RECIMA21 - REVISTA CIENTÍFICA MULTIDISCIPLINAR
ISSN 2675-6218

SECURITY STANDARDS FOR LOW-END IOT DEVICES: A COMPARATIVE REVIEW

PADRÕES DE SEGURANÇA PARA DISPOSITIVOS IOT LOW-END: UMA REVISÃO COMPARATIVA

ESTÁNDARES DE SEGURIDAD PARA DISPOSITIVOS IOT DE GAMA BAJA: UNA REVISIÓN COMPARATIVA

Rodrigo Roner Tertulino da Silva[1], Nuno Antunes[2]

e3122437
https://doi.org/10.47820/recima21.v3i12.2437

RECEIVED: 11/14/2022 APPROVED: 12/15/2022 PUBLISHED: 12/22/2022

ABSTRACT

The Internet of things allows people and objects to be connected anytime, anywhere, with any object to anyone, using any path/network and any service. Thus, it leads to a challenging heterogeneity of components and networks. Different operating systems were developed for low-end IoT devices with stringent requirements mainly imposed by the low ability to process and store information compared to a conventional machine. Hence, the OS should be able to perform tasks as efficiently as possible. In heterogeneous networks, as in the case of IoT, it is more complex to guarantee the security and privacy of systems that are part of this ecosystem. The core functionality of IoT is based on exchanging information between hundreds or even millions of objects with the Internet. This work performs a comparative review of the leading security features available in low-end IoT-oriented OS, including Contiki, RIOT-OS, TinyOS, and FreeRTOS.

KEYWORDS: IoT. Security. Contiki. TinyOS. RIOT-OS. FreeRTOS.

RESUMO

A Internet das coisas permite que pessoas e objetos estejam conectados a qualquer momento, em qualquer lugar, com qualquer objeto a qualquer pessoa, usando qualquer caminho/rede e qualquer serviço. Assim, leva a uma heterogeneidade desafiadora de componentes e redes. Diferentes sistemas operacionais foram desenvolvidos para dispositivos IoT de baixo custo com requisitos rigorosos impostos principalmente pela baixa capacidade de processar e armazenar informações em comparação com uma máquina convencional. Consequentemente, o sistema operacional deve ser capaz de executar tarefas da forma mais eficiente possível. Em redes heterogêneas, como no caso da IoT, é mais complexo garantir a segurança e a privacidade dos sistemas que fazem parte desse ecossistema. A funcionalidade principal da IoT é baseada na troca de informações entre centenas ou até milhões de objetos com a Internet. Este trabalho realiza uma revisão comparativa dos principais recursos de segurança disponíveis em sistemas operacionais de baixo custo orientados para IoT, incluindo Contiki, RIOT-OS, TinyOS e FreeRTOS.

PALAVRAS-CHAVE: IoT; Securança. Contiki; TinyOS. RIOT-OS. FreeRTOS.

RESUMEN

El Internet de las cosas permite que las personas y los objetos se conecten en cualquier momento, en cualquier lugar, con cualquier objeto a cualquier persona, utilizando cualquier ruta / red y cualquier servicio. Por lo tanto, conduce a una heterogeneidad desafiante de componentes y redes. Se desarrollaron diferentes sistemas operativos para dispositivos IoT de gama baja con requisitos estrictos impuestos principalmente por la baja capacidad de procesar y almacenar información en

[1] Student doctoral program at the Department of Informatics and Engineering of the University of Coimbra (UC). Professor at the Federal Institute of Education, Science, and Technology of Rio Grande do Norte (IFRN), Brazil. Master's degree at the State University of Rio Grande do Norte (UERN), Brazil. Researcher on privacy and security in Healthcare (EHR) systems.

[2] Assistant Professor at the Department of Informatics and Engineering of the University of Coimbra (UC). PhD from UC. Researcher in software security and dependability within the Centre for Informatics and Systems of University of Coimbra (CISUC).

comparación con una máquina convencional. Por lo tanto, el sistema operativo debe ser capaz de realizar tareas de la manera más eficiente posible. En redes heterogéneas, como en el caso de IoT, es más complejo garantizar la seguridad y privacidad de los sistemas que forman parte de este ecosistema. La funcionalidad principal de IoT se basa en el intercambio de información entre cientos o incluso millones de objetos con Internet. Este trabajo realiza una revisión comparativa de las principales características de seguridad disponibles en sistemas operativos orientados a IoT de gama baja, incluidos Contiki, RIOT-OS, TinyOS y FreeRTOS.

PALABRAS CLAVE: IoT. Seguridad. Contiki. TinyOS. RIOT-OS. FreeRTOS.

INTRODUCTION

The Internet of Things (IoT) is a paradigm that is based on a world with different physical objects embedded with sensors and actuators, connected by wireless networks over the Internet, providing support to a network of intelligent objects capable of performing various processes, such as: capturing environment variables, temperature and reacting to external stimuli (SILVA *et al.,* 2021). These objects or things, as they are called, interconnect with each other and with other resources (physical or virtual) and can be controlled through the Internet, providing the emergence of a wide range of possible applications which can obtain data, services, and available operations (ATZORI *et al.* 2010)**.** By 2020, there will be between 50 and 100 billion Internet-connected devices, between smartphones and PCs (CHARITH *et al*., 2015).

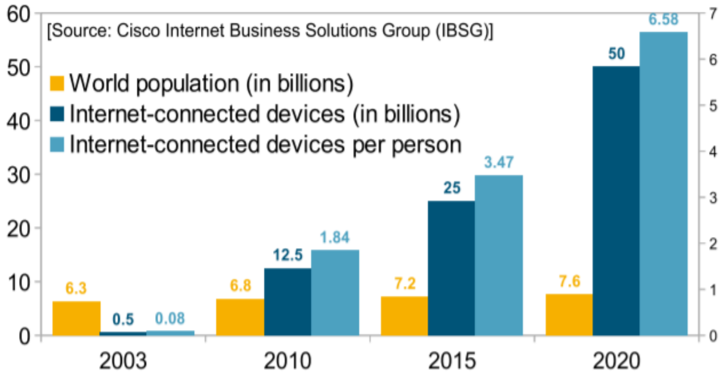

Fig. 1. The Growth of Interconnected Devices by the Year 2020 (adapted from Charith *et al.* 2015)

The Internet of things allows people and objects to be connected anytime, anywhere, with any object to anyone, using any path/network and any service. Besides that, it implies an approach with elements of content convergence, repositories, computing, communication, and connectivity, as there is continuous interconnectedness between people and objects and/or between objects and objects (VERMESAN *et al*., 2011).

SOs must support low-memory hardware architectures, heterogeneous network and communication capabilities, efficient power management, and real-time operations (RTOS). During the construction phase of SOs, technical resources must provide for their structural design, escalation, memory allocation, buffer management, network support, and a programming model. Properties not involving technical issues in the SOs project focus on standards issues, certifications, documentation, licensing, code maturity, and operating system service providers.

In this paradigm, there are operating systems for IoT. Different from the operating systems for standard computers (Desktops). IoT operating systems must meet different prerequisites, mainly due to the low ability to process and store information compared to a conventional operating system. In this sense, SOs should be able to perform tasks more optimally and efficiently as possible (UMER *et al*. 2019).

In the connection between operating systems and IoT, Security presents a significant challenge for IoT-based deployments. In heterogeneous networks, as in the case of IoT, it is more complex to guarantee the security and privacy of systems that are part of this ecosystem. The core functionality of IoT is based on exchanging information between hundreds or even millions of Objects with the Internet (AL-FUQAHA *et al.,* 2015). A problem already considered typical in relation to security in IoT refers to the lack of open standards for distributing encrypted keys between devices (ISHAQ *et al.,* 2013).

While many surveys are being conducted over IoT, there is a need for much more effort to make it safer. The growing attention of governments, companies, and industries has led to various security-related research projects on IoT devices. Thus, it is necessary to keep devices available on already working devices (HOEPMAN; JACOBS, 2007).

In this context, this article aims to research the main security features available in low-end IoT operating systems. The analyzed systems will be Contiki, RIOT-OS, TinyOS, and FreeRTOS.

RELATED WORK

This article (Hahm *et al.* 2016) analyzed the main requirements that SOs must satisfy to be executed on Low-end IoT devices and which SOs would be good candidates. The authors demonstrated in the article that SOs could become a system for open-source IoT devices with standard features for all IoT devices. Whereas the article (YOUSAF et al. 2019) features a brief general vision of the different SOs for IoT. At the same time, it compares the hardware supported by the various SOs presented in the article and future directions of search. In addition, This article (ABERBACH *et al.,* 2017) presents a comparative study between known IoT operating systems. As a result, details are demonstrated as the advantages and disadvantages of each, analyzed specific requirements that an operating system for IoT should provide for use in equipment with memory restriction, low power storage capacity, and little processing power. Finally, the authors concluded that RIOT-OS is the most efficient operating system to be utilized in IoT devices compared with the others cited in the article.

All the approaches in the analyzed articles are directed at the analysis of practical aspects in the operation of the OSs and the hardware where this OSs will be executed. In a way, aspects such as security features were not mentioned in the analyzed articles. The contribution of this work is to survey the low-end IoT OSs, and which ones have the most security features and functionality for these devices.

RESEARCH STRUCTURE

The primary purpose of this review is to allow the reader to understand better which security protocols are available for these OSs. The remainder of this research paper is organized as follows. In Section II, we present some related security issues on IoT devices. The chosen methodology is presented in Section III. Section IV provides insight into which features and SOs will be analyzed. Section V reviews the SOs compared in this paper, highlighting the security features of each SO. Section VI are demonstrated the challenges and roadmap of our research. Finally, we present the conclusion at the end of this review.

2. IOT DEVICES SECURITY

Due to the structure of how these IoT devices are developed, mainly by low-resource hardware, it is challenging to implement a security standard for these devices. IoT OSs are also limited precisely by resource constraints, often including the ability to process limited power and memory. These features make it difficult to apply many traditional security features (SHA *et al.,* 2018).

Thus, it is simple to realize how much the ethical challenge satisfies the security requirements of IoT devices. Thus, to achieve a high level of security, innovations have been required for these devices since their construction, including algorithms and protocols of security. Also, protocols provide privacy in communication between the devices and mechanisms of security to protect the systems physically (SABRI; KRIA, 2019).

Besides that, solutions of security based on IP, including IPSec, SSL, HTTPS, and SSH, have challenges working on devices with little ability, such as intelligent sensors that do not directly support protocols based on IP. In other words, when devices with a more remarkable power of processing make the possible implementation of resources more sophisticated in security. Thus, when the device has little power for processing and memory, makes challenging to incorporate security in this type of device (SHARMA *et al*., 2018).

As a result, the devices of IoT are classified into two categories: high-end IoT and low-end IoT. Devices high-end contain more power processing and energy, such as smartphones and Raspberry Pi. On the other hand, low-end devices are restricted due to their limited resources. So, a system operating traditionally, such as Linux, no can run on devices with limited resources. Like this, IoT can only reach all its potential once an operating system default support running these devices at low cost in a heterogeneous network (HAHM *et al*., 2016).

3. METHODOLOGY

This article discusses the operating systems for IoT aimed at low-end devices. In this regard, security features implemented by default or through third-party solutions such as encryption support: IPsec/6LowPAN, SSL/TLS, and DTLS will be reviewed. The OSs that will be compared will be Contiki, TinyOS, FreeRTOS, and RIOT-OS. The different dimensions of their security resource management approaches are studied, so which OSs have the most available security resources must be analyzed. We will also analyze the vulnerabilities found in the CVE and GitHub base for these OSs. Hence, it allows us to compare better which OSs are constant growth and which are becoming obsolete. In figure 2, we can understand which flow will be followed in this article.

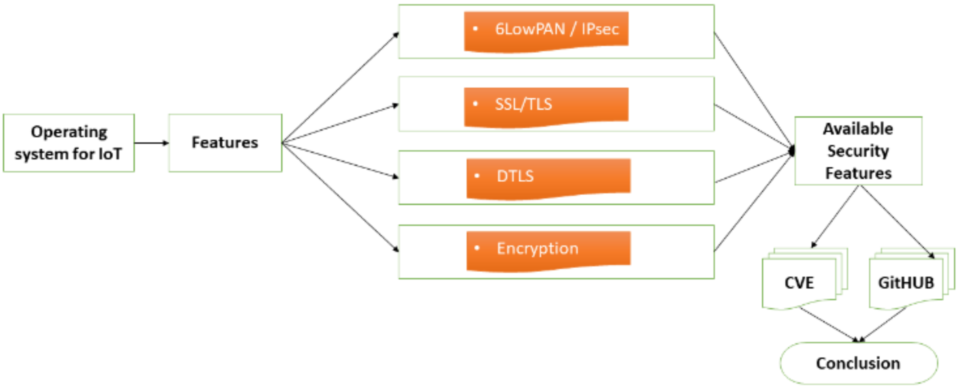

Fig. 2. Research flow

4. FEATURES ANALYZED

The essential security services provision includes confidentiality, integrity, and authentication. These services can be implemented through cryptographic mechanisms, such as block cipher, hash functions, or signature algorithms. There should be a critical management infrastructure for these mechanisms to handle cryptographic keys (HEER et al., 2011).

In the case of IoT, however, security should not only be locked into the required security services but also how they will be made available in the system as a whole and how security features are performed. Because it is a device with low processing power, it is always a challenge to provide security features for these devices. In this article, the safety features analyzed will be demonstrated Table I.

6LowPAN/ IPSec	It is designed to enable IPv6 on IEEE 802.15.4 networks. It uses IPsec, which has two protocols: Encapsulating Security Payload (ESP), which provides authentication, data confidentiality, and message integrity. An authentication Header (AH) provides authentication and data integrity, not confidentiality.
Encryption	Encryption provides an essential toolset to protect data, transactions, and privacy.
SSL/TLS	Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) are encryption protocols designed for the Internet. Enable secure communication between devices.
DTLS	DTLS is derived and inherits some features of TLS. Allows them to reuse TLS security features and make use of UDP. With the advent of CoAP as a specialized web transfer protocol for low-end devices, DTLS is the preferred IoT security protocol for use at the application layer.

TABLE. I: Features analyzed

4.1 CONTIKI

Contiki is an open-source, highly portable, multitasking operating system for embedded systems. Contiki is written in C language and is designed for small-memory microcontrollers. Contiki code is available under a BSD license on GitHub[1]. A typical Contiki configuration uses two kBytes of RAM and 40 kBytes of ROM. Contiki is built for different hardware platforms. Contiki is an open-source operating system for low-end IoT devices (HAHM *et al*., 2016).

It has several network stacks, including the popular µIP stack, known as the TCP/IP implementation for Contiki, with support for IPV4/IPv6, 6LoWPAN, RPL, and CoAP (RUSSEL; DUREN, 2018). Supports IEEE 802.15.4 standard encryption, using AES 128 symmetric key with CBC–CS. ContikiSec is a network layer that provides security and encryption for wireless sensor networks under the Contiki operating system. It supports three security modes: confidentiality only (ContikiSec-Enc), authentication only (ContikiSec-Auth), and encryption authentication (ContikiSec-AE). ContikiSec offers a programming model that allows choosing from three levels of security, depending on the application's need (JSANG *et al*., 2009).

4. 2 RIOT-OS

RIOT-OS is an operating system for low-cost IoT devices. RIOT runs on low-memory devices in the order of 10kByte. Unlike other OSs, RIOT takes a deliberately similar approach to Linux's GNU philosophy regarding code license, vendor independence, and transparency. From a technical perspective, however, RIOT is written from scratch and differs from Linux in terms of operating system architecture. It has support for languages like C and C ++. The source code is available on GitHub[2]. It

[1] to see https://github.com/contiki-os/contiki
[2] to see https://github.com/RIOT-OS/RIOT

supports multi-threading as well. Thus, it is a modular and real-time operating system (BACCELLI *et al*. 2018).

RIOT has several network stacks, including its implementation of the 6LoWPAN stack, as well as 6LoWPAN, RPL, CoAP, and TCP/UDP support, as well as full IPv6 support. As for encryption, it provides a collection of block ciphers with different modes of operation and cryptographic hash algorithms (RUSSEL; DURE., 2018). RIOT supports WolfSSL is a lightweight TLS/SSL library. It adds security, authentication, integrity, and confidentiality to network communications (WOLFSSL, 2021). RIOT was developed in 2012 and has been growing as OS for IoT devices. It has a large open-source community (HAHM *et al.,* 2016).

4.3 FREERTOS

FreeRTOS is free software for RTOS and is a real-time operating system for low-end IoT devices. One of the main features of a multitasking operating system is real-time. The source code is available on GitHub[3]. A real-time operating system is nothing more than software that manages the resources of a computer system to ensure that all events are handled according to their time constraints and managed as efficiently as possible. A real-time operating system has as its main feature its response time, to the detriment of performing hundreds of tasks simultaneously. Response time can be slower than possible, but it must be predictable (TAN *et al*., 2009).

Unlike other low-and IoT OSs, it does not have a network stack. As such, using other developers' stacks to provide connectivity between devices, WolfSSL can provide security, authentication, integrity, and confidentiality to network communications (WOLFSSL 2021). As with RIOT-OS, it is using WolfSSL is optional in RIOT and is only required to implement SSL/ TLS. In the case of FreeRTOS, this is not optional. FreeRTOS was developed in 2002 and is also considered an open-source RTOS most used for IoT devices (HAHM *et al*., 2016).

4.4 TINYOS

TinyOS is an open-source operating system developed for low-power wireless devices such as those used in wireless sensor networks, ubiquitous computing, home networks, smart buildings, and smart meters. Its source code is available online under the BSD license on GitHub[4]. Unlike FreeRTOS, TinyOS does not support real-time applications. It also does not work with multitasking, users, or file system (HICHAM *et al.,* 2017). TinyOS was developed in 2000 and is still one of the most widely used OSs for wireless sensor networks (HAHM *et al*., 2016).

The included BLIP network stack implements 6LoWPAN and IPv6. To incorporate more security in communications. TinySec is designed to be implemented in conjunction with TinyOS. Through TinySec, it can provide confidentiality, integrity, and authentication. Encryption is provided by the Skipjack algorithm used with the CBC operation mode (DENER *et al*., 2014).

[3] to see https://github.com/FreeRTOS/FreeRTOS
[4] to see https://github.com/tinyos/tinyos-main

5. RESULTS

In this section, a set of standards and criteria has been compared. The results are shown in table II.

SOs	Encryption	Integrity	Authentication	Confidentiality	6LowPAN / IPsec	SSL/TLS	DTLS
Contiki	✓	✗	ContikiSec-Auth	ContikiSec-Enc	✓	✓	✓
RIOT-OS	✓	WolfSSL	WolfSSL	WolfSSL	✓	✓	✓
FreeRTOS	✓	WolfSSL	WolfSSL	WolfSSL	✓	✓	✗
TinyOS	✓	TinySec	TinySec	TinySec	✓	✗	✗

Table II: Comparison of the main security standards found

In our research, looking for the Common Vulnerabilities and Exposures (CVE) database, we found some vulnerabilities related to the searched OSs, as shown in Figure 3.

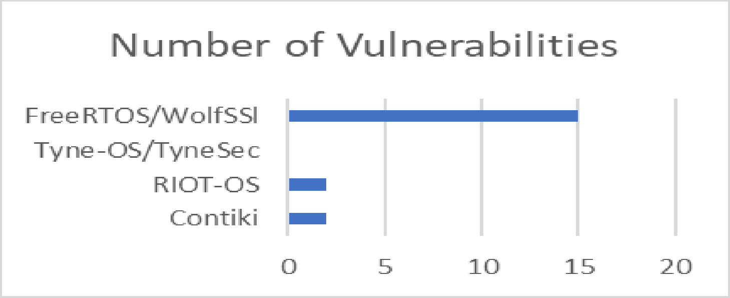

Fig. 3. Vulnerabilities found in CVE

It is noted that WolfSSL has a higher number of reported vulnerabilities in the CVE base. Thus, it is the application that implements security in FreeRTOS. At the same time, we can see that TinyOS and TinySec have no reported vulnerabilities, which makes us think that they are safer and that TinyOS and TinySec are out of use (TINYOS 2021). For this reason, we have no vulnerabilities for these OSs.

We also researched GitHub. GitHub is a project and code version management system and a social networking platform designed for developers. In our search, we look for "issues" that have the following "is issue security is: open" search method when checking each SO, it is clear that RIOT-OS has a higher number of reported security issues, as shown in figure 4. At the same time, TinyOS and TinySec, as explained earlier, are in disuse. Hence, it only shows that the development communities for Contiki and RIOT-OS SOs are very active, making it possible for the system to improve constantly.

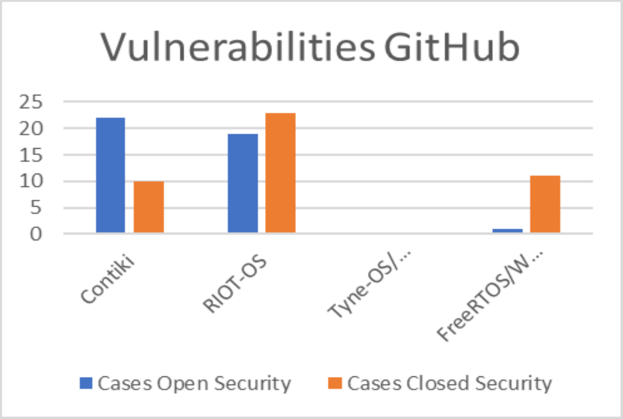

Fig. 4. Vulnerabilities found in GitHub

6. CHALLENGES AND RESEARCH ROADMAP

IoT devices present high-security risks due to their characteristics, such as being dynamic and due to their constant mobility. In addition, IoT devices are highly heterogeneous in communications, protocols, systems, and hardware. Regarding security, the analyzing OSs have means of providing security, either with their cryptographic means or by using a third-party solution. What was noticed is that using encryption as RSA has become very costly to run on devices with limited computing resources, most of the time requiring the use of embedded encryption processors (SHARMA *et al.,* 2017). For this reason, the most used encryption on these devices is encryption using symmetric keys and block encryption.

Hence, one of the critical challenges for IoT device developers is understanding the different forms of interaction between the various protocol models available for IoT devices while choosing the best approach for implementing security in these protocols. There are several options to enable communication between devices. Availability and data security on devices is a challenge on IoT devices (SABRI; KRIA 2019).

In addition, to encryption to protect data during transmission, data security requires policies to control access to data stored on devices. Availability is intended, among other things, to ensure that information is always available for the appropriate use. Thus, other challenges include device-constrained features such as low power storage, battery life, low bandwidth, heterogeneous hardware platforms, and security methodologies that may impede device efficiency.

CONCLUSION

Therefore, we can conclude from the information gathered that, by analyzing the features analyzed and relating the OSs together with the entire OS development community, we can infer that RIOT-OS has a more significant number of security features available. Thus, it can provide all features by analyzing such as encryption, 6LowPAN/IPsec SSL/TLS, and DTLS. Its most vital point is how it provides integrity, authentication, and confidentiality through a third-party implementation: WolfSSL.

Our research found that almost all low-end IoT-OS have a minimum of features implemented, some with more features, others with fewer. However, it is essential to note that, for lower memory and processing devices, we did not identify in our research.

The European Union Agency for Network and Information Security (ENISA) reviews best practices to protect the lifecycle of IoT products and servers (KAVALLIERATOS *et al.,* 2019). Whereas, among their best practices, we can highlight security measures that can help ensure communications security by properly implementing protocols: encrypt communication, industrial segment-based plants, isolate safety networks from business and control networks, avoid the ones with known vulnerabilities (e.g., Telnet, SNMPv1 or v2). Thus, ensure security capabilities and interoperability between protocols when implementing different protocols for various devices within the same system. Hence, limiting the number of protocols implemented within a given environment and disabling unused default network services.

Consequently, ensure a safe environment for key exchange and management, avoiding sharing cryptographic keys across multiple devices. Ensure proper and effective use of cryptography to protect confidentiality, authenticity, and/or integrity of data and information (including control messages) in transit and at rest. Ensure the proper selection of standard and robust encryption algorithms and vital keys and disable insecure protocols. Verify the robustness of the implementation.

In future work, we plan to survey a more significant number of OSs and make a comparison so that we can pen test those OSs to analyze how safe each OS is and if their security implementations are in line with safety assumptions.

REFERENCES

AL-FUQAHA, A.; GUIZANI, M.; MOHAMMADI, M.; ALEDHARI, M; AYYASH, M. "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications." **IEEE Communications Surveys & Tutorials**, v. 17, n. 4, p. 2347-2376, 2015. doi: 10.1109/COMST.2015.2444095

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The Internet of Things: A survey. **Computer Networks**, v. 54, Issue 15, p. 2787-2805, 2010. ISSN 1389-1286. https://doi.org/10.1016/j.comnet.2010.05.010.

BACCELLI, E. *et al*. "RIOT: An Open Source Operating System for Low-End Embedded Devices in the IoT." **IEEE Internet of Things Journal**, v. 5, n. 6, p. 4428-4440, Dec. 2018. doi: 10.1109/JIOT.2018.2815038

DENER, M. Security Analysis in Wireless Sensor Networks. **International Journal of Distributed Sensor Networks**, 2014. Doi: https://doi.org/10.1155/2014/303501

HAHM, Oliver et al. “Operating Systems for Low-End Devices in the Internet of Things: A Survey.” **IEEE Internet of Things Journal**, v. 3, p. 720-734, 2016. DOI:10.1109/JIOT.2015.2505901

HAHM, Oliver; BACCELLI, Emmanuel; HAUKE; Petersen; TSIFTES, Nicolas. Operating Systems for Low-End Devices in the Internet of Things: A Survey. **IEEE Internet of Things Journal**, v. 3, n. 5, p.720-734, 2016. ff10.1109/JIOT.2015.2505901ff. ffhal-01245551f

HEER, T.; GARCIA-MORCHON, O.; HUMMEN, R.; KEOH, S.L.; KUMAR, S.S.; WEHRLE, K. Security Challenges in the IP-based Internet of Things. Wirel. **Person. Commun**., v. 61, p. 527–542, 2011.

HICHAM, A.; SABRI, A.; JEGHAL, A.; Tairi, H. "A Comparative Study between Operating Systems for the Internet of Things (IoT)," **Trans. Mach. Learn. Artif. Intell**., v. 5, n. 4, 2017.

HICHAM, Aberbach; SABRI; Abdelouahed, ADIL; Jeghal; TAIRI, H. A Comparative Study between Operating Systems (Os) for the Internet of Things (IoT). **Transactions on Machine Learning and Artificial Intelligence**, v. 5, 2017. Doi: 10.14738/tmlai.54.3192.

HOEPMAN, J. H.; JACOBS, B. Increased security through open source Commun. ACM, v. 50, n. 1, p. 79-83, 2007. doi: http://doi.acm.org/10.1145/1188913.1188921

ISHAQ, I. *et al.* "IETF standardization in the field of the Internet of Things (IoT): A survey," **J. Sens. Actuator Netw**., v. 2, p. 235–287, 2013.

JSANG, A.; MASENG, T.; KNAPSKOG, S. J. **Identity and Privacy in the Internet Age**: 14th Nordic Conference on Secure IT Systems, NordSec 2009, Oslo, Norway: Proceedings LNCS sublibrary: Security and cryptology, 2009.

KAVALLIERATOS, Georgios; CHOWDHURY, Nabin; KATSIKAS, Sokratis; GKIOULOS, Vasileios; WOLTHUSEN, Stephen. Threat Analysis for Smart Homes. **Future Internet**, v. 11, p. 207, 2019. 10.3390/fi11100207.

LÉVY-BENCHETON, C.; DARRA, E. TÉTU, G.; DUFAY, G.; ALATTAR, M. "Security and resilience of smart home environments: Good practices and recommendations." **Eur. Union Agency Netw. Inf. Security, Athens, Greece, Rep.**, Dec. 2015. Available: https://www.enisa.europa.eu/publications/security-resilience-good-practices. doi: 10.2824/360120.

OUADJAOUT, A.; MINÉ, A.; LASLA, N.; BADACHE, N. "Static analysis by abstract interpretation of functional properties of device drivers in TinyOS," **J. Syst. Softw**., v. 120, p. 114–132, 2016.

PERERA, Charith; CHI HAROLD, Liu; JAYAWARDENA, Srimal; CHEN, Min. A Survey on Internet of Things From Industrial Market Perspective. **IEEE ACCESS**., v. 2, p. 1660-1679, 2015. Doi: 10.1109/ACCESS.2015.2389854.

RUSSELL, Brian, VAN DUREN, Drew. Practical Internet of Things Security: Design a Security Framework for an Internet Connected Ecosystem. 2nd ed. [S. l.]: Packt Publishing, 2018.

SABRI, C.; KRIAA, L.; AZZOUZ, S. L. "Comparison of IoT constrained devices operating systems: A survey." **Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA**, v. 2017. p. 369–375, Octob. 2018.

SHA, Kewei; WEI WEI, T.; YANG, Andrew; WANG, Zhiwei; SHI, Weisong. On security challenges and open issues in Internet of Things. **Future Generation Computer Systems-The International Journal of eScience**, v. 83, p. 326–37, 2018.

SILVA, Tertulino da; LIMA, R. R.; LEITE, C. Desenvolvimento Seguro de Aplicações Web. **RECIMA21 - Revista Científica Multidisciplinar**, v. 2, n. 3, p. 128–149, 2021. ISSN 2675-6218. https://doi.org/10.47820/recima21.v2i3.156

SINGH, S.; SHARMA, P. K.; MOON, S. Y. *et al.* Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions. **J Ambient Intell Human Comput**, 2017. DOI: https://doi-org.ez139.periodicos.capes.gov.br/10.1007/s12652-017-0494-4

TAN, S. L.; TRAN NGUYEN, B. A. Survey and performance evaluation of real-time operating systems (RTOS) for small microcontrollers. **IEEE Micro**, p. 1–14, 2009. doi: 10.1109/mm.2009.56. TinyOS Documentation Wiki. http://tinyos.stanford.edu/tinyos-wiki/index.php/TinySec. Accessed: 21/10/2021

UMER, T.; REHMANI, M. H.; KAMAL, A. E.; MIHAYLOVA, L. Information and resource management systems for Internet of Things: energy management, communication protocols and future applications. **Future Gener Comput Syst**, v. 92, p. 1021- 1027, 2019.

VERMESAN, O.; FRIESS, P.; GUILLEMIN, P.; GUSMEROLI, S.; SUNDMAEKER, H.; BASSI, A.; I. JUBERT, S.; MAZURA, M.; HARRISON, M.; EISENHAUER, M.; DOODY, P. Internet of Things Strategic Research Roadmap, Cluster of European Research Projects on the Internet of Things. **CERP-IoT**, 2011.

WOLFSSL. **State of the art networking security for embedded systems**. [S. l.]: WOLFSSL, 2021. https://www.freertos.org/FreeRTOS-Plus/WolfSSL/WolfSSL.html. Accessed: 07 jul. 2021.

ZIKRIA, Yousaf; KIM, Sung; HAHM, Oliver; AFZAL, Muhammad Y.; AALSALEM, Mohammed. Internet of Things (IoT) Operating Systems Management: Opportunities, Challenges, and Solution. **Sensors**, v. 8, p. 1-10, 2019. Doi: 10.3390/s1908