International Journal of Electronics and Communication Engineering (IJECE)
ISSN 2278-9901
Vol. 2, Issue 2, May 2013, 161-168
© IASET

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

A SECURED FIRMWARE UPDATE PROCEDURE TO PREVENT CROSS CHANNEL SCRIPTING ATTACK IN EMBEDDED DEVICES

M. KAMESWARAO[1] & P. BHAVYA SREE[2]

[1]Associate Professor, Department of Electronics & Computer Engineering, K. L. University, Vaddeswaram, Guntur, Andhra Pradesh, India

[2]Department of Electronics & Computer Engineering, K. L. University, Vaddeswaram, Guntur, Andhra Pradesh, India

ABSTRACT

Many embedded systems are complex, and it is often required that the firmware updates in these systems will become necessary due to bug fixes, improved functions, extensions, and parameter changes. For confidentiality it is important that these systems only accept firmware approved by the firmware producer. In this work we propose a framework that only accepts approved firmware and protects against unauthorized or vulnerable firmware updates in embedded systems. In this work the secured framework for web interface of embedded devices defines a protocol for data exchange between web interface and the firmware repository and verifies the authentication of firmware update using data authentication code.

KEYWORDS: Data Authentication Code, Cross Channel Scripting Attack, Firmware Update, Embedded Devices

INTRODUCTION

More and more devices in our modern world are equipped with a huge number of embedded systems. Today's embedded systems increasingly face the requirement for heightened security. Embedded systems typically allow downloading of updated program and data code via a boot loader. The authenticity of the software and a secure boot process must be ensured, since an increasing number of embedded devices are used in security-sensitive applications .Hence, any local or remote tampering of these devices must be prevented[1, 2]. Most contemporary embedded devices, such as wireless routers, digital cameras, and digital photo frames, have Web based management interfaces that allow an administrator to perform management tasks on the device from a Web browser connecting to the device's Web server. Many of these devices are vulnerable to Cross Site Scripting type attacks whereby some malicious JavaScript code can be injected in the Web pages stored on the device.

When such infected pages are opened by the administrator, the malicious script is executed with admin privileges, and it can potentially fully compromise the embedded device. To prevent counterfeiting or unauthorized access, software which is typically stored in reprogrammable flash memory must be updated securely. During the boot process where the software is typically signed at a secure back end server and then installed using a boot loader the system must verify the authenticity of the new firmware by checking the digital signature[3]. The new firmware must be executed by the device only if this verification is successful. This paper presents an overview of firmware modification attacks, a general strategy that is well-suited to the exploitation of embedded devices and suggests a security framework that prevents unauthorized firmware modification attacks.

The paper is organized as follows: Section 2 provides a brief overview of embedded systems. Section 3 describes the security requirements of various embedded systems. Section 4 reviews firmware modification attack. Section 5 provides an overview of the various cryptographic techniques used to provide security to embedded devices. Section 6 presents proposed security framework to prevent unauthorized firmware modification attack. Section 7 concludes with best practices in firmware modification.

EMBEDDED SYSTEMS MODEL

An embedded system is defined as: "A combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function"[4].The software that run on these systems and that provide the system’s functionality by connecting the various pieces of hardware together is called firmware. Figure 1 depicts the components most common in an embedded system [5].

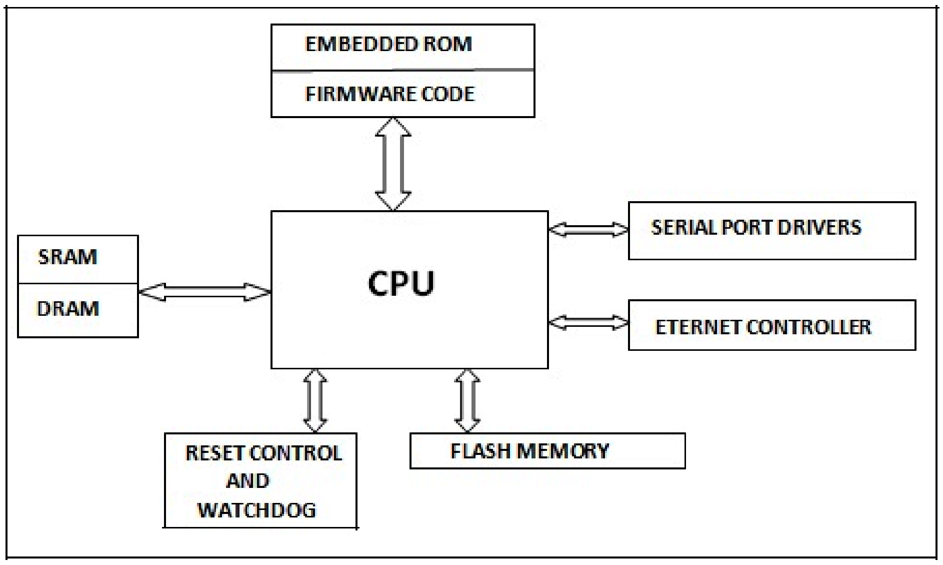

Figure 1: System Model Showing Various Parts of an Embedded System

The CPU is the very heart of the system, and is in charge of performing actions based on the instructions it is fed with. These instructions are called machine code. The CPU in an embedded system is normally referred to as either a microcontroller or a microprocessor. The system memory is used for storage of machine code and data. System memory is split into volatile and non-volatile memory. Volatile memory only retains its data while the memory is powered and is often referred to as RAM. Non-volatile memory stores the bits in such a way that it will keep the data even without power. Firmware is normally stored in non-volatile memory since it needs to be accessible on startup even after a power loss. Read-Only Memory (ROM) is a memory that is programmed during manufacture and that cannot be changed once programmed. Programmable ROM (PROM) is a more flexible type of memory that can be programmed, but only once. Erasable Programmable ROM (EPROM), on the other hand, allows for reprogramming by providing a method of completely erasing the content of the memory. This erasure is done with ultra-violet light in a device called programmer. Electrically Erasable PROM (EEPROM) and flash memory are both non-volatile memory types. EEPROM can be programmed byte by byte; flash is typically erased and programmed on a sector basis. Random Access Memory (RAM) is volatile memory that is readable and writable from the CPU. Two fundamental types of RAM exist: Static RAM (SRAM) and Dynamic RAM (DRAM). Of these, SRAM is the most interesting since it is typically used for storage of secret key material. External communication is the way in which the system communicates with the outside world, and the communication channel on which the firmware image is transferred during updates. In many embedded systems the interface to the outside world is a serial port. The serial device on an embedded system consists of two parts: a serial protocol and a physical interface. The physical interface converts voltage-signals from the CPU to signals suitable to send on the interface. Some processor chips have a portion of the Ethernet interface built in on the chip, but the most common method of providing this connectivity is by adding a separate Ethernet device to the system.

SECURITY REQUIREMENTS OF EMBEDDED SYSTEMS

Several functional security primitives have been proposed in the context of network security. These include various cryptographic algorithms used for encrypting and decrypting data, and for checking the integrity of data. The main goals of security in embedded system are confidentiality, message authentication and data integrity [6, 7, 8]. Confidentiality ensures that only authorized parties can understand a message and can be achieved through the use of ciphers, using a cryptographic algorithm. Cryptography is the science of encoding data so that it is difficult to decode it without knowing a secret key. The firmware might be stored in encrypted form and only decrypted when it is to be executed, or it might be decrypted during the firmware update process. (Data) Integrity is concerned with ensuring that a message has not been corrupted or modified while in transit. This is important since a malicious user cannot be allowed to alter the firmware originally delivered by the firmware producer.

FIRMWARE UPDATE

Even though firmware is not designed to be changed, updates in a boot loader or firmware are usually needed to correct bugs or to add new functionalities. Indeed, there are firmware's that are not correctly designed and so contain bugs which can sometimes be critical. Sometimes, the firmware is just too old and does not comply with the client desires any more. In all these situations, a new firmware is needed. There are plenty of methods for updating firmware. Traditional remote firmware updating systems rely on a boot loader application that executes on a system reset and selects the application to run or executes an application update routine [9, 10].

A remote server sends new firmware to the system to program into its local memory. In most cases, when the system update process is started, the main application is halted. The main application code is then erased and reprogrammed. There is only one copy of the main application, therefore if there were undetected errors when the new code was received and programmed, the system may not operate correctly and could stop operating until a new application is downloaded. The more classical are using the UART or the USB port. But, if the system has an Ethernet controller, the update can be made on a network and on Internet too [11]. For the mobile phones for example, the method used is called Update Over-The-Air. However, the firmware update operation is critical because if an unexpected problem occurs during the transmission and the updating process has not been designed accordingly, all the system can be blocked.

FIRMWARE MODIFICATION ATTACK

Firmware modification attacks aim to inject malware into the target embedded device. They can be carried out either as standalone attacks or as secondary attacks following initial exploitation using traditional attack vectors. Standalone firmware modification attacks manipulate firmware update features instead of exploiting flaws in the victim software. Bojinov et al [12] showed that many commercially available embedded devices with networking capabilities (e.g., wireless routers, printers, network-attached storage devices, but also modern cameras and digital photo frames) are vulnerable to a special form of Cross Site Scripting attack [13]. They called this special type of attack Cross Channel Scripting (or XCS for short). The threat comes from the fact that these embedded devices have a Web based management interface that allows an administrator to perform management tasks remotely on the device from a browser connecting to the device's Web server. This can be exploited by injecting malicious JavaScript code in the device, which is executed by the browser of the administrator when he performs management tasks on the device and opens the page that contains the injected code. Such malicious code can be injected in the device via any of its non-Web based interfaces, such as NFS or SNMP, hence the name Cross Channel Scripting. Figure 2 illustrates XCS attack.

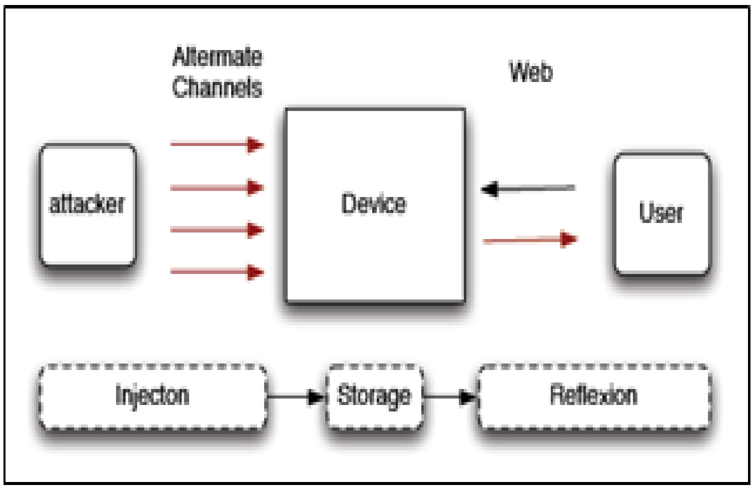

Figure 2: Cross Channel Scripting (XCS) Attack

A Cross-Channel Scripting attack comprises two steps, as shown in Figure 2. In the first step the attacker uses a non-web communication channel such as FTP or SNMP to store malicious JavaScript code on the server. In the second step, the malicious content is sent to the victim via the Web interface. XCS vulnerabilities are prevalent in embedded devices since they typically expose multiple services beyond HTTP [14]. XCS bugs are harder to detect than XSS and CSRF since they involve multiple communication channels.

SECURITY ENHANCED FIRMWARE UPDATE PROCEDURE

The purpose of the secure update system is to prevent malicious users from installing a firmware that is not approved by the manufacturer. Verifying the firmware means validating that the firmware provided for update is approved. Following are various classes of cryptographic techniques and algorithms that are used for firmware verification in this work.

Symmetric Cryptography

Symmetric ciphers can provide confidentiality, integrity and authentication [15, 16]. In a symmetric cipher, the encryption and decryption can be performed using the same secret. A key-management problem exists in that the secret key must be known to all parties that are to understand the message. The proposed procedure for firmware updates uses a trusted third party for secure exchange between firmware repository and embedded device as shown in figure 3.

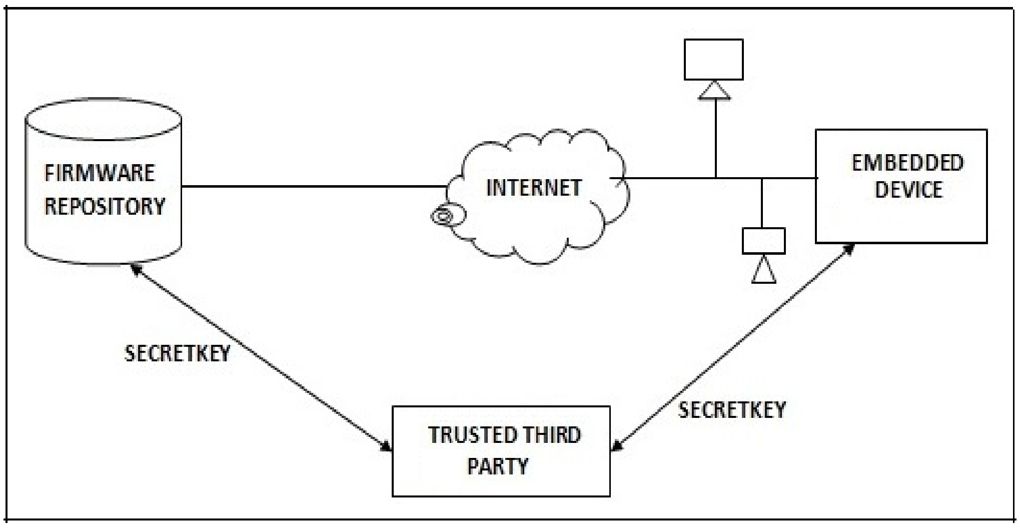

Figure 3: Secure Key Exchange Using Trusted Third Party for Firmware Update

One-Way Hash Functions

One-way hash functions, also known as message digests, cryptographic checksums and message integrity check (MIC) provide message integrity. A one-way function is a function that is easy to calculate but hard to reverse. A hash function will take a variable length input string called a pre-image, such as a firmware image, and convert it into a fixed length output string called a hash value [17, 18]. The firmware verification check using hash functions can be performed as follows: The manufacturer computes the hash value. This hash value is transferred or stored securely in the embedded system that is to be updated. When the system receives an updated firmware a hash value is computed and compared to the hash value from the manufacturer. If they match, the integrity of the firmware is verified.

Data Authentication Code (DAC)

A Data Authentication Code (DAC) provides both integrity and authentication and is a function that is based on a secret key, thus using symmetric cryptography. A DAC based on cryptographic hash functions, such as the one-way hash functions, are called keyed-hash based DACs.

Our proposed update procedure is illustrated in Figure 4

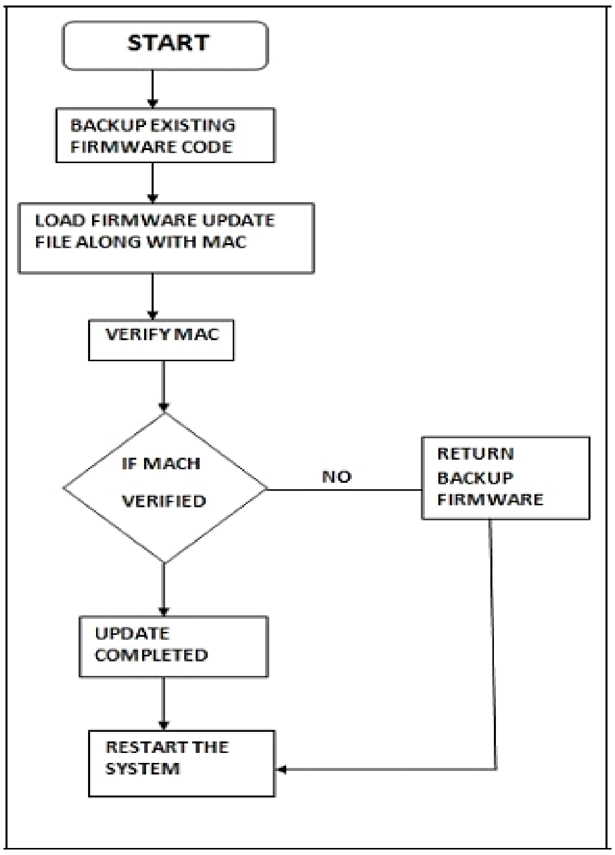

Figure 4: Security Enhanced Firmware Update Procedure

Step 1: In the first step of the procedure the system prepares for a firmware update by taking actions such as saving important state information and backup existing firmware code.

Step 2: The firmware update file is downloaded to the device. This file comprises of new firmware and encrypted DAC (Data Authentication Code) .DAC is calculated as follows:

DAC=C_k (FW)

Where C- One way Hash Function, FW –Firmware, k-Shared Secret key

Step 3: Once the update file has been loaded into the system, the validity of DAC is checked. The primary purpose of this step is to assure that only approved firmware can be installed. The update procedure is aborted if the verification fails.

The steps for verifying the DAC is as follows:

- Decrypt the DAC calculated by the firmware repository using shared secret key.
- Compute the DAC of the firmware sent in the update file and check whether it matches the DAC in the previous step.

Step 4: If the DAC computed does not match the original DAC in the firmware update file the backup firmware is reloaded else the new firmware has successfully been programmed into flash memory and the system is restarted.

CONCLUSIONS

Usually, a boot loader is built into the firmware to update the program. However, in most cases there are no mechanisms implemented to avoid downloading a manipulated program that alters the device's behavior in a manner not authorized by the manufacturer. The presented procedure is an efficient countermeasure to manipulation attacks. Such procedure can be implemented in a variety of applications such as the automotive domain, the aeronautical domain, and even the mobile phone industry. The conclusion was that firmware verification would best be done using hash functions and DAC. The system uses a symmetric key approach between firmware repository and embedded device. It is sufficient to distribute the secret key by a trusted third party between the firmware repository and the embedded device. We believe that the security of the proposed update system is good as long as the attacker doesn't have physical access to the device.

REFERENCES

1. David Brenan, "Embedded system update", http://www.techrepublic.com/article/the-flash-rom-boot-loader-performs-critical-actions/5034893, 2003.
2. Salecker Juergen," Embedded system update", http://hillside.net/europlop/ europlop2006/workshops/E2.pdf, 2006.
3. Safe upgrade of embedded systems. http://mind.be/content/Presentation_Safe-Upgrade.pdf, 2012.
4. M.Barr, "Memory types," Embedded Systems Programming, vol. May, pp.103–104, 2001.
5. E.Sutter, Embedded Systems Firmware Demystified. CMP Books, 2002.
6. S.Ball, Embedded Microprocessor Systems: Real World Design, Second Edition. Newnes, 2000
7. J.H. Chiloyan, "Firmware recovery," U.S. Patent 7 043 664, May, 2006.
8. J.Ganssle, The Firmware Handbook. Newnes, 2004.
9. L.Hars, "Secure firmware update procedure for programmable security devices," U.S. Patent 20 060 005 046, January, 2006.
10. Grossman, J., Hansen, R., Petkov, P., Rager, A., and Fogie, S. Cross site scripting attacks: XSS Exploits and defense. Syngress, Elsevier, 2007.
11. S.Fogie, J. Grossman, R. Hansen, A. Rager, and P. Petkov. XSS Exploits: Cross Site Scripting Attacks and Defense. Syngress, ISBN 1597491543, 2007.

12. H.Bojinov, E. Bursztein, and D. Boneh. XCS: cross channel scripting and its impact on web applications. Conference on Computer and Communications Security, Proceedings of the 16th ACM conference on Computer and communications security, pages 420- 431. Chicago, USA, 2009.

13. J.de Haas. (2007) Side channel attacks and countermeasures for embedded systems. [Online]. Available: https://www.blackhat.com/presentations/bh-usa-07/De_Haas/ Presentation/bh-usa-07-de_haas.pdf

14. C.Paar. (2005, January) Applied cryptography and data security. Accessed 2008-02-19. [Online]. Available: www.crypto.ruhr-uni-bochum.de/imperia/md/content/lectures/notes.pdf

15. The Keyed-Hash Message Authentication Code (HMAC), National Institute of Standards and Technology Std., 2002. [Online]. Available: http: //csrc.nist.gov/publications/fips/fips198/fips-198a.pdf

16. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996

17. B.Schneier, Applied Cryptography, Second Edition. John Wiley & Sons Inc 1996.

18. FIPS 140-2, Security requirements for cryptographic modules, May 2001,Available at http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf