PADRÕES DE SEGURANÇA PARA DISPOSITIVOS IOT DE BAIXO NÍVEL: Uma Revisão Comparativa

Laura Regina Morais de Oliveira
lrmo@a.recife.ifpe.edu.br
Me. Anderson Luiz Sousa Moreira
anderson.moreira@recife.ifpe.edu.br

RESUMO

A Internet das Coisas (IoT) permite que pessoas e objetos se conectem a qualquer hora, em qualquer lugar, para qualquer finalidade para qualquer pessoa, usando qualquer caminho / rede e qualquer serviço. Diferentes sistemas operacionais foram desenvolvidos para dispositivos IoT low-end, com estritos requisitos impostos principalmente pela baixa capacidade de processar e armazenar informações quando comparado a uma máquina convencional. Assim, o sistema operacional (SO) deve ser capaz de executar tarefas da forma mais eficiente possível. Em redes heterogêneas, como no caso da IoT, não é simples garantir a segurança e privacidade dos sistemas que fazem parte deste ecossistema. A principal funcionalidade da IoT é baseada na troca de informações entre centenas ou até milhões de objetos com a Internet. Este trabalho tem como objetivo realizar uma análise comparativa dos principais recursos de segurança disponíveis no sistema operacional orientado para IoT, como Contiki, RIOT-OS, TinyOS e FreeRTOS. Assim, também analisaremos as vulnerabilidades encontradas no CVE e Base de segurança GitHub para esses sistemas operacionais. Esses sistemas operacionais foram escolhidos pela razão de serem os sistemas operacionais mais usados no universo IoT (Operational Systems; Top Used IoT Operational System**;** 2021).

Palavras-chave: Contiki. RIOT-OS. TinyOs. FreeRTOS. GitHub. CVE.

ABSTRACT

The Internet of Things (IoT) allows people and objects to be connected anytime, anywhere, for any purpose to any person, using any path/network and any service. Different operating systems were developed for low- end IoT devices, with strict requirements imposed mainly by the low capacity to process and store information when compared to a conventional machine. Thus, the operating system (OS) must be able to perform tasks as efficiently as possible. In heterogeneous networks, as in the case of IoT, it is not simple to guarantee the security and privacy of the systems

Instituto Federal de Pernambuco. Campus Recife. Curso Superior Tecnológico em Análise e Desenvolvimento de Sistemas. 9 de agosto de 2021.

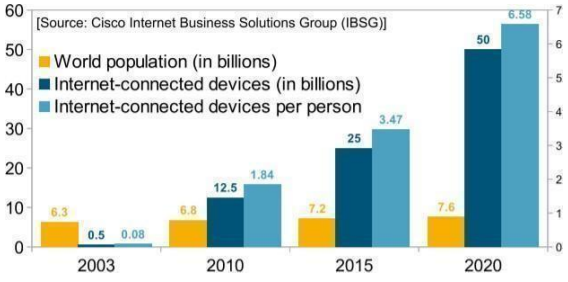
that are part of this ecosystem. IoT's main functionality is based on the exchange of information between hundreds or even millions of objects with the Internet. This work objective performs a comparative analysis of the main security features available in the IoT-oriented operating system such as Contiki, RIOT-OS, TinyOS, and FreeRTOS. Hence, we will also analyze the vulnerabilities found in the CVE and GitHub Security Base for these OSs.

Keywords: Contiki. RIOT-OS. TinyOs. FreeRTOS. GitHub. CVE.

1 INTRODUÇÃO

A IoT é um paradigma que se baseia em um mundo com diferentes objetos físicos embutidos com sensores e atuadores, conectados por redes sem fio pela Internet, fornecendo suporte para uma rede de objetos inteligentes, capazes de realizar diversos processos, tais como: capturando variáveis de ambiente, temperatura e reagindo a estímulos externos. Esses objetos, ou coisas como são chamados, se interconectam entre si e com outros recursos (físicos ou virtuais) e podem ser controlados pela Internet, proporcionam o surgimento de uma ampla gama de aplicações possíveis, que podem obter dados, serviços e operações disponíveis (ATZORI; IERA; MORABITO, 2010).

Figura 1 - O crescimento dos dispositivos interconectados até o ano de 2020.



Fonte: Adaptado de CXSECURITY, 2021.

Estima-se que até 2020 (Figura 1) haverá entre 50 e 100 bilhões de dispositivos conectados à Internet entre smartphones e PCs (VULNERABILITIES, 2021).

O conceito permite que pessoas e objetos sejam conectados a qualquer hora, em qualquer lugar, com qualquer objeto para qualquer pessoa, usando qualquer caminho/rede e serviço. Logo, implica uma abordagem com convergência de conteúdo, repositórios, computação, comunicação e conectividade, pois há interconexão contínua entre pessoas e objetos e/ou entre objetos (MAPPED, 2020).

As SOs precisam oferecer suporte a pouca memória, arquiteturas de hardware, rede heterogênea e recursos de comunicação, gerenciamento de energia eficiente e sistemas operacionais em tempo real (RTOS). Durante a fase de construção de SOs, os recursos técnicos devem fornecer design estrutural, escalonamento, alocação de memória, gerenciamento de buffer, suporte de rede e um modelo de programação. Propriedades que não envolvem questões técnicas no projeto de SOs focam em questões de padrões, certificações, documentação, licenciamento, maturidade de código e

provedores de serviços de sistema operacional (RENAUX, 2016).

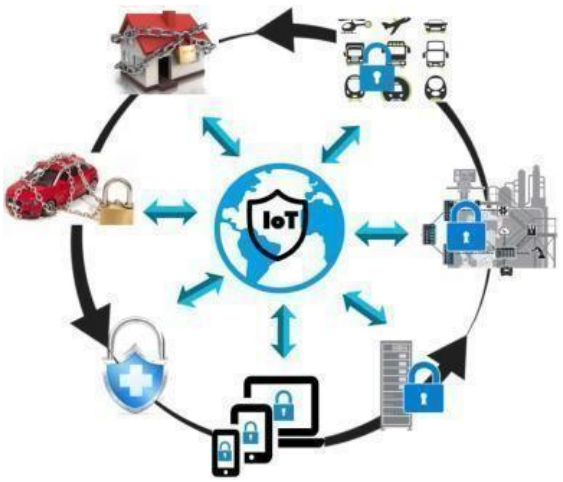
Nesse paradigma, existem sistemas operacionais para IoT. Diferente dos sistemas operacionais para computadores padrão (Desktop). Assim, os sistemas operacionais para IoT devem atender a diferentes pré-requisitos, principalmente devido à baixa capacidade de processar e armazenar informações quando comparados a um sistema operacional convencional. Nesse sentido, os SOs devem ser capazes de realizar as tarefas da forma mais otimizada e eficiente possível (GANTER, 1999).

Na conexão de sistemas operacionais e IoT, a segurança apresenta um desafio significativo para implantações baseadas em IoT. Em redes heterogêneas, como no caso da IoT, não é simples garantir a segurança e a privacidade dos sistemas que fazem parte desse ecossistema. A funcionalidade central da IoT é baseada na troca de informações entre centenas ou até milhões de objetos com a Internet (AL-FUQAHA et al., 2015).

Um problema considerado típico da segurança da IoT refere-se à falta de padrões abertos para distribuição de chaves criptografadas entre dispositivos (MATHEU-GARCÍA, 2019).

Para tornar um sistema IoT idealmente seguro, os métodos de segurança específicos do fornecedor (setas azuis na Figura 2) devem ser substituídos por modelos de segurança regulamentados globalmente usados em todas as plataformas (setas pretas). Embora muitas pesquisas estejam sendo conduzidas na IoT, é necessário muito mais esforço para torná-la mais segura. A crescente atenção de governos, empresas e indústrias levou a uma ampla variedade de projetos de pesquisa relacionados à segurança em dispositivos IoT (AL-FUQAHA et al., 2015).

Figura 2 - Segurança IoT ideal.



Fonte: Adaptado de YOUSEFNEZHAD, et al 2020.

Há uma expectativa de que os dispositivos IoT possam se comunicar obedecendo a altos padrões de segurança e privacidade. Portanto, uma condição e um desafio ao mesmo tempo para SOs para IoT é fornecer os meios necessários para implementar requisitos de segurança e privacidade, por exemplo, bibliotecas de criptografia e protocolos de segurança. Não menos importante, é necessário manter os dispositivos sempre disponíveis nos dispositivos que já estão funcionando (DENNEDY; FOX; FINNERAN, 2014).

Este artigo tem como objetivo pesquisar os principais recursos de segurança disponíveis em sistemas operacionais IoT low-end neste contexto. Os sistemas analisados serão Contiki, RIOT-OS, TinyOS e FreeRTOS.

Esses sistemas operacionais foram escolhidos pela razão de serem

os sistemas operacionais mais usados no universo IoT (Operacional Sistemas; Sistema operacional de IoT mais usado**;** 2021).

2 ANÁLISE DE DADOS

A análise feita neste artigo considerou a segurança do CVE e do Github como fonte de dados, onde:

CVE ou *Common Vulnerabilities and Exposures* é uma lista pública de falhas de segurança que contém informações catalogadas de vários tipos de vulnerabilidades. O site da CVE ajuda os profissionais de TI a priorizar e resolver vulnerabilidades de segurança para tornar os sistemas de computador mais seguros. O CVE foi lançado em 1999 pela corporação MITER para identificar e categorizar vulnerabilidades em software e firmware. Esta entidade fornece um dicionário gratuito para as organizações melhorarem sua segurança cibernética. O MITER é uma organização sem fins lucrativos que opera centros de pesquisa e desenvolvimento financiados pelo governo federal nos Estados Unidos.

O objetivo do CVE[1] é facilitar o compartilhamento de informações sobre vulnerabilidades conhecidas entre as organizações. Ele faz isso criando um identificador padronizado para uma determinada vulnerabilidade ou exposição. Identificadores CVE ou nomes CVE permitem segurança profissional para acessar informações sobre ameaças cibernéticas específicas em várias fontes de informação usando o mesmo nome comum.

O CVE é patrocinado pelo Departamento de Segurança Interna dos Estados Unidos (DHS), Agência de Segurança Cibernética e Infraestrutura (CISA) e US-CERT.

O objetivo do CVE é facilitar o compartilhamento de informações sobre vulnerabilidades conhecidas entre as organizações. Ele faz isso criando um identificador padronizado para uma determinada vulnerabilidade ou exposição. Identificadores CVE ou nomes CVE permitem segurança profissional para acessar informações sobre ameaças cibernéticas específicas em várias fontes de informação usando o mesmo nome comum. O CVE é patrocinado pelo Departamento de Segurança Interna dos Estados Unidos (DHS), Agência de Segurança Cibernética e Infraestrutura (CISA) e US-CERT.

Enquanto GitHub Security[2] é uma licença que permite que profissionais de TI e empresas adicionem recursos de segurança em seus aplicativos / repositórios onde esses repositórios podem ser públicos ou privados. Se o incidente representar um risco para usuários ou clientes externos, o GitHub se compromete a comunicar esse risco, junto com um resumo do incidente, a todas as partes afetadas da maneira mais rápida e abrangente possível.

A equipe de resposta a incidentes de segurança (SIRT) do GitHub identifica, faz a triagem e responde a eventos de segurança de forma rápida e eficaz. O SIRT monitora os sistemas e recursos internos antes que os incidentes aconteçam e responde às

[1] O NVD é o repositório governamental dos EUA de dados de gerenciamento de vulnerabilidade baseados em padrões representados usando o protocolo de automação de conteúdo de segurança (SCAP)- https://nvd.nist.gov/.

[2] GitHub Site de Incidentes - https://github.com/security/incident-response.

questões de segurança relatadas por nossos usuários ou por terceiros.

3 CONTIKI

Contiki é um sistema operacional de código aberto, altamente portátil e multitarefa para sistemas embarcados e esse sistema operacional foi escrito em linguagem C, que é projetado para microcontroladores com uma pequena quantidade de memória. O código do Contiki está disponível sob licença BSD no GitHub e sua configuração usa 2 kBytes de RAM e 40 kBytes de ROM. Contiki é construído para diferentes plataformas de hardware e é um sistema operacional de código aberto para dispositivos IoT de baixo custo (BROWSING…, 2021).

Possui várias pilhas de rede, incluindo a pilha µIP popular conhecida como implementação TCP/IP para Contiki, com suporte para IPV4/IPv6, 6LoWPAN, RPL e CoAP (ADD..., 2020). Suporta criptografia padrão IEEE 802.15.4, fazendo uso de chave simétrica AES 128 com CBC – CS.

ContikiSec é uma camada de rede que fornece segurança e criptografia para redes de sensores sem fio no sistema operacional Contiki. Ele oferece suporte a três modos de segurança: somente confidencialidade (ContikiSec-Enc), autenticação somente (ContikiSec-Auth) e autenticação de criptografia (ContikiSec-AE). A ContikiSec oferece um modelo de programação que dá a opção de escolher entre três níveis de segurança, todos dependendo da necessidade da aplicação (CVE…, 2020).

3.1 Vulnerabilidades

Dentre as principais vulnerabilidades encontradas na utilização do sistema operacional Contiki podemos destacar o fato de problemas relacionados a problemas de rede. Especificamente usando protocolos TCP/IP ou IPV6 para conectar o dispositivo com o mundo externo, como: Out-of-Bound e cabeçalhos de verificação de vulnerabilidades.

A vulnerabilidade Out-of-Bounds foi identificada nos produtos Siemens e está incluída no pacote de vulnerabilidades “AMNESIA:33[3]” a partir desses dispositivos IoT. Isso acontece quando o protocolo TCP/IP não executa a soma de verificação corretamente e pode abrir espaço para o invasor acionar um ataque de negação de serviço da mesma rede à qual o dispositivo IoT está conectado, enviando um pacote IP criado.

Portanto, tome cuidado ao usar um dispositivo Siemens, pois esses dispositivos possuem um conjunto de vulnerabilidades denominado “AMNÉSIA: 33”. É um conjunto de vulnerabilidades encontradas em dispositivos Siemens. Mesmo assim, tendo pacotes de atualizações publicados, o aplicativo pode enfrentar alguns outros problemas além dessa verificação de soma incorreta.

Enquanto a vulnerabilidade de verificação de cabeçalhos leva a uma negação de serviço de um dispositivo confiável e torna possível para um dispositivo não confiável acessar o

[3] AMNESIA:33 – Descrição sobre o conjunto de 33 vulnerabilidades que afetam os sistemas de código aberto ao usar protocolos TCP / IP https://www.forescout.com/research-labs/amnesia33/.

dispositivo IoT devido a essas inconsistências nas conexões IPV6. Isso pode acontecer quando o dispositivo IoT usa o protocolo IPv6 para comunicar o sistema com o mundo externo.

Se um dispositivo não estruturado puder acessar com êxito um dispositivo IoT por meio dessa vulnerabilidade de cabeçalho, isso poderá corromper a memória do dispositivo, criar loops infinitos no dispositivo, abrir espaço para dados não autorizados e/ou corromper o cache DNS. Considere as recomendações abaixo para minimizar o risco de exploração ao usar o Contiki:

Limitar a exposição de todos os dispositivos e/ou sistemas do sistema de controle e garantir que eles não acessem a Internet;

Isole a rede comercial do sistema de controle que possui conexão com a Internet;

Localize as redes do sistema de controle e dispositivos remotos atrás de firewalls.

Quando o acesso remoto for necessário, use métodos seguros, como Redes Privadas Virtuais (VPNs) e certifique-se de usar a versão mais atual disponível para evitar novas vulnerabilidades. Reconhecendo que VPNs podem ter vulnerabilidades, é importante que o sistema seja atualizado para a versão mais recente disponível. Reconheça também que a VPN é tão segura quanto os dispositivos conectados. E lembre-se de que as VPNs também podem ter suas próprias vulnerabilidades.

4 RIOT - OS

RIOT-OS é um sistema operacional para dispositivos IoT de baixo custo. O RIOT é executado em dispositivos com pouca memória na ordem de 10kByte. Ao contrário de outros sistemas operacionais, o RIOT tem uma abordagem deliberadamente semelhante à filosofia GNU do Linux em termos de licença de código, independência de fornecedor e transparência. No entanto, de uma perspectiva técnica, RIOT foi escrito do zero e difere do Linux em termos de arquitetura do sistema operacional. Possui suporte para linguagens como C e C ++. O código-fonte está disponível no GitHub[4]. Ele também suporta multi-threading; é modular e RTOS.

RIOT tem várias pilhas de rede, incluindo sua implementação da pilha 6LoWPAN, 6LoWPAN, RPL, CoAP, suporte TCP/UDP e IPv6 suporte completo. Quanto à criptografia, ela fornece uma coleção de cifras de bloco com diferentes modos de operação e algoritmos de hash criptográficos (ADD…, 2020). RIOT suporta WolfSSL como uma biblioteca TLS/SSL leve. É usado para adicionar segurança, autenticação, integridade e confidencialidade às comunicações de rede (WOLFSSL State of the art, 2016) O RIOT foi desenvolvido em 2012 e vem crescendo como um sistema operacional para dispositivos IoT. Tem uma grande comunidade de código aberto (BROWSING…, 2021).

4.1 Vulnerabilidades

Se a aplicação IoT depende muito das conexões de internet para funcionar corretamente, fique atento

[4] O sistema operacional Contiki - Descrição sobre o sistema e informações: https://github.com/contiki-os/contiki.

para escolher o RIOT-OS. Porque o RIOT-OS tem uma vulnerabilidade bastante crítica cercada pela conexão de internet com o mundo externo, como por exemplo: servidor DNS de estouro de buffer e verificação de falta de limites para datagramas ICMP.

O estouro de buffer do servidor DNS consiste em permitir um ataque através da rede onde o invasor pode ter acesso e controlar o dispositivo através do envio de uma solicitação que irá travar a verificação de limites do servidor e causar um estouro de ponto. Este cenário pode permitir que o invasor tenha acesso à parte comercial do sistema e afetar a funcionalidade correta do dispositivo.

A falta de verificação de limites para datagramas ICMP ocorre nos datagramas ICMP durante o uso do protocolo RPL (protocolo de roteamento IPv6 para redes de baixa potência e com perdas). O RPL consiste em criar uma árvore de tentativas de conexões de sequência de tentativa e erro e considera as baixas capacidades de hardware do conjunto de dispositivos envolvidos na transmissão.

Além da utilização do IPV6 como protocolo principal em suas conexões os datagramas ICMP que são responsáveis por alertar a aplicação sobre erros no processo de entrega de pacotes não funcionam corretamente, pois não há uma verificação confiável nos limites dos pacotes que o IPV6 está circulando entre os dispositivos IoT e o mundo externo.

Isso pode levar a uma parada total do aplicativo ou o aplicativo pode não ser capaz de se conectar a qualquer outro dispositivo porque nunca saberá se aquela rota não está funcionando corretamente porque nunca receberá uma mensagem de erro por meio do protocolo ICMP.

Considerando o cenário apresentado neste estudo, especificamente o RIOT-OS pode ser considerado o sistema operacional mais seguro para ser usado em plataformas IoT. Ele tem a lista de vulnerabilidades mais curta no CVE e até mesmo o Buffer Overflow do servidor DNS pode ser controlado usando uma das bibliotecas abaixo:

- Internet Software Consortium (ISC);
- Berkeley Internet Name Domain (BIND);
- DNS resolver library (libbind);
- Berkeley Software Distribution (BSD);
- DNS resolver library (libc).

5 FREERTOS

FreeRTOS é um software livre para RTOS e é um sistema operacional em tempo real para dispositivos IoT de baixo custo. Tendo como uma das principais características ser um sistema operacional multitarefa é o tempo real. O código-fonte está disponível no GitHub[5]. Um sistema operacional em tempo real é um software que gerencia os recursos de um sistema de computador para garantir que todos os eventos sejam tratados de acordo com suas restrições de tempo e gerenciados da forma mais eficiente possível. Tem como característica principal, o seu tempo de resposta, em detrimento da realização de centenas de tarefas simultaneamente. O tempo

[5] Download FreeRTOS versão 10.4.3 - https://www.arduinolibraries.info/libraries/free-rtos.

de resposta não precisa ser necessariamente o mais rápido possível, mas deve ser previsível (RENAUX et al., 2016).

Ao contrário de alguns outros sistemas operacionais IoT e de baixo nível, ele não tem sua pilha de rede. Como tal, usando pilhas de outros desenvolvedores para fornecer conectividade entre dispositivos, WolfSSL pode ser usado para fornecer segurança, autenticação, integridade e confidencialidade para comunicações de rede (WOLFSSL…, 2016). Tal como acontece com o RIOT-OS, o uso de WolfSSL é opcional no RIOT e só é necessário para implementar SSL/TLS. No caso do FreeRTOS, isso não é opcional.

O FreeRTOS foi desenvolvido em 2002 e também é considerado um RTOS de código aberto mais comumente utilizado para dispositivos IoT (BROWSING…, 2021).

5.1 Vulnerabilidades

O sistema FreeRTOS contém uma vulnerabilidade crítica em seu gerenciamento de fila nas versões anteriores de 10.4.3. Ele não cria mecanismos de verificação para os limites das filas usadas para trocar informações entre tarefas do sistema ou outros dispositivos. Pode abrir espaço para um invasor travar o sistema, vazar informações do sistema de memória para o mundo externo e acessar remotamente o dispositivo, comprometendo-o completamente.

Há uma segunda vulnerabilidade que devemos observar de perto, é a verificação de limites insuficientes durante o gerenciamento de memória heap em FreeRTOS. O gerenciamento deficiente da memória pode permitir que um invasor execute código arbitrário, altere o fluxo do sistema, leia informações confidenciais ou provoque o travamento de um sistema.

Porque quando o gerenciamento de memória heap é usado, é possível que o aplicativo leia e grave dados em qualquer espaço de memória disponível no sistema. Mas com as amplas possibilidades de troca de dados em qualquer espaço de memória, é difícil gerenciar a mudança de dados durante a execução das tarefas.

Considere as recomendações abaixo para minimizar o risco de exploração se usar FreeRTOS:

- Crie uma verificação adicional para os limites dos pacotes;
- Crie uma verificação adicional de limites de heap.

6 TINYOS

TinyOS é um sistema operacional de código aberto desenvolvido para dispositivos sem fio de baixo consumo de energia, como redes de sensores sem fio, computação ubíqua, redes domésticas, edifícios inteligentes e medidores inteligentes. Seu código-fonte está disponível online sob a licença BSD no GitHub4. Ao contrário do FreeRTOS, o TinyOS não oferece suporte a aplicativos em tempo real. Também não funciona com multitarefa, usuários ou sistema de arquivos (HICHAM et. al., 2017). O TinyOS foi desenvolvido em 2000 e ainda é um dos sistemas operacionais mais usados para redes de sensores sem fio (BROWSING…, 2021).

A pilha de rede BLIP incluída implementa LoWPAN e IPv6. Para incorporar mais segurança nas comunicações. O TinySec foi projetado para ser implementado em conjunto com o TinyOS. Por meio do TinySec, é possível fornecer confidencialidade,

integridade e autenticação. A criptografia é fornecida pelo algoritmo Skipjack usado com o modo de operação CBC (TINYSEC, 2009).

6.1 Vulnerabilidades

Como se sabe, os dispositivos embarcados usam gerenciamento de sistema de memória otimizado enquanto trocam informações entre suas tarefas ou quando se comunicam com o mundo externo. O problema reside quando o dispositivo precisa se comunicar por meio de nós de sensores sem fio com outros dispositivos. Abre espaço para uma vulnerabilidade no sistema TinyOS onde o gerenciamento de memória pode misturar os pacotes recebidos das tarefas e dos dispositivos wireless.

Pode permitir que um invasor acesse o dispositivo por meio desse erro na etapa de verificação de pacotes na entrada do nó sem fio e possibilitando a execução de código malicioso no dispositivo IoT ou acessando partes confidenciais do aplicativo. É crucial identificar como os pacotes de comunicação estão sendo tratados e criar condições seguras para gerenciar essas trocas de pacotes para garantir que pacotes externos de um invasor não entrem no dispositivo IoT e assumam o controle do dispositivo, fazendo-o travar ou ter comportamento indesejado.

Existe outra vulnerabilidade realmente crítica encontrada no sistema TinyOS, chamada stack smashing. Essa vulnerabilidade força a pilha do sistema a estourar, colocando mais dados na pilha do que sua capacidade de retenção. É um espaço aberto para que as condições de dados de controle do aplicativo sejam executadas de forma imprevisível e levem a um código malicioso injetado a ser executado no dispositivo.

Considere as recomendações abaixo para minimizar o risco de exploração ao usar o TinyOS:

- Impedir a substituição de dados de controle;
- Impedir a execução de código injetado;
- Endereço e código aleatórios;
- Crie condições para detectar bugs na comunicação da rede.

Como pudemos ver, o universo IoT abre espaço para possibilidades de criação de aplicativos para dispositivos limitados que podem criar sistemas vulneráveis dependendo do sistema operacional escolhido para rodar no dispositivo Iot. Vulnerabilidade facilmente contornada como servidor DNS de estouro de buffer configurado com condições de segurança restritas no RIOT-OS e, por outro lado, tendo problemas ao tentar criar condições para verificar se o estouro de inteiro não acontecerá no FreeRTOS.

7 ANÁLISE CRÍTICA

A análise de vulnerabilidade usada neste trabalho é baseada em boletins oficiais e artigos publicados nos bancos de dados CVE e GitHub, que não atualizam informações com frequência.

Considerando esse cenário para validar o estudo e gerar discussões dos sistemas coletados, foi necessário obter resultados de 3 anos atrás até hoje. A partir dessa análise, é possível afirmar que os sistemas operacionais citados obedecem a uma escala de vulnerabilidade.

Considerando uma escala de 1 a 5 em que 1 é menos crítico e 5 é alto

crítico, temos o Contiki e o FreeRTOS empatados em primeiro lugar, com ambas as vulnerabilidades principais classificadas como 5. O segundo lugar vai para o TinyOS e, por último, mas não menos importante, para o FreeRTOS e, consequentemente, considerado o sistema operacional mais seguro em comparação com os outros três sistemas IoT analisados neste artigo.

A escala utilizada para classificar os sistemas operacionais analisados neste artigo foi baseada na pontuação crítica de vulnerabilidades de segurança do CVE e Github (CVE, 2021; CXSECURITY, 2021).

A Tabela 1 mostra as vulnerabilidades classificadas por seu status de criticidade.

Tabela 1 -Rank das Vulnerabilidades

Sistema IoT	Vulnerabilidade	Status Crítico
Contiki	Out-of-Bound	5
Contiki	Headers Check	5
FreeRTOS	Integer Overflow	5
FreeROTS	Insufficient Bounds Check of Heap Memory Management	5
TinyOS	Stack Smashing	5
TinyOS	Packages Networks Mixed with Communications Packages of the Firmware	4
RIOT-OS	Lack of Bounds Check for ICMP Datagrams	4
RIOT-OS	Buffer Overflow DNS Server	3

Fonte: Autor, 2021.

8 CONCLUSÃO

Podemos concluir da informação recolhida que de certa forma, ao analisar as funcionalidades analisadas e relacionar os SOs com toda a comunidade de desenvolvimento de SO, podemos inferir que o RIOT-OS possui um maior número de funcionalidades de segurança disponíveis. Ele pode fornecer todos os recursos por meio de análise, como criptografia, 6LowPAN/IPsec, SSL/TLS e DTLS. Seu ponto mais forte é como ele fornece integridade, autenticação e confidencialidade por meio do uso de uma implementação de terceiros: WolfSSL.

Os sistemas operacionais analisados possuem um mini de segurança implementado, de forma nativa ou por meio de soluções de terceiros, apesar das restrições de processamento, armazenamento e memória de baixo consumo de energia. A Agência da União Europeia para Rede e Segurança da Informação (ENISA)[6] analisa as melhores práticas para proteger o ciclo de vida dos produtos e servidores IoT.

Porém, em relação às melhores práticas, podemos destacar as medidas de segurança que podem ajudar a garantir a segurança das comunicações.

Por exemplo: implementar corretamente os protocolos:

[6] IoT e infraestruturas inteligentes - https://www.enisa.europa.eu/topics/iot-and-smart-infrastructures.

criptografar a comunicação, plantas baseadas em segmentos industriais, isolar as redes de segurança das redes comerciais e de controle, evitar aquelas com vulnerabilidades conhecidas (por exemplo, Telnet, SNMPv1 ou v2). Assim, a garantia de recursos de segurança e interoperabilidade entre os protocolos ao implementar protocolos diferentes para vários dispositivos no mesmo sistema, limita o número de protocolos implementados em um determinado ambiente e desativa os serviços de rede padrão que não são usados.

Garanta um ambiente seguro para troca e gerenciamento de chaves. Assim, compartilhando chaves criptográficas em vários dispositivos. Garanta o uso adequado e eficaz da criptografia para proteger a confidencialidade, autenticidade e / ou integridade dos dados e informações (incluindo mensagens de controle) em trânsito e em repouso. Garanta a seleção adequada de algoritmos de criptografia padrão e fortes e chaves fortes e desative os protocolos inseguros. Verifique a robustez da implementação.

Em trabalhos futuros, pretendemos fazer uma pesquisa sobre um número mais significativo de sistemas operacionais e fazer uma comparação para que possamos realizar um pen test desses sistemas operacionais para analisar o quão seguro é cada sistema operacional e se suas implementações de segurança estão realmente em linha com suposições de segurança.

REFERÊNCIAS

ADD. **Assert for addition overflow on queue creation (#225).** Github.com, 2020. Homepage. Disponível em: <https://github.com/FreeRTOS/FreeRTOS-Kernel/commit/47338393f1f79558f6144213409f09f81d7c4837> Acesso em: 5 de maio de 2021.

ALABA, Fadele Ayotunde; OTHMAN, Mazliza; HASHEM, Ibrahim Abaker Targio; ALOTAIBI, Faiz. **Internet of Things security: a survey**. Journal Of Network And Computer Applications, [S.L.], v. 88, p. 10-28, jun. 2017. Elsevier BV. Disponível em: <http://dx.doi.org/10.1016/j.jnca.2017.04.002>. Acesso em: 10 de maio de 2021.

AL-FUQAHA, Ala; GUIZANI, Mohsen; MOHAMMADI, Mehdi; ALEDHARI, Mohammed; AYYASH, Moussa**. Internet of Things: a survey on enabling technologies, protocols, and applications.** Ieee Communications Surveys & Tutorials, [S.L.], v. 17, n. 4, p. 2347-2376, 2015. Institute of Electrical and Electronics Engineers (IEEE). Disponível em: <http://dx.doi.org/10.1109/comst.2015.2444095>. Acesso em: 6 de junho de 2021.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. **The Internet of Things: a survey.** Computer Networks, [S.L.], v. 54, n. 15, p. 2787-2805, out. 2010. Elsevier BV. Disponível em: <http://dx.doi.org/10.1016/j.comnet.2010.05.010>. Acesso em: 5 de abril de 2021.

BACCELLI Emmanuel, HAHM Oliver, GÜNES Mesut, WÄHLISCH Matthias, SCHMIDT Thomas. **RIOT OS: Towards an OS for the Internet of Things**. In: The IEEE International Conference on Computer Communications (INFOCOM), 32., 2013, Turin. Anais. Italy: IEEE. Disponível em: <https://hal.inria.fr/hal-00945122>. Acesso em: 30 de junho de 2021.

BELLEZA, Rafael Raymundo; PIGNATON, Edison. **Performance study of real-time operating systems for internet of things devices.** Iet Software, [S.L.], v. 12, n. 3, p. 176-182, jun. 2018. Institution of Engineering and Technology (IET). Disponível em: <http://dx.doi.org/10.1049/iet-sen.2017.0048>. Acesso em: 08 de junho de 2021.

BERTOLOTTI, Ivan Cibrario; KASHANI, Gilda Ghafour Zadeh. **On the performance of open-source RTOS synchronization primitives.** In: 2015 Ieee international forum on research and technologies for society and industry leveraging a better tomorrow (RTSI), 1., 2015, Turin. Anais. Italy: IEEE. p. 398-402.

BROWSING; **security vulnerabilities in the GitHub Advisory Database.** Github, 2021. Homepage. Disponível em: <https://docs.github.com/en/code-security/supply-chain-security/managing-vulnerabilities-in-your-projects-dependencies/browsing-security-vulnerabilities-in-the-github-advisory-database>. Acesso em: 13 de junho de 2021.

CVE. **common vulnerabilities and exposures, the de facto international standard for vulnerability identification and naming**. Homepage. cve.mitre.org, 2021. Disponível em: <https://cve.mitre.org/index.html>. Acesso em: 28 de junho de 2021.

CXSECURITY. **Vulnerability cve-2021-31571.** Cxsecurity, 2021. Homepage. Disponível em: <https://cxsecurity.com/cveshow/CVE-2021-31571/>. Acesso em: 26 de junho de 2021.

DENNEDY, Michelle Finneran; FOX, Jonathan; FINNERAN, Thomas R. **Technology Evolution, People, and Privacy.** The Privacy Engineer's Manifesto, [S.L.], v. 1, n. 1, p. 3-24, 12 fev. 2014. Disponível em: <http://dx.doi.org/10.1007/978-1-4302-6356-2_1>. Acesso em: 15 de maio de 2021.

GANTER, Bernhard; WILLE, Rudolf. **Formal Concept Analysis: mathematical foundations.** Berlin: Springer Book Archive, 1999. 284 p.

HICHAM, Aberbach; SABRI, Abdelouahed; JEGHAL, Adil; TAIRI, Hamid. **A Comparative Study between Operating Systems (Os) for the Internet of Things (IoT).** Transactions On Machine Learning And Artificial Intelligence, [S.L.], v. 5, n. 4, p. 280-290, agust 31, 2017. Scholar Publishing. Disponível em: http://dx.doi.org/10.14738/tmlai.54.3192. Acesso em: 06 de junho de 2021.

IMPROVE. **Heap2 bounds checking (#224).** Github.com, 2020. Homepage. Disponível em: <https://github.com/FreeRTOS/FreeRTOS-Kernel/commit/c7a9a01c94987082b223d3e59969ede64363da63>. Acesso em: 07 de maio de 2021.

KOVACS, Eduard; **Freertos vulnerabilities: Expose Many Systems to Attacks.** Securityweek, 2018. Homepage. Disponível em: <https://www.securityweek.com/freertos-vulnerabilities-expose-many-systems-attacks>. Acesso em: 02 de julho de 2021.

MAPPED. **vulnerability in RIOT-OS.** GitHub, 2020. Homepage. Disponível em: <https://github.com/FreeRTOS/FreeRTOS/commit/b877e4ec478de2c24d07ab46241070d7c66f375c?branch=b877e4ec478de2c24d07ab46241070d7c66f375c>. Acesso em: 25 de maio de 2021

MATHEU-GARCÍA, Sara N.; HERNÁNDEZ-RAMOS, José L.; SKARMETA, Antonio F.; BALDINI, Gianmarco. **Risk-based automated assessment and testing for the cybersecurity certification and labelling of IoT devices**. Computer Standards & Interfaces, [S.L.], v. 62, p. 64-83, fev. 2019. Elsevier BV. Disponível em: <http://dx.doi.org/10.1016/j.csi.2018.08.003>. Acesso em: 21 de maio de 2021.

NIST. **National vulnerability database.** Nvd.Nist.gov. 2021. Homepage. Disponível em: <https://nvd.nist.gov/vuln/detail/CVE-2021-31571>. Acesso em: 08 de junho de 2021.

OUADJAOUT, Abdelraouf; MINÉ, Antoine; LASLA, Noureddine; BADACHE, Nadjib. **Static analysis by abstract interpretation of functional properties of device drivers in TinyOS.** Journal Of Systems And Software, [S.L.], v. 120, p. 114-132, October, 2016. Elsevier BV. Disponível em: <http://dx.doi.org/10.1016/j.jss.2016.07.030>. Acesso em: 06 de junho de 2021

RENAUX, Douglas P.B.; POTTKER, Fabiana; SOARES, Claudio E.; VALERIO, Cristiano C. **A State-Based Function-Queue Software Architecture for Electric Motor Control.** In: Ieee international symposium on real-time distributed computing (ISORC), 19., 2016, York: IEEE. p. 229-236.

RIOT-OS. **Project Team.** Riot-os.org, 2021. Homepage. Disponível em: <https://www.riot-os.org>. Access on: 21 de junho de 2021.

TINYSEC. **Tinyos.stanford.edu**. 2009. Homepage. Disponível em: <http://tinyos.stanford.edu/tinyos-wiki/index.php/TinySec>. Acesso em: 25 de junho de 2021.

VULMON. **Vulmon Search is a vulnerability search engine**. cve-2021-31572. Vulmon, 2021. Homepage. Disponível em: <https://vulmon.com/vulnerabilitydetails?qid=CVE-2021-31572>. Acesso em: 08 de junho de 2021.

WOLFSSL. **State of the art networking security for embedded systems.** Openrtos, 2016. Homepage. Disponível em:<http://www.openrtos.org/FreeRTOS-Plus/WolfSSL/WolfSSL.shtml>. Acesso em: 29 de junho de 2021.

YOUSEFNEZHAD, Narges; MALHI, Avleen; FRÄMLING, Kary. **Security in product lifecycle of IoT devices: a survey.** Journal Of Network And Computer Applications, Amsterdã, v. 171, n. 102779, 01 december 2020. Elsevier BV. Disponível em: <http://dx.doi.org/10.1016/j.jnca.2020.102779>. Acesso em: 22 de junho de 2021.