



Notas Técnicas

CBPF-NT-003/22

junho 2022

Comunicações seguras entre dispositivos IoT utilizando o ESP32a

Alan Franco Rosa, David Vaz Teixeira, e Nilton Alves Júnior

Comunicações seguras entre dispositivos IoT utilizando o ESP32

Alan Franco Rosa^{*,*}, David Vaz Teixeira^{†,†} and Nilton Alves Júnior^{‡,‡}

Laboratório de Instrumentação, Informação e IoT (Lab3I)

Centro Brasileiro de Pesquisas Físicas (CBPF),

Rua Dr. Xavier Sigaud 150,

CEP: 22290-180, Urca, RJ, Brasil

Submetido em 20 de maio de 2022

Aceito em 25 de maio de 2022

Resumo: Neste trabalho foi desenvolvida uma metodologia para aumentar a segurança na comunicação em *Internet of Things* (IoT) que utilizem o ESP32. No projeto desenvolvido foram implementadas criptografias na camada de comunicação, utilizando o protocolo TLSv1.2, além da criptografia do *firmware*, para mitigar ataques físicos com o intuito de inserir um *firmware* modificado.

Palavras chave: Segurança, Instrumentação, IoT, ESP32.

Abstract: In this work, a methodology was developed to increase communication security in *Internet of Things* (IoT) projects that uses ESP32. In this project we have implemented encryption in the layer communication, using the TLSv1.2 protocol, besides firmware cryptography, to mitigate physical attacks that have the objective to inserting a modified firmware.

Keywords: Security, Instrumentation, IoT, ESP32.

I. INTRODUÇÃO

Com a crescente quantidade de equipamentos conectados à Internet, também aumentam os números relativos a ataques que exploram as vulnerabilidades desses dispositivos. Com isso, o padrão IEEE 2413-2019 [1] destacou a importância da segurança, definindo-a como parte da natureza dos dispositivos e aplicações em *Internet of Things* (IoT).

No ambiente IoT, geralmente as questões relacionadas com a segurança são negligenciadas, seja pela limitação de *hardware*, pelas dificuldades técnicas envolvidas ou por se julgar desnecessário. A grande quantidade de tecnologias que podem ser utilizadas também dificultam a determinação de uma infraestrutura padronizada [2], sendo necessário que se avalie cada caso.

Em 2018 foi publicado pelo *Open Web Application Security Project* (OWASP) [3] o *Top Ten* das vulnerabilidades mais comuns em IoT, dentre as quais podem ser destacadas as que envolvem a transferência de dados sem criptografia e dispositivos fisicamente vulneráveis. Com a preocupação de melhorar a segurança em dispositivos IoT houve um trabalho desenvolvido pelo *National Institute of Standards and Technology* (NIST) dos Estados Unidos, que buscou definir quais orientações as empresas devem seguir para coibir ataques a esses tipos de dispositivos [4]. Em 12 de Maio de 2022 o presidente dos EUA assinou a Ordem Executiva 14028, resultante desse projeto, para apoiar a segurança cibernética, proteger a infraestrutura crítica e as redes do governo federal [5].

O presente trabalho implementa formas de mitigar ataques que envolvam a interceptação de dados e a alteração do *firmware* em dispositivos IoT que utilizam o ESP32. É esperado que essas implementações sirvam de orientação para interessados em desenvolver projetos semelhantes. Também é importante observar que as soluções desenvolvidas apenas acrescentam novas camadas de segurança, mas existem ataques mais sofisticados que podem ser feitos em laboratórios, como os ataques de *Fault Injection and eFuse protection* [6], *Fault Injection and Secure Boot* [7], *Zero PMK Installation* [8], *EAP DoS* [9] e *Secure Boot Bypass* [10].

* Endereço Eletrônico: alanfr@cbpf.br

† Endereço Eletrônico: davidvaz@cbpf.br

‡ Endereço Eletrônico: naj@cbpf.br

II. ASPECTOS DE SEGURANÇA

Em um ataque do tipo *Man-in-the-middle* (Figura 1), o invasor busca interceptar a comunicação do dispositivo com o servidor [11], para que as informações enviadas sejam adulteradas ou descobertas, de forma passiva (*Eavesdropping*), possibilitando que o atacante amplie o controle da rede através de técnicas de pivoteamento ou de movimento lateral. No ambiente de IoT essa é uma vulnerabilidade passível de ser explorada, porque em geral os dados enviados pelos dispositivos são considerados como de baixa criticidade, fazendo com que os usuários não se preocupem com a criptografia da informação transmitida, devido ao fato de geralmente os dispositivos utilizados possuírem limitações de *hardware* que dificultam a implementação de metodologias seguras, em razão de essas exigirem mais recursos de processamento e memória.

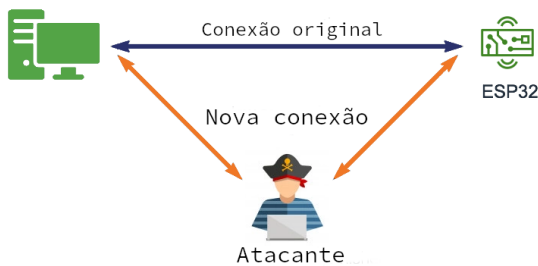


Figura 1. Ataque *Man-in-the-middle*.

Além da segurança associada ao dispositivo, também existe a necessidade de se manter seguro o servidor que faz a comunicação. A escolha do protocolo de comunicação é outra tarefa crítica no sistema, pois ela deve levar em consideração a complexidade para implementação, taxa de transferência necessária e quantidade de dados enviadas por cada dispositivo. A escolha mais comum para os projetos IoT acaba sendo o protocolo *Message Queuing Telemetry Transport* (MQTT) ao invés do tradicional HTTP, pois é um protocolo que possui a natureza de ser utilizado para a coleta de dados entre muitos dispositivos que se conectam a uma infraestrutura computacional para processamento e armazenamento [12].

A característica que torna o MQTT um protocolo amplamente utilizado é o fato de ele necessitar de uma quantidade menor de pacotes para realizar e manter a comunicação. Porém, foi verificado [13] que para sistemas onde a conexão entre os dispositivos e o servidor não é reaproveitada (os dados não são enviados na forma de *stream*), o custo no tráfego acaba sendo muito alto, tornando o protocolo menos eficiente que o HTTP. Sendo assim, devido a natureza das conexões entre os dispositivos de interesse serem de forma intermitente, a utilização de um protocolo com arquitetura *publish/subscribe* não é relevante para a grande maioria dos projetos desenvolvidos pelo laboratório. O protocolo HTTP foi o escolhido por também possuir suporte a comunicações seguras através de criptografia TLS (HTTPS), além de ser

compatível com outros sistemas e aplicações.

III. METODOLOGIA

O ambiente utilizado está esquematizado na Figura 2. O servidor foi implementado em uma máquina virtual (VM Server), utilizando a distribuição Rocky Linux em um sistema VMWare. A *Application Programming Interface* (API) que recebe os dados obtidos pelo ESP32 foi elaborada em Python. A segurança é feita através de um firewall com políticas de filtragem *stateful*, baseada em iptables, sistema de detecção de intrusão (IDS) e a utilização do SELinux, para, em caso de invasão, conter o escalonamento de privilégios. A comunicação com a API é feita através de um proxy, que utiliza um *Web Application Firewall* (WAF). Um bom ponto de partida para implementar uma configuração minimamente segura do servidor web é seguir as recomendações do *Mozilla SSL Configuration Generator* [14].

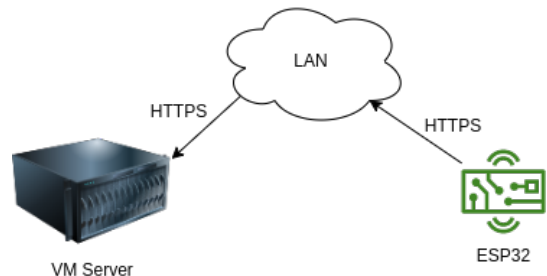


Figura 2. Topologia do ambiente.

A comunicação entre o ESP32 e o servidor é estabelecida através de conexão WiFi, com autenticação baseada na especificação WPA2-PSK. O protocolo de criptografia escolhido foi o TLS v1.2, por ser amplamente suportado pelos sistemas e por muitas bibliotecas disponíveis para o ESP32.

Para captura e análise dos dados de rede foram utilizadas os *softwares* tcpdump [15] e Wireshark [16]. As interfaces de desenvolvimento utilizadas (IDE) foram a do Arduino e o esp-idf.

IV. RESULTADOS

IV.1. Criptografia na comunicação

Inicialmente, para comparação, a comunicação entre os dispositivos foi feita utilizando o protocolo HTTP e todo o tráfego da comunicação foi interceptada com o *tcpdump*. A segurança da comunicação, nesse caso, fica dependente da rede e de suas políticas de acesso.

Na Figura 3 é possível observar o resultado da interceptação, com o fluxo da informação não criptografada.

tografada, sendo possível identificar quando os dados são enviados (POST) e a rota utilizada (/medidores/).

```

74 45898 → 8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_
58 8000 → 45898 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS
60 45898 → 8000 [ACK] Seq=1 Ack=1 Win=29200 Len=0
418 POST /medidores/ HTTP/1.1 (application/json)
54 8000 → 45898 [ACK] Seq=1 Ack=365 Win=30016 Len=0
76 8000 → 45898 [PSH, ACK] Seq=1 Ack=365 Win=30016 Len=22
60 45898 → 8000 [ACK] Seq=365 Ack=23 Win=29200 Len=0
500 HTTP/1.1 201 Created (application/json)
60 45898 → 8000 [ACK] Seq=365 Ack=469 Win=30016 Len=0
60 45898 → 8000 [FIN, ACK] Seq=365 Ack=469 Win=30016 Len=0
54 8000 → 45898 [FIN, ACK] Seq=469 Ack=366 Win=30016 Len=0
60 45898 → 8000 [ACK] Seq=366 Ack=470 Win=30016 Len=0
74 45900 → 8000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_

```

Figura 3. Intercepção da comunicação não criptografada (HTTP) entre o ESP32 e o servidor, utilizando o Wireshark.

Ao analisar o fluxo, fica claro a facilidade de recuperar informações sensíveis, como o formato em que os dados são enviados (JSON), o token de validação e as informações enviadas (Figura 4).

```

Internet Protocol Version 4, Src: 152.84.1.1, Dst: 152.84.1.1
Transmission Control Protocol, Src Port: 45898, Dst Port: 8000, Seq: 1, Ack: 1, Len: 364
Hypertext Transfer Protocol
    p: .E, 1 .....E
    ..dn@0 .....T 6.T
    ..J@0 .....P
    ..Y..PO ST /medi
    ..dores/ HTTP/1.1
    ..Host: 152.84.1.1
    ..User-Agent: curl/7.6
    ..Content-Type:
    ..application/json
    ..Authorization:
    ..Token d 95ff5aeb
    ..9df6eeb3
    ..Content-Length:
    ..154
    ..["local"
    .."Local-Teste"
    .."name": "Teste"
    .."ip": "0.10."
    .."mac": "00
    ..:24:
    .."rede": "Cbpfi
    ..ot", "ve rsao": "
    ..3.2.1"
    .."compila
    ..do": "06 /10/2021
    ..")

```

Figura 4. Dados não criptografados enviados pelo ESP32 para a API.

A programação para a comunicação através do protocolo HTTPS foi implementada de duas formas: a primeira foi com a utilização da IDE do ESP32 [17] (esp-idf), utilizando a biblioteca mbedtls [18]; a segunda foi através da biblioteca WiFiClientSecure [19], utilizando a interface do Arduino.

Com a comunicação segura, foi possível observar que o tráfego de dados é feito de forma criptografada, com a utilização do protocolo TLSv1.2. No fluxo de dados que está na Figura 5 fica claro a diferença em relação aos dados não criptografados (Figura 3), sendo possível notar a etapa do *handshake* da comunicação entre o ESP32 e o servidor.

```

TCP 60 61412 → 4242 [SYN] Seq=0 Win=5744 Len=0 MSS=1436
TCP 58 4242 → 61412 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS
TCP 60 61412 → 4242 [ACK] Seq=1 Ack=1 Win=5744 Len=0
TLSv1.2 357 Client Hello
TCP 54 4242 → 61412 [ACK] Seq=1 Ack=304 Win=30016 Len=0
TLSv1.2 1589 Server Hello, Certificate, Server Key Exchange, Server
TCP 60 61412 → 4242 [ACK] Seq=304 Ack=1536 Win=4209 Len=0
TCP 60 [TCP Window Update] 61412 → 4242 [ACK] Seq=304 Ack=1536
TLSv1.2 161 Client Key Exchange
TLSv1.2 60 Change Cipher Spec
TCP 54 4242 → 61412 [ACK] Seq=1536 Ack=417 Win=30016 Len=0
TLSv1.2 99 Encrypted Handshake Message
TLSv1.2 105 Change Cipher Spec, Encrypted Handshake Message
TLSv1.2 357 Application Data
TLSv1.2 577 Application Data
TCP 54 4242 → 61412 [ACK] Seq=1587 Ack=1288 Win=32160 Len=0
TLSv1.2 1477 Application Data
TCP 60 61412 → 4242 [ACK] Seq=1288 Ack=3010 Win=5744 Len=0
TCP 60 61412 → 4242 [FIN, ACK] Seq=1288 Ack=3010 Win=5744 Len=0
TCP 54 4242 → 61412 [FIN, ACK] Seq=3010 Ack=1289 Win=32160 Len=0
TCP 60 61412 → 4242 [ACK] Seq=1289 Ack=3011 Win=5743 Len=0

```

Figura 5. Intercepção da comunicação criptografada (HTTPS) entre o ESP32 e o servidor, utilizando o Wireshark.

```

No. Time Source Destination Protocol Length Info
920 68.960835 152.84.1.1 152.84.1.1 TCP 54 4242 → 61412 [ACK]
921 68.985765 152.84.1.1 152.84.1.1 TLSv1.2 1471 Application Data
922 RR 993112 152.84.1.1 152.84.1.1 TCP RR 61412 → 4242 [ACK]
Content Type: Application Data (23)
Version: TLS 1.2 (0x0303)
Length: 1412
0190 88 88 ee 78 4a cd e8 0e 30 9f 5d be e5 fd d1 e7 ...xJ...0...
0190 c1 53 ab 9c 7d fe b3 eb 90 25 e3 48 d2 cc 6e 3d ...S)...%H..f=
01a0 ca bc 25 4d b0 4a 73 a3 b3 39 c8 79 da 58 b0 8f ...%MJs...y.X-
01b0 61 30 80 7e bd 83 ac 3a af 38 89 f6 c7 0e 50 84 ...a0...-8...Pd
01c0 52 d0 80 8b 9d bc 5e ff 9b 90 c0 37 aa 2a be a6 ...R...A...7...
01d0 5c 75 08 23 25 d1 42 db 6d 21 6d d6 4a 58 f5 53 ...u...%B...mImJX.S
01e0 14 ed 91 b1 98 d7 24 d8 ef 1c 01 dc ef 8e 29 56 ...S...$...4...S...
01f0 32 1e 3b 12 65 b1 a8 18 33 79 46 70 14 85 ff 61 2;...e...3yF...V
0200 4e f0 a1 06 be a0 57 6d f0 79 40 af bb 99 00 35 N...vm...y0...S
0210 c1 1e 80 ae ec ff 9f f0 40 4b f3 e2 c2 1c 75 10 ...@K...u...
0220 f1 33 11 d1 d5 17 3d 89 6b 24 f1 10 42 f9 12 a7 ...3...=kS...B...
0230 24 e4 d9 d0 22 43 e3 c1 ba 34 cd d3 90 fb e9 e4 ...S...C...4...S...
0240 80 05 02 f1 6f 0a aa 68 02 e9 a1 49 1b 50 06 20 ...o...h...I.P...
0250 f7 7a 03 f3 bc d5 2c 46 c4 7c ed e7 c2 b9 6e db ...Z...;F...n...
0260 14 9b 87 19 a6 98 87 12 b3 a1 aa ef ca 9d 22 c1 ...z...;F...n...
0270 79 7a f0 00 55 13 38 0f 64 6f ae c1 90 c5 18 0e ...yz...U...8...do...
0280 d4 a5 39 a6 7f 83 3a 35 20 1c b3 ca db bf 73 ff ...9...S...S...S...
0290 2e 6b 93 2f f0 8d 4c 68 60 55 03 12 2b 4a e6 08 ...k...;Lh...U...+...d
02a0 84 10 7a 7c 73 6e 47 09 fc 9f ad cf 56 1b 64 aa ...z...;sn0...;V...d
02b0 00 b7 cd d3 fb a2 41 75 0d 00 b9 8f ae bc 8f 2c ...-...Au...;...
02c0 09 f1 46 10 81 ec de 66 77 fc 59 be 68 9b 8f a8 ...F...f...W...Y...h...
02d0 1d b6 8e 62 ae ae cf 5e b7 95 3c 5c 8f 4e cd 94 ...b...A...<...N...

```

Figura 6. Dados criptografados através do protocolo TLSv1.2.

Analisando o fluxo *Application Data* apresentado na Figura 6 é possível observar a melhoria na segurança, já que não é mais possível identificar informações sensíveis na comunicação, em relação ao que está na Figura 4.

IV.2. Criptografia do *firmware*

A taxonomia de uma vulnerabilidade de *hardware* pode ser dividida entre a sua natureza e o seu domínio [20], como pode ser visto na Figura 7.

Caso uma vulnerabilidade de *hardware* possua natureza intencional, significa que ela foi implantada pelo fabricante/desenvolvedor de forma a permitir o acesso ao equipamento. Caso essa natureza não seja intencional, ela pode ser devido a falha do equipamento ou devido a lógica não segura de implementação (*bug*).

Por outro lado, uma vulnerabilidade de domínio lógico é implementada na fase inicial do projeto, enquanto a do

domínio físico é implantada no final do projeto, na parte de design do equipamento.

Dentre as diversas categorias de ameaças, o objetivo com a criptografia do *firmware* é mitigar os ataques de Engenharia reversa e de modificação do *firmware* [21]. Estes tipos de vulnerabilidades estão relacionadas ao domínio físico, onde um atacante que possua acesso ao *hardware* seja capaz de modificar o *firmware* e colocar um outro dispositivo semelhante no lugar.

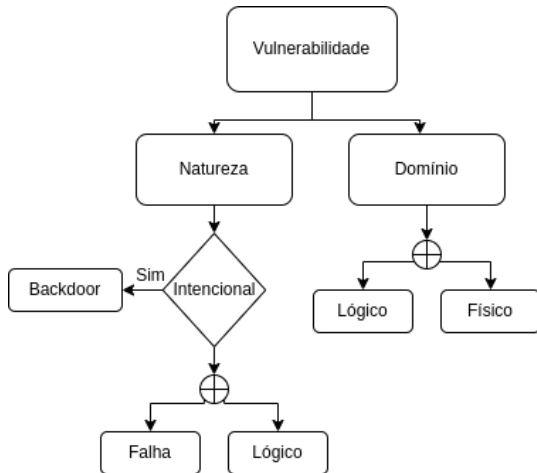


Figura 7. Taxonomia de uma vulnerabilidade de *hardware*.

Como muitas vezes os dispositivos ficam instalados em ambientes isolados e sem vigilância, um invasor de posse do ESP32 pode fazer o *dump* dos dados da memória *flash* e ter acesso a informações sensíveis, além de ser capaz de gerar e implantar um novo *firmware* capaz de se comportar como o original, porém alterando as informações. Por exemplo, um atacante pode mascarar o envio de dados de temperatura, fazendo com que o dispositivo envie informações de que o ambiente está dentro dos limites de temperatura adequados, quando, na verdade, está ocorrendo um incêndio.

Um exemplo de ataque foi o causado pelo *worm* Stuxnet [22]. A dinâmica envolvia infectar o sistema supervisor de uma usina de enriquecimento de urânio do Irã, que era executado em um sistema SCADA, mascarar os valores exibidos e acelerar as centrífugas utilizadas no projeto nuclear iraniano. Com isso foi possível danificar os equipamentos utilizados, prejudicando o programa nuclear daquele país.

Na Figura 8 é possível ver a informação contida no *firmware* a, partir do endereço 0001:0A40, não criptografado, obtida através da extração dos dados da memória. São visíveis os dados da chave utilizada para comunicação com a API, assim como o endereço do servidor e as rotas utilizadas.

Já na Figura 9 está o trecho com os dados para autenticação Wifi. Por questões de segurança algumas partes das informações foram omitidas, mas é possível identificar o SSID no qual o ESP32 deve se conectar (CBPF-IoT), além de trechos da senha de acesso utilizada (-i2-7-).

```

0001:0A40 4C 36 34 78 65 50 6D 41 3D 3D 0A 2D 2D 2D 2D L64xePmA==,-----
0001:0A50 45 4E 44 20 43 45 52 54 49 46 49 43 41 54 45 2D END CERTIFICATE-
0001:0A60 2D 2D 2D 2D 0A 00 68 74 74 70 73 3A 2F 2F 31 35 -----,https://15
0001:0A70 32 3A 34 32 34 2.84.424
0001:0A80 32 2F 6D 65 64 69 64 61 73 2F 00 68 74 74 70 73 2/medidas/.https
0001:0A90 3A 2F 2F 31 35 152.84.4242/medidores
0001:0AA0 38 3A 34 32 34 32 2F 6D 65 64 69 64 6F 72 65 73 8:4242/medidores
0001:0AB0 2F 00 33 2E 30 2E 38 00 4C 61 62 33 69 00 4C 61 /.3.0.8.Lab3i.La
0001:0AC0 62 33 69 2D 00 43 42 50 b3i-1.1.1.1.CBP
0001:0AD0 46 2D 49 6F 54 00 39 32 33 63 35 36 62 39 32 36 F-IoT_923c56b926
0001:0AE0 66 30 66 36 65 65 31 61 63 35 38 30 62 34 39 63 f0f6ee1ac580b49c
0001:0AF0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 4b0
0001:0B00 00 00 80 3F CD CC CC 3D 6F 12 83 3A 95 BF D6 33 ...?iii=o...i03
0001:0B10 7D 1D 90 26 B3 CE 01 0C CD CC CC 3D 0A D7 23 3C }.&^i..iii=.x#<

```

Figura 8. Trecho do *firmware* não criptografado com os dados da apikey.

```

0000:B160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000:B170 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0000:B180 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....
0000:B190 FF FF FF FF FF FF FF FF 08 00 00 00 43 42 50 46 .....
0000:B1A0 2D 49 6F 54 00 00 00 00 00 00 00 00 00 00 00 .....
0000:B1B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000:B1C0 63 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000:B1D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000:B1E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000:B1F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000:B200 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF .....

```

Figura 9. Trecho do *firmware* não criptografado com os dados de acesso wifi.

O ESP32 possui mecanismos de segurança que permitem a criptografia do *firmware* através de seus eFuses [23].

O procedimento adotado nesse projeto foi o de gerar uma chave que deve ser gravada na memória do ESP32. Sendo assim, somente de posse dela que será possível gerar um novo *firmware* capaz de ser executado no ESP32 (Figura 10).

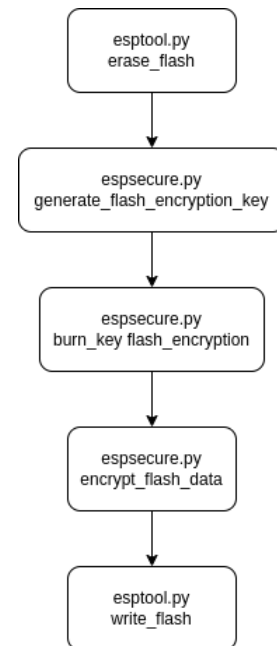


Figura 10. Fluxograma do procedimento de encriptação do *firmware*.

-
- [1] IEEE. Standard for an architectural framework for the internet of things (iot) ieee 2413. *IEEE-2413 Working Group. Technical Report*, 2019.
 - [2] RUSSELL, B.; DUREN, D. V. *Practical internet of things security*. [S.l.]: Packt Publishing Ltd, 2018.
 - [3] OWASP Internet of Things Project. 2018. https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top10. [Online; Maio-2022].
 - [4] IOT Device Criteria. 2021. <https://www.nist.gov/itl/executive-order-improving-nations-cybersecurity/iot-device-criteria>. [Online; Maio-2022].
 - [5] EXECUTIVE Order on Improving the Nations’s Cybersecurity. 2022. <https://www.cisa.gov/executive-order-improving-nations-cybersecurity>. [Online; Maio-2022].
 - [6] CVE-2019-17391 - Fault Injection and eFuse protections. 2019. <https://nvd.nist.gov/vuln/detail/CVE-2019-17391>. [Online; Maio-2022].
 - [7] CVE-2019-15894 - Fault Injection and Secure Boot. 2019. <https://nvd.nist.gov/vuln/detail/CVE-2019-15894>. [Online; Maio-2022].
 - [8] CVE-2019-12587 - Zero PMK Installation. 2019. <https://nvd.nist.gov/vuln/detail/CVE-2019-12587>. [Online; Maio-2022].
 - [9] CVE-2019-12586 - EAP DoS. 2019. <https://nvd.nist.gov/vuln/detail/CVE-2019-12586>. [Online; Maio-2022].
 - [10] CVE-2018-18558 - Secure Boot Bypass. 2018. <https://nvd.nist.gov/vuln/detail/CVE-2018-18558>. [Online; Maio-2022].
 - [11] BUTUN, I.; ÖSTERBERG, P.; SONG, H. Security of the internet of things: Vulnerabilities, attacks, and countermeasures. *IEEE Communications Surveys & Tutorials*, IEEE, v. 22, n. 1, p. 616–644, 2019.
 - [12] GENG, H. *Internet of things and data analytics handbook*. [S.l.]: John Wiley & Sons, 2017.
 - [13] HTTP vs. MQTT: A tale of two IoT protocols. 2018. <https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols>. [Online; Maio-2022].
 - [14] MOZILLA SSL Configuration Generator. 2021. <https://ssl-config.mozilla.org/>. [Online; Maio-2022].
 - [15] TCPDUMP. 2021. <https://www.tcpdump.org/>. [Online; Maio-2022].
 - [16] WIRESHARK. 2021. <https://www.wireshark.org/>. [Online; Maio-2022].
 - [17] ESPRESSIF - HTTPS server. 2021. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_https_server.html. [Online; Maio-2022].
 - [18] MBED TLS. 2021. <https://github.com/ARMmbed/mbedtls>. [Online; Maio-2022].
 - [19] WIFICIENTSECURE. 2021. <https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFiClientSecure>. [Online; Maio-2022].
 - [20] PRINETTO, P.; ROASCIO, G. Hardware security, vulnerabilities, and attacks: A comprehensive taxonomy. In: *ITASEC*. [S.l.: s.n.], 2020, p. 177–189.
 - [21] TEHRANIPOOR, M.; WANG, C. *Introduction to hardware security and trust*. [S.l.]: Springer Science & Business Media, 2011.
 - [22] THE real story of STUXNET. 2013. <https://spectrum.ieee.org/the-real-story-of-stuxnet>. [Online; Maio-2022].
 - [23] ESPRESSIF Docs: Flash Encryption. 2018. <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/flash-encryption.html>. [Online; Maio-2022].
 - [24] MCKAY, K.; COOPER, D. *Guidelines for the selection, configuration, and use of Transport Layer Security (TLS) implementations*. [S.l.], 2017.
 - [25] SSL/TLS Vulnerabilities. 2021. <https://www.hhs.gov/sites/default/files/securing-ssl-tls-in-healthcare-tlpwhite.pdf>. [Online; Maio-2022].
 - [26] SALMON, A.; LEVESQUE, W.; MCLAFFERTY, M. *Applied Network Security*. [S.l.]: Packt Publishing Ltd, 2017.
 - [27] VANHOEF, M.; PIESSENS, F. Key reinstallation attacks: Forcing nonce reuse in wpa2. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. [S.l.: s.n.], 2017, p. 1313–1328.
 - [28] XU, L. et al. Fpga based blockchain system for industrial iot. In: *IEEE. 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. [S.l.], 2020, p. 876–883.

Notas Técnicas é uma publicação de trabalhos técnicos relevantes, das diferentes áreas da física e afins, e áreas interdisciplinares tais como: Química, Computação, Matemática Aplicada, Biblioteconomia, Eletrônica e Mecânica entre outras.

Cópias desta publicação podem ser obtidas diretamente na página web <http://revistas.cbpf.br/index.php/nt> ou por correspondência ao:

Centro Brasileiro de Pesquisas Físicas
Área de Publicações
Rua Dr. Xavier Sigaud, 150 – 4^o andar
22290-180 – Rio de Janeiro, RJ
Brasil
E-mail: alinecd@cbpf.br/valeria@cbpf.br
<http://portal.cbpf.br/publicacoes-do-cbpf>

Notas Técnicas is a publication of relevant technical papers, from different areas of physics and related fields, and interdisciplinary areas such as Chemistry, Computer Science, Applied Mathematics, Library Science, Electronics and Mechanical Engineering among others.

Copies of these reports can be downloaded directly from the website <http://notastecnicas.cbpf.br> or requested by regular mail to:

Centro Brasileiro de Pesquisas Físicas
Área de Publicações
Rua Dr. Xavier Sigaud, 150 – 4^o andar
22290-180 – Rio de Janeiro, RJ
Brazil
E-mail: alinecd@cbpf.br/valeria@cbpf.br
<http://portal.cbpf.br/publicacoes-do-cbpf>