JEAS

Journal of Engineering and Applied Sciences, Volume 17, Number 2, December 2020, 196-206

JOURNAL OF ENGINEERING
AND
APPLIED SCIENCES

Review of embedded systems security

Duru C.C.[1*], Azubogu A.C.O. [2], Aniedu A.N. [2]
[1]Department of Electrical and Electronic Engineering, Imo State University Owerri, Imo State.
[2]Department of Electronic and Computer Engineering, Nnamdi Azikiwe University Awka, Anambra State.
[*]Corresponding Author's E-mail: chukwuemekacduru@gmail.com

Abstract
Embedded system security has received less attention when compared with other areas of device security application partly because of the architecture and their single user nature. They have successfully found their relevance in critical aspects of the society like, healthcare, transportation, energy, economy with life threatening consequences in most cases if they are compromised. This paper provides a review of these systems and how best it should be architected. It shows their vulnerability and the importance of providing adequate security to mitigate against them. It further presents some security concerns in embedded systems and shows sequence of steps required for security implementation in such systems.

Keywords: Embedded Systems, Vulnerability, Mitigation, Security.

1. Introduction

Embedded systems are special purpose computer systems that are built to carry out specific tasks. At their core is the central processing Unit (CPU) where every other module like memory, analogue to digital converter (ADC), digital to analogue converter (DAC), timers, counters, signal conditioning and processing, interfacing standards are built around to make up a full functional system (Wu, Obeng, Wang, & Kulas, 2013). They are designed to carry out dedicated tasks while fulfilling real-time processing requirements (Massa & Barr, 2009).

Proliferation of these devices have been promoted by the growth and advancements in microprocessors, Real Time Operating Systems (RTOSs) and the recent Advanced Reduced instruction set computers Machines ARM compatible operating systems like Android (Ghafoor, Jattala, Durrani, & Tahir, 2014). Their main characteristic lies in the technologies surrounding their implementation (both digital hardware and software), and the complexity/strictness of their nonfunctional requirements (Marinov & Pavlov, 2015).

Nowadays, embedded systems are ubiquitous in different areas ranging from industrial automation, home automation, health care, transportation (Borges & Rodrigues, 2011) as a trend, it is rather quite difficult to find any application without one or more embedded system these days (Noergaard, 2005). They are known to share common OS and CPU platforms implying that any algorithm that is able to crack any of these devices can be used to compromise hundreds of different devices of a given class simultaneously.

Most developers assume that their devices are immune to attacks since they have unique features (use of flash storage and non-x86-based processors) from that of the desktop computers. Contrary to this, most embedded system lack the five essential operating system security features as stated by Stammberger & Semp (Stammberger & Semp, embedded.com, 2016) including;

- Application-kernel separation
- Memory protection domains
- Restricted code execution on the system stack
- File system access protection
- Randomization of process information

These among other things usually make most embedded systems more vulnerable than most desktop systems. Another feature that makes these devices more prone to malicious attacks is the availability of debug shells built into the system during production and the fact that most of the operating systems used on these devices are open source creating a good avenue for cryptanalysis and malicious reverse engineering of the system.

Although majority of developers are lured into the common believe that security is a thing of less concern in this field, based on the fact that embedded systems are of little interest to hackers. While this may be true, the gap is closing quite fast as no fewer than 120,000 new malware signatures designed for embedded systems are identified every week signifying that attacks on embedded systems is increasing rapidly (Stammberger & Semp, embedded.com, 2016).

The two major factors that enable adversaries to target these grades of systems include their complex nature and the fact that they are always connected to the internet with the aim of exploiting the vulnerability in these devices to steal important data or even destroy the whole system (Ali Alheeti, Ehsan, & McDonald-Maier, 2014). The number of attack on the embedded system were not as much as it is now because in the past these systems were independent this trend has changed due to the growing use of internet-connected devices (Nilsson & Larson, 2008). This and more have made the security of the embedded system, a serious problem (Clark, et al., 2013).

Most Internet of Things solutions especially in the Industrial parlance will have embedded systems as their cornerstone. Owing to this, key players in the sector especially in the areas of hardware and software development are aiming to bring these transformations into their products to take advantage of the increasing IoT deployment. The areas include Real Time Operating Systems (RTOS), microprocessors and microcontrollers, memory footprints, networking, open source communication etc.

2.0 Security Concerns in Embedded Systems

It is estimated that the market for the overall embedded system will grow with a compound annual growth rate (CAGR) of 22.5% to reach $226 billion through 2020 (MindCommerce, Embedded Systems and the Internet of Things (IoT), 2015). This growth brings about the need for flexibility and function consolidation especially in terms of security. Since these systems play crucial roles in our day to day activities with increased complexity, network and permitting functional extensibility through their respective softwares, their security should be a core concern. Implementing security at software level alone has shown to cause lots of overheads (Wang, et al., 2018).

Complexity, extensibility and connectivity are the major factors that hamper the management of software error control (Kocher, Lee, McGraw, & Raghunathan, 2004). This security flaw is used by adversaries to gain logical control of the device by leveraging on any programming error in firmware, operating system (OS) and applications running on these embedded systems. OS functionalities are performed by most the firmware of most embedded systems owing to the fact that most embedded systems do not have separate operating systems. Adversaries can leverage on this by sending fake inputs or packets that are wrongly processed by the software causing buffer overflow and subsequently hijacking the control sequence (Cai & Zuhairi, 2017).

Security is not always strictly enforced in these systems making them vulnerable to a plethora of known and unknown threats (Ott & Mahapatra, 2019). This problem gets more compounded due to the resource constrained nature of these grade of devices making it difficult to implement current and ubiquitous mitigation technique like address space layout, control flow integrity, randomization, memory permission, against security breaches (Wetzels, 2017).

Embedded system security requirement vary according to their unique operation and the application area. Notwithstanding, these system should be able to carry out its design goals, prevent attacks and operate with some level of resilience when under attack (Vai, et al., 2015). Unfortunately, most systems see data integrity, confidentiality and authentication as the basic security requirement and are never strictly enforced. It is also a requirement for these systems to have secure storage to handle user information and data, secure network access, availability and tamper resistance (Kocher, Lee, McGraw, & Raghunathan, 2004).

Violation of authentication, integrity and confidentiality of the information being handled by these systems may constitute a huge threat to life and industrial espionage. For instance, malicious access to pacemaker or the control of a nuclear reactor or car driving system may threaten human life and health respectively. Their design goal due to this is limited to providing as much hardware and software resources as the manufacturer deems fit for the specific task they would be performing throughout their lifecycle. Also, since most of these devices are autonomous having the responsibility of authentication and malicious modification prevention, proper care should be taken during manufacturing and lifetime of the system to provide these devices with enough resources to handle security and privacy issues.

2.1 Process Management

A thread of processes is usually associated with different users in a typical operating system. The processes initiate system calls, request I/O systems and memory resources. Most processes could belong to system with root privilege hence if compromised would threaten the overall functionality of the system. Restricted Privilege subsystem has been proposed by Shukla (Shukla, Singh, Choi, Kwon, & Hahm, 2013). They have based their work by copying the Linux OS inheriting the fact that the highest privilege system on the Linux system is the root user and that in most cases group IDs belonging to the same User ID share the same privileges. This implies that any member of a group belonging to a process with root privilege will also have the same privilege as the root user. The security inefficiency here is that if a member of the group is compromised, then the attacker automatically gains root access that can be used to breach the entire system. Shukla argued that new processes should not be given same root privilege as the parent application. Since complete removal of root user privileges is not possible, restricted privileges is proposed in their work for new processes even though they may belong to a root user.

3.0 Security Issues in Real Time Embedded Systems

Real-time embedded systems are designed to be in constant communication with their environment providing real time data collection and analysis. They are fundamental part of many sensor networks that capture changes occurring in their environments and employing algorithms to interpret such changes to enable the actuators give proper reaction or compensation to such change (Jingjing, 2011).

Adversaries may also exploit limited battery lives of these devices by making numerous request to keep the device active at all times hence, causing battery drainage and subsequently reducing the system lifetime (Martin, Thomas; Hsiao, Michael; Ha, Dong; Krish, Jayan, 2005). These requests cause excessive workload on the processor which in turn affects the temperature of the device. A good defense against it is to take control of the temperature regulation scheme such that an irregular increase in temperature would activate the device overheat security measures thereby temporarily halting the activities of the malicious subjects. These measures could be in the form of intermittently shutting down the device or forcing a restart (Dadvar & Skadron, 2005).

The kind of application to be run on any embedded system and the services that would be offered determine the security requirement of such system (Galbraith, 2012)**.** Analyzing these different application areas and looking into their security needs demands the following security requirement for embedded systems (Dhillon & Kalra, 2016).

i. Confidentiality: This refers to ability to protect unwarranted exposure of information to an adversary.
ii. Integrity: Preventing adversary from unauthorized modification of information and data in the system.
iii. Availability: Information must be available to be accessed by authorized users any time it’s needed.
iv. Authentication: Users associated with any particular system must be validated prior to access approval.
v. Non-repudiation: This feature prevents participants from denying processes or transactions they initiated.
vi. Dependability: Systems resources, quality of service, real time data collection and analysis must be dependable in a good embedded system design.
vii. Privacy: Users personal information must be protected from access by adversaries
viii. Safety: An embedded system must be able to carry out its operations without harm to the user or environment.

The categories of different types of attack on embedded systems are listed as follows (Papp, Ma, & Buttyan, 2015)

i. Control hijacking attack: An adversary can gain control of communication between two legitimate users by altering the normal program control flow to a malicious one generated by it.
ii. Reverse engineering attack: An adversary is able to access sensitive information stored in the firmwares of these embedded system while they are online or offline. This enables him to exploit security loophole to compromise user data and passwords stored in these firmware.

iii. Malware attack: Malicious codes can by launched by adversaries on embedded systems thereby changing their behavior subsequently leading to severe consequences for time critical application.
iv. Injecting crafted packets: The attack allows the adversary to inject fake packet into legitimate packet stream between authorized communicating parties. A typical instance is the replay attack. Such malicious packet can be used to gain access to the system.
v. Eavesdropping attack: Communication between two legitimate parties can be silently listened into. Information extracted from this act could be used to compromise the privacy of the entire systems.
vi. Brute-force attack: Here, the attacker attempts what is referred to as 'hit and trial' attack method with the aim of extracting sensitive information like user IDs and passwords if there is a successful breach.

Denial of Service: The adversary tends to overwhelm the device with data and system resource request call that it becomes unavailable to service request from other legitimate users.

3.1 Bottom-Top Security Implementation in Embedded Systems

Embedded systems require a multilevel approach to creating effective protection from the time of power-up, entrusted connection setting and binding to a solid authentication system for attack resistance. The following bottom-top approach to security should be employed to achieve a secure system for critical applications;

a. Secure Booting: For embedded systems security, this step is critical and is seen as a necessary part of embedded systems anti-malware fortress (Padoin, The whys and hows of secure boot, 2017).
The secure boot process is a vital first step in securing any embedded system, a necessary part of the application's anti-malware fortress. Boot time security like insuring that the right firmware is loaded is very important in these embedded devices. During this period, authenticity and integrity of the software is carried out.

Secure booting also refers to the process whereby operating system (OS) boot images and codes are authenticated against the hardware before they are allowed to be used in the boot process (Padoin, The whys and hows of secure boot, 2017). The hardware for secure boot system is implemented in such a way that only validated codes from the trusted security credentials are allowed this guarantees booting of intended OS and not a maliciously tempered version of it (Rashmi & Karthikeyan, 2018).

The ability to attest, authenticate and authorize are the fundamental building blocks of system confidentiality and integrity to achieve this, an inherently trusted root is required "root-of-trust" that verifies the authenticity of the BIOS (Basic Input Output System) to know if it has been altered before passing on control to the next component. This process goes on till all system components are authenticated. The guidelines for this were published by the National Institute of Standards and Technology (NIST) (Cooper, Polk, Regenscheid, & Souppaya, 2011).

A typical instance of commercial boot-time security is the Trusted Platform Module (TPM), which is now seen as the standard of choice for implementing trust in computing systems (Fuchs & Schreiner, 2009). TPM was designed for PC and Servers as a result have been modified to MTM (Mobile Trusted Module) adding series of new command and modules for embedded and mobile platforms. This increases the trust in embedded system computing and reduces risk associated with such systems (Kai, Xin, & Guo, 2012). A good advantage of implementing secure boot is to restrict the end users from running custom ROMs that may compromise the overall system. Secure boot is also usually implemented in situations where third party bootloader is not permitted for a particular device. Like in IP camera where a compromise in the bootloader data can compromise the stored visuals which can in turn be used in blackmailing the affected customer.

b. Access Control: Access control which is an interaction between a system and entities demanding access to its resources is an important part of most embedded systems. It detects and identifies the presence of a user, uniquely classifies it with variety of authentication indices, logs this action and grant access (Sruthy & George, 2017).

All access control schemes must have policies referred to as access control policies that serve as their building block. These policies have the following indices; subject, object, action and sign (Asija &

Nallusamy, 2017). The user is termed the subject. The object is the resource that is being requested, the action denotes the kind of operations to be carried out on the object such as read, write, delete, modify etc and the sign determines if the action is permitted for the particular subject or not (Samarati & Capitani, 2014)

One major feature of embedded systems is that they are always-online devices hence, trade-offs are usually made between the complexity of authentication and access control schemes to be implemented which goes a long way in deemphasizing the need for strict security in these systems. A few requirements for a secure and manageable access control scheme for embedded systems are listed thus (Naedele, 2006);

i. Access revocation: Options to change or revoke access rights of a user to an embedded device should also be integrated in these systems during manufacturing.
ii. Centralized user management: A centralized user and access rights management system should be integrated into this system to help in access right modification and revocation without reconfiguring the whole device.
iii. Individual accountability: There should be a means of holding users accountable to access made to these devices meaning that shared or group credentials or should be eliminated.
iv. Protocol security: This is an important feature for the security of transmitted credentials. The implemented protocols on these devices should be able to guard against modification of these credentials.
v. Session protection: Since most of these devices have remote accesses enabled in them, there should be ways of ensuring the security of the sessions created by the protocols during the access period to ensure confidentiality and privacy of transmitted data.

Since embedded systems contain a plethora of resources such as; communication, files, memory, processes and so many other interconnecting devices, a critical design goal is to ensure that applications are able to access resources they need and are denied access to resources they don’t need (Kleidermacher & Kleidermacher, How to Article, 2013). Majority of security problems encountered today in this field are actually caused by poor access control architecture and/or implementation within the operating system or poor implementation of the access control facilities by the embedded system designer (Iskhakov, Shelupanov, & Mitse, 2018). The National Institute of Standards and Technology have also stipulated some primitives for any access control design which must include the following or more; Subject, Object, Action, Capability and Privileges (Hu & Scarfone, 2012).

Access control can be divided into two classes namely; Discretionary Access Control (DAC) (Sandhu & Samarati, 1994) and Mandatory Access Control (MAC) (Sandhu, 1993). Both play a vital role in secure embedded system design (EDN, Embedded Systems Security – Part 2: Access control and capabilities, 2013). These models have been discovered to have a number of flaws which led to the proposal of some other modifications like Role-Based Access Control (RBAC), Attribute Based Access Control (ABAC) and Risk Based Access Control (R-BAC) (Younis, Kifayat, & Merab, 2015).

The DAC lacks control over the information flow in the system because this scheme allows information to be copied from one object to the other. For the MAC, the problem of information flow control is solved by classifying subjects and objects according to their security hierarchy. Here requests made from a subject for access to a particular object are validated once the relationship between them is satisfied. MAC has a drawback which is the inability to change the assigned privilege level (Dundua & Rukhaia, 2019).

A typical instant of DAC is the UNIX file system where a process or thread has the sole discretion to modify the permissions and access levels to files that belongs to it thereby permitting or denying access to the file to other processes. A more critical approach to access control is the MAC system which is managed by the system and can’t be modified by processes or users. Access control here is provided by the kernel and cannot be bypassed by an application code. There are more guarantees when using MAC because the effectiveness of DAC is solely dependent on the credibility of the applications using them.

For a stronger embedded system security, both MAC and DAC should be combined to provide formidable security architecture for the overall system. For instance, mandatory access control system can be used to divide system resources into different security domains in such a way that no application can access data in a domain higher than it hence, preventing unauthorized access and modification of data. While discretionary access control system can be employed to provide more dynamic sharing of resources in the same security domain.

Access control should be implemented based on the principles of least privilege where the minimal access is granted to parties in a particular security domain. This allows for minimal effect to data once there is a malicious breach which will affect only information that the particular node or system has access to and not the whole data in that security domain.

c. Device Authentication: Authentication is an important feature of any secure system. Authentication guarantees that the right user is granted access or authorization to the system resources. A good analogy is when accessing cooperate network where a username and password is required. It is the responsibly of the network server to authenticate the users input like login ID and Password and grant access to the network resources depending on the outcome of the authentication process. With authentication, Login ID can also be grouped in classes thereby giving different IDs to different users according to their roles or status. IDs here determine the extent and part of the system the user is able to access after authorization.

   Most embedded systems employ data encryption (cryptography), near-field communication for key distribution purposes and a key infrastructure to provide effective device authentication. These mechanisms must be chosen in such a way that they can be accommodated by the high computational constraint nature of these devices and also within users' expectation (Romann & Salomon, 2014). Users of embedded devices expect immediate systems response hence; care should be taken in choosing the encryption schemes implemented on these devices.

   Hashing algorithms could also be used to encrypt data stored in embedded devices as a major requirement for information security and device authentication. A hash can be seen as a one-way function that inputs message of an arbitrary length and converts them to hash values H(m) by employing some internal techniques (Menezes, van Oorschot, & Vanstone, 2001). System-on-Chip (SoC) solutions have been employed in the implementation of cryptographic schemes on embedded systems. These SoC consist of embedded CPU and on-chip bus, memories, controllers and dedicated coprocessor to accelerate the execution time of these cryptographic algorithms (Lu, Han, Zeng, Li, Mai, & Zhao, 2008).

   It is important guarantee that the right device is authenticated and that they have not been compromised since most embedded devices are known to be autonomous and do not require operators to enter any registration prior to authorization to the overall network. The hash of the device requesting access is checked against the original hash stored in the network prior to authorization. This can go a long way in detecting compromised devices and access would subsequently be denied to these devices which can be further isolated from the overall network.

d. Firewall: They are the first defense strategy deployed at edge of the network or access points for protecting network and server resources from malicious access (Salah, Elbadawi, & Boutaba, 2012). These tools can be hardware or software based and the protected system can be a typical PC, network equipment or embedded device.

   There are different variants of firewalls which are based on the network resource under consideration. These variants are (Krit & Haimoud, 2017) ;

   i. Packet filtering firewall also referred to as network or stateless firewall. Being stateless, it treats each packet as an individual packet by looking at each incoming packet and allowing or rejecting it based on some laid down features. These features could be the type of protocol (IP, TCP and UDP), source or destination IP address, source and destination port, and traffic direction.
   ii. Circuit-level gateways: This takes leverage of the handshaking process of TCP to determine whether a session is being initialized by a legitimate user or not network access is granted based on the outcome of this check.
   iii. Stateful filters: It keeps an instance of all connection in its cache and compares this with a new connection to know if it's a part of an existing connection, a start of a new connection or an

invalid request. Stateless filter firewalls take note of the different components of a TCP connection such as SYN and ACK bits to determine the legitimacy of a request.

iv. Application Layer Firewalls: Also referred to ac application proxy firewall filters messages at the application layer. Operating at the application latter means that is checks data from different application to determine if they can be malicious hence blocking them. By combining the features of both packet filtering firewalls and circuit-level firewalls they provide improved data security at the expense of network performance.

v. Multilayer inspection firewalls: Stateful multilayer Inspection Firewall is a combination of all the firewalls that we have studied till now hence, they can filter packets at the network layer using ACL (Access Control List), verify session for legitimacy on session layers and can equally authenticate packet on the application layer (ITA, Firewall and types, 2019). Algorithms and complex security schemes that are protocol dependent can also be implemented in this type of firewall making it more secure than the previously discussed types.

Failure to integrate firewall into embedded systems is tantamount to leaving ones door unlocked hence, creating high vulnerability to attacks. A compromised device can even be used to take down the overall network. For business setting, these compromised devices can be used to delete customer's profiles, product specification and recipe and even supply and production lines thereby compromising a brand name or business's integrity.

e. Patches and Updates: Software patches and updates are avenues device manufactures employ in closing up security loopholes discovered during the lifetime of their systems firmware. In embedded systems, firmwares are dedicated softwares that are found in the Read Only Memory that oversee the device hardware. A common update model allows the bootloader to apply the same mechanism used in the boot sequence process enabling it to flash and validate a new firmware image (Falas, Konstantinou, & Michael, 2019). Higher operations like device initialization, basic functionality control and execution of compiled binary programs are supported by the firmware. The lowest abstraction layer accessible by the programmer on an embedded device is the firmware hence; malicious access could bring the entire system to a halt (Konstantinou, Keliris, & Maniatakos, 2016).

Poor update or patches mechanism can be exploited by adversaries to gain full access to the device. These patches are usually provided online via device manufacturers' web server making them vulnerable to web crawlers that may aggregate critical equipment information for malicious purposes (Costin, Zaddach, & Francillon, 2014).

It is always recommended for there to be proper checks on the validity of the updates and their source. Crypto-bootloaders were invented to aid in the security of firmware updating these systems have inherent vulnerabilities in their design making them prone to both invasive and non-invasive attacks (Konstantinou & Maniatakos, 2019).

Security is one of the major considerations when developing a device update framework, The software update mechanism should be built in such a way that malicious parties cannot truncate the update process and use the avenue to install their own files to the device or modify its software.

4.0 Conclusion

Embedded system security is quite critical due to the numerous critical areas of our lives they find their application. It is an encompassing process that requires bottom top approach for proper implementation. At the lowest level is to ensure that the correct boot data is loaded and access to important system files properly managed. Authentication of users (devices) prior to network and data access is also performed in real-time during the lifetime of these systems. This helps to detect compromised devices and prevent or deny access to the network or data to that device. At the higher level is the design of firewalls to detect and prevent unwanted packets which could be as a result of malicious activities. Proper firewall implementation as we can see can prevent the devices from unknowingly accessing problematic services, unauthorized traffic and serve as a good security audit point for embedded systems. Finally, patches and updates should be provided for these systems throughout their lifetime. This is important if security loopholes or vulnerability is to be removed once the devices have been deployed to the fields.

References

Ali Alheeti, K. M., Ehsan, S., & McDonald-Maier, K. D. 2014. An Assessment of Recent Attacks on Specific Embedded Systems. *Fifth International Conference on Emerging Security Technologies* (pp. 88 - 93). Alcala de Henares, Spain: IEEE.

Asija, R., & Nallusamy, R. 2017. Security and Complexity Analysis of User and Data based Access Control (UDBAC) Model. *International Conference on Current Trends in Computer, Electrical, Electronics and Communication (ICCTCEEC-2017)* (pp. 358-366). Mysore, India: IEEE.

Borges, R. W., & Rodrigues, E. L. 2011. Embedded System Design: An Overview of Brazilian Development. *IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications Workshops* (pp. 141 - 146). Busan, South Korea: IEEE.

Cai, L. Z., & Zuhairi, M. F. 2017. Security Challenges for Open Embedded Systems. *International Conference on Engineering Technology and Technopreneurship (ICE2T)* (pp. 1-6). Kuala Lumpur, Malaysia: IEEE.

Clark, S. S., Ransford, B., Rahmati, A., Guineau, S., Sorber, J., Fu, K., et al. 2013. WattsUpDoc: Power Side Channels to Nonintrusively Discover Untargeted Malware on Embedded Medical Devices. *USENIX Workshop on Health Information Technologies.* Washington, D.C.: USENIX.

Cooper, D., Polk, W., Regenscheid, A., & Souppaya, M. 2011. *BIOS Protection Guidelines.* Retrieved 02 08, 202, from SP 800-147: https://csrc.nist.gov/publications/detail/sp/800-147/final

Costin, A., Zaddach, J., & Francillon, A. 2014. A large-scale analysis of the security of embedded. *23rd USENIX Security Symposium* (pp. 95 - 110). San Diego, CA: USENIX.

Dadvar, P., & Skadron, K. 2005. Potential Thermal Security Risks. *IEEE SEMI-THERM Symposium* .

Dhillon, P. K., & Kalra, S. 2016. Elliptic Curve Cryptography for Real Time Embedded Systems in IoT Networks. *5th International Conference on Wireless Networks and Embedded Systems (WECON)* (pp. 1-6). Rajpura, India: IEEE.

Dundua, B., & Rukhaia, M. 2019. Towards Integrating Attribute-Based Access Control into Ontologies. *IEEE 2nd Ukraine Conference on Electrical and Computer Engineering* (pp. 1052-1056). Lviv, Ukraine: IEEE.

EDN. 2013. *Embedded Systems Security – Part 2: Access control and capabilities*. Retrieved 02 13, 2020, from edn.com: https://www.edn.com/embedded-systems-security-part-2-access-control-and-capabilities/

Falas, S., Konstantinou, C., & Michael, M. K. 2019. A Hardware-based Framework for Secure Firmware Updates on Embedded Systems. *International Conference on Very Large Scale Integration* (pp. 198 - 203). Cuzco, Peru, Peru: IEEE.

Fuchs, A., & Schreiner, F. 2009. *How to Use the TPM: A Guide to Hardware-Based Endpoint Security.* Retrieved 02 09, 2020, from Trusted computing group: http://www.trustedcomputinggroup.org/resources/how_to_use_the_tpm_a_guide_to_hardwarebased_endpoint_security

Galbraith, S. D. 2012. *Mathematics of public key cryptography.* Cambridge: Cambridge University Press.

Ghafoor, I., Jattala, I., Durrani, S., & Tahir, C. M. 2014. Analysis of OpenSSL Heartbleed vulnerability for embedded systems. *IEEE International Multi Topic Conference 2014.* Karachi, Pakistan: IEEE.

Hu, V. C., & Scarfone, K. 2012. *Guidelines for Access Control System Evaluation Metrics.* Retrieved 02 13, 2020, from National Institute of Standards and Technology: http://csrc.nist.gov/publications/nistir/ir7874/nistir7874.pdf.

Iskhakov, S., Shelupanov, A., & Mitse, A. 2018. Internet of Things: Security of Embedded Devices. *3rd Russian-Pacific Conference on Computer Technology and Applications (RPC)* (p. 6). Vladivostok, Russia: IEEE.

ITA, T. 2019. *Firewall and types*. Retrieved 02 08, 2020, from Cisco Community: https://community.cisco.com/t5/security-documents/firewall-and-types/ta-p/3112038

Jingjing, M. 2011. Analysis of Embedded Real-Time System. *Communications in Computer and Information Science , 215,* 429–433.

Kai, T., Xin, X., & Guo, C. 2012. The Secure Boot of Embedded System Based on Mobile Trusted Module. *Second International Conference on Intelligent System Design and Engineering Application* (pp. 1331 - 1334). Sanya, Hainan, China: IEEE.

Kleidermacher, D., & Kleidermacher, M. 2013. *How to Article*. Retrieved 07 09, 2019, from EDN Network: https://www.edn.com/design/systems-design/4406717/Embedded-Systems-Security---Part-2--Access-control-and-capabilities.

Kocher, P., Lee, R., McGraw, G., & Raghunathan, A. 2004. Security as a new dimension in embedded system design. *41st annual Design Automation Conference* (pp. 753-760). San Diego, CA, USA: Design and Automation Conference.

Kocher, P., Lee, R., McGraw, G., Raghunathan, A., & Ravi, ,. S. 2004. Security as a new dimension in embedded system design. *41st Annual Design Automation Conference* (pp. 753-760). San Diego, CA, USA: Association for Computing Machinery.

Konstantinou, C., & Maniatakos, M. 2019. Hardware-Layer Intelligence Collection for Smart Grid Embedded Systems. *Journal of Hardware and Systems Security , 3,* 1-15.

Konstantinou, C., Keliris, A., & Maniatakos, M. 2016. Taxonomy of Firmware Trojans in Smart Grid Devices. *EEE Power and Energy Society General Meeting (PESGM)* (pp. 1 - 5). Boston, MA, USA: IEEE.

Krit, S.-d., & Haimoud, E. 2017. Overview of firewalls: Types and policies: Managing windows embedded firewall programmatically. *International Conference on Engineering & MIS (ICEMIS)* (pp. 1-7). Monastir, Tunisia: IEEE.

Lu, R., Han, J., Zeng, X., Li, Q., Mai, L., & Zhao, J. 2008. A Low-Cost Cryptographic Processor for Security Embedded System. *Asia and South Pacific Design Automation Conference* (pp. 113-114). Seoul, South Korea: IEEE.

Marinov, M. T., & Pavlov, T. M. 2015. Rule-based decision support tools for injection moulding. *KES Journal* , 97 - 107.

Martin, Thomas; Hsiao, Michael; Ha, Dong; Krish, Jayan. 2005. Denial-of-Service Attacks on Battery-powered Mobile Computers. *3rd IEEE Conference on Pervasive Computing and Communications Workshops.* Kauai Island, HI, USA: IEEE.

Massa, A., & Barr, M. 2009. *Programming Embedded Systems, 2nd Edition* . O'Reilly Media : Sebastopol, California, United States.

Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. 2001. *Handbook of Applied Cryptography.* CRC Press.

MindCommerce. 2015. *Embedded Systems and the Internet of Things (IoT)*. Retrieved July 08, 2019, from MARKET RESEARCH BLOG: https://blog.marketresearch.com/embedded-systems-and-the-internet-of-things-iot

Naedele, M. 2006. An Access Control Protocol for Embedded Devices. *4th IEEE International Conference on Industrial Informatics* (pp. 565-569). Singapore, Singapore: IEEE.

Nilsson, D. K., & Larson, U. E. 2008. Secure Firmware Updates over the Air in Intelligent Vehicles. *ICC Workshops - 2008 IEEE International Conference on Communications Workshops* (pp. 380-384). Beijing, China: IEEE.

Noergaard, T. 2005. *Embedded Systems Architecture - A Comprehensive Guide for Engineers and Programmers.* Oxford, UK: Elsevier.

Ott, K., & Mahapatra, R. 2019. Continuous Authentication of Embedded Software. *18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering* (pp. 128-135). Rotorua, New Zealand, New Zealand: IEEE.

Padoin, N. 2017. *The whys and hows of secure boot*. Retrieved 07 09, 2019, from embedded.com: https://www.embedded.com/design/safety-and-security/4458717/The-whys-and-hows-of-secure-boot

Papp, D., Ma, Z., & Buttyan, L. 2015. Embedded Systems Security: Threats, Vulnerabilities, and Attack Taxonomy. *13th Annual Conference on Privacy, Security and Trust (PST). 1*, pp. 145-152. Izmir, Turkey: IEEE.

Rashmi, R., & Karthikeyan, A. 2018. Secure boot of Embedded Applications – A Review. *2nd International conference on Electronics, Communication and Aerospace Technology (ICECA 2018)* (pp. 291-298). Coimbatore, India: IEEE.

Romann, R., & Salomon, R. 2014. Salted Hashes for Message Authentication – Proof of concept on Tiny Embedded Systems. *IEEE Symposium on Intelligent Embedded Systems (IES)* (pp. 42-46). Orlando, FL, USA: IEEE.

Salah, K., Elbadawi, K., & Boutaba, R. 2012. Performance Modeling and Analysis of Network Firewalls. *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT , 9*, 12 - 21.

Samarati, P. D., & Capitani, S. D. 2014. *Healthcare Security.* Retrieved 02 13, 2020, from cisco.com: https://www.cisco.com/c/dam/global/en_ca/solutions/strategy/healthcare/assets/docs/health_security_impgd.pdf

Sandhu, R. S. 1993. Lattice-Based Access Control Models. *IEEE Computer Society Press* , 9-19.

Sandhu, R. S., & Samarati, P. 1994. Access control: Principles and practice. *IEEE Communications Magazine* .

Shukla, H., Singh, V., Choi, Y.-H., Kwon, J., & Hahm, C.-h. 2013. Enhance OS Security by restricting privileges of vulnerable application. *IEEE 2nd Global Conference on Consumer Electronics (GCCE)* (pp. 201-211). Tokyo, Japan: IEEE.

Sruthy, S., & George, S. N. 2017. WiFi enabled home security surveillance system using Raspberry Pi and IoT module. *IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)* (pp. 244 - 249). Kollam, India: IEEE.

Stammberger, K., & Semp, M. 2016. *embedded.com*. Retrieved 07 10, 2019, from Embedded systems: https://www.embedded.com/design/prototyping-and-development/4008289/Turning-up-the-heat-on-hackers-with-embedded-firewalls

Vai, M., Nahill, B., Kramer, J., Geis, M., Utin, D., Whelihan, D., et al. 2015. Secure Architecture for Embedded Systems. *IEEE High Performance Extreme Computing Conference (HPEC)* (pp. 1 - 5). Waltham, MA, USA: IEEE.

Wang, X., Xu, B., Wang, W., Zhang, Z., Zhang, X., Hao, Q., et al. 2018. An Architectural-Enhanced Secure Design in Embedded System. *IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)* (pp. 100-103). Beijing, China, China: IEEE.

Wetzels, J. 2017. *Identifying & addressing challenges in embedded binary security.* Eindhoven, Netherlands: Eindhoven University of Technology.

Wu, X., Obeng, M., Wang, J., & Kulas, D. 2013. A survey of techniques to add audio module to embedded systems. *Proceedings of IEEE Southeastcon* (pp. 1 - 5). Jacksonville, FL, USA: IEEE.

Younis, Y. A., Kifayat, K., & Merab, M. 2015. A Novel Evaluation Criteria to Cloud Based Access Control Models. *11th International Conference on Innovations in Information Technology (IIT)* (pp. 68-73). Dubai: IEEE.