal of Information Processing Society of Japan, 40(6):2596–2606, June 1999. (in Japanese)

[13] J.-H. Choi et al., "A low power TLB structure for embedded systems," Computer Architecture Letters, vol. 1, no. 1 (2002).

[14] B. Egger, S. Kim, C. Jang, J. Lee, S. L. Min, H. Shin, "Scratchpad memory management techniques for code in embedded systems without an MMU," IEEE Transactions on Computers, vol. 59, no. 8 (August 2010), pp. 1047–1062.

[15] A. Francillon, D. Perito, C. Castelluccia, "Defending embedded systems against control flow attacks," Proceedings of the First ACM Workshop on Secure Execution of Untrusted Code, Chicago, Illinois, USA, November 2009, pp. 19–26.